UTRECHT UNIVERSITY

Multi-label text classification of news articles for ASDMedia

Author:
Frank van Meeuwen
ICA-3154025

Supervisors:
dr. A.J. Feelders
prof. dr. A.P.J.M. Siebes
ir. J. Hoeve

in the
Faculty of Science
Department of Information and Computing Sciences

August 2013

UTRECHT UNIVERSITY

Abstract

Faculty of Science
Department of Information and Computing Sciences

by Frank van Meeuwen

With the continuously increasing amount of online information, there is a pressing need to structure this information. *Text classification* (TC) is a technique which classifies textual information into a predefined set of categories.

This thesis describes a case study on classifying news articles on two different datasets collected by the business-to-business news publisher ASDMedia. The goal is to find out if it's possible to use a machine learning (ML) approach to TC to construct a classification system that can be used in a semi-automatic setting. Two main challenges of the cases are that news articles are potentially labeled with multiple categories (multi-label) and the dataset is very imbalanced.

For analytical purposes, we restrict ourselves to ML algorithms that generate humanly interpretable models, namely *decision trees*. We applied state-of-the-art techniques to solve the above mentioned challenges and conduct various experiments. Our focus is on 1) Finding the best feature representation of news articles and 2) Trying out techniques to exploit structures within the class labels; namely *classifier chains* (CC) and *hierarchical top-down classification* (HTC).

By using the optimized feature representation and by applying the CC technique we managed to improve the results substantially for both datasets from a default setup. The best settings reached a *Micro-F1* value of **.625** and **.752** for both ASDMedia datasets.

We can conclude that our constructed classification system is suited to be part of a semi-automated system. However, advisable is to collect more data for the minority categories. Although HTC looked promising and saves a lot of CPU-time, the actual performance was considerably lower than not using it.

Acknowledgements

Writing this master thesis was a time-consuming and challenging task for me. Thanks to the feedback and support of several people it was possible to finish this master thesis.

First of all I would like to thank my supervisor, Ad Feelders at Utrecht University, for guiding me through this project and giving me enough freedom. All the discussions and feedback helped me understood the complex field of machine learning and led me to the right track. I also would like to thank prof. dr. A.P.J.M. Siebes at Utrecht University, for reviewing my work as second supervisor.

Secondly I am gratefully to the owners of ASDMedia, for providing me with the opportunity of doing this case study and sharing to me their domain expertise. I would like to thank in particular my supervisor there, Jol Hoeve, who had many fruitful ideas and supported me during the project.

I also would like to thank my friends and family for supporting me and providing feedback on my thesis. Special thanks for my girlfriend Ederlyn for bearing with me all the time and sharing ideas. Thanks to David-Jan, Jeroen and Ederlyn for proofreading my thesis and providing me with valuable feedback.

Contents

List of Figures

List of Tables

Abbreviations

AFP	**A**ssociated **F**oreign **P**ress
AI	**A**rtificial **I**ntelligence
ANOVA	**An**lysis **o**f **Va**riance
ASD	**A**erospace & **D**efence
BC	**B**inary **C**lassification
BR	**B**inary **R**elevance
CC	**C**lassifier **C**hains
DAG	**D**irected **A**cyclic **G**raph
DR	**D**imensionality **R**eduction
DT	**D**ecision **T**ree
ECC	**E**nsemble of **C**lassifier **C**hains
FC	**F**lat **C**lassification
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
GEW	**G**lobal **E**nergy **W**orld
HC	**H**ierarchical **C**lassification
HMC	**H**ierarchical **M**utli-label **C**lassification
HTC	**H**ierarchical **T**op-Down **C**lassification
IDF	**I**nverse **D**ocument **F**requency
IR	**I**nformation **R**etrieval
LP	**L**abel **P**ower-set
MC	**M**ulti-label **C**lassification
ML	**M**achine **L**earning
MTC	**M**ulti-label **T**ext **C**lassification
NDCS	**N**ews **D**ata **C**onversion **S**ystem

NLP	**N**atural **L**anguage **P**rocessing
SC	**S**ingle-label **C**lassification
SVM	**S**upport **V**ector **M**achine
TC	**T**ext **C**lassification
TF	**T**erm **F**requency
TF-IDF	**T**erm **F**requency - **I**nverse **D**ocument **F**requency
TN	**T**rue **N**egative
TP	**T**rue **P**ositive

Chapter 1

Introduction

1.1 Motivation

In the past decades information that individuals and companies share online has grown and continues to grow rapidly. In addition, the number of people that have access to the Internet continues to grow day by day since 1995 [2]. With all this, often unstructured, online information available, there is an increasing need for techniques that give structure to this information.

An example of such a technique is text classification (TC), which is a technique to classify textual information into a predefined set of categories. Examples of interesting sources for TC are e-mail, news feeds, reviews, forums, blogs and social media. TC has been applied to categorize news articles [3–5] , filter spam e-mail [6, 7] and authorship attribution of texts [8].

In the '80s, when constructing an automatic classification system this was done by building an *expert system*. The system consists of simple 'if-then-else' rules, which are used to classify a piece of data into a category. In other words, the *domain knowledge* of experts is translated to rules which are easily interpretable by a computer system. Although this approach can achieve good performance, a clear drawback is that the rules have to be constructed manually. Usually this is done by people other than the domain-experts, which have explicit knowledge about constructing those rules. This makes it difficult to expand to a new domain or to add new categories.

The machine learning (ML) approach to automatic TC solves this problem. The only thing needed is an example database with pre-classified examples. Instead of manually creating rules, now a model is created automatically by learning from examples. This makes the ML approach more scalable and flexible than manually constructing the model, while achieving the same or even better performance.

1.2 Background

Automatic TC is widely studied in the literature and the ML approach is applied with success. Since the early 90's research has begun and various learning algorithms have been researched, such as: probabilistic methods, DTs/ decision rules, neural networks, nearest neighbor and support vector machines (SVMs). An overview of the ML approach to TC is given in figure 1.1. Consider here a collection of news articles about sports and the problem of classifying a new document with the right type of sport. The ML algorithm will need the training documents with class labels to learn from. The documents are translated to a set of features for each document. With this information the algorithm can construct a model, which can predict the category of previously unseen documents. We will briefly explain the different parts of the learning process next.

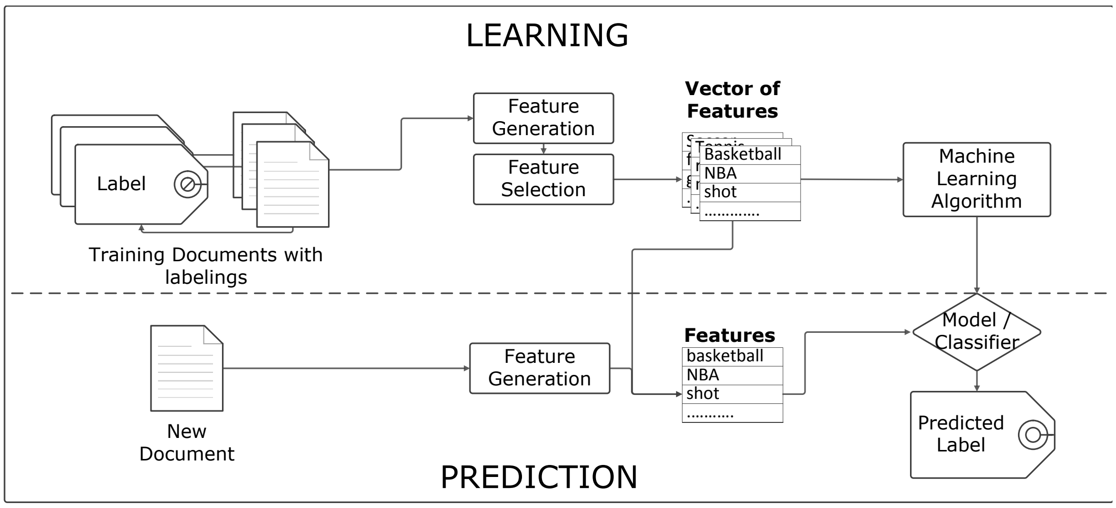


FIGURE 1.1: The supervised learning process applied to TC

1.2.1 Feature generation

In this part of the learning process the textual data (documents) have to be transformed to features. A feature is a distinct property of the data. For example some features of a car are it's color, the manufacturer, the type of engine, etc. For textual data usually a collection of terms and associated weights are used as features. There are different approaches on how to identify terms and how to calculate the weights.

A simple approach, called the *set of words* model identifies terms for each word occurring in the collection of documents. The weight for a term is one if it occurs in the document, otherwise the weight is zero. Note that with this model the order of the words is not maintained. Another possibility is to, in addition to one term for each word, also generate a term for each sequence of two words. Such a sequence of two words is called a 2-gram. In general we can talk of *n-grams*, where n is the number of words. In figure 2.9 you can see an example of how 1-grams and 2-grams are generated from a sentence.

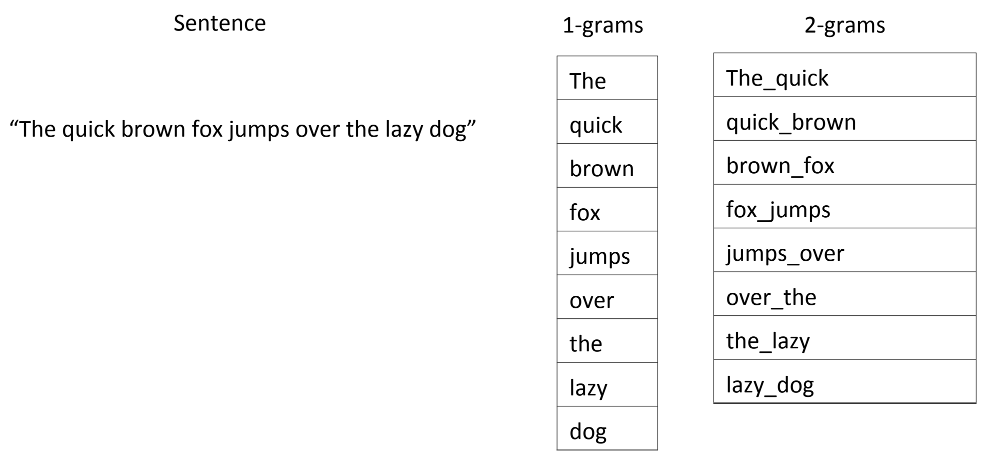


FIGURE 1.2: An example of how 1-grams and 2-grams are generated from a sentence

1.2.2 Feature selection

In the previous section we described how the features are generated from a collection of documents. Depending on how many documents there are available to learn from and the number of words in each document, the number of features that are generated can be enormous. Especially, when in addition to 1-grams (single words), also 2-grams are collected. Most learning algorithms cannot cope with this large number of features. To reduce the term space, *feature selection* is applied.

A popular method that reduces the term space by approximately a factor 10, is the removal of very rare words. Because of their low frequency they probably are not distinctive anyway. Usually this method is used in combination with a statistical approach. There exists many statistical approaches, a popular one being the *Chi Squared* statistic. They all roughly indicate how distinctive a term is to a certain class label. A term can be *positive* distinctive if it co-occurs often with a class label and it does not occur in documents without that class label. If it's the other way around, the term is *negative* distinctive. The number of features that eventually are feed to the learning algorithm are also an important factor. Either too many features, or too few features will result in bad performance. However what the optimal number of features is different for each classification problem.

1.2.3 Categories

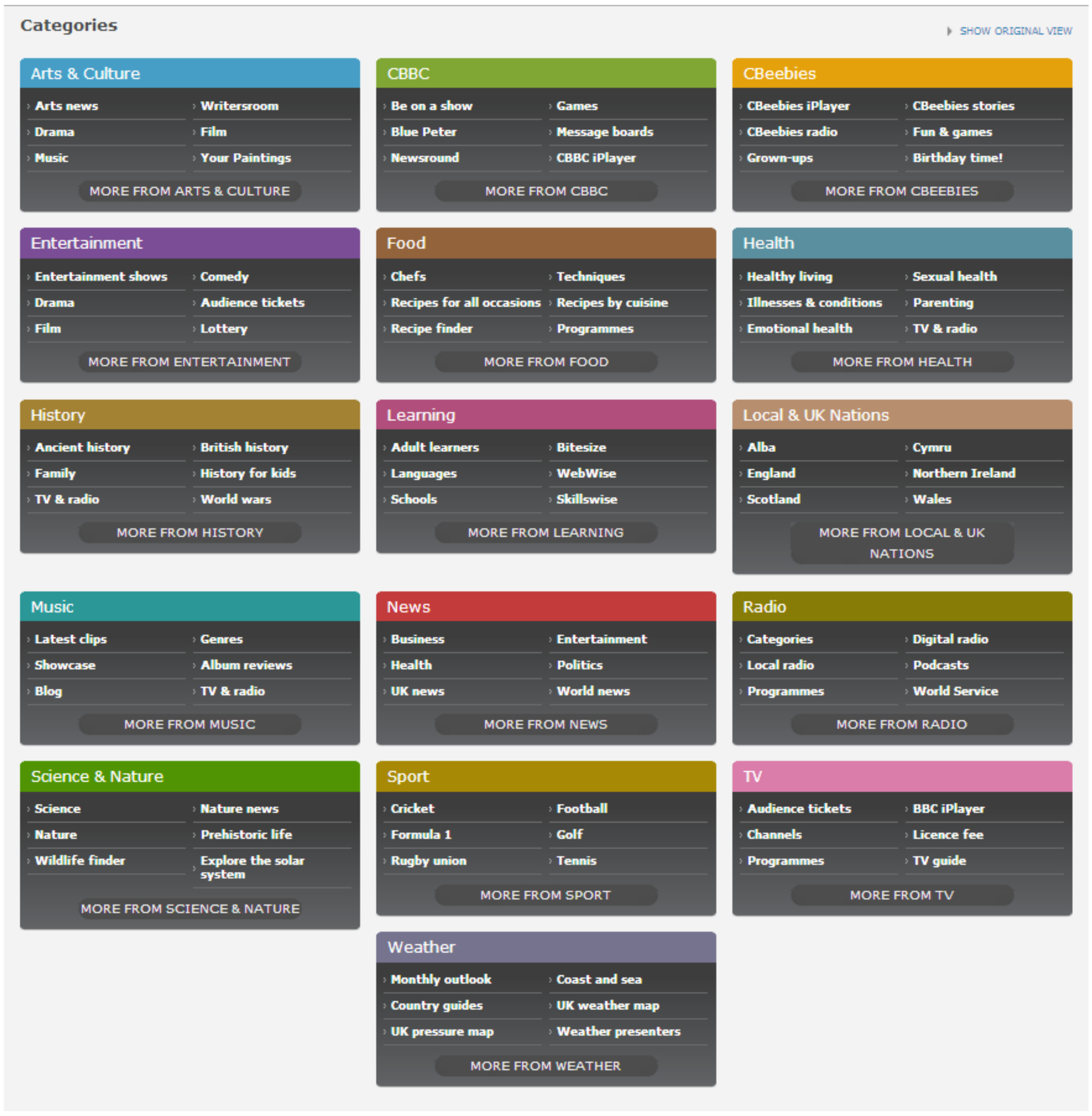
The labellings of categories to examples is an important aspect of the learning process. When the labellings are inconsistent, it will be impossible for a ML algorithm to create a model with good performance. A big difference is whether the examples are labeled with multiple categories (multi-label) or only with a single category (single-label). A multi-label classification (MC) problem is much more involved than a single-label one, which we will explain later in the thesis. An interesting aspect of MC is that the class labels themselves can be correlated. This *internal structure* of class labels can be useful for a learning algorithm.

Sometimes the categories are organized with an *external structure*, such as for example a hierarchy. This actually happens often in the case of topic classification. A good example are the topics from the BBC (`www.bbc.co.uk`), where the topics are organized in a hierarchical way. Figure 1.3 shows the general groups of categories in which more specific topics resides. For example the group Sport, which has topics like 'Cricket', 'Formula-1' and 'Tennis'.

Koller and Sahami [9] were the first to introduce an approach that exploits the hierarchical structure of topics in 1997. They did this by decomposing the problem into multiple sub-problems, one for each node in the hierarchy tree. For each sub-problem a simple classifier is trained, each having its own specialized set of features for that category. This has the advantage that the number of features for each sub-problem can be reduced significantly. Another advantage is that the number of classes and training examples for a sub-problem can be reduced. Nowadays hierarchical classification (HC) is widely studied and has been showed to improve the classification performance in many cases [10]. Another great advantage of HC is that it reduces the learning and predicting time considerably.

1.2.4 Bringing theory to practice

During my part-time job at *ASDMedia*, we were discussing plans for automating parts of their work. A part of their business is to maintain Business-to-Business news websites in the 'Aerospace & Defence' market (`www.asdnews.com`) and in the 'Energy & Resources' market (`www.globalenergyworld.com`). The most work for them lies in the retrieval of news articles and labeling them with the right categories. As a technical artificial



Categories

SHOW ORIGINAL VIEW

Arts & Culture
- Arts news
- Writersroom
- Drama
- Film
- Music
- Your Paintings

MORE FROM ARTS & CULTURE

CBBC
- Be on a show
- Games
- Blue Peter
- Message boards
- Newsround
- CBBC iPlayer

MORE FROM CBBC

CBeebies
- CBeebies iPlayer
- CBeebies stories
- CBeebies radio
- Fun & games
- Grown-ups
- Birthday time!

MORE FROM CBEEBIES

Entertainment
- Entertainment shows
- Comedy
- Drama
- Audience tickets
- Film
- Lottery

MORE FROM ENTERTAINMENT

Food
- Chefs
- Techniques
- Recipes for all occasions
- Recipes by cuisine
- Recipe finder
- Programmes

MORE FROM FOOD

Health
- Healthy living
- Sexual health
- Illnesses & conditions
- Parenting
- Emotional health
- TV & radio

MORE FROM HEALTH

History
- Ancient history
- British history
- Family
- History for kids
- TV & radio
- World wars

MORE FROM HISTORY

Learning
- Adult learners
- Bitesize
- Languages
- WebWise
- Schools
- Skillswise

MORE FROM LEARNING

Local & UK Nations
- Alba
- Cymru
- England
- Northern Ireland
- Scotland
- Wales

MORE FROM LOCAL & UK NATIONS

Music
- Latest clips
- Genres
- Showcase
- Album reviews
- Blog
- TV & radio

MORE FROM MUSIC

News
- Business
- Entertainment
- Health
- Politics
- UK news
- World news

MORE FROM NEWS

Radio
- Categories
- Digital radio
- Local radio
- Podcasts
- Programmes
- World Service

MORE FROM RADIO

Science & Nature
- Science
- Nature news
- Nature
- Prehistoric life
- Wildlife finder
- Explore the solar system

MORE FROM SCIENCE & NATURE

Sport
- Cricket
- Football
- Formula 1
- Golf
- Rugby union
- Tennis

MORE FROM SPORT

TV
- Audience tickets
- BBC iPlayer
- Channels
- Licence fee
- Programmes
- TV guide

MORE FROM TV

Weather
- Monthly outlook
- Coast and sea
- Country guides
- UK weather map
- UK pressure map
- Weather presenters

MORE FROM WEATHER

FIGURE 1.3: The hierarchical topic categories from the BBC website (source `http://www.bbc.co.uk/a-z/`

intelligence (AI) student the question of automating the categorization of the news articles was very interesting.

Apart from facilitating news, *ASDMedia* also sells market reports and event tickets. These three units, news, reports and events could all benefit from a system which automatically retrieves them and labels them with the right category. In the end the plan is to build a semi-automatic system which automatically retrieves news, reports and events and in addition performs some extra tasks on those units. One of the tasks is classification, but also determining whether a unit is relevant to show on the corresponding websites is an important task. Furthermore the text and layout of some units has to be adjusted such that it looks good and can be showed on the website. The idea is that the semi-automatic system supports a human user in those tasks. Figure 1.4

shows roughly the process of the intended News Data Conversion System (NDCS). The goal of this research is to find out if it's possible to use the ML approach to TC for the classification task of the intended NDCS system. The training data for our experiments are the manually labeled news articles from the two news websites mentioned earlier. We will refer to the dataset containing news from the 'Aerospace & Defence' market as the 'ASD' dataset. The dataset containing news from the 'Energy & Resources' market is referred to as the 'GEW' dataset. We will study the ML approach applied to those two datasets.

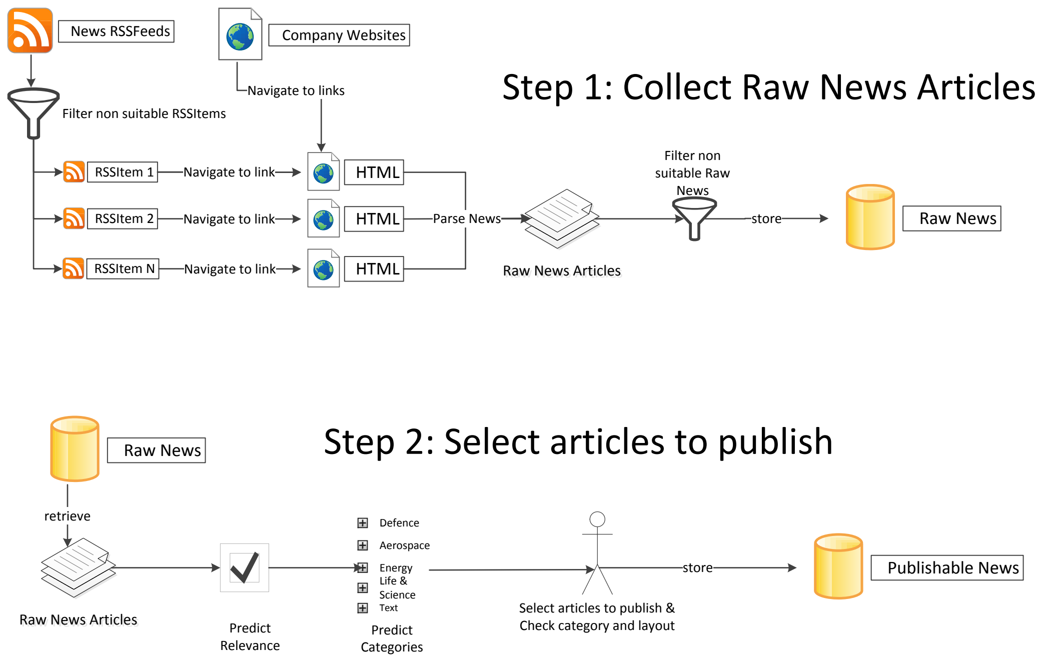


FIGURE 1.4: A flowchart of the intended news data conversion system

1.3 Problem statement

A lot of research is done on the ML approach to TC. There are different forms of TC and we will be concerned with *Topic Categorization*. This is a form of TC where the goal is to determine the topic(s) of a piece of textual information.

Our research will use two datasets collected by the business-to-business news publisher ASDMedia. One dataset contains news articles about the *Aerospace & Defence* market and the other about the *Energy & Resources* market, which we will refer to as the 'ASD' and 'GEW' dataset respectively. Each article can be labeled with multiple categories, which makes them both a multi-label classification (MC) problem. We will refer to these two problems as the ASDMedia classification problem.

In our case we restricted ourselves to algorithms that generate models that are easily interpretable by humans and decided to use decision trees (DTs). This has the advantage that the models can be analyzed, even by people which are not familiar with ML. It should be noted however that this is not the best performing classification algorithm for TC. It is not the goal of our research to address the question whether resulting models are humanly interpretable. Instead we restrict ourselves to DTs, which are known to produce the best humanly interpretable models.

The goal of this research is to find out if it's possible to construct a classification system that can classify news articles with reasonable performance, while using a ML algorithm with the above mentioned restriction. Our first subgoal will be to find out which approach of translating a news article into features is the most suitable for our problem. The other goal is to find out if we can improve the performance of our classification system by exploiting the structure of the categories.

The main question we will address during this research is the following:

Is it possible to develop an automatic classification system with the following features to solve the ASDMedia classification problem?

- Use ML to generate the model
- The ML algorithms are restricted to those which generate humanly interpretable models
- The classifier achieves a reasonable performance

The performance of the classifier must be good enough to be able to use it in a semi-automatic system. This means that the result of the classifier is verified by a human, before the choice is final. If the result is wrong the human can adjust the categories. Because there is always a human verification, we intentionally are vague about what a reasonable performance is. Later in the introduction we will discuss this semi-automatic system in more detail.

In order to find the best solution for the ASDMedia classification problem, we will address two questions. The first one deals with the representation of news articles as features for the classification model. We will try various approaches looking at both the *generation of features* and the *selection of features.*

The second question deals with the structure of the categories. A distinction is made between the external structure (such as defining a hierarchy for the categories) and internal structure (such as correlations between categories). We will try approaches that exploit this structure and hope to improve the performance of our classification system. While doing these experiment we will continue each experiment with the best setting from the previous experiment. This way we will attempt to find the best combination of settings for both ASDMedia datasets.

1.4 Challenges

One of the challenges of the ML approach to text classification is how to translate the textual information into the features that eventually can be used by a ML algorithm. This is what we refer to as *feature generation*.

Perhaps in an ideal world the true semantics of the text is understood and only the relevant concepts are used as features. In practice just using each word as a separate feature already works quite well. However most approaches will generate an enormous number of features, which not all ML algorithms can handle well. In order for them to work, only the most promising features are selected to feed to the algorithm.

Topic categorization is a part of TC that addresses the problem of determining the topic(s) of a text document. In the ASDMedia cases a document can belong to multiple categories, making it a MC problem. This is often the case in *topic categorization*. Most ML algorithms however, can only handle single-label problems. Furthermore the number of distinct categories tends to be very large and unbalanced. This makes *Topic Categorization* a challenging problem to research.

1.5 Outline

In chapter 2 a literature study relevant to our problem will be discussed. First an introduction will be given to Single-Label Classification (SC) and MC in general, after which we will discuss HC. Then we will discuss TC and give a small introduction on DTs. We will end the chapter with a section on the evaluation of classifiers.

In Chapter 3 we will analyze the data we had available to learn from. First we will show which attributes of a news article are stored in the database. Then the existing categories are discussed and we will end the chapter with a manually constructed hierarchy for the categories.

Chapter 4 will give an overview of the experimental setup of this research. First the basic settings are explained, which are settings that are the same for all experiments. Then we will explain in detail the methodology of the experiments.

In Chapter 5 the results will be presented and discussed in detail. Some interesting findings are made and when problems are found with the designed experiments, attempts are made to solve them. In Chapter 6 we will conclude that we can use the ML approach to TC for our cases, to generate DTs that can achieve reasonable performance. We will end with a discussion on future research.

Chapter 2

Introduction to multi-label classification in the text domain

This chapter gives an introduction to *multi-label classification* (MC), starting by first explaining the single-label classification (SC) case. We will also mention the topic of *hierarchical classification* (HC), which can also either be single or multi-label. In addition we describe techniques that can improve MC. Then we describe specifics for the text classification (TC) domain. We continue with a brief introduction on *decision tree* (DT) induction, the algorithm for generating DTs from examples. We will end this chapter with a detailed explanation on how to evaluate classifiers.

2.1 Inductive learning and classification

Machine learning (ML) is a branch of artificial intelligence (AI) concerned with the design of computer programs that can improve their performance by learning from empirical data. Mitchell [11] describes the ML field by the central question it studies: ‘How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?’.

Inductive learning is the area in ML that specifically deals with learning models from observations. Within this area, learning tasks are often characterized in terms of the feedback given back to the learner [12], as follows:

- **Supervised learning:** The learner is provided with the desired output for each observation. The goal is to learn the function that predicts the correct output value when provided with a possibly previously unseen observation.

- **Unsupervised learning:** No output is provided to the learner. The goal is to discover patterns and regularities in the data.
- **Reinforcement learning:** This is a special case of supervised learning, where the learner is provided with a reward after each action.

In *supervised learning*, when the task is to learn a discrete-valued function, it is called *classification.* When learning a continuous-valued function the task is called *regression. Clustering* on the other hand is an *unsupervised learning* task which finds groups of similar objects in the data.

The set of assumptions made to be able to learn is called the *inductive bias.* A famous inductive bias is *Occam's Razor*, which states that when choosing from multiple hypotheses, we should pick the simplest one consistent with the data. We will continue with explaining in detail what classification is in the next section.

2.1.1 Classification

Classification is the task of assigning observations to one of several predefined categories also called classes. It is a supervised learning task of learning a discrete-valued function that maps an observation to the correct class label. An observation consists of a vector of attributes $\mathcal{F}$, called *features.*

The set of observations or examples with for each entry the correct class label is called the *training set.* More formally it contains pairs $\langle x, c\rangle \in \mathcal{O} \times \mathcal{C}$, where $\mathcal{O}$ is the domain of observations and $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is the set of predefined *classes* or *categories.* When an observation $x \in \mathcal{O}$ is labeled to a class $c_i \in \mathcal{C}$ this observation is called a *positive example* of c_i, otherwise it's called a *negative example* of c_i [1]. The classifier is evaluated on a *test set* which is disjoint from the training set. This is important because if you use the same data to train and to test on, the performance assessment can be too optimistic. We will tell more about evaluating classifier in section 2.7. Later the classifier (also known as hypothesis or model) can be used to predict unknown instances.

Definition 2.1. Classification
Classification is the task of approximating the unknown *target function* $\breve{\Phi} : \mathcal{O} \to \mathcal{C}$ by means of a function $\Phi : \mathcal{O} \to \mathcal{C}$ called the *classifier*, which assigns each observation $x \in \mathcal{O}$ to one of the predefined class labels $c \in \mathcal{C}$.

The supervised learning method can now be formally defined as a function $\Gamma : \mathcal{O} \times \mathcal{C} \to \Phi$ which, given the training-set, produces as output the classifier Φ. The whole supervised learning process is presented in figure 2.1. First the data is split into two disjoint sets the training and the test set. Then feature generation and selection is applied. The arrow from the feature generation & selection in the training phase to the test phase indicates that some information is passed from the training to the test phase. For example the features that are selected in the train phase should be the same as in the test phase.

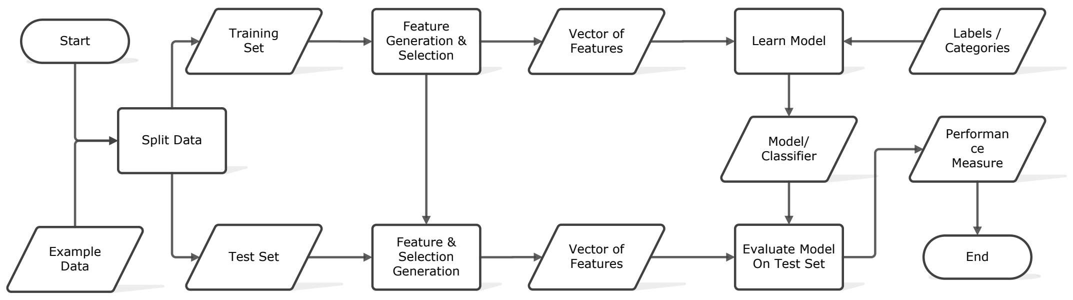


FIGURE 2.1: The Supervised Learning Process

The constraint of assigning exactly one class label to each observation $x \in \mathcal{O}$ is called SC. The class labels are assumed to be mutually exclusive. A special case of SC is *binary classification* (BC), where each $x \in \mathcal{O}$ must be assigned to either the class c_i or to its complement $\bar{c}_i$.

Not having this *single-label* constraint is termed MC. Here the classes are not disjoint and an observation $x \in \mathcal{O}$ can belong to multiple classes $c \in \mathcal{C}$. We will further explain this in the next section.

2.2 The multi-label classification problem

As mentioned before with MC an observation can belong to multiple classes. Not only does the training data has examples with multiple labels, the classifier has to be able to predict a single record into multiple classes as well. The training procedure has to be adjusted to be able to handle multiple labels. The definition of classification can be extended to allow for the multi-label case by having the classifier output a (possibly empty) set of categories, member of the power set of categories as in definition 2.2.

Definition 2.2. Multi-Label Classification
Multi-Label Classification is the task of approximating the unknown *target function* $\breve{\Phi} : \mathcal{O} \rightarrow \mathcal{P}(\mathcal{C})$ by means of a function $\Phi : \mathcal{O} \rightarrow \mathcal{P}(\mathcal{C})$ called the *classifier*, which assigns each observation $x \in \mathcal{O}$ to zero or more predefined class labels from $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$.

2.2.1 Multi-label approaches

Most traditional learning algorithms are developed for SC problems. Therefore a lot of approaches in the literature transform the multi-label problem into multiple single-label problems, so that the existing single-label algorithms can be used. There are quite some different approaches and we will give an overview of them here.

A clear distinction can be made between *algorithm dependent* and *algorithm independent* approaches also called *problem transformation* approaches. Algorithm independent problems transform the problem to multiple single-label problems, which can be further divided into *instance-based* and *label-based* methods.

When the transformation is based on instances, i.e. simply eliminating all instances with multiple labels, it is termed *instance-based. Label-based* methods transform the multi-label problem to one or more single-label problems by only looking at the class labels [13]. A hierarchical structure representing the approaches, based on the figure by Carvalho and Freitas [13] , is shown in figure 2.2.

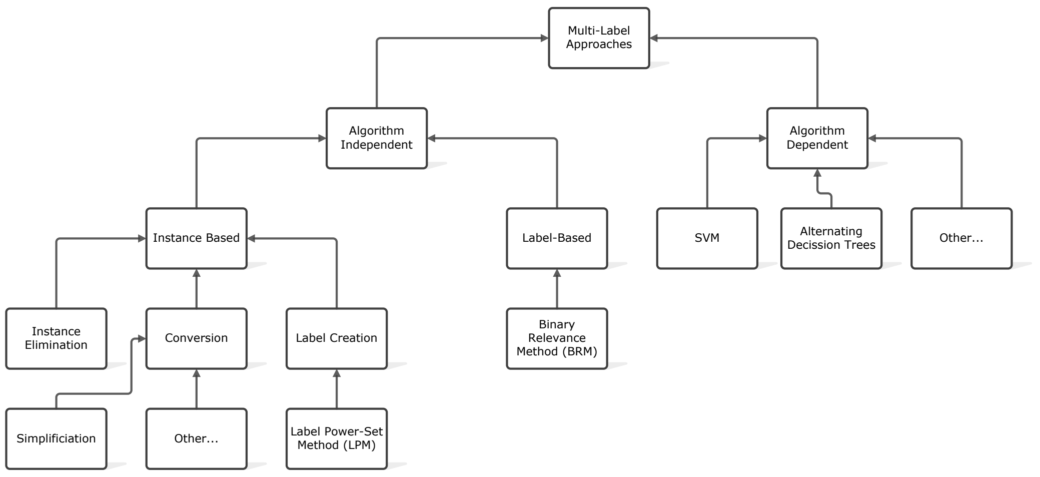


FIGURE 2.2: Multi-Label approaches

Most approaches have to choose between at one hand the computational complexity and on the other hand taking into account the correlation between class labels. The simplest method is probably *instance elimination*, which simply ignores all multi-label instances. Another simple method called *simplification*, randomly selects one label from multiple labels and uses that to train on. The two previously mentioned approaches however do not give good results.

A promising *algorithm independent* method, which also takes into account a hierarchy, not discussed here is a method by Bi and Kwok [14] that transform the instances to multiple single-label problems using kernel dependency estimation. Here the labels are transformed to a low-dimensional space while preserving their hierarchical structure. We will discuss two *algorithm independent* approaches here which are commonly used in the literature and have good performance, namely the *binary relevance* (BR) method and the *label power-set* (LP) method.

2.2.1.1 Binary relevance method

A popular *label-based* approach is the BR method aka *one-against-all* approach. In this case an *ensemble* of single-label binary classifiers is trained, one for each class. Each classifier predicts either the membership or the non-membership of one class. The union of all classes that were predicted is taken as the multi-label output. This approach is popular because it is easy to implement, however it ignores the possible correlations between class labels. This method needs $|\mathcal{C}|$ classifiers and has a low computational complexity relative to other multi-label approaches.

2.2.1.2 Label power-set method

An *instance-based* approach called *multiplicative conversion* by Carvalho and Freitas [13] does take possible correlations between class labels into account. More commonly this approach is called the LP method, because it considers each member of the power set of labels in the training set as a single label.

This method needs worst case $2^{|\mathcal{C}|}$ classifiers, and has a high computational complexity. In an extensive comparison with other *algorithm independent* approaches, this method scored best, followed by the one-against-all method [15]. However when the number of classes increases the number of distinct label combinations can grow exponentially. This easily leads to combinatorial explosion and thus computational infeasibility. Furthermore, some label combinations will have very few positive examples.

2.2.2 Techniques to improve the binary relevance method

Because of its low computational complexity and relatively good performance the BR method is popular in the literature. However because it completely ignores the possible correlations between class labels, techniques have been introduced to enhance this method by making use of some correlations between class labels.

2.2.2.1 Stacking

Godbole and Sarawagi [16] provided two improvements for a Support Vector Machine (SVM) learner, one improvement being algorithm independent. This improvement is a technique based on *stacked generalization* or simply called *stacking* by Wolpert [17].

The classification problem is divided into two levels. The first level being the classification problem with the normal feature space $|\mathcal{F}|$. The second level classifiers are trained on an extended training-set which contains all predictions of the level-1 classifiers used on the training-set. So the training-set of level-2 consists now of $|\mathcal{F}| + |\mathcal{C}|$ attributes. The process is illustrated in figure 2.3. The level-2 classifiers can take into account label dependency and improve the BR method, by doubling the computational time using $2|\mathcal{C}|$ classifiers.

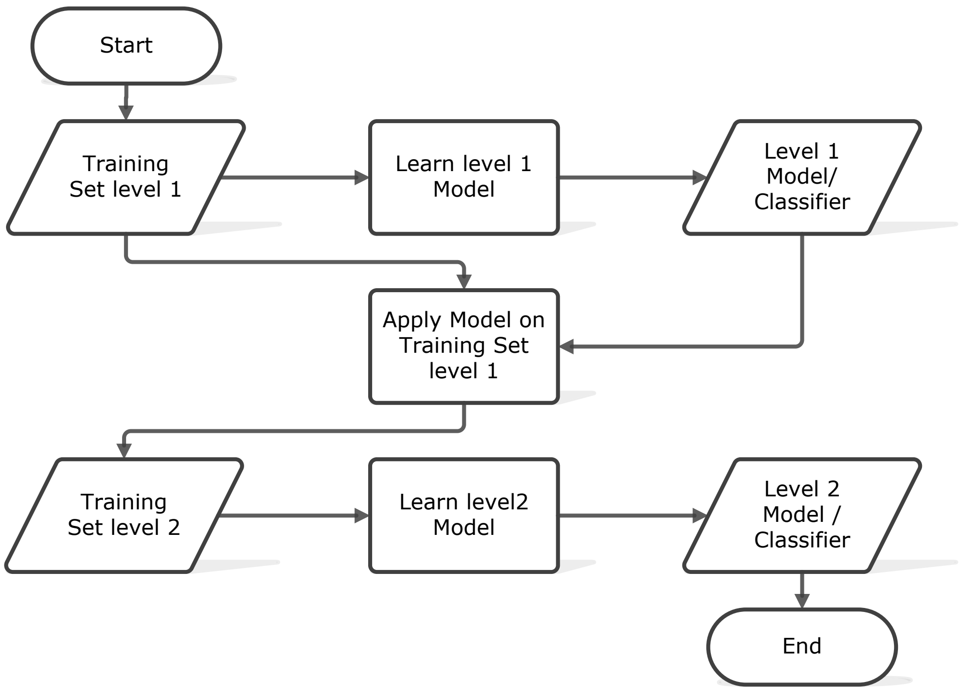

Start
Training Set level 1
Learn level 1 Model
Level 1 Model/ Classifier
Apply Model on Training Set level 1
Training Set level 2
Learn level2 Model
Level 2 Model / Classifier
End

FIGURE 2.3: The learning process of stacking

2.2.2.2 Chaining

Read et al. [18] introduced a technique called *chaining*. A chain of binary classifiers $C_0, C_1, \ldots, C_{|\mathcal{C}|}$ is constructed, where a classifier C_i uses the predictions of all the classifier C_j, where $j < i$. This way the method, also called *classifier chains* (CC), can take into account label correlations. The total number of classifiers needed for this approach is equal to the number of classes, but the training of the classifiers is more involved. This process is illustrated in figure 2.4, with a classification problem of three categories $\{C1, C2, C3\}$ chained in that order.

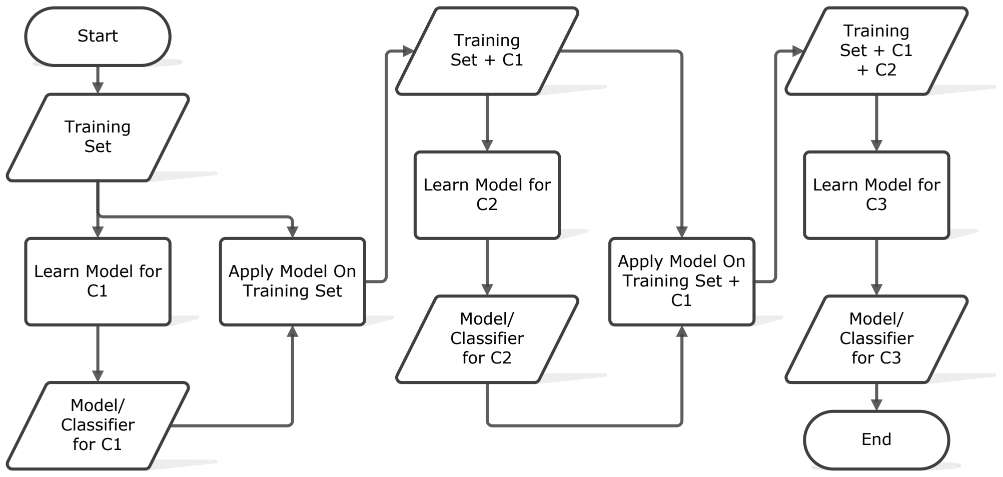


FIGURE 2.4: The learning process of chaining

The order of the classifiers in the chain clearly can have a great impact on the accuracy of the method. Read et al. [18] solved this problem by creating ensembles of chain classifiers (ECC), each chain having a random order and a subset of the training data. This ensemble approach is a state-of-the-art method in MC.

Dembczynski et al. [19] presented *probabilistic chain classifiers* where a CC is put in a probabilistic framework. Therefore it uses a classifier which has as output an approximation of the probability that an instance belongs to a class label. They calculate all joint probabilities of classes by using the chain rule, whose computational complexity is very high.

Zatagoza et al. [20] introduced an improvement of the ECC by building a probabilistic network that models the label dependencies. They also build an ensemble of chains but instead of picking the chain order totally random, they first build a probabilistic network and picked orders consistent with that network.

2.2.3 Multi-label statistics for datasets

The number of labels that each example has and it's proportion of the total number of labels differs per dataset. These two parameters describe more or less the multi-labeledness of the dataset and can influence the performance of the classification. We will show here two statistics, the *label cardinality* and *label density* described by Tsoumakas et al. [21]. The label cardinality is the average number of labels per example. The label density is the average number of labels per example divided by the total number of labels.

Definition 2.3. $LabelCardinality = \frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} |\breve{\Phi}(x)|$

Definition 2.4. $LabelDensity = \frac{1}{|\mathcal{O}|} \sum_{x \in \mathcal{O}} \frac{|\breve{\Phi}(x)|}{|\mathcal{C}|}$

Another measure of importance is the number of *distinct* label sets. This is mainly for algorithm transformation methods that operate on subsets of labels, such as the LP method which we have discussed in section 2.2.1.2.

2.3 Common classification issues

This section explains two issues which may be present in a classification problem, namely *overfitting* and a *skewed class distribution*. *Overfitting* is something every ML algorithm has to deal with. A *skewed class distribution* can be a property of the training data.

2.3.1 Overfitting

Overfitting basically means that the classifier has modeled the training-data too well and does not perform well on previously unseen data. The errors a classifier makes on training-data are called *training-errors* and the estimated errors on previously unseen samples are called *generalization-errors*. A good model has both a low training-error as well as a low generalization-error. *Underfitting* occurs when a model has both high training-error as generalization-error. *Overfitting* on the other hand, generally occurs when a model has low training-error but high generalization-error [22].

Overfitting basically means that the model fails to generalize well and fits too good on the training data. In other words the model has adjusted itself to the noise in the training data. A good model finds a balance between training and generalization errors. See figure 2.5, the green line clearly overfits and fails to generalize well.

Overfitting can occur when the criterion the model is optimized on when learning, is not the same as the one it is evaluated on when testing. But most often occurs when there is much noise in the training data.

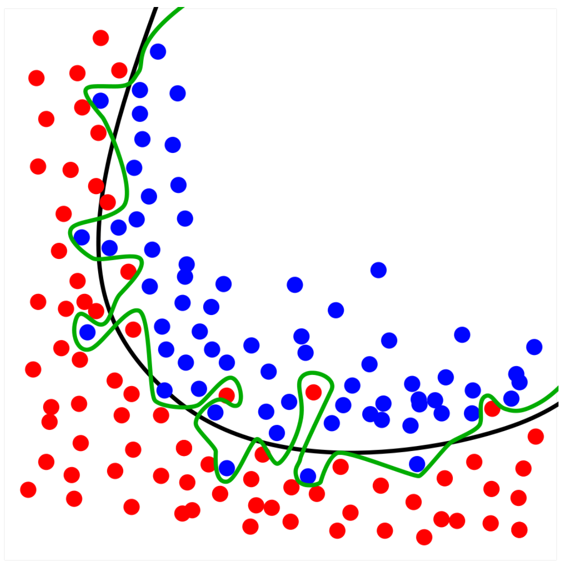


FIGURE 2.5: An example of overfitting, the green line fits the data points too well (source: By Chabacano (Own work), 2008, Overfitting.svg , via Wikimedia Commons, Available from `http://commons.wikimedia.org/wiki/File:Overfitting.svg`)

2.3.2 Skewed class distribution

A skewed class distribution can occur in both SC and MC problems. Even when there is an uniform class distribution, by using the BR method and the number of classes is high, the training data for each classifier will be skewed. Lets illustrate this with an example.

Example 2.1. *A skewed class distribution*
Lets look at a MC problem with N classes $\{c_1, \ldots, c_N\}$, using the BR method. There are N classifiers trained $\{C_1, C_2, \ldots, C_N\}$, where classifier C_i predicts the membership or non-membership of an observation to the class c_i.

Suppose that $N = 100$ and suppose that the classes are uniformly distributed in the training and test data. Then the training and test data for classifier C_j will have 99% of the observations labeled $\bar{c}_j$ and only 1% of the observations labeled c_j.

In the example above a classifier C_i which simply assigns all observations to $\bar{c}_i$, the *majority class*, achieves an accuracy of 99%.

By taking the accuracy measure, misclassifying a fraction of the majority class has more impact than misclassifying the same fraction of the minority class, simply because there are more observations that belong to the majority class. Consider for example the classifier C_i from example 2.1. Misclassifying 50% of the records that belong to c_i will have an impact of 0.5% on the total accuracy. On the other hand, misclassifying 50% of the records that belong to $\bar{c}_i$ will have an impact of 49.5% on the total accuracy.

Sometimes this is not what you want. In example 2.1 the classifiers should be punished more for misclassifying an observation that belongs to the minority class. In order to prevent the classifier in predicting all instances to the majority class. There are broadly two techniques to balance the class distribution, namely *sampling* and *example weights*. Balancing the class distribution can be done with sampling by over-sampling the minority class or by under-sampling the majority class. Another option is to use *example weights*, which assigns a weight to an example based on which class it has labeled. By making sure the weights sum up to the same value for each class, one can balance the class distribution. The ML algorithm has to be able to incorporate this weighting mechanism. When changing the way the classification algorithm trains this should also be reflected in the evaluation measure which the algorithm is evaluated on. Different measures will be discussed in section 2.7.

2.4 Hierarchical classification

This section will give an overview of the literature on HC. First we will explain the hierarchical class structure and how ML algorithms can exploit this structure.

2.4.1 Hierarchical structured classes

Structured classification is a form of classification where the class labels have a structure. HC is a type of structured classification where the classes form a hierarchy. An observation that belongs to a class, automatically belongs to its superclasses. This is called the *hierarchy constraint* [23].

Wu et al. [24] defined a class taxonomy for a tree structured hierarchy as a strict partially ordered set $(\mathcal{C}, \prec)$ where $\mathcal{C}$ is the set of classes and $\prec$ represents the 'IS-A' relation. The 'IS-A' relation is defined to be anti-reflexive and transitive. Later Silla and Freitas [10] extended this relation to be also asymmetric and noted that it also holds for directed acyclic graphs (DAG). In this case a class can have multiple superclasses.

Definition 2.5. Hierarchical Class Taxonomy

- There is one greatest element 'R'.
- $\forall c_i \in \mathcal{C}$, if $c_i \prec c_j$ then $c_j \not\prec c_i$ (asymmetric)
- $\forall c_i \in \mathcal{C}, c_i \not\prec c_i$ (irreflexive)
- $\forall c_i, c_j, c_k \in \mathcal{C}$, $c_i \prec c_j$ and $c_j \prec c_k \Rightarrow c_i \prec c_k$ (transitive)

According to the hierarchical constraint a HC problem is by definition multi-label. Since an observation which belongs to a class automatically belongs to its superclass, each observation has multiple class labels. However in the literature, when talking about hierarchical multi-label classification (HMC), it indicates that training instances can have multiple class labels not related by the 'IS-A' relation.

Figure 2.6 gives an example of an hierarchical DAG, here the countries China and India have two parents.

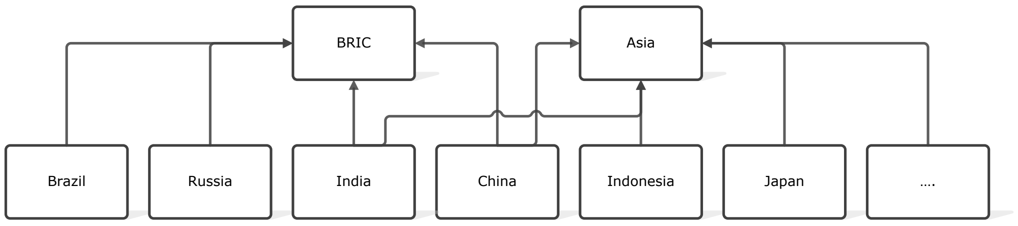


Figure 2.6: An example of a hierarchical DAG of grouping countries

2.4.2 Exploiting the hierarchy

Ignoring the class hierarchy is termed *flat classification* (FC), because only predictions at the leaf nodes of the hierarchy are made. This basically flattens down the hierarchy.

The hierarchy can be exploited by a 'divide and conquer' approach splitting the classification problem in multiple sub-problems. Each sub-problem dealing with another part of the hierarchy. There are multiple ways to split the hierarchy, which will be explained in the next section. The idea is that each sub-problem is simpler than the original and thus can be solved more efficiently. There are a few advantages to this approach:

- These sub-problems may have different partitions of the data set which reduces the number of classes which in turn may reduce the *skewness* of the class distribution.
- A more specialized set of features can be used for each sub-problem which can lead to better results [9].
- Some sub-problems can be ruled out, thereby saving computation time.

Disadvantages are:

- How to combine mixed results from different levels in the hierarchy? The hierarchy constraint cannot be violated.

- Some classes may have very few training instances to learn from. Note however that the positive instances for a class are the same as in FC.

Exploiting the hierarchical structure can be done broadly in two distinct ways: *Globally* a.k.a the *Big-Bang* approach or *locally.* It has been shown by many that exploiting the structure can improve the classification result [9, 14, 23–25]. Global means that one classifier is trained which can deal with the whole hierarchical structure, usually adapting an existing algorithm. A state of the art example is an adaptation of the C4.5 DT by Blockeel et al. [23].

Local means that multiple classifiers, usually one for each class in the hierarchy, are combined. A popular approach is the 'top-down' approach, which starts at the highest level of the hierarchy, and if a membership of a class is predicted, the classifiers for its children are used to classify further down the hierarchy. See for example figure 2.7, where the dashed lines indicate each sub-problem and the shaded nodes are the predicted ones. Consider this a SC problem, starting at the root node with a multi-class classifier at each parent node. The first problem is to predict the class $C1$ or $C2$ or $C3$. Then another classifier predicts $C3.1$ or $C3.2$ until finally we end with the predicted category $C3.2.3$. Note that instead of a multi-class classifier at the parent node, the BR method can also be used.

A disadvantage of this *top-down* approach is the so called *blocking problem.* When a classifier higher in the hierarchy predicts the non-membership for an example, the classifiers lower in the hierarchy will not be applied. This is not a problem when the prediction is correct. However, when the classifier higher in the hierarchy predicts a *false negative* (it incorrectly predicts a *positive example*), then the example is 'blocked'. The *Recall* of the classifiers at higher levels in the hierarchy should be very high, because otherwise they drag the whole system down.

Another approach by Valentini [26] in the gene domain is a combined 'bottom-up' and 'top-down' approach, called the *True Path Rule Algorithm.* The classification starts at the bottom level, with local classifiers which propagate their answer to a higher level if the classification is positive. When a classification is negative, the classification is propagated downward to stay consistent with the hierarchy. A local classifier there combines positive classifications from descendants, negative classifications from ancestors and its own prediction to determine the membership of a class.

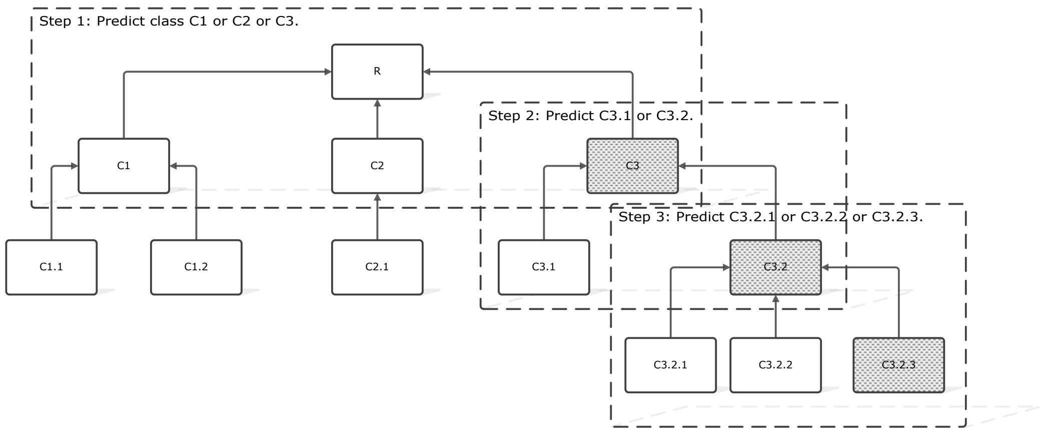

Step 1: Predict class C1 or C2 or C3.
Step 2: Predict C3.1 or C3.2.
Step 3: Predict C3.2.1 or C3.2.2 or C3.2.3.
R
C1
C2
C3
C1.1
C1.2
C2.1
C3.1
C3.2
C3.2.1
C3.2.2
C3.2.3

FIGURE 2.7: An example of 'top-down' approach, the filled nodes are the predicted ones

2.4.3 How to partition the training data?

There are many possible ways to partition the training data for each local classifier. For text classification, an extensive study by Fagni and Sebastiani [27] showed that the *sibling* approach seems to work best. A class has positive examples of those instances labeled with that class or its descendants. And negative examples of those instances not labeled with its class or descendants, but labeled by its siblings or their descendants. Figure 2.8 shows which training data to use when using the sibling approach, to learn a classifier for the category C1.1. All examples labeled with C1.1 are used as positive examples, as opposed to all examples labeled with C1.2 or C1.3 (and not with C1.1) which are used as negative examples.

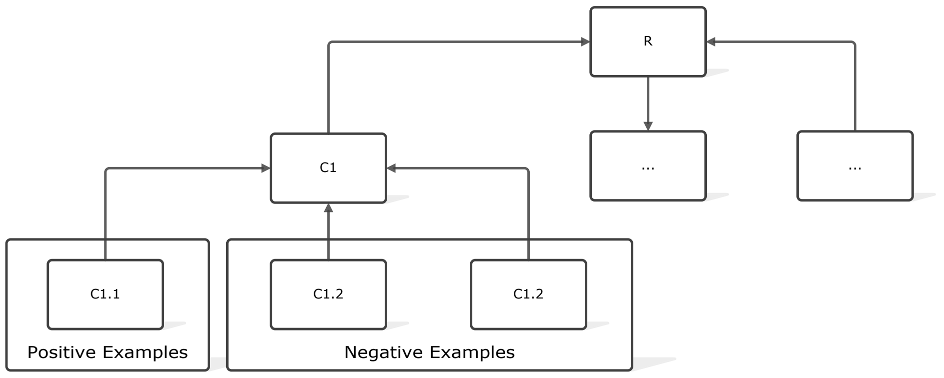


FIGURE 2.8: A part of a hierarchy where the training data is indicated when using the sibling approach for learning a classifier for the category C1.1

In domains with many features, like in text classification or music genre classification, it may help to also apply different *Feature Selection* at different levels in the hierarchy. Because the most-discriminating features tend to be different at each level in the hierarchy. This was shown by Silla and Freitas [25] to improve classification results in music genre classification. Also Koller and Sahami [9] show that using a smaller specialized set of features at each level in the hierarchy improves accuracy and reduces the risk of *overfitting* in the text domain, as opposed to using more less specialized features.

2.5 Text classification

In this section we describe the specifics of classification in the text domain. Mainly this has to do with how a textual document is transformed into a set of *features* $\mathcal{F}$, which can be used in classification. This transformation which we will call *data preprocessing* consists of two steps: *feature generation* and *feature selection.* We explain those steps later.

TC can be formally defined by extending the previous definition of (multi-label) classification 2.2, by replacing the observations by documents. Here we will give the definition of multi-label text classification (MTC), the same way a definition for single-label text classification can be defined.

Definition 2.6. Multi-Label Text Classification
Multi-Label text classification is the task of approximating the unknown *target function* $\breve{\Phi} : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$ by means of a function $\Phi : \mathcal{D} \rightarrow \mathcal{P}(\mathcal{C})$ called the *classifier*, which assigns each document $x \in \mathcal{D}$ to zero or more predefined class labels from the set $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$.

The set of documents in TC provided with class labels is also called the *initial corpus* Ω.

2.5.1 Data preprocessing

In this section we will explain both the *feature generation* and *feature selection* methods in detail. The features that are generated usually correspond with the words in the text and associated weights. How the words are extracted from the text is not a trivial task and is explained in section 2.5.1.2 about *term identification.* After this step many features may be extracted, which can be problematic. This is because many classifiers can't deal well with that many features and usually they are reduced by applying *Feature Selection* [1].

2.5.1.1 Feature generation

Many techniques used for *document indexing* in the field of information retrieval (IR) can be used to generate features. Document indexing has been studied extensively already in the late 50's [1]. This technique deals with the representation of text and this is usually done by a set of terms with associated weights. The weights roughly indicate the contribution of the term to the meaning of the document. There are various ways of defining what a term exactly is and how to calculate the associated weight. Much research has been done, starting with the *set of words* model where each word is identified as a term and the weight is 1 if the word occurs in the document and 0 if not. Later this was extended by the *bag of words* model which computes the weight by calculating the term frequency [28].

Research has been done on using phrases as terms, instead of single words. In the IR field a phrase consisting of n words, in exact that order, is called a *n-gram.* This can be extended for any n, however the number of features will grow very fast and may lead to an explosion of features. In figure 2.9 you see the example from the introduction on how 1-grams and 2-grams are generated from a sentence.

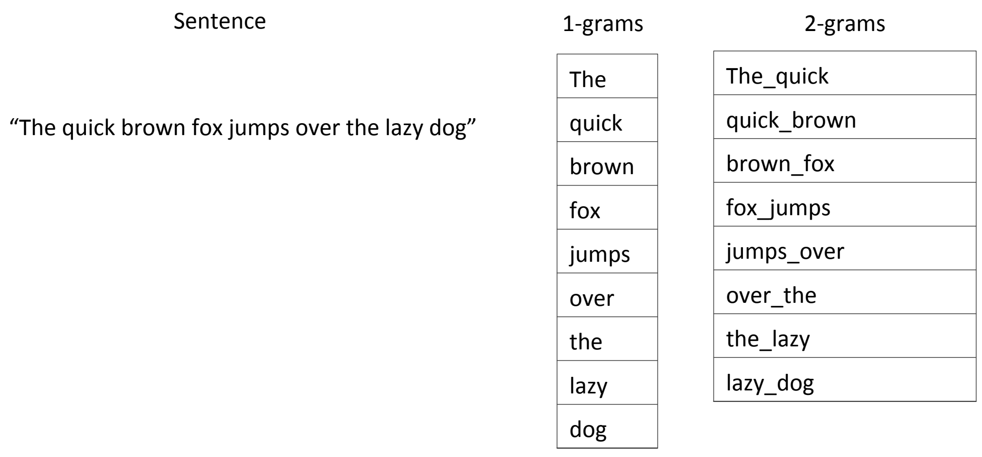


FIGURE 2.9: An example of how 1-grams and 2-grams are generated from a sentence

Although *n-grams* are widely used in the IR field, their usage in TC is not clear. There are mixed results and there is no consensus on whether using phrases improves classification performance [1, 4, 29, 30]. Other options are to use *natural language processing* (NLP) techniques to extract language semantics and encode that as feature. A comprehensive study by Moschitti [4] showed no advantage of using a more complex model than the *bag of words* model when using a SVM or Rochio classifier.

Nowadays, usually the *term frequency - inverse document frequency* (TF-IDF) weighting is used [31]. This measure takes into account a collection of documents and attempts to measure the discriminative power of a term inside that collection. The term frequency (TF) is divided by number of times the word occurs in the other documents. This is more formally defined as:

Definition 2.7. $\textit{TF-IDF}(t_k, d_j) = \textit{TF}(t_k, d_j).log\frac{|\Omega|}{\textit{DF}(t_k)}$

where $\textit{TF}(t_k, d_j)$ denotes the TF of term t_k in document d_j and $\textit{DF}(t_k)$ denotes the *document frequency* (DF) of term t_k, which is the number of documents where t_k occurs at least once. The function represents that the more often a term occurs in a document, the more it is a representative of its content and the more documents a term occurs in, the less discriminating it is. The weights of *TF-IDF* are often normalized by *cosine normalization*, where $\mathcal{T}$ is the number of *terms*.

Definition 2.8. Cosine normalization $w_{kj} = \frac{\textit{TF-IDF}(t_k, d_j)}{\sqrt{\sum_{s=1}^{|\mathcal{T}|}(\textit{TF-IDF}(t_s, d_j))^2}}$

Usually the full-text of a document is indexed, but in some cases indexing only the title or a part of the full-text gives better performance [32]. Another method that influences the calculated scores is to apply weights on different types of text, like the title and the full-text. Please note that apart from selecting terms as features, a document can contain other information. For a news article this can be the source of the news, or perhaps the author. Sometimes the length of the document can also be a useful feature.

2.5.1.2 Term identification

Before the words in the text can be used as terms, first we need to be able to identify the words or *tokens* from the text stream. This is done by a process called *tokenization.* Another step, which depends on the application, is to apply *stemming* which converts different tokens of the same semantics into one standard form. Tokens which are not needed can be removed from the text, which is done by the processes called filtering. We will explain these processes next.

Tokenization This process splits the text into a list of tokens by searching for specific token delimiters. Some delimiters like white space and the newline character are easy to determine. However some character are ambiguous and can be both delimiter and part of a token depending on its context. A period for example can be part of a token as in an abbreviation, but at the end of a sentence it is a token delimiter.

Stemming There exist different kinds of stemming which basically transform words with roughly the same semantics to one standard form. Inflectional stemming is a form of stemming which reduces all inflected variants of words in different grammatical contexts to one standard. Plurals forms can be transformed to singular form, like ‘books’ to ‘book’. The past tense of words can be transformed to present tense, like ‘ate’ to ‘eat’. The same holds for other inflected forms of words. Here a word can also be ambiguous, the context is needed to determine the semantic of a word. Another method which is similar to stemming, is to find synonyms of words and transform them to one form. Not in all classification cases stemming improves the result, as for example author classification relies on the subtly different writing styles of authors.

Filtering In this process certain words are removed from the text. Often so called stop words are removed, which are words that almost never have a predictive capability, such as ‘a’ and ‘the’ in English. This is usually a straightforward task.

2.5.1.3 Feature selection

The high dimensionality of the term space may be problematic for TC. Therefore, before learning a classifier, often *dimensionality reduction* (DR), is applied to the term space. Another benefit is that DR reduces *overfitting*. Experiments have shown that to avoid overfitting, training examples roughly proportional to the number of terms are needed. Around 50-100 examples are needed per term according to Fuhr and Buckley [33]. There are two ways of using DR, *globally* and *locally*, the first reduced the set of terms chosen for the classification for all categories, the latter chooses a reduced set of terms per category. The resulting terms can also either be selected from the original terms by *term selection* or extracted from the terms by *term extraction*.

Term selection Term selection has been shown by Yang and Pedersen [5] to increase the effectiveness a bit ($< 5\%$), depending on the classifier, the aggressiveness and the DR technique. An approach called *filtering* by John et al. [34], is to keep the terms that receive the highest score according to a function that measures the importance of the term for the TC task. There are different functions to measure the importance of a term, where the *document frequency* has been shown to reduce the term space by a factor ten without loss of effectiveness [5]. Many authors remove terms that occur in at most one-three training documents.

Other more sophisticated functions used are the DIA association factor, chi-square, NGL coefficient, information gain, mutual information, odds ratio, relevancy score and GSS coefficient. A study by Rogati and Yang [35] compared around 100 methods of feature selection for text classification among different classifiers. There main results were that usually a combination of methods works best and the best combinations differ for each type of classifier. However they state that:

- Almost all top performers had χ^2 as a component.
- Eliminating the low document frequency words boosts the performance.
- Combining good methods with little or no correlation improved the result.

Term extraction Term extraction creates a new set of synthetic terms, from the original set of terms. First a function that extracts the synthetic terms from the original set of terms is needed and secondly a function that converts the old representation of a document in the new one. There has been experimented with *term clustering* which clusters terms that are related to each other on some similarity measure and uses a synthesized term to represent that cluster. Slonim [36] found an improvement of the effectiveness of classification by clustering per category. He used the co-occurrence of words contributing to the same category as similarity measure.

2.5.2 Text classifiers

Sebastiani [1] extensively compared classifiers in the TC domain and found the following results:

- Boosting-based classifier committees, support vector machines, example-based methods, and regression methods deliver top-notch performance.
- Neural networks and on-line linear classifiers work very well, although slightly worse than the previously mentioned methods.
- Batch linear classifiers (Rochio) and Naive Bayes classifiers seem to be the worst of the learning-based classifiers.
- The data collected was unable to say something sufficient about DTs.

2.5.2.1 Support vector machines

Most classification algorithms cannot handle a large number of features well. One notable exception is the SVM, introduced to the field of TC by Joachims [37] in 1998, which is now one of the state-of-the-art algorithms in TC. It attempts to find the surface σ_i from all the surfaces from $|\mathcal{T}|$-dimensional space, that separates the positive from the negative examples by the widest possible margin. The best decision surface is determined by only a small set of training examples, called the support vectors. It offers advantages over other classifiers because:

- Term selection is often not needed, a SVM can scale up to considerable dimensionality's and is robust to overfitting.
- No effort in parameter tuning, as there is a theoretical 'default', which is shown to provide the best effectiveness.

It's power and suitability for TC is that it can handle large number of features. For most other classification algorithms a feature selection procedure is done to reduce the number of features.

2.6 Decision tree induction

In this section we will discuss the DT learning algorithm and the product of this algorithm, a DT classifier. Because the model is easily interpretable by humans it makes it great for analytical purposes. We will first discuss the DT classifier and then describe the basics of the learning algorithm.

2.6.1 The decision tree classifier

A DT classifier uses a tree model to predict the class of an example. The tree consists of one *root node*, which is where the classifier starts. The other nodes are either *leaf nodes*, when they have no branches or *internal nodes.* The internal nodes and the root node represent a feature and a test that has to be performed on that feature. For each possible outcome of the test, the node has a branch that leads to the next node. The *leaf nodes* eventually indicate a class.

A DT classifier predicts the class of an example by following a path from the *root node* of the tree until it encounters a *leaf node.* At every node (except leaf nodes) a test is performed to choose which branch to follow to the next node. When a *leaf node* is encountered, the classifier predicts the class that the *leaf nodes* indicates.

Take a look at figure 2.10 where a DT is displayed which decides whether to play tennis outside. Assume the following conditions: Outlook=Sunny and Humidity=High. To decide whether to play tennis, we start at the root node of the tree which is Outlook and follow the branch labeled 'Sunny' to the internal node labeled 'Humidity'. Here we follow the branch labeled 'High' to the leaf node labeled 'no'. This means we will decide not to play tennis under these conditions.

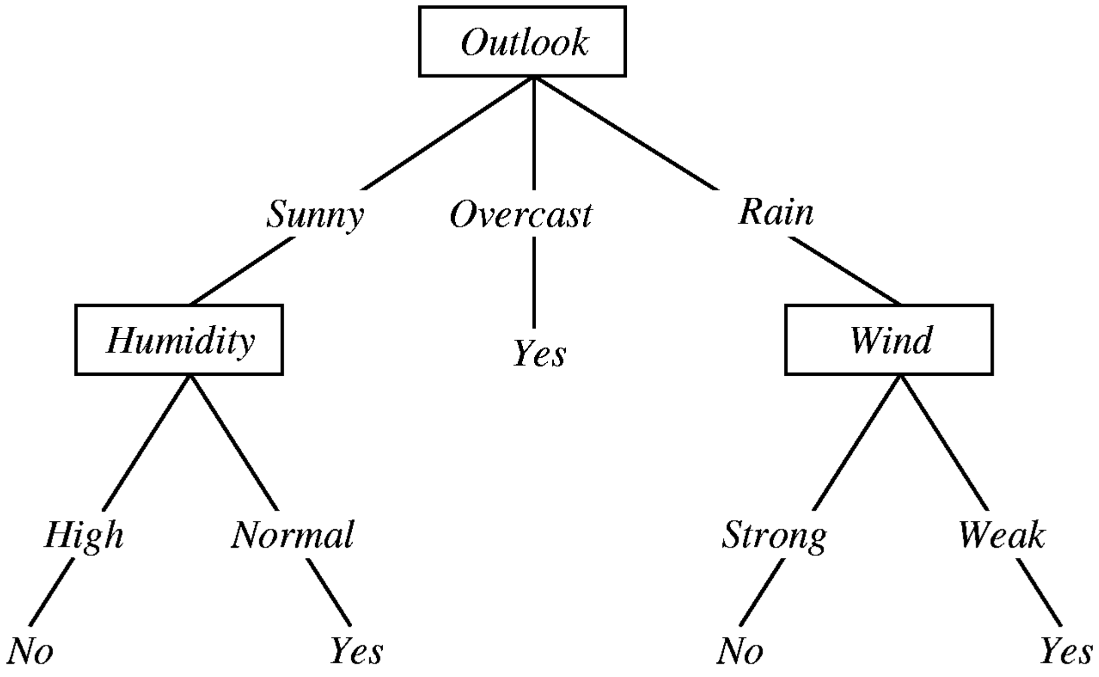


FIGURE 2.10: An example decision tree which decides whether to play tennis outside or not (source `http://jmvidal.cse.sc.edu/talks/decisiontrees/allslides.html`)

The whole tree could also be represented as **if . . . then . . . else . . .** rules. The following rules would represent the same DT:

```
IF Outlook = Sunny THEN
{
        IF Humidity = High THEN
                No
        IF Humidity = Normal THEN
                Yes
}
IF Outlook = Overcast THEN
        Yes
IF Outlook = Rain THEN
{
        IF Wind = Strong THEN
                No
        IF Wind = Weak THEN
                Yes
}
```

2.6.2 The learning algorithm

The goal of the learning algorithm is to construct a DT based on the examples in the training data. However the algorithm should avoid to build a tree that overfits the training data. That is why the optimal tree is the smallest DT which best distinguishes between classes. In theory, finding this optimal DT for a given classification problem is a search problem, which is shown to be *NP-complete* [38]. The search space consists of all possible ways to construct a DT from the given set of attributes, which grows exponentially, when the number of attributes increases. Usually a greedy search strategy is applied which constructs an accurate, although sub-optimal, DT. [22]

2.6.2.1 Hunt's algorithm

Hunt's algorithm is the basis of many existing DT induction algorithms, such as ID3, and C4.5. When constructing the tree, each node t is associated with a subset of the training data $D_t \subset \mathcal{O}$. Let $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ denote the set of predefined *classes*. Then starting with the root node, the following is a recursive definition for constructing a DT according to *Hunt's algorithm*:

- If $D_t = \emptyset$ then t is a *leaf node* identifying the majority class c_m of the parent node.
- If $D_t \neq \emptyset$ and all records in D_t belong to the same class c_i, then t is a *leaf node* identifying the class c_i.
- If $D_t \neq \emptyset$ and records in D_t belong to a mixture of classes, then a single attribute and corresponding attribute test is selected to partition the data further in disjoint sets. Each outcome of the test is mutually exclusive and for each outcome a child node is created. The algorithm is then recursively applied for each child node and corresponding partition of the training data.

  A special case is when no attributes can be found to split the data further, for example when they all have identical values. In this case the node becomes a leaf node which identifies the majority class c_m of the examples associated with this node.

2.6.2.2 Stopping criteria and pruning

An important decision for this algorithm is when to stop splitting nodes, which is called *pre-pruning*. Otherwise the DT may overfit the data. Another way to make the tree smaller is to prune the DT afterward, called *post-pruning*. There exists different ways of doing this, for example, the C4.5 algorithm uses *Error-based pruning*, while the CART algorithm uses *Cost-complexity pruning*. The reader is referred to [39] for a detailed explanation about *Error-based pruning* and to [40] for more information on *Cost-complexity pruning*.

Example (pre-pruning) criteria are:

- **MinimalLeafsize:** Determines the minimal number of training examples a leaf node should have when deciding to create this leaf node as a result of splitting it's parent node.
- **MinimalSplitSize:** Determines the minimal number of training examples a node should have to be able to create child nodes.
- **MaxTreeDepth:** Determines the maximum depth of the DT.
- **MinimalGain:** Determines the minimal gain in impurity measure the split should achieve.

2.6.2.3 Selecting the best split

The algorithm needs to select the best attribute and corresponding test to partition the data in the optimal way. There are many measures available to select the best split. Most of them use the class distribution of the child nodes, that would be created if the parent node was split. The class distribution can be expressed as a level of impurity, which is zero when all examples belong to the same class. The optimal split is then the split that results in the least impure child nodes. Different measures are *Gini index*, *information gain* and *classification error*. Impurity measures as information gain and Gini index tend to favor attributes that have a large number of distinct values, which may cause problems.

2.6.2.4 How to split the data

The way to split the data based on the values of a single attribute depends on the type of the feature. When the feature is binary, the data is partitioned in two sets, one where the feature is true, and the other where the feature is false. However when splitting nominal (categorical) attributes, this can be done in different ways. For example, the C4.5 algorithm splits the data with one branch for each possible value of the feature, called a multi-way split. The CART algorithm always makes binary splits, by grouping multiple values of a feature together. Continuous attributes can be split by first sorting the values and then selecting a point x, which lies between two adjacent values, to split the data on. For example, one partition contains the data where the attribute value $< x$ and the other partition contains data where the value $\geq x$.

2.7 Evaluating classifiers

The performance of classifiers is usually evaluated experimentally rather than analytically [1]. To get an unbiased estimate of the performance, the classifier is applied on previously unseen data, the test set. This is important because a classifier that performs well on the training set may have been subject to *overfitting*, and therefore will not perform well when used on previously unseen data. There are different methods to evaluate the performance of classifiers, which all divide the data into a training and test set and calculate the performance by using an evaluation measure. We will first give an overview of common evaluation methods. This is followed by a section on *sampling*, which describes the way examples are selected from a dataset. Then we will discuss the different evaluation measures for SC and then MC problems. There exist also evaluation measures for HC problems, but they fall outside the scope of this thesis.

2.7.1 Evaluation methods

There exist a couple of evaluation methods, which we will discuss here briefly. The way the data is selected from the original data is determined by the sampling technique used, which we will discuss in the next section.

- **Holdout Method:**
  This method is the classic evaluation method, which is simple but has some clear drawbacks. The original data is split into two disjoint sets, the training and test set. The classifier is trained on the training set and evaluated on the test set. The proportion of the training set is typically two thirds for training and one third for testing. The obvious limitation to this method is that fewer examples are available for training because they are reserved for testing. Another limitation is that the classifier depends on the composition of the training and test data [22].

- **Random Subsampling:**
  This method repeats the *holdout method* several times to improve the estimation of the classifiers performance, by taking the average. However it still suffers from the same limitations as mentioned by the *holdout method* [22].

- **Cross-Validation:**
  This approach partitions the data into k equal partitions and uses each partition and thus each record exactly once for testing. The other $k-1$ times, the partitions are used for training. The advantage is that each record is used exactly once for testing and the evaluation measures can be combined to obtain a good estimate of the average performance of the k classifiers. This method does not suffer from the limitations of the *holdout method* [22].

2.7.2 Sampling

Regardless of which evaluation method is used, there are different ways to partition the data into a training and test set. *Sampling* is a technique to select a subset from a dataset. In the case of splitting the data into a disjoint train and test set, sampling is done without replacement. One way of selecting records is to randomly select each record also called *random sampling.* A problem with this method is that, although probably with a low chance, this can lead to a training set which is quite different than the test set.

Another sampling approach called *stratified sampling* covers this problem by maintaining the class distribution of the original data set when selecting a subset. This has been shown to improve *cross validation* with random sampling in both bias and variance [41]. In SC, *stratified sampling* is a straight-forward task. However, stratifying a multi-labeled data set is more involved. This is because there are multiple interpretations of how to maintain the same class distribution in the selected sample.

One way to look at it is to maintain the same distribution for each combination of labels. This has similarities with the LP method, which trains a classifier for each combination of labels. Another, more relaxed approach is to only look at the distribution of all single-labels, which is similar to the BR Method. This method looks only at each single category and aims to distribute the positives examples for that category evenly.

Sechidis et al. [42] compared the two approaches with *random subsampling.* Their conclusion was that *random subsampling* performs consistently worse than both multi-label stratification approaches. The label combination method achieves low variance for datasets where the number of distinct label sets is low compared to the total number of examples in the dataset. The main problem with this method occurs when the number of examples for a certain label set is fewer than the number of folds you want to divide the data in. In those cases the label sets can't get evenly distributed. The other approach, which looks at positive examples for labels achieves low variance in cases where the number of distinct label sets is large compared to the number of total examples.

Perhaps a method combining both approaches will be ideal, however we did not come across such a method in the literature. The next example will show a case with many distinct label-sets, where the approach that looks at positive examples works better.

Example 2.2. *Applying Multi-Label Stratification*
In table 2.1a a dataset is shown with the occurrences of each label combination, that serves as a toy example. Suppose we want to select (without replacement) four folds of the dataset. Lets first look at the approach to maintain the same class distribution by looking at the label combinations. An immediate problem is that the label combinations $\{c1, c2, c3\}$ *and* $\{c1, c3\}$ *occur less than four times, namely two, so we can only put it in two folds. For the label combination* $\{c2, c3\}$ *there are six examples, which means we can put them into two folds once and also in two folds twice. How do we decide which folds get the example twice? Table 2.1b shows the positive examples for each label. Creating four folds with the same distribution of positive and negative examples for each category is simpler, because there are enough positive examples for each category.*

Table 2.1: A toy example set

label combination	occurrence
$\{c1\}$	8
$\{c2\}$	2
$\{c3\}$	10
$\{c1, c2\}$	4
$\{c2, c3\}$	6
$\{c1, c3\}$	2
$\{c1, c2, c3\}$	2

(a) Label Combinations

positive example for	occurrence
c1	16
c2	14
c3	20

(b) Positive Examples

2.7.3 Evaluation measures

The evaluation measures for SC are usually different than for MC. This is because in the multi-label case when predicting two of the three labels correctly this is better than predicting no labels at all. These differences are all captured in the evaluation measures and are explained in the next sections.

2.7.3.1 Basic measures

We will first explain some basic measures used mainly for SC. Here the evaluation is straightforward, the predicted value is either the correct one, or not. We define a function $Wrong$ which is 1 when the target function $\breve{\Phi}$ and the classifier Φ do not output the same class as follows:

Definition 2.9.

$$Wrong(x) = \begin{cases} 1 & \text{if } \breve{\Phi}(x) \neq \Phi(x) \\ 0 & \text{if } \breve{\Phi}(x) = \Phi(x). \end{cases}$$

We can define the error rate for a set of observations D by taking the sum of wrong predictions and dividing it by the total number of observations as follows:

Definition 2.10. Error Rate $Err(D) = \dfrac{\sum_{x \in D} Wrong(x)}{|D|}$

The accuracy of a classifier is then 1 minus the Error Rate.

Confusion matrix Before explaining other measurements we will first explain a *confusion matrix* for a BC problem, see table 2.2. Consider the prediction of the membership for a certain class c_i. Two types of errors can be made, misclassifying an actual positive example, called *False Negative* (FN) which corresponds with the Type I error in statistics. The other error is predicting c_i for an actual negative example which is called a *False Positive* (FP) which corresponds with the Type II error in statistics.

Class		Predicted	
c_i		positive (c_i)	negative ($\bar{c}_i$)
Actual	positive (c_i)	True Positives (TPs)	False Negatives (FNs)
	negative ($\bar{c}_i$)	False Positives (FPs)	True Negatives (TNs)

Table 2.2: The structure of a Confusion Matrix

Now the accuracy of c_i can be defined in terms of *True positives* (TPs) and *True Negatives* (TNs) divided by the total number of examples $|D|$. In the definitions below for $TPs, TNs, FPsandFNs$ the parameters (D, c_i), which resembles the set of observations D and the class label c_i are omitted for clarity purposes.

Definition 2.11. Accuracy $Acc(D, c_i) = \dfrac{TPs + TNs}{|D|}$

Straightforward *error rate* can now be defined in terms of FPs and FNs.

Definition 2.12. Error Rate $Err(D, c_i) = \dfrac{FPs + FNs}{|D|}$

An interesting measure we can now define is *precision*, also called the *positive prediction value.* This expresses the estimated probability that an example which is predicted as *positive* is correctly classified.

Definition 2.13. Precision $Prc(D, c_i) = \dfrac{TPs}{TPs + FPs}$

On the other hand we can also define *recall* also called the *sensitivity*, or the *true positive rate*. This expresses the estimated probability that an example which actual value is *positive* is correctly predicted.

Definition 2.14. Recall $Rcl(D, c_i) = \dfrac{TPs}{TPs + FNs}$

Figure 2.11 shows a Venn diagram, with the left circle containing all the actual positive examples and the right circle containing the predicted positive examples. This figure can help understanding the *precision* and *recall* measures. *Precision* is the percentage of TPs of the *predicted positive examples* and *recall* is the percentage of TPs of the *actual positive examples.*

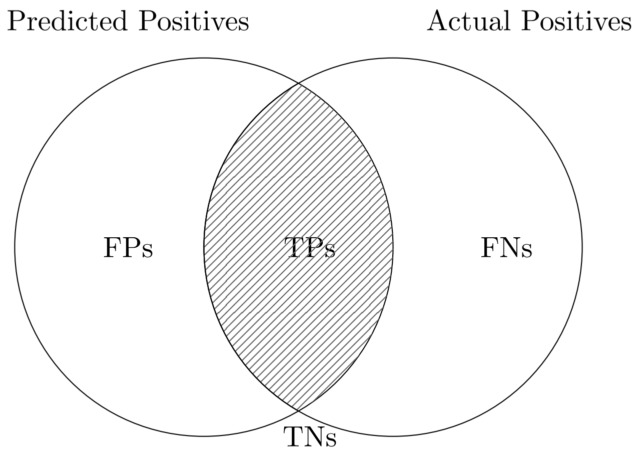


FIGURE 2.11: A Venn diagram of actual positive examples and predicted positive examples, illustrating the precision and recall measures

Table 2.3 shows the precision and recall of some trivial binary classifiers taken from the paper of Sebastiani et al. [1]. The trivial rejector always classifies the non-membership for a class, as opposed to the trivial acceptor which always classifies the membership for a class. The trivial 'YES' Collection is a test set which contains only positive examples and the trivial 'NO' Collection is a test set which contains only negative examples. The C-Precision and C-Recall are the precision and recall calculated by switching the negative and positive classes.

Note that previously defined measures are for one specific class label c_i. To measure a *multi-class* classifier we have to average out the classes somehow. There are two different methods of doing this called *microaveraging* and *macroaveraging*. In the first case all TPs, TNs, FPs and FNs for each class are summed up and then the average is taken. This treats each single example of the same importance. The latter applies the measure for each class and then takes the average of that, which treats each class of the same importance. There can be quite a difference in performance when using micro- or macroaveraging, especially with datasets that have a skewed class distribution.

Table 2.3: Trivial cases in Text Classification, taken from [1]

	Precision $\frac{TPs}{TPs+FPs}$	Recall $\frac{TPs}{TPs+FNs}$	C-Precision $\frac{TNs}{FPs+TNs}$	C-Recall $\frac{TNs}{TNs+FNs}$
Trivial Rejector $TPs = FPs = 0$	undefined	$\frac{0}{FNs} = 0$	$\frac{TNs}{TNs} = 1$	$\frac{TNs}{TNs+FNs}$
Trivial Acceptor $FNs = TNs = 0$	$\frac{TPs}{TPs+FPs}$	$\frac{TPs}{TPs} = 1$	$\frac{0}{FPs} = 0$	undefined
Trivial 'YES' Collection $FPs = TNs = 0$	$\frac{TPs}{TPs} = 1$	$\frac{TPs}{TPs+FNs}$	undefined	$\frac{0}{FNs} = 0$
Trivial 'No' Collection TPs=FNs=0	$\frac{0}{FPs} = 0$	undefined	$\frac{TNs}{FPs+TNs}$	$\frac{TNs}{TNs} = 1$

Definition 2.15. Microaveraging Precision $Prc^{micro}(D) = \dfrac{\sum_{c_i \in \mathcal{C}} TPs(c_i)}{\sum_{c_i \in \mathcal{C}} TPs(c_i) + FPs(c_i)}$

Definition 2.16. Microaveraging Recall $Rcl^{micro}(D) = \dfrac{\sum_{c_i \in \mathcal{C}} TPs(c_i)}{\sum_{c_i \in \mathcal{C}} TPs(c_i) + FNs(c_i)}$

Definition 2.17. Macroaveraging Precision $Prc^{macro}(D) = \dfrac{\sum_{c_i \in \mathcal{C}} Prc(D, c_i)}{|\mathcal{C}|}$

Definition 2.18. Macroaveraging Recall $Rec^{macro}(D) = \dfrac{\sum_{c_i \in \mathcal{C}} Rcl(D, c_i)}{|\mathcal{C}|}$

Utility In some classification applications the different misclassification costs should not be considered equal. Custom utilities $u_{TP}, u_{FP}, u_{FN}, u_{TN}$ can be defined to express the gain or loss brought about by a TP, FP, FN and TN respectively. In general cases $u_{TP} = u_{TN} > u_{FP} = u_{FN}$ which just states that a misclassification should bring less gain than a correct classification. But utilities are particularly useful in applications where this is not the case. For example when evaluating a classifier to filter spam e-mail messages. A legitimate e-mail message predicted to be spam (FN) should be avoided. But the other way around, classifying a spam email as legitimate (FP) is tolerable. In this case the utilities are $u_{FN} < u_{FP} < u_{TP} = u_{TN}$ [1].

2.7.3.2 Multi-label classification measures

In MC a misclassification is no longer a hard *wrong* or *right*. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them. There have been some proposals in the literature for measures that capture this. Tsoumakas divides them into two groups: *label based* and *example based* measures [21]. *Label based* measures use BC measures per label and average them out using micro- or macroaveraging. Just like we showed in the previous section. On the other hand example based measures look at each example and their set of predicted labels.

A popular *example based* measure is the Hamming-loss [43]. Let D be a set of observations and Δ the *symmetric distance* between the output of the target function $\breve{\Phi}$ and the classifier Φ. The symmetric distance Δ contains the classes that do occur in one set but not in the other. For example, $\{A, E, G\} \; \Delta \; \{E, F, G\} = \{A, F\}$. It's a loss function so the better the prediction the lower the value for *Hamming-Loss*.

Definition 2.19. Hamming-Loss $Hml(D) = \frac{1}{|D|} \sum_{x \in D} \frac{|\breve{\Phi}(x) \Delta \Phi(x)|}{|C|}$

Godbole and Sarawagi [16] have used adapted multi-label versions of *precision*, *recall*, *F1* and accuracy as follows.

Definition 2.20. Multi-label Precision $Prc_{ml}(D) = \frac{1}{|D|} \sum_{x \in D} \frac{|\breve{\Phi}(x) \cap \Phi(x)|}{|\breve{\Phi}(x)|}$

Definition 2.21. Multi-label Recall $Rcl_{ml}(D) = \frac{1}{|D|} \sum_{x \in D} \frac{|\breve{\Phi}(x) \cap \Phi(x)|}{|\Phi(x)|}$

Definition 2.22. Multi-label F1 $F1_{ml}(D) = \frac{1}{|D|} \sum_{x \in D} \frac{2|\breve{\Phi}(x) \cap \Phi(x)|}{|\breve{\Phi}(x)| + |\Phi(x)|}$

Definition 2.23. Multi-label Accuracy $Acc_{ml}(D) = \frac{1}{|D|} \sum_{x \in D} \frac{|\breve{\Phi}(x) \cap \Phi(x)|}{|\breve{\Phi}(x) \cup \Phi(x)|}$

Combining measures Sebastiani [1] argued that in MC, neither precision nor recall alone can effectively measure the performance of a classifier in isolation of each other. He notes that this does not hold for SC where *precision* and *recall* are dependent of each other. A classification to the wrong category (decreasing precision), automatically means it's not classified to the right category (decreasing *recall*). Therefore in SC either *precision* or *recall* can be used for measuring the performance. In MC however this is not the case.

Take for example the trivial acceptor from table 2.3, this classifier will obtain a *recall* of 1 for c_i, because it has no FNs. In contrast its *precision* will depend on the number of FPs. In practice, when tuning a classifier, obtaining a higher *precision* yields a lower *recall* [1]. Therefore a combination of both measures should be used. A number of methods have been proposed in the literature, we will list the most popular three [3]:

- Break-Even Point: The Break-Even Point, proposed by Lewis [44], between the *precision* and *recall* curve, in other words the point on the curve where they are equal. Sometimes this point does not exist and an interpolated point has to be calculated.

- F_β Measure: Proposed by Rijsbergen [45] is a measure which uses a value β which indicates the importance of *precision* or *recall*. Using $\beta = 0$ indicates only *recall* is relevant and $+\infty$ indicates that only *precision* is important. Usually a value of 1 is chosen, which assigns equal importance to both *precision* and *recall*, called the F1 measure.

  Definition 2.24. $F_\beta = \dfrac{(\beta^2 + 1) \cdot Prc \cdot Rcl}{\beta^2 \cdot (Pcr + Rcl)}$

- Average 11-point precision: The classifier is tuned such that the *recall* takes up the eleven values $\{0.0, 0.1, \ldots, 1.0\}$ and the *precision* at each point is calculated and averaged out over those points. This method is mainly used in document ranking applications [1].

Chapter 3

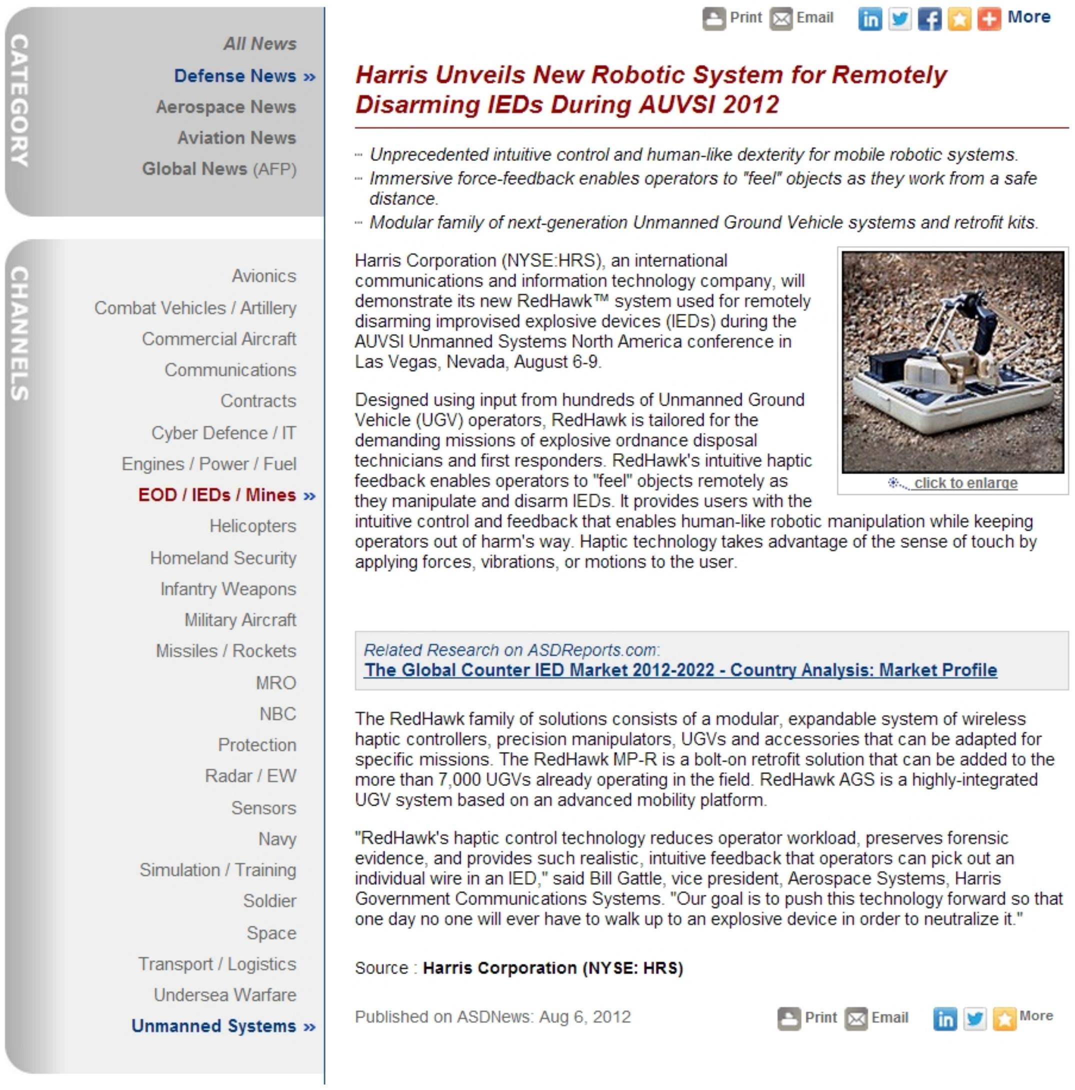
The data

In this chapter we will describe the data that was available for classification. We will provide some interesting statistics about the data and show which choices were made regarding which attributes and data to select. Furthermore, the transformation of the data labels into a hierarchical structure will be explained.

3.1 Introduction

The data we use in our experiments is a manually labeled database of news articles by ASDMedia. The articles are labeled with categories which provide information about the market the article is for and the channel the article is in. Basically the data consists of news articles for two different markets, namely news about the 'aerospace and defense market' and news about the 'energy and resources' market which we will refer to as 'ASD' and 'GEW' dataset respectively. The news is a selection of news articles issued from industry, government, and the largest, global press agencies, such as the Associated Foreign Press (AFP). Articles are focused on contract announcements, new developments and hot topics in the marketplace. All the news is in the English language.

Take a look at the example news article in figure 3.1. We can clearly see the title and text of the article along with the source, which is presented below the text. On the left of the article we can see a list of categories and channels, where a category or a channel is highlighted when it is labeled to this news article. This specific article is labeled with the category 'Defense News' and with the Channels 'EOD / IEDs / Mines' and 'Unmanned Systems'. EOD stands for 'Explosive Ordinance Disposal' and IEDs stands for 'Improvised Explosive Device'. The article is about an unmanned robotic system that can remotely disarm IEDs, which justifies the labellings of the channels. In the



Harris Unveils New Robotic System for Remotely Disarming IEDs During AUVSI 2012

- *Unprecedented intuitive control and human-like dexterity for mobile robotic systems.*
- *Immersive force-feedback enables operators to "feel" objects as they work from a safe distance.*
- *Modular family of next-generation Unmanned Ground Vehicle systems and retrofit kits.*

Harris Corporation (NYSE:HRS), an international communications and information technology company, will demonstrate its new RedHawk™ system used for remotely disarming improvised explosive devices (IEDs) during the AUVSI Unmanned Systems North America conference in Las Vegas, Nevada, August 6-9.

Designed using input from hundreds of Unmanned Ground Vehicle (UGV) operators, RedHawk is tailored for the demanding missions of explosive ordnance disposal technicians and first responders. RedHawk's intuitive haptic feedback enables operators to "feel" objects remotely as they manipulate and disarm IEDs. It provides users with the intuitive control and feedback that enables human-like robotic manipulation while keeping operators out of harm's way. Haptic technology takes advantage of the sense of touch by applying forces, vibrations, or motions to the user.

Related Research on ASDReports.com:
The Global Counter IED Market 2012-2022 - Country Analysis: Market Profile

The RedHawk family of solutions consists of a modular, expandable system of wireless haptic controllers, precision manipulators, UGVs and accessories that can be adapted for specific missions. The RedHawk MP-R is a bolt-on retrofit solution that can be added to the more than 7,000 UGVs already operating in the field. RedHawk AGS is a highly-integrated UGV system based on an advanced mobility platform.

"RedHawk's haptic control technology reduces operator workload, preserves forensic evidence, and provides such realistic, intuitive feedback that operators can pick out an individual wire in an IED," said Bill Gattle, vice president, Aerospace Systems, Harris Government Communications Systems. "Our goal is to push this technology forward so that one day no one will ever have to walk up to an explosive device in order to neutralize it."

Source : **Harris Corporation (NYSE: HRS)**

Published on ASDNews: Aug 6, 2012

Figure 3.1: An example news article from the website `www.asdnews.com`

next sections we will tell you more about the attributes of a news article that can be used for automatic classification and about the category and channel labels.

The 'ASD' dataset contains news articles that have been labeled since 1 January 2008 and it contains roughly 22,600 records available for classification. This makes the dataset considerably larger than the 'GEW' dataset, because this one has been collected since February 2011 and has roughly 6,000 records available for classification. The manual labellings are done by one person who started the task from the beginning. The articles we will use for classification were extracted from the database on 4 December 2012.

The task of labeling an article with the right channels is pretty difficult and is not clear-cut. The example article might as well be labeled with the channel 'sensors', because of the sensors in the robot that provide the feedback system. In this case there is chosen to exclude the sensors label, because the main message of the article is not about sensors. However, labeling the article with 'sensors' is not clearly a mistake.

3.2 Attributes for classification

The news articles are stored in a database as a collection of attributes. In table 3.1 you see a selection of the attributes that are potentially interesting to use in the learning process. MAX,AVG and Distinct stands for the maximum, average and distinct number of examples with the same value for that attribute respectively.

TABLE 3.1: The attributes of a news article from both datasets up to 4-12-2012

ASD dataset of total 27,130 examples					
Attribute	**Missings**	**MAX**	**AVG**	**Distinct**	**Description**
Title	0(0%)	6	1	26,953	The title of the news article
Full-text	0(0%)	1	1	27,130	All the text except the title
Source	4,481(17%)	1,525	17	1,276	The news source
Issue place	14,762(54%)	1,079	6	2,049	The issue location
GEW dataset of total 6,261 examples					
Attribute	**Missings**	**MAX**	**AVG**	**Distinct**	**Description**
Title	0(0%)	3	1	6,207	The title of the news article
Full-text	0(0%)	1	1	6,260	All the text except the title
Source	14(0%)	2,585	5	1,196	The news source
Issue place	712(11%)	215	4	1,333	The issue location

The title and full-text are clearly available to use for text classification. However for the source and *issue place* attribute this is not so clear. Especially in the 'ASD' dataset there are a lot of missing values for those attributes, this is because those attributes have not been collected from the start. The 'GEW' dataset, which is newer, has less missing values, but still 11% for *issue place*.

Interesting is that there are some news articles that have the same title, but have a different full-text. On further investigation we found out that these are actually the same articles stored on a different time in the database. There are around 150 of those duplicates, where the full-text is just a little bit different. More interesting is that identical articles are sometimes not labeled with the same channels. This raises questions about the consistency of the labellings.

Another interesting fact is the maximum examples with the same source attribute. For the 'GEW' dataset these are 2,585 articles with the same source, around 41% of the dataset. This actually is the source 'AFP', where a lot of articles come from.

Domain experts have indicated that the source attribute could be correlated to the label but the issue place shouldn't. A news source could have a certain selection of categories it produces news about. Therefore we will investigate if the source attribute contributes to the performance of the classification system. To handle the missing values, we will mark missing values with a '?', which is then handled by the decision tree (DT) learning algorithm as a single source. The *issue place* will not be investigated due to the large number of missing values and the judgment of domain experts.

To give some more insight into the text fields, in table 3.2 we show some statistics about them. We provided the average number of characters in the title and the full-text. The number of distinct words showed is after removing very rare words. Interesting is that the 'GEW' dataset has almost half of the distinct words as the 'ASD' dataset, which is probably because the size of the 'GEW' dataset is around 4.3 times smaller as the 'ASD' dataset.

Table 3.2: Some text statistics about both datasets

Dataset	**Avg. title characters**	**Avg. full-text characters**	**Distinct words**
ASD	57	2,228	23,127
GEW	58	2,441	11,487

In table 3.3 and 3.4 we show the top most occurring words of the 'ASD' and 'GEW' dataset respectively. These words have already been stemmed. Interesting to see is that the words 'said', 'quot' and 'year' are in both datasets in the top 10 words.

Table 3.3: The top 10 most occurring words in the 'ASD' dataset

Word	**Total Occurrences**	**Document Occurrences**
system	67,344	15,496
said	52,326	19,463
aircraft	39,039	10,944
air	33,097	10,434
support	26,432	11,227
quot	25,372	3,295
force	25,123	9,575
year	23,343	12,560
program	22,718	9,397
provide	22,484	12,226

3.3 Labeled categories

Each news article in the datasets is labeled with two different categories. One category represents the general market which the article is for, which we will refer to as *market category*. The other is a more specialized category which tells what the news article is

Table 3.4: The top 10 most occurring words in the 'GEW' dataset

Word	**Total Occurrences**	**Document Occurrences**
energy	17,691	4,127
quot	14,476	1,874
said	13,736	4,461
pow	12,720	2,909
solar	12,490	1,528
project	9,586	2,607
oil	9,458	2,032
gas	8,523	2,058
company	7,963	3,432
year	7,472	3,601

about, which we will refer to as the *channel category.* This *channel category* can be seen as a topic of the news article but also as a more specialized market within the more general *market category.* Each news article can be labeled with one or more *market categories* and also with one or more *channel categories.*

3.3.1 Market categories

We will show examples of all the market categories occurring in the 'ASD' and 'GEW' datasets in table 3.5 and 3.6 respectively. Each article in the 'ASD' dataset has on average 1.27 *market categories* as opposed to articles in the 'GEW' dataset, having on average 1.05 *market categories.*

Table 3.5: The *market categories* of all news articles from the 'ASD' dataset up to 4-12-2012

ASD dataset of total 27130 examples Each article having on average **1.27** *market categories*		
Market category	**Examples**	**Percentage**
Aviation	6,321	23.30%
Defence	16,536	60.95%
GlobalNews	1,747	6.43%
Space	543	2.00%
Aerospace	9,294	34.26%
Financial	18	0.06%

Not all of these *market categories* are useful, mainly because of the low occurrence. Some of them had a temporary existence and some are actually indicating something else than the market the news article belongs to. The 'GlobalNews' in the 'ASD' dataset and 'Global' in the 'GEW' dataset are examples of a *market category* which is actually not indicating a market. The category was originally intended to indicate news articles originating from the 'AFP'. Interesting to note is that the *GEW* dataset, with on average

Table 3.6: The *market categories* of all news articles from both datasets up to 4-12-2012

GEW dataset of total 6261 examples Each article having on average **1.05** *market categories*		
Market category	**Examples**	**Percentage**
Sustainable	3,452	55,13%
Global	126	2,01%
Distribution	237	3,79%
Traditional	2,743	43,81%

1.05 market categories per article is hardly *multi-label*. This is because the 'Sustainable' and 'Traditional' categories almost exclude each other.

3.3.2 Channel categories

The *channel categories* are the more specialised categories which are supplied in addition to the *market categories*. Some of the *channel categories* can belong to only one *market category*, but it is also possible that it belongs to multiple *market categories*.

We will show examples of *channel categories* in table 3.7 for the ASD dataset and table 3.8 for the GEW dataset.

3.4 A hierarchical labeling

The *market-* and *channel categories* are clearly correlated. For example an news article in the 'ASD' dataset labeled with the *channel category* 'Homeland Security' will belong to the 'Defense' *market category*. Because a news article is multi-label it may also belong to other *market categories* but also to other *channel categories*. Because both objectives need to be classified, we decided to create a new category structure which encapsulates both categories. It is clear that the *channel categories* belong to one or more *market categories*, the first being *more-specific* than the latter. Therefore these two categories are a good candidate to be merged into a hierarchical category structure. Now because a *channel category* can belong to more than one *market category* theoretically this is an *overlapping hierarchy* or *DAG based* hierarchy. However in practice working with such a hierarchy brings many implementation difficulties. Therefore we decided to force the structure to be Tree-based, meaning that each node in the hierarchy has at most one parent.

In order to do this the *channel categories* that belong to more than one *market category* are split into different *market categories*, one for each parent. For example the *channel*

Table 3.7: The *channel categories* of all news articles from the 'ASD' dataset up to 4-12-2012

ASD dataset of total 27130 examples Each article having on average **1.75** *channel categories.*		
Channel category	**Examples**	**Percentage**
Infantry Weapons	693	2.55%
Commercial Aircraft	5,652	20.83%
Avionics	578	2.13%
Military Aircraft	3,225	11.89%
Communications	3,249	11.98%
Protection	968	3.57%
Cyber Defence / IT	541	1.99%
Contracts	5,331	19.65%
Helicopters	1,740	6.41%
EOD / IEDs / Mines	341	1.26%
Transport / Logistics	753	2.78%
Combat Vehicles / Artillery	1,257	4.63%
Missiles / Rockets	1,743	6.43%
MRO	1,350	4.98%
Navy	2,012	7.42%
NBC	300	1.11%
Sensors	1,652	6.09%
Simulation / Training	2,018	7.44%
Soldier	608	2.24%
Space	3,814	14.06%
Unmanned Systems	1,658	6.11%
Undersea Warfare	473	1.74%
Engines / Power / Fuel	1,874	6.91%
Radar / EW	1,103	4.07%
Homeland Security	1,970	7.26%
Global	1,761	6.49%
Disasters / Accidents	699	2.58%
Financial / Economical	171	0.63%

categories 'Soldier' can occur only within the *market category* 'Defence' which means it doesn't need to be split. But the *channel category* 'Contracts', occurs in all the available *market categories* of the 'ASD' dataset. This one will be split into three new categories, one for 'Defense', one for 'Avionics' and one for 'Aviation'.

Tables 3.10 and 3.11 show the resulting category structure which we will use for the classification, *channel categories* and *market categories* not used anymore have been removed. We used lvl1, lvl2 and lvl3 to indicate the depth of the category in the hierarchy tree and assigned a unique code to each category. At the top-level of the hierarchy is the choice between the 'ASD' or 'GEW' dataset, which resembles the lvl1 category. This top-level decision is actually a single-label classification (SC) problem which we will not address in this research. The news is already collected separately

Table 3.8: The *channel categories* of all news articles from the 'GEW' dataset up to 4-12-2012

GEW dataset of total 6261 examples Each article having on average **1.20** *channel categories.*		
channel category	**Examples**	**Percentage**
Natural Gas & LNG	305	4.87%
Oil and Gas	1,843	29.44%
Refinery	48	0.77%
Fuel & Retailing	28	0.45%
Mining	37	0.59%
Wind	502	8.02%
General News	586	9.36%
Geothermal Energy	62	0.99%
Nuclear	549	8.77%
Solar	1,338	21.37%
Bio	445	7.11%
Hydro	169	2.70%
Pipelines	122	1.95%
Storage	209	3.34%
Power Generation	267	4.27%
Utilities	38	0.61%
Clean Transport	341	5.45%
Smart Grids	249	3.98%
Contracts	97	1.55%
Energy Efficiency	112	1.79%
Exploration	142	2.27%

because the decision is easily made based on the publisher of the news. Our research will treat each dataset separately and focus on the MC problem within each dataset. The lvl2 categories correspond to the *market categories* of each dataset and the lvl3 categories correspond to the *channel categories* of each dataset. Note that by predicting the lvl3 categories, the corresponding lvl2 categories can be deduced.

3.4.1 Multi-label statistics

To tell something about the multi-labeledness of the datasets some statistics are presented in table 3.9. For a full explanation of the measures the reader is referred to section 2.2.3. For reference we added the statistics of the Reuters dataset taken from the paper by Tsoumakas et al. [21]. This is a similar dataset often used in multi-label text classification. The statistics are the averages of the five 'rcv1v2' datasets from Reuters.

When you look at the label cardinality of the 'GEW' and 'ASD' dataset there is quite a difference. This is explainable because the two lvl2 categories in the 'GEW' dataset are 'Traditional Energy & Exploration' and 'Sustainable Energy'. The two categories are almost mutually exclusive, which we also observed when looking at the *market categories*. When we look at the 'ASD' dataset, the lvl2 categories are much more overlapping. The 'Aerospace' lvl2 category is about manufacturing of aircrafts, space vehicles, guided missiles. 'Aviation' on the other hand is news about existing aircrafts, airlines and airports. An article about a company manufacturing parts of aircrafts for a specific airline will be labeled with both categories.

When comparing the two datasets with the Reuters dataset we can see some similarities and differences. The main difference is that the Reuters dataset has more labels and a higher label cardinality. The latter means that on average an example from the Reuters dataset is labeled with more categories than examples from the 'ASD' or 'GEW' dataset. Interesting to see is that the 'ASD' dataset has more distinct label sets than the Reuters and 'GEW' dataset.

We suspect that the 'ASD' dataset is more difficult, in the multi-label sense, than the 'GEW' dataset and suspect that it is harder to classify correctly.

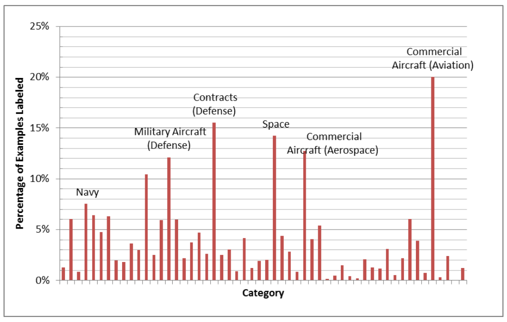
Table 3.9: Multi-label statistics of different datasets

Dataset	**Examples**	**Labels**	**Cardinality**	**Density**	**Distinct Label set**
ASD	26042	54	2.1757	0.040	1972
GEW	5965	24	1.2055	0.050	184
Reuters	6000	101	2.6508	0.026	937

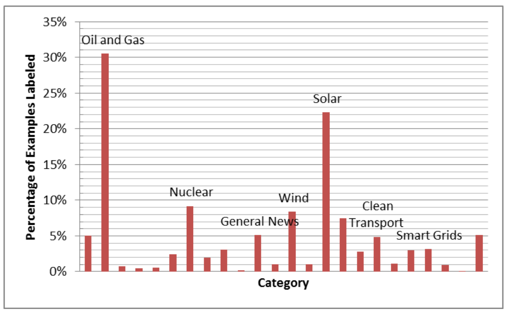
3.4.2 Skewed distribution of categories

The class distribution of both datasets is very skewed. To illustrate this, graph 3.2a and 3.2b show the categories and the percentage of examples labeled with them for the 'ASD' and 'GEW' dataset respectively. Tables 3.10 and 3.11 show all the data, note that in a multi-label setting the percentages don't necessarily sum up to 100. There are single categories that are labeled to 20% of the examples, as is the case with *commercial aircraft (aviation)* in the 'ASD' dataset. And for the 'GEW' dataset there are even two dominating categories such as *Solar* 20% and *Oil and Gas* 30%. On the other hand there are also categories labeled to less than 1% percent of the data.

Because we are planning to use the 'one-against-all' approach, one classifier is trained for each category. A classifier has to distinguish that category from all the other categories. Therefore it is expected that a classifier for categories with few positive examples may not perform that well. Because of a practical limitation we require our categories to have at least 20 examples. This is because, when using 10-fold cross validation, the chance



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 3.2: This graph shows for each lvl3 category, the percentage of total examples labeled with it.

of having a fold with no positive examples is very small. So the following categories in tables 3.10 and 3.11 will be excluded from our experiments:

- NBC under Aviation with 3 positive examples
- Utilities under Traditional Energy & Exploration with 10 positive examples
- Utilities under Sustainable Energy with 8 positive examples

Table 3.10: The hierarchical categorization for the 'ASD' dataset up to 4-12-2012

ASD dataset of total 26,042 examples				
lvl2	**lvl3**	**Category**	**Examples**	**Percentage**
1010	-	Defence	16,411	63.017%
	101010	EOD / IEDs / Mines	330	1.267%
	101014	Homeland Security	1,569	6.025%
	101018	NBC	224	0.860%
	101020	Navy	1,969	7.561%
	101024	Missiles / Rockets	1,677	6.440%
	101028	Combat Vehicles / Artillery	1,244	4.777%
	101030	Simulation / Training	1,641	6.301%
	101034	Cyber Defence / IT	516	1.981%
	101038	Undersea Warfare	467	1.793%
	101040	Protection	953	3.659%
	101044	MRO	775	2.976%
	101048	Communications	2,722	10.452%
	101050	Transport / Logistics	655	2.515%
	101054	Sensors	1,552	5.960%
	101058	Military Aircraft	3,152	12.104%
	101060	Unmanned Systems	1,558	5.983%
	101064	Soldier	571	2.193%
	101068	Radar / EW	976	3.748%
	101070	Helicopters	1,220	4.685%
	101074	Infantry Weapons	685	2.630%
	101078	Contracts	4,043	15.525%
	101080	Space	650	2.496%
	101084	Engines / Power / Fuel	787	3.022%
	101088	Avionics	230	0.883%
1020	-	Aerospace	9,809	37.666%
	102010	Military Aircraft	1,084	4.163%
	102014	Sensors	324	1.244%
	102020	Unmanned Systems	499	1.916%
	102024	Simulation / Training	535	2.054%
	102030	Space	3,706	14.231%
	102034	Engines / Power / Fuel	1,139	4.374%
	102040	Helicopters	745	2.861%
	102044	Radar / EW	220	0.845%
	102050	Commercial Aircraft	3,324	12.764%
	102054	Communications	1,061	4.074%
	102060	Contracts	1,414	5.430%
	102064	Protection	42	0.161%
	102070	Transport / Logistics	123	0.472%
	102074	Avionics	385	1.478%
	102080	Missiles / Rockets	114	0.438%
	102084	Cyber Defence / IT	57	0.219%
	102090	MRO	538	2.066%
1030	-	Aviation	6,253	24.011%
	103010	Avionics	331	1.271%
	103014	Simulation / Training	309	1.187%
	103020	Engines / Power / Fuel	809	3.107%
	103024	Radar / EW	132	0.507%
	103030	MRO	570	2.189%
	103034	Homeland Security	1,569	6.025%
	103040	Contracts	1,024	3.932%
	103044	Communications	195	0.749%
	103050	Commercial Aircraft	5,215	20.025%
	103060	Transport / Logistics	77	0.296%
	103070	Helicopters	628	2.411%
	103080	NBC	3	0.012%
	103090	Sensors	324	1.244%

Table 3.11: The hierarchical categorization for the 'GEW' dataset up to 4-12-2012

GEW dataset of total 5,965 examples				
lvl2	**lvl3**	**Category**	**Examples**	**Percentage**
2010	-	Traditional Energy & Exploration	2,946	49.388%
	201010	Natural Gas & LNG	300	5.029%
	201014	Oil and Gas	1,820	30.511%
	201020	Refinery	45	0.754%
	201024	Fuel & Retailing	27	0.453%
	201030	Mining	36	0.604%
	201034	Exploration	142	2.381%
	201040	Nuclear	547	9.170%
	201044	Pipelines	119	1.995%
	201050	Power Generation	186	3.118%
	201054	Utilities	10	0.168%
	201060	General News	306	5.130%
	201064	Energy Efficiency	62	1.039%
2020	-	Sustainable Energy	3,227	54.099%
	202010	Wind	499	8.365%
	202014	Geothermal Energy	61	1.023%
	202020	Solar	1,329	22.280%
	202024	Bio	443	7.427%
	202030	Hydro	165	2.766%
	202034	Clean Transport	288	4.828%
	202040	Energy Efficiency	68	1.140%
	202044	Storage	178	2.984%
	202050	Smart Grids	187	3.135%
	202054	Power Generation	56	0.939%
	202060	Utilities	8	0.134%
	202064	General News	309	5.180%

Chapter 4

Methodology

In this chapter we will describe the experimental setup. First we will describe the basic experimental setup and justify the choices we made during that setup. Finally we will describe each experiment that we will conduct in detail.

Recall that the main question we will address during this research is the following:

Is it possible to develop an automatic classification system with the following features to solve the ASDMedia classification problem?

- Use Machine Learning (ML) to generate the model
- The ML algorithm is restricted to those which generate humanly interpretable models
- The classifier achieves a reasonable performance

With the ASDMedia classification problem we actually indicate two problems, one for each dataset. One dataset contains news articles about the 'Aerospace & Defence' market and the other about the Energy market, which we will refer to as the 'ASD' and 'GEW' dataset respectively. Each article can be labeled with multiple categories, which makes them both a Multi-Label Classification (MC) problem. We will investigate various techniques and settings which we explain next. By doing these experiments we also try to reach the best combination of settings for both classification problems. We apply a kind of greedy search strategy, and continue each experiment with the best result from the previous experiment. We are aware of the fact that we do not cover all possible combinations of settings, so may not find the real optimal setting. However we must exclude some combinations due to practical limitations.

4.1 The basic setup

In this section we will explain the basics of our classification system. In our experiments we will compare different settings and approaches with each other, but what they have in common will be described in this section. Al our experiments are done with the RapidMiner software [46]. One of the largest open-source data-mining systems, originally developed by YALE University.

4.1.1 The multi-label problem

To tackle the MC problem we have chosen to use the *binary relevance* (BR) a.k.a *one-against-all* approach. As discussed in section 2.2.1 this approach is widely used in the literature but may not take into account label correlations. Theoretically there is a better approach, the *label power-set* (LP) method, but this approach is computationally infeasible. The number of distinct label sets for the 'ASD' dataset are 1972, see table 3.9. This would mean training 1972 different classifiers that would require a lot of computation time. For the 'GEW' dataset it might be possible with only 184 distinct label set. However we want to use the same solution for both datasets.

4.1.2 The skewed class distribution problem

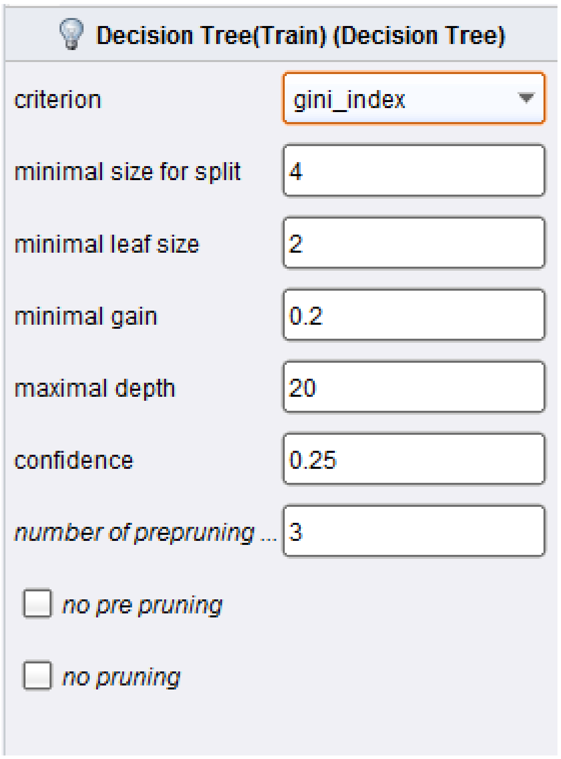
A discussed in section 2.3.2 a skewed class distribution in combination with the BR method can cause problems in the test and training phase. In the training phase the main problem is that a model will be learned that just assigns every example to the majority class, thereby achieving a high accuracy. In the test phase this can be solved by using an appropriate performance measure which we will discuss later in section 4.1.4. For the training phase, we have to prevent the classification algorithm from learning a model that just assigns every example to the majority class. One solution is to give each example a weight, so that the sum of the weights of positive examples equals the sum of weights of negative examples for a class. This has the effect of balancing the positive and negative examples for a classifier. Other possibilities are oversampling the minority class and under sampling the majority class, or both.

In an extensive study by Mccarthy, Zabar and Weiss [47] they compared over sampling, under sampling and example weights. In almost all cases under sampling scored worse than over sampling and example weights. Between oversampling and under sampling there was no clear winner, but in small datasets oversampling achieved a better performance. We have chosen to use example weights because it is one of the best methods to deal with a skewed class distribution.

4.1.3 Classification algorithm

One of the constraints was, when looking at solutions of our problem, that the resulting classifier is humanly interpretable. This is so that domain experts can read the learned model and compare it with what they think the model should look like. This constraint also prevents us from using techniques that generate multiple models and average the result, like boosting. Therefore we have chosen to use a *decision tree* (DT). An alternative for that would be a *rule based* classifier, which is as expressive as a DT. More specifically we use an algorithm that is similar to Quinlan's C4.5 in RapidMiner, see figure 4.1 for the parameters we used.

We used the default values for the parameters, except for the splitting criterion for which we used the *Gini index*. This decision was based on a small test we performed with other splitting criteria. The confidence is one of the most important parameters which influences the post-pruning of the DT. This pruning is used to make the DT smaller and prevent overfitting. We excluded experimenting with the DT parameters in our research due to a lack of time.


Decision Tree(Train) (Decision Tree)

criterion	gini_index
minimal size for split	4
minimal leaf size	2
minimal gain	0.2
maximal depth	20
confidence	0.25
number of prepruning ...	3

☐ *no pre pruning*

☐ *no pruning*

FIGURE 4.1: A form to alter the decision tree parameters in RapidMiner

4.1.4 Validation & evaluation setup

To evaluate and compare our experiments we will use the 10-fold cross validation setup. To partition the data into 10 folds, we used the iterative stratification algorithm by Sechidis et al. [42]. For MC problems this is not a trivial task, see section 2.7 for more details. Briefly, this method divides the dataset into 10 folds, while trying to maintain the label distribution of the original dataset. It does so by distributing the positive and negative examples for each category evenly among the 10 folds.

For our experiments we will use the micro-averaged F1 measure as the main performance measure, which is the harmonic mean of precision and recall. The F1 measure is defined in definition 2.24 in section 2.7.3.2. To obtain the micro-averaged F1, the micro-averaged precision and micro-averaged recall should be used in the formula which are defined in definition 2.15 and 2.16 respectively in section 2.7.3.1.

The reason why we have chosen for the micro-averaged F1 measure is because we have a skewed class distribution and we use the one-against-all method. Accuracy is not an adequate performance measure in this situation, see section 2.3.2. The precision and recall measures are more suitable in this setting, because they measure the positive class. However they should not be used in isolation of each other, see section 2.7.3.2. The F1 measure is such a combination of precision and recall, which is widely adopted in the literature.

We have chosen to use micro-averaging F1 instead of the macro-averaging F1. This is because the main goal is to predict as many labels from as many previously unseen documents as possible. If we would adopt macro-averaging, then the categories with very few examples are treated of equally importance as the categories with more examples. In our case, some categories have so few examples that they are already expected to perform very badly. Therefore we want to set the focus on the number of documents that a category is labeled on instead.

4.1.5 Feature generation

To transform our dataset to a matrix with features that the classification algorithm can use, *document indexing* is applied. In the general setup both the *full-text* and the title of a news article are used to extract features from. We will also experiment with using parts of the *full-text*. A screenshot of the document indexing procedure from *RapidMiner* is showed in figure 4.2, which shows the individual steps that are done. More background information about each step is given in section 2.5.1.1

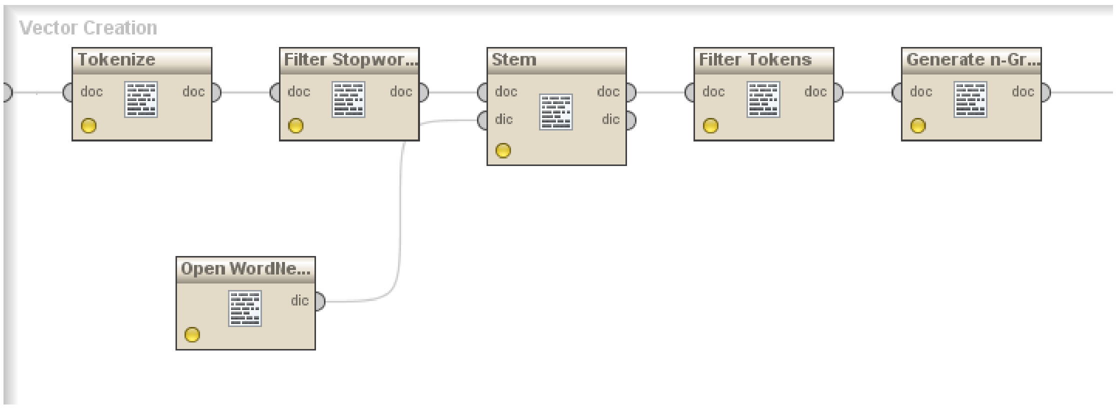


FIGURE 4.2: The document indexing procedure in RapidMiner which iterates over all documents and applies these steps

1. **Tokenization:** First the text is *tokenized* on non-letters. This means that on every character that is not a letter, like for example a space or a punctuation mark, the text is split. What remains is a list of tokens that resemble individual words.

2. **Filter stopwords:** Then from that list of words the stopwords are filtered out. Those stopwords are a built-in list of basic English stopwords.

3. **Stemming:** Now, using the *WordNet* dictionary stemming is applied. Wordnet [48] is an electronic lexical dictionary maintained by Princeton University.

4. **Filter tokens:** In this step tokens are removed that have less than three characters. Because we have split on non-characters, there can be left overs from incorrect splits at punctuation marks. Besides most tokens less than three characters are probably not really descriptive in the English language.

5. **Generate n-grams:** Depending on the setup, either 1-grams or up to 2-grams are generated from the words. These are finally used as features.

Now for each feature that is generated the *term frequency* (TF) and the *inverse document frequency* (IDF) are calculated. These features that occur in less than 4 documents are removed from the list. This was indicated by Sebastiani [1] as an empirical method widely used to remove the very rare words. The rest of the features is kept, also the very common words. As was indicated by Sebastiani it was a common mistake to remove the common words. Furthermore we noticed that for a category that contains only 1 percent positive examples, removing rare words that occur in less than 1 percent of the document would be a disaster. Very discriminating words that occur in all of the positive examples could also be removed. This could make it very difficult or maybe even impossible to learn a good classifier for those categories.

Finally for each feature the *TF-IDF* is calculated and the result is a *term matrix*, containing the documents as rows and the features as columns with as value the *TF-IDF* scores. Other options would be to just calculate the TF or even simpler to have only a boolean flag indicating whether or not the feature occurs in the document. The reason we have chosen for *TF-IDF* is because it is widely adopted in the text classification literature. The generated *term matrix* will be used to perform the feature selection on.

4.1.6 Feature selection

Our feature selection mechanism consists of two parts. First of all we perform feature selection based on the word frequency and we filter stopwords. This type of feature selection is done *globally*, for each category the same. The next part of our feature selection is done *locally*, for each category on its own. [1]

To decide which measure to use in our feature selection we applied a few measures and reported the top 500 features. We tested *Chi Squared*, *Information Gain* and *Gini Index*. We let domain experts look at the features that were selected and they found the *Chi Squared* measure to select the most promising features. To give an idea of which features score the highest *Chi Squared* statistics, in tables 4.1 and 4.2 you can see the top 10 features for two arbitrary categories for both the 1-gram and up to 2-gram settings.

A form of feature selection which is not considered part of feature selection occurs inside the DT induction algorithm. The algorithm will select a subset of the available features to split the data on. Recall that we also use the *Gini Index* as the method to select the feature that generates the best split in the DT. The tree is also pruned which can remove certain features again.

Table 4.1: The top 10 features from the category EOD / IEDs / Mines from the 'ASD' dataset

1-grams		**2-grams**	
Feature	**χ^2 weight**	**Feature**	**χ^2 weight**
explosive	1	explosive	1
improvised	0.817	explosive_device	0.871
ied	0.647	improvised	0.855
counter	0.358	improvised_explosive	0.850
device	0.305	ied	0.720
disposal	0.277	device_ied	0.374
bomb	0.261	counter	0.369
aircraft	0.259	device	0.341
threat	0.224	disposal	0.317
mine	0.224	bomb	0.290

Table 4.2: The top 10 features from the category Natural Gas & LNG from the 'GEW' dataset

1-grams		**2-grams**	
Feature	χ^2 **weight**	**Feature**	χ^2 **weight**
gas	1	gas	1
natural	0.413	natural	0.449
LNG	0.279	natural_gas	0.429
liquefy	0.244	LNG	0.268
solar	0.208	liquefy	0.248
cubic	0.185	cubic	0.242
pow	0.136	liquefy_natural	0.213
foot	0.097	solar	0.209
reserves	0.089	cubic_foot	0.167
shale	0.0815	pow	0.138

4.2 Experiment I: the feature representation of news articles

The first experiment is about how a news article eventually is translated to features used by the ML algorithm. We can make a distinction between *feature generation* and *feature selection*. The first is about how a news article is translated to features and the second is how we select from those features the ones to feed to the machine learning algorithm. We would like to find out which representation is the most suitable for these news articles. The best representation is selected to use in the second experiment.

4.2.1 Experiment variables

First we will give an overview of all settings we came up with that can be varied. After each item in parenthesis are showed the different variants. Later we will further discuss those settings in more detail and further explain the experimental setup.

- Selection of Features
  - **Number of features** We will vary the number of features that is selected which have the highest Chi Squared value. (40, 80 or 160)
  - **Custom blacklist:** We will try out a custom-made list of features that should not be selected. (include or exclude)
- Generation of Features
  - **Word modeling:** We will vary between using the *TF-IDF* of words (1-gram) as features or use the *TF-IDF* of tokens consisting up to two words (2-gram) as features. (1-gram or 2-gram)
  - **Text selection:** We will always use the title of a news article as input for feature generation but we will vary between selecting a part of the full-text. (no full-text, 500 characters, 1000 characters, all full-text)
  - **Attribute weighting:** We will apply different weights to features originating from the title and features originating from the full-text of the news article. (title/full-text ratio: 0.25, 0.5, 1.0, 2.0 and 4.0)
  - **The source attribute:** We will try out including the source attribute as additional feature. (include or exclude)

When we change the way features are generated, likely there is also another optimal way to select those features. Ideally we would like to test each setting in combination with each other, by using factorial design. But this would lead easily to a combinatorial explosion. ($3 * 2 * 2 * 4 * 3 * 2 = 288$ runs) Therefore we decided to exclude investigating *the source attribute* and the *attribute weighting* from the factorial experiment. This means that we run the factorial experiment with the source attribute excluded and an equal weighting among the title and full-text attributes. We will investigate these two settings after the factorial experiment, in isolation of each other. Knowing that if we had performed them in combination of the other settings in the first experiment, we may have had different results. However we have to make this limitation to avoid the combinatorial explosion.

To justify the exclusion of the source attribute, recall that it is a single nominal feature which has little to do with the text selection or feature representation. However it may have impact on both settings. When including the source attribute we will extend the feature space by one extra feature. This feature is not subject to feature selection.

The reason why we exclude the *attribute weighting* is mainly to save time. We cannot argue that this method is independent of *feature selection* and it depends a lot on the *text selection* choice.

4.2.2 The factorial experiment

The main experiment will be conducted according to the basic setup explained in section 4.1. We will try out every combination of the following settings: *number of features*, *custom blacklist*, *feature representation* and *text selection*. We will discuss them first in some detail and also explain our motives for varying these settings.

4.2.2.1 Number of features

Feature selection is a very important part of the classification process. We already have chosen our algorithm to rank our features to be the chi squared statistic, see section 4.1.6. However the number of features that are selected is also very important. Because we use *local* feature selection, generally less features are needed to achieve good performance, as was showed by Koller and Sahami [9]. However the ideal number of features is dependent on a lot of things like for example the dataset, the feature generation method and classification algorithm used. We have chosen to vary the number of features to be 40, 80 or 160 which we will refer to as **Features:40**, **Features:80** and **Features:160** respectively.

4.2.2.2 Custom blackList

The idea of a custom blacklist comes from a domain expert who looked at the features being selected and observed a number of features that should not have any predictive power. To get an idea which features are in the blacklist, you can see a part of the blacklist in table 4.3. These words are actually very common words in news articles that do not say anything specific about the markets of 'Aerospace & Defence' or 'Energy & Resources'. The idea is that those words are subtracted from the available features for feature selection. Such that there are better (from a domain expert perspective) features available to learn the model from. The blacklist is constructed by looking at the top 180 features that were being selected for a setting where the *word modeling* is 1-gram and the *text selection* is the title and the whole full-text. Per category words are selected to add to the blacklist, to receive a global blacklist which is the same for both datasets. In total the blacklist contains **104** words. We will refer to either including or excluding the blacklist as **Blacklist:yes** and **Blacklist:no**.

TABLE 4.3: A couple words from the global blacklist of features

Terms	
get	kit
use	getting
Sgt	corporal
wednesday	said
keep	tell
using	copy
guy	nyse
province	sunday
lieutenant	saying

4.2.2.3 Word modeling

The way the text is transformed to features is an important step in the document indexing procedure, see 2.5.1.1 for more information. We have chosen to test the 1-gram and up to 2-gram representation, although in the literature there are mixed results on using the 2-gram representation. However this can be different per domain and in our case domain experts think that certain two-word combinations may be very distinctive. We will refer to 1-gram versus up to 2-gram as **WordModeling:1-gram** and **WordModeling:2-gram** respectively.

4.2.2.4 Text selection

Domain experts indicated that when they classified news articles, they primary look at the title and if that is not specific enough they look at the first paragraph of the news article. Seldom they read the entire article. As is also the case with feature selection, less information is sometimes better. This is because there is probably also a lot of noise in the extra text. Another advantage is selecting only a part of the text is that it will reduce the computational complexity. Because of the way the data is stored, it was hard to distinguish the first paragraph from the complete full-text. Therefore we took from each article the words that occurred within the first 500 characters of the full-text. This 500 characters is close to the average length of the first paragraph. We decided to vary the text selection in four cases: selecting only the title, selecting the title and the words within the first 500 characters, selecting the title and the words within the first 1000 characters and the complete text. We will further refer to this settings as **TextSelection:Title**, **TextSelection:T+500c**, **TextSelection:T+1000c** and **TextSelection:All** respectively.

4.2.3 Side experiments

The winning setting from the main experiment will be used to conduct the two side experiments on. These are two experiments which will test the contribution of the source attribute and find the right attribute weighting. Finally the winning setting from the factorial experiment and the winning setting from the side experiments will be used in the second experiment. We will discuss the two settings in detail here.

4.2.3.1 Attribute weighting

This side experiment will try to find the best weighting between the title and the full-text. Domain experts have indicated that the title is the most important part of the text. To apply the weights, the calculation of the *TF-IDF* score is adjusted. When counting the term occurrences, the weight associated with the attribute where the term originates from is first applied. This results in a different TF value, while the IDF remains the same. In the factorial experiment the title and full-text both have equal weights, but in this experiment we will vary both weights. We will try different *title/full-text weight ratios*, namely: 0.25, 0.5, 1.0, 2.0 and 4.0. We will refer to these settings as **AttributeWeighting:0.25**, **AttributeWeighting:0.5**, **AttributeWeighting:1**, **AttributeWeighting:2** and **AttributeWeighting:4** respectively.

4.2.3.2 The source attribute

This sub-experiment will test the contribution of the source attribute to the classification process. As domain experts have indicated, the source attribute may be correlated to the category of the news article. To test this we simply add the source attribute as an additional nominal feature, regardless of the feature selection method. By adding a nominal feature, the problem is not strictly a text classification (TC) problem anymore. In the 'ASD' dataset there are some missing values for the source attribute, which is handled by the DT learning algorithm. We will test the performance of either including or excluding the source attribute, which we will further refer to as **SourceAttribute:yes** and **SourceAttribute:no** respectively.

4.3 Experiment II: exploiting the label structure

The purpose of the second experiment is to find out if we can exploit the structure within the labels. We will look at both the *external* and *internal* structure of the labels. External means that the structure is imposed on the labels from the outside, such as a hierarchy. A good example of an internal structure is the structure formed by the correlations between labels. The winning settings from the first experiment are used for this experiment. First we will list the experiments we came up with to perform. Later we will describe them in more detail.

4.3.1 Experiment outline

We will list here the different sub-experiments we came up with. Both experiments will be conducted according to the basic experimental setup, unless otherwise stated. Furthermore the winning settings from the first experiment are used here, there will not be experiments to find out the best settings in those setups. The approach here is compared to the approach we used in experiment I.

- **Chaining** We will try out the chaining technique in a flat (not using the hierarchy) classification (FC) problem to exploit the internal structure of the class labels.
- **Hierarchical top-down classification** We will try out a hierarchical top-down classification (HTC) system, to exploit the external structure of the class labels.

4.3.2 Classifier chains

In this experiment we will look at the *classifiers chains* (CCs) technique explained in section 2.2.2.2. We will use the winning settings from experiment I and compare the result with the approach when chaining is not applied. This technique is used to exploit the correlations between class labels and thus exploits the internal structure of our labels.

4.3.2.1 The order of the chain

In his paper Read et al. [18] created an ensemble of models with a random chaining order. In our case we restrict ourselves to one model only, so that it stays easy interpretable by humans. Because of this restriction we need to stick to one order of the CC.

Chaining attempts to exploit correlations between class labels. We observe that since correlation between two events is symmetric, if there are only two nodes in the chain, the order does not matter. Both nodes can be predicted by the other node just as good as the other way around. However when there are three or more variables, then the order does matter. Now we suddenly have more combinations, take for example as variables A, B and C. We can calculate *multiple correlations* between all three variables $C|AB$, $B|AC$ and $A|BC$. In this case however those correlations are not symmetric, and thus the order does matter. If $C|AB$ gives the highest measure of correlation it makes sense to put C at the end of the chain so that B and A can help predicting C. Now the order of B and A can still be determined in a recursive fashion.

With this idea in mind we use statistical multiple regression to determine the order of the chain as mentioned above. The *multiple correlation coefficient* determines how well a variable can be predicted by it's predictors.

4.3.2.2 Feature selection

The 'ASD' and 'GEW' have 53 and 22 categories respectively. The last classifier in the chain would then receive 52 and 21 extra features respectively. Because it is such a large number, we decided to perform an extra feature selection specially for the chaining features. This feature selection works the same as the normal feature selection, it uses the *Chi Squared* measure and selects the top x number of features. We decided to let x be proportional to the number of categories in the dataset minus one, because this is the maximum number of features a classifier can get. We decided to use the following percentages for x: 0%, 10%, 20%, 40% and 100%. For the 'ASD' datasets this means we will experiment with selecting 0, 5, 10, 21 and 52 chaining features. For the 'GEW' dataset we will experiment with selecting 0, 2, 4, 8 and 21 chaining features. When 0 chaining features are selected the classifiers will be the same as when no chaining is done. We will refer to these settings as **ChainingFeatures:**x, where x indicates the number of chaining features selected.

4.3.3 Hierarchical top-down classification

In this experiment we will try to exploit the hierarchy of the class labels. First we describe the hierarchical structure used. Then we will describe in more detail how the classification process looks like and what the differences are compared to FC. For more information about hierarchical classification (HC) the reader is referred to section 2.4.

4.3.3.1 The hierarchical structure

The hierarchy we use is explained in section 3.4. At the highest level (level-1) of the hierarchy is the single-label choice between the 'ASD' or the 'GEW' dataset. Because these datasets are already collected separately, we will just look at the hierarchy within each datasets. We will train for each dataset a multi-label top-down HC system, with for each dataset the winning settings from the first experiment. We will validate the two classification systems separately. So basically when we talk about the hierarchy in the context of top-down classification we mean the hierarchical structure formed by the level-2 and level-3 labels within one dataset. All level-2 and level-3 categories are showed in table 3.10 and 3.11 in section 3.4.

4.3.3.2 Hierarchical classification in contrast with flat classification

Hierarchical multi-label classification (HMC) has some differences with the FC setup we had in experiment I. First of all more classifiers are trained, namely for those labels in the second level of the hierarchy. For the classifiers at level 3 of the hierarchy the training data is adjusted. The positive examples stay the same, but for the negative examples only those examples that are labeled with a descendant from the same parent are included in the training data. This partitioning is according to the *sibling* approach, as explained in section 2.4.3. The *document indexing* and *multi-label stratification* remain the same, such that the same document term matrices are used as with FC. However when *feature selection* is performed and when a classifier is trained, only a subset of the training data is used.

The test phase is also different, because we have to apply the top-down approach. This means that we start with the classifiers at the highest level of the hierarchy, in this case the level-2 categories. When the membership of a category at level-2 is predicted, we go further down the hierarchy and use the classifiers of all the children of that category. When the non-membership of a level-2 category is predicted we will not use it‘s children classifiers. When a positive example for a level-3 category is negatively classified by the corresponding level-2 classifier, then the level-3 classifier will not get the chance to classify this example correctly. This is called the *blocking problem*.

There exist many different hierarchical evaluation measures. These approaches usually measure the performance of the whole classification system given the hierarchy. For example a misclassification to a category closer in the hierarchy is less problematic than a misclassification to a category far away in the hierarchy. However since we are comparing the hierarchical top-down approach with the flat approach, we need the same performance measures as we used in FC. It might be interesting however to report also the performances of the classifiers at the highest level of the hierarchy. Because if those classifiers don't perform well, they drag the whole system down.

4.3.3.3 Feature selection

Koller and Sahami [9] researched the number of features needed for HTC compared to FC. In their paper 'Hierarchically classifying documents using very few words' they showed that the classifiers that compose a hierarchy need less features than in FC for TC problems. Please note that they used a *Naive Bayes* and *Probabilistic* classifier. They showed that words that can distinguish higher levels in the hierarchy are different for the words that distinguish lower levels in the hierarchy. A classifier in a hierarchy only has to distinguish its category from it's siblings, as opposed to a flat classifier, which has to distinguish it's category from all other categories in the dataset. This is why the hierarchical classifiers probably need fewer features than the flat classifiers.

In their paper they compared a HTC approach versus a FC approach. The HTC approach applied a feature selection for each node in the hierarchy separately and the FC approach applied feature selection globally. This is different than our experimental setup, because we apply local feature selection per classifier, also in the FC setup.

However, we decided to experiment with the number of features selected in the HTC experiment. Because of the results from Koller and Sahami [9] we decided to test if we needed fewer features in the HTC approach than our FC approach.

Chapter 5

Results & discussion

In this chapter we will present the results of our experiments. We will first briefly discuss the performance measures we selected and some descriptive statistics of the generated decision tree (DT) models. We will then present the results of the first experiment on ‘the feature representation of news articles’ and we will then show the results of the second experiment on ‘exploiting the label structure’.

5.1 Performance measures

Recall that we use the *Micro-F1* measure as our main performance measure. For an explanation of performance measures refer to section 2.7 and for a justification on our choice of measurement refer to section 4.1.4. We also included the measures *Micro-Precision* and *Micro-Recall*, because the *Micro-F1* measure is based on those two measures. Aside from that we also included two special measures, *AtLeast1Good* and *NonePredictedAtAll*. Those measures are not normally found in the literature, but we included them to make the results more concrete. *AtLeast1Good* is the percentage of records where at least one correct prediction is made, *NonePredictedAtAll* is the percentage of records where no labels are predicted. Those measures are example-based measures, such as the *Hamming-Loss*, which we also included. We use this measure to compare our *label-based* F1 measure with a good *example-based* measure used in the *multi-label classification* (MC) literature. Note that the *Hamming-Loss* is a loss function, which value decreases when the performance increases.

5.2 Decision tree models

Aside from measuring the performance of machine learning (ML) algorithms, it is also interesting to analyze the generated DT models. For an introduction on DT, please refer to section 2.6. An example of a DT model generated for the 'Geothermal Energy' category is shown in figure 5.1. We deliberately selected a small model, so it would fit on paper. The *term frequency - inverse document frequency* (TF-IDF) scores of features are used to split the data. The tree will predict the membership to the category 'Geothermal Energy'.

In order to give some descriptive statistics about the DT generated, we calculated the number of *nodes* and the number of *distinct features* in the model. The number of *nodes* is calculated by counting all the nodes in the tree, including the leaf nodes. This measures the complexity of the model. The number of *distinct features* of the model indicates how many features that were selected are eventually used in the model. In the example model in figure 5.1, the model has 35 nodes with 15 distinct features. In this model there are two features that are used twice, namely 'energy' and 'pow' (pow remains after stemming the word 'power').

5.3 The feature representation of news articles

This experiment, as described in section 4.2, consists of three parts where in total six experiment variables are investigated. The factorial experiment looks at *word modeling*, *text selection*, *number of features* and *feature blacklist* in combination with each other. After that we will show results of the other two parts which look at the inclusion of the *source attribute* of the news article and the *attribute weighting* technique with the best results obtained from the first part. At the end we will combine the best settings from the three parts and evaluate the performance of that setting. This setting will be used as the basis of the next experiment, 'exploiting the label structure'.

5.3.1 The factorial experiment

5.3.1.1 Feature explosion

We performed the factorial experiment with some limitations, as we discovered that some settings would take too long to perform. Mainly because the number of features that was generated was too large. The number of documents in the dataset of course also contributes to the duration of the experiments. The most cpu-time for each experiment

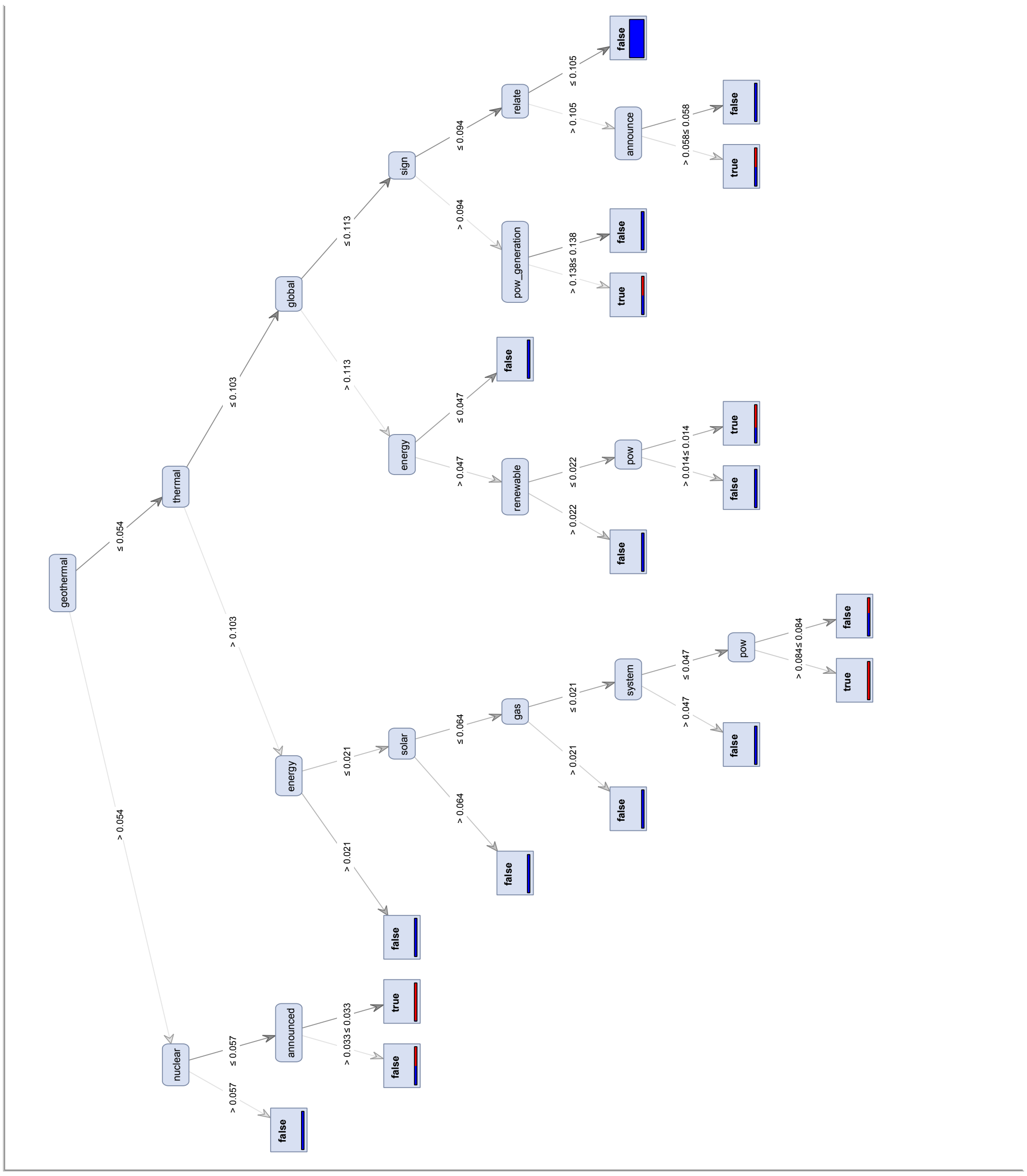
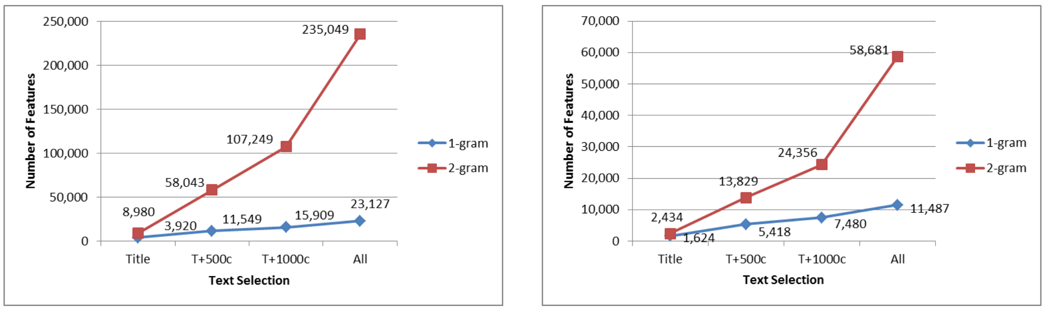


FIGURE 5.1: An example Decision Tree Model for the category 'Geothermal Energy'

is spent on calculating the *chi squared* value for each feature, in order to apply *feature selection*.

In general, an experiment on the 'ASD' dataset takes considerably longer than the same experiment on the 'GEW' dataset. This is mainly because of two reasons, first the number of documents is 4.3 times the number of documents in the 'GEW' dataset. Secondly, the number of features generated from these documents is two to four times the size of the number of features in the 'GEW' dataset, depending on whether 1-grams or up to 2-grams are used as features.

We found out that when using 2-grams and the whole text of the news article, a single experiment would take us seven days of CPU-time when using the 'ASD' dataset. Even when using only the first 1000 characters, the experiment would take two days of CPU-time when using the 'ASD' dataset. Therefore we decided to exclude the combination *TextSelection:All* and *WordModeling:2-gram* from the factorial experiment and the whole *TextSelection:T+1000c* setting from experiments on the 'ASD' dataset.



(A) The 'ASD' dataset (B) The 'GEW' dataset

FIGURE 5.2: The number of features generated with different Text Selection and Word Modeling settings for both datasets

The reason why those experiments take considerably longer is because of the so called *feature explosion*. When using up to 2-grams in combination with a large amount of text, the feature space begins to explode. To illustrate this, figures 5.2a and 5.2b show the number of features generated for each combination of *WordModeling* and *Text Selection* for the 'ASD' and 'GEW' dataset respectively. The number of features shown is after removing the very rare words (features that occur in less than four documents).

One would expect to see twice as many features when using up to 2-grams, because the number of 2-grams extracted from a sentence of distinct words is one less than the number of 1-grams that is extracted. (See section 2.5.1.1 for more information about feature generation) However, recall that in a certain domain D, only a finite number N_1 of words occur that are not very rare. The number of 2-grams N_2 that occur in D is much higher, because a lot of combinations are possible. Only a small fraction of the combinations will occur, but N_2 will have an upper limit equal to the number of combinations (with repetition) of two words from N_1, namely N_1^2.

5.3.1.2 All results

Table 5.2 and 5.3 show all the results of the factorial experiment with the 'ASD' and 'GEW' dataset respectively. The best performance value for each measure is shown in bold. We wanted to use the best experiment setup with the highest *Micro-F1* value as the best setting for further experiments. Unfortunately, both winning experiments use 2-grams and we found out that running the remaining experiments with 2-grams on the 'ASD' dataset would take too much time too perform. Luckily settings that use 2-grams for the 'GEW' dataset still run in decent time. We have to restrict ourselves to 1-grams for the 'ASD' dataset. Luckily the second best setting is a good candidate to select. Except for the WordModeling variable, it is identical to the best setting and furthermore also has only a slightly lower *Micro-F1* value. Because of this we selected the second-best setting from the 'ASD' dataset for further experiments and the best setting from the 'GEW' dataset. Table 5.1 shows the default, winning and selected settings for both datasets.

TABLE 5.1: The default,winning and selected settings for both datasets

	'ASD' dataset			'GEW' dataset		
	Default	**Winner**	**Selected**	**Default**	**Winner**	**Selected**
Rank	8th	1st	2nd	35th	1st	1st
Features	80	80	80	80	40	40
WordModeling	1-gram	2-gram	1-gram	1-gram	2-gram	2-gram
Blacklist	no	yes	yes	no	yes	yes
TextSelection	All	T+500c	T+500c	All	T+500c	T+500c
Micro-F1 value	.598	.605	.604	.704	.731	.731

Remarkable is that only the number of features is different for both the winning datasets. The *Micro-F1* values for experiments performed on the 'ASD' dataset are considerably lower than on the 'GEW' dataset with the same settings. Also remarkable is that the difference between the default and best setting in the 'GEW' dataset is much greater than in the 'ASD' dataset. This is because the TextSelection variable has a greater effect in this dataset.

Table 5.2: The results of the 'feature representation of news articles' experiment with the 'ASD' dataset

id	**Features**	**WordModeling**	**Blacklist**	**TextSelection**	**Micro-Precision**	**Micro-Recall**	**Micro-F1**	**AtLeast1Good**	**NonePredictedAtAll**	**HammingLoss**
1	80	2-gram	yes	T+500c	.625	.586	**.605**	.845	.069	.0314
2	80	1-gram	yes	T+500c	.620	.588	.604	.847	.064	.0317
3	80	2-gram	no	T+500c	.626	.580	.602	.840	.070	.0315
4	40	1-gram	no	All	.598	.600	.599	.853	.055	.0330
5	160	2-gram	yes	T+500c	.612	.586	.599	.849	.058	.0322
6	160	1-gram	yes	T+500c	.601	.596	.599	.856	.051	.0328
7	40	1-gram	yes	T+500c	.633	.567	.598	.829	.079	.0313
8	80	1-gram	no	All	.584	**.612**	.598	**.866**	.041	.0338
9	80	1-gram	no	T+500c	.619	.575	.596	.840	.069	.0320
10	160	2-gram	no	T+500c	.607	.585	.596	.845	.062	.0326
11	40	2-gram	no	T+500c	.640	.557	.595	.820	.085	.0311
12	40	1-gram	yes	All	.596	.594	.595	.851	.057	.0332
13	40	2-gram	yes	T+500c	.641	.555	.595	.820	.086	**.0310**
14	40	1-gram	no	T+500c	.631	.560	.594	.825	.081	.0315
15	80	1-gram	yes	All	.586	.602	.593	.863	.042	.0338
16	160	1-gram	no	T+500c	.598	.589	.593	.848	.057	.0332
17	160	1-gram	no	All	.583	.598	.590	.862	.041	.0341
18	160	1-gram	yes	All	.577	.595	.586	.859	**.040**	.0345
19	160	1-gram	no	Title	.655	.442	.528	.682	.197	.0325
20	160	2-gram	no	Title	.654	.441	.527	.684	.196	.0325
21	160	1-gram	yes	Title	.660	.431	.522	.671	.212	.0324
22	160	2-gram	yes	Title	.662	.426	.518	.667	.217	.0325
23	80	1-gram	no	Title	.674	.414	.513	.656	.231	.0323
24	80	1-gram	yes	Title	.675	.412	.512	.656	.231	.0323
25	80	2-gram	no	Title	.672	.411	.510	.651	.233	.0324
26	80	2-gram	yes	Title	.673	.408	.508	.648	.236	.0324
27	40	1-gram	no	Title	.673	.395	.498	.628	.255	.0327
28	40	1-gram	yes	Title	.675	.393	.497	.628	.256	.0327
29	40	2-gram	no	Title	**.676**	.385	.491	.615	.270	.0328
30	40	2-gram	yes	Title	.676	.383	.489	.612	.272	.0329

TABLE 5.3: The results of the 'feature representation of news articles' experiment with the 'GEW' dataset

id	**Features**	**WordModeling**	**Blacklist**	**TextSelection**	**Micro-Precision**	**Micro-Recall**	**Micro-F1**	**AtLeast1Good**	**NonePredictedAtAll**	**HammingLoss**
1	40	2-gram	yes	T+500c	.759	.705	**.731**	.807	.107	**.0284**
2	40	1-gram	yes	T+500c	.748	.713	.730	.813	.100	.0288
3	80	2-gram	yes	T+500c	.736	.722	.729	.821	.091	.0293
4	40	1-gram	no	T+500c	.739	.715	.727	.814	.100	.0294
5	80	1-gram	yes	T+500c	.731	.722	.727	.822	.090	.0297
6	80	2-gram	no	T+500c	.722	.727	.725	.826	.084	.0302
7	160	1-gram	yes	T+500c	.722	.725	.724	.824	.086	.0303
8	80	1-gram	no	T+500c	.721	.726	.723	.826	.085	.0304
9	160	1-gram	no	T+500c	.716	.730	.723	.828	.085	.0306
10	40	2-gram	no	T+500c	.743	.704	.723	.804	.106	.0295
11	40	2-gram	yes	T+1000c	.739	.706	.722	.804	.102	.0297
12	40	1-gram	yes	T+1000c	.732	.711	.722	.810	.094	.0300
13	160	2-gram	yes	T+500c	.712	.731	.721	.828	.077	.0309
14	80	2-gram	yes	T+1000c	.725	.715	.720	.813	.090	.0304
15	160	2-gram	no	T+500c	.705	.734	.719	**.833**	.076	.0313
16	40	1-gram	no	T+1000c	.728	.709	.718	.807	.095	.0304
17	80	1-gram	yes	T+1000c	.712	.724	.718	.826	.081	.0311
18	40	2-gram	no	T+1000c	.729	.706	.717	.801	.105	.0305
19	80	2-gram	no	T+1000c	.713	.720	.716	.821	.078	.0312
20	160	2-gram	yes	T+1000c	.705	.728	.716	.826	.076	.0315
21	80	1-gram	no	T+1000c	.706	.724	.715	.824	.076	.0315
22	80	2-gram	yes	All	.695	.732	.713	.831	.079	.0322
23	160	2-gram	yes	All	.693	.733	.713	.833	.074	.0323
24	160	1-gram	yes	T+1000c	.701	.724	.712	.822	.082	.0320
25	40	2-gram	yes	All	.706	.718	.712	.816	.097	.0317
26	40	2-gram	no	All	.702	.720	.711	.816	.090	.0320
27	160	2-gram	no	T+1000c	.689	.733	.711	.829	.072	.0327
28	40	1-gram	yes	All	.701	.717	.709	.811	.094	.0322
29	80	1-gram	yes	All	.681	**.736**	.708	.830	.073	.0332
30	80	2-gram	no	All	.688	.726	.707	.825	.079	.0329
31	40	1-gram	no	All	.697	.716	.707	.809	.092	.0325
32	160	1-gram	no	T+1000c	.686	.726	.706	.825	.074	.0331
33	160	2-gram	no	All	.683	.729	.705	.828	.078	.0333
34	160	1-gram	yes	All	.686	.725	.705	.826	.071	.0332
35	80	1-gram	no	All	.681	.730	.704	.826	.072	.0335
36	160	1-gram	no	All	.680	.730	.704	.830	**.070**	.0335
37	80	1-gram	yes	Title	.808	.607	.693	.697	.227	.0294
38	80	1-gram	no	Title	**.810**	.603	.691	.695	.228	.0294
39	80	2-gram	no	Title	.799	.608	.690	.699	.220	.0298
40	160	1-gram	no	Title	.801	.606	.690	.697	.220	.0297
41	80	2-gram	yes	Title	.795	.608	.689	.699	.219	.0300
42	40	1-gram	yes	Title	.810	.599	.689	.690	.234	.0296
43	160	1-gram	yes	Title	.805	.602	.689	.692	.226	.0297
44	40	1-gram	no	Title	.809	.598	.688	.690	.233	.0297
45	160	2-gram	yes	Title	.793	.607	.688	.697	.218	.0301
46	160	2-gram	no	Title	.792	.607	.688	.696	.217	.0302
47	40	2-gram	no	Title	.802	.597	.684	.686	.233	.0301
48	40	2-gram	yes	Title	.799	.598	.684	.686	.235	.0302

We will describe the results from tables 5.2 and 5.3, by looking at each experiment variable.

Number of features We can see that in the 'ASD' dataset the top three experiments all use 80 features. If we look at the *text selection* column we can see that when only the title is used, 160 features score the best. This is followed by using 80 features and at last using 40 features. This hints at an interaction between the *number of features* and *text selection* variable, which we will investigate further in the next section. When the title and the first paragraph is selected, it is not so clear what the best number of features is. However using 80 features seems to perform well in combination with this *text selection*.

Looking at the 'GEW' dataset, we see that things are a bit different. It seems that when using the title and the first paragraph using 40 or 80 features scores better than using 160 features. When using only the title, 80 or 160 features seems to score better.

Word modeling Looking at the 'ASD' dataset, we can't see which *word modeling* setting performs better. Note that the experiments with 2-grams and using the whole text are excluded from the results, so it may look like 1-gram is better represented in the top results. When looking at the 'GEW' dataset we also see 1-gram and 2-gram alternating in the top experiments.

Feature blacklist The *blacklist* variable has some impact on the Micro-F1 value, but it is not clear whether using or not using it leads to better results. However, for the 'ASD' and 'GEW' dataset using the blacklist appears in all of the top two and top three results respectively.

Text selection The fourth variable, *text selection*, clearly has a great impact on the results. In both datasets we see that using only the title of a news article at the bottom of the table and using the title plus the first paragraph in the top results. Looking closer we can see that using only the title does have a higher Micro-Precision than using in addition (a part of) the full-text. However it does have a lower Micro-Recall. This indicates that using only the title is very accurate but not good enough to classify all articles.

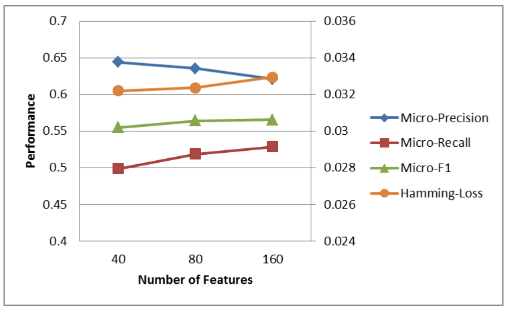
We will look closer at each of the experiment variables and their effect in the next sections. We will perform an analysis of variance (ANOVA) on each variable it's main effect on the *Micro-F1* performance value. In addition we will perform an ANOVA to detect possible interactions between variables. ANOVA is a widely-used hypothesis test that can compare the means of several groups. This makes it a more general test than the *t-test*, which is limited to only two groups. In this application a group is the collection of experiments that share the same value for the tested variable.

Basically an ANOVA tests whether a null hypotheses should be rejected or not. The null hypothesis is that there is no difference between the means of the groups. When it should be rejected, it means that the variable its effect is statistically significant. To decide this a $p-value$ is calculated, which indicates the probability that the same results were obtained assuming the null-hypothesis is true. If the $p - value$ is less than α, here chosen to be 0.05, it indicates that the null hypothesis should be rejected.

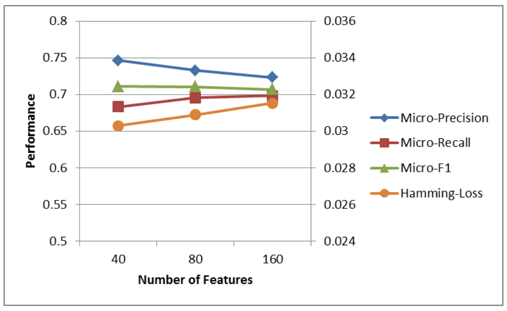
We calculated the ANOVA for the main effect and the interaction effect using the 'One-Way' and 'Two-Way' layout respectively as described by DeGroot and Schervish in [49].

5.3.1.3 The number of features

Figure 5.3a and 5.3b display a graph of the average results for the 'ASD' and 'GEW' dataset respectively. In tables 5.4a and 5.4b the average performances are displayed.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.3: The average performances of experiments with on the x-axis the number of features

In this and the remaining graphs the performance measures; *Micro-Recall*, *Micro-Precision* and *Micro-F1* are displayed on the left y-axis while the *Hamming-Loss* is displayed on the right y-axis. The explanatory variable is displayed on the x-axis.

It is remarkable to see that the *Hamming-loss* is affected by the number of features. It seems that the lower the number of features, the lower the *Hamming-loss* for both datasets. However our main criterion is *Micro-F1*, which seems to do the opposite for the 'ASD' dataset and increases when the number of features increases. However for the 'GEW' dataset, the *Micro-F1* value is the same when using 40 or 80 features and decreases when using 160 features.

Table 5.4: The average measures for the *Number of Features* variable

	Number of Features		
Measure	**40**	**80**	**160**
Micro-Precision	**.644**	.635	.621
Micro-Recall	.499	.519	**.529**
Micro-F1	.555	.564	**.566**
AtLeast1Good	.748	.771	**.782**
NonePredictedAtAll	.150	.129	**.113**
Hamming-Loss	**.0322**	.0324	.0329

(a) The 'ASD' dataset

	Number of Features		
Measure	**40**	**80**	**160**
Micro-Precision	**.747**	.733	.723
Micro-Recall	.683	.696	**.698**
Micro-F1	**.711**	.711	.707
AtLeast1Good	.779	.793	**.795**
NonePredictedAtAll	.132	.117	**.113**
Hamming-Loss	**.0303**	.0309	.0315

(b) The 'GEW' dataset

Interesting to observe in both datasets is the effect of the number of features on *Micro-Precision* and *Micro-Recall.* When the number of features increases, the *Micro-Precision* decreases and the *Micro-Recall* increases.

We performed an ANOVA on the main effect of the number of features variable for the *Micro-F1* value and obtained a p value of 0.85 and 0.70 for the 'ASD' and 'GEW' dataset respectively. This indicates that the main effect is not statistically significant.

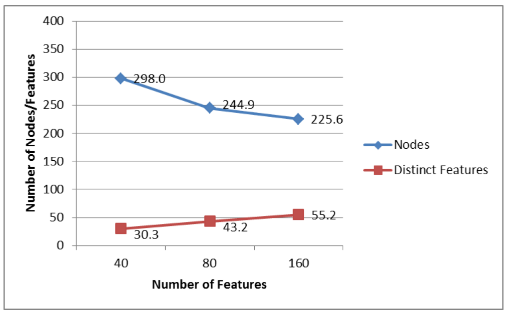
Discussion

Recall that the total feature selection process consists of first filtering very rare terms. Afterward, the top x features with the highest *Chi Squared* value are selected to feed to the DT learning algorithm. Finding out the best x value for both datasets is the purpose of including this variable. The DT itself has it's own selection procedure, which is why the number of features that end up in the generated DT can be less than the features fed to the learning algorithm. More background information on feature selection can be found in section 2.5.1.3 and more details on the experimental setup and justifications can be found in section 4.2.2.1.

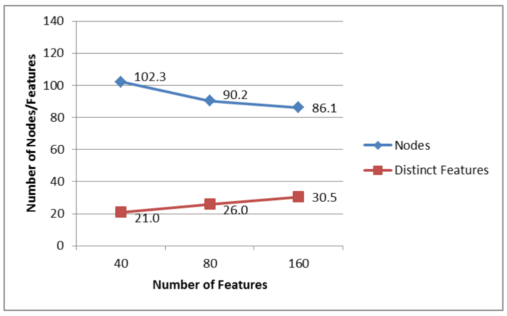
It seems that the number of features has a positive effect on the *Micro-Recall* and a negative effect on the *Micro-Precision.* In practice increasing the *Micro-Recall* can often decrease the *Micro-Precision.* In addition it seems that the best performing number of features is different in both datasets.

A reason why *Micro-Recall* increases and *Micro-Precision* decreases could be because the generated DT uses not necessarily more, but different features, which in combination with each other can possibly cover a wider range of the positive examples.

Suppose we want to learn a classifier which predicts whether a text is describing the sport *basketball*. Due to feature selection, the 2-gram 'slam dunk' is excluded from the selected features. Suppose there are some examples which can only be identified correctly by this term, then by including it we are able to correctly predict more positive examples, and thus increasing the *Micro-Recall*.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.4: The average number of nodes and distinct features in the generated decision trees with on the x-axis the number of features

To take a closer look at the DTs generated, figure 5.4 shows the number of nodes and the number of distinct features in the generated DTs. In both datasets, the number of distinct features used increases when more features are selected, which is as expected. However the number of nodes in the tree decreases when more features are used. It could be the case that with more features available, less splits are needed, because the extra features produce better splits of the data.

The reason why the best performing number of features for the 'GEW' dataset is lower than the 'ASD' dataset is probably because the size of the 'GEW' dataset is 4.3 times smaller than the 'ASD' dataset.

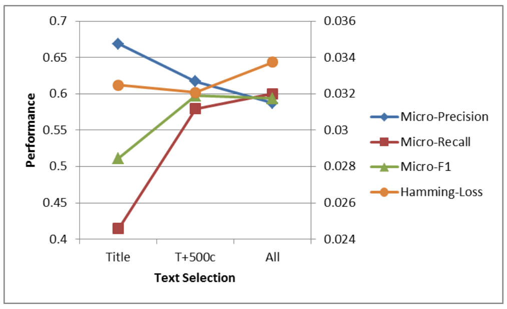
The optimal number of features is known to vary depending on many things, such as which learning algorithm is used, the size of the dataset, the way the features are generated etc. It is hard to say something in general about the number of features, but Yang and Pederson showed that with *Chi Squared* feature selection one can reduce the term space by a factor of 100 without loss of effectiveness [5]. According to Sebastiani when using local feature selection, typical number of features are between 10 and 50 features [1].

Because we only tested 40, 80 or 160 features, it could be entirely possible that the real optimal number of features for each dataset is quite different. Especially for the 'GEW' dataset, for which the best performing number of features is 40, it could be the case that using 10 features even scores better.

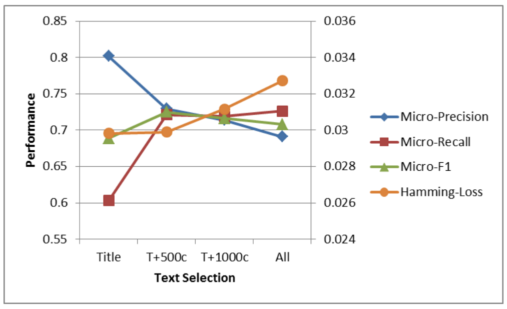
For future research it would be interesting to study a wider range of numbers, to select from the *Chi Squared* ranked features.

5.3.1.4 Text selection

As discussed before, the text selection variable has the most effect on the performance from all the variables in this experiment. Figures 5.5a and 5.5b show the performances and table 5.5a and 5.5b show the average measures for the text selection variable for the 'ASD' and 'GEW' dataset respectively.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.5: The average performances of experiments with on the x-axis the text selection method

We can see that using the title and the first paragraph (T+500c) results in the highest *Micro-F1* values for both datasets. The *Hamming-Loss* seems to indicate the same effect. Except for the 'GEW' dataset, where the *Hamming-Loss* of using only the title is just a little higher than using the title and the first paragraph. Using the title gives a considerably lower *Micro-F1* value than using in addition more full-text in both datasets.

Interesting to observe are the *Micro-Precision* and *Micro-Recall* values. When using only the title, the *Micro-Precision* is a lot higher than the *Micro-Recall*, but when more parts of the full-text are used the *Micro-Precision* decreases and the *Micro-Recall* increases.

Remarkable is that the improvement in *Micro-F1* performance when using the title and the first paragraph as opposed to the title and all the full-text, is much greater in the 'GEW' dataset than in the 'ASD' dataset. In the 'ASD' dataset there is an improvement of 0.003 while there is an improvement of 0.017 in the 'GEW' dataset.

We calculated the ANOVA for the main effect of the text selection variable and obtained p values of 9.240×10^{-12} and 2.218×10^{-26} for the 'ASD' and 'GEW' dataset respectively. This means that the main effect of the text selection variable is statistically significant.

Table 5.5: The averages measures for the Text Selection variable (The experiments with WordModeling:2-gram are excluded from the calculation for the 'ASD' dataset.)

	Text Selection		
Measure	**Title**	**T+500c**	**All**
Micro-Precision	**.669**	.617	.587
Micro-Recall	.415	.579	**.600**
Micro-F1	.511	**.597**	.594
AtLeast1Good	.653	.841	**.859**
NonePredictedAtAll	.230	.067	**.046**
Hamming-Loss	.0325	**.0321**	.0337

(a) The 'ASD' dataset

	Text Selection			
Measure	**Title**	**T+500c**	**T+1000c**	**All**
Micro-Precision	**.802**	.729	.714	.691
Micro-Recall	.603	.721	.719	**.726**
Micro-F1	.689	**.725**	.716	.708
AtLeast1Good	.694	.820	.817	**.823**
NonePredictedAtAll	.226	.091	.085	**.081**
Hamming-Loss	**.0298**	.0299	.0312	.0327

(b) The 'GEW' dataset

Discussion

The purpose of this investigation is to find out which parts of the text are the best to be used for classification. Details on the experimental setup and justifications can be found in section 4.2.2.4.

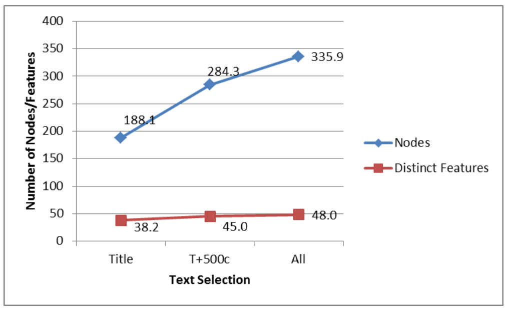
One of the key results is that using the title in combination with the first 500 characters of the text gives the best performance. This is also what we expected, because domain experts have indicating the title and the first paragraph to be the most descriptive. The experts indicated they look first at the title and when they are still not sure, they look further at the full-text. First they read the first paragraph, which usually is enough to classify the article but when they are still not sure they look at the full-text. Why the learning algorithm benefits from excluding a part of the full-text is possibly because the extra full-text will be misleading. It might contain terms that indicate another category, but are not the main subject(s) of the news article. By removing the noise we prevent the algorithm from modeling this noise. We also discovered that in a fraction of the news articles, the headlines were present in the first part of the full-text. Because these headlines often summarize the article and contain important keywords this can also be an explanation why the title and first paragraph performs well.

Nomoto and Matsumoto [32] studied the effect of selecting parts of the text for news articles of different length. They use a different model and different datasets, so it's hard to compare the results. However by testing various different approaches in discarding rear text, such as keeping the first i words from the text. Most of them resulted in better performance than using the whole full-text.

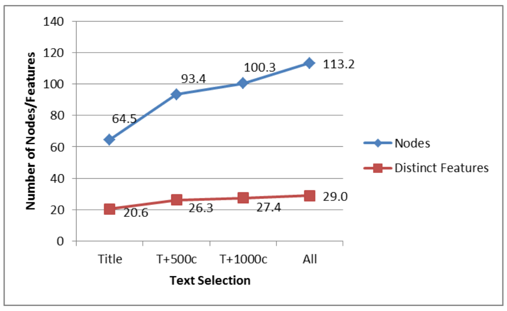
Why the text selection has a greater effect on the ‘GEW’ dataset than the ‘ASD’ dataset remains speculation. It could be due the fact that the ‘ASD’ dataset is more multi-label than the ‘GEW’ dataset. Therefore ‘GEW’ could benefit more from using fewer text, as it has less categories to deduce from the text.

Another result is that the *Micro-Recall* seems to increase when more parts of the full-text is used, at the cost of the *Micro-Precision*. By using the title we can see a very high *Micro-Precision* and low *Micro-Recall*. This could mean that the title can very accurately predict a fraction of the positive examples, which is not enough to be a good classifier. By using (a part of) the article-text we can increase the *Micro-Recall* and cover a larger fraction of the positive examples, but unfortunately not so precise as the fraction we could classify by using only the title. Therefore the *Micro-Precision* decreases and the *Micro-Recall* increases. This is exactly what domain experts have indicated.

We included figure 5.6 to show the influence of the *text selection* variable on the generated DTs. Looking at only the title and using (a part of) the full-text we see a big difference in number of nodes and number of distinct features used. Using (a part of) the full-text in addition with the title causes a more complex DT to be generated, with more distinct features. This also explains the behavior of the *Micro-Recall* and *Micro-Precision* values. Given two DTs with the same performance, then usually the most simple tree of the two is preferred, analogous to Occam’s Razor. The performance of using the title and first paragraph and the performance of the title and all the full-text are close in the ‘ASD’ dataset. However we can see that the complexity of the generated decision trees is higher when using all the text. That is another reason why the title and first paragraph text selection should be preferred.



(A) The ‘ASD’ dataset



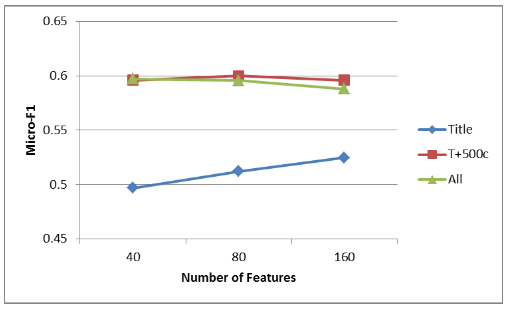
(B) The ‘GEW’ dataset

FIGURE 5.6: The number of nodes and distinct features in the generated decision tree models for both datasets

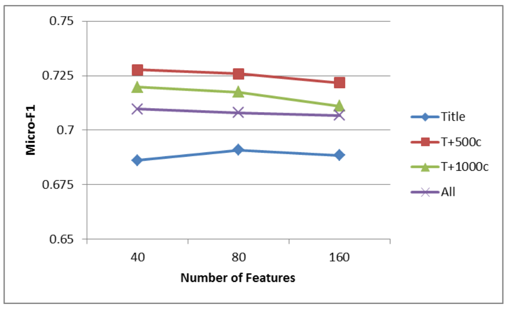
A clear drawback of this research is that we could not identify paragraphs from the text, and thus used complete words that occur in the first 500 characters to represent a paragraph. We also found that some articles contain headlines in the beginning of the text. It would be a good idea to structure the data, so that we could further exploit the headlines or paragraphs.

5.3.1.5 Interaction between number of features and text selection

We studied all the interactions between the variables of this factorial experiment and found out an interaction between the number of features and the text selection. In figures 5.7a and 5.7b we can see how the two variables affect each other. For the 'ASD' set we excluded experiments that use 2-grams, because the combination of 2-grams and all the text is excluded from the factorial experiment.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.7: The interaction between the number of features and the text selection method. The performance measure used is the *Micro-F1 value*.

When you look closely at the graph in figure 5.7a you can see two types of lines. One line keeps increasing when the number of features increases, which is when only the title is used. All other lines follow the same pattern, they change only a little bit when going from 40 to 80 features and then decrease when going from 80 to 160 features.

For the 'GEW' dataset in figure 5.7b the line that represents using only the title, increases when going from 40 to 80 features and decreases when going from 80 to 160 features. In contrast the other lines keep decreasing while more features are used.

If we generalize both datasets, we can say that using only the title has a higher best number of features than when using the title and (a part of) the full-text.

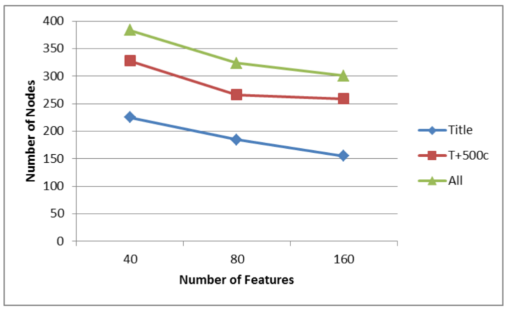
We performed an ANOVA on the interaction effect and obtained a p value of 3.189×10^{-4} and 0.014 for the 'ASD' and 'GEW' dataset respectively. This means the interaction in both datasets is statistically significant. The interaction seems to be more present in the 'ASD' dataset.

Discussion

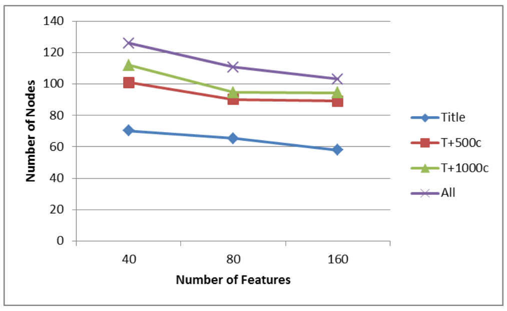
One of the key results is that we showed there is an interaction between the number of features and the amount of text that is selected. It also seems that when using only the title, a higher optimal number of features is needed than when using more full-text.

What we would expect is that less features are needed when using only the title, because the total number of features available is much less than when using (a part of) the full-text. Figure 5.2 shows the total number of features for each *text selection* method. However it seems the results are the opposite. One reason for this could be that the words in the title are far more descriptive than words in the full-text. Because of this, the generated DT is less likely to begin modeling noise than DTs generated with (a part of) the full-text, when the number of features increases.

Another explanation or perhaps in conjunction with the previous one is the fact that the generated DT when using only the title is much simpler than when using (a part of) the full-text. This could also be a reason why it can handle more features, because it will not come overly complex. To support this, we will show the effect of the interaction on both the number of nodes and on the number of distinct features of the generated DT models. The effect on the number of nodes is shown in figure 5.8 and the effect on the number of distinct features is showed in figure 5.9.



(A) The 'ASD' dataset

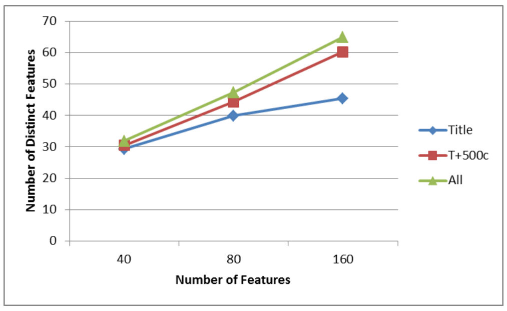


(B) The 'GEW' dataset

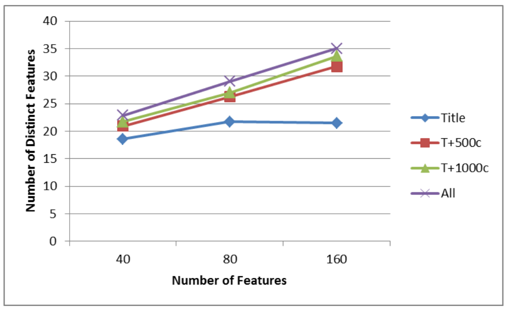
Figure 5.8: The interaction between the number of features and the text selection method on the number of nodes in the decision tree model

We can see that for the models that use only the title, the number of nodes (complexity) keeps decreasing at 160 features, whereas the other models decrease only a little bit. Also the distinct number of features is much less increasing as with the models that use (a part of) the full-text. The reason for this could be just because the model also has less nodes, so less room for new features.

As already noted before the different number of features that were tested on is quite small and the real optimal number of features could be very different. It could be the case that the real optimal number of features for when using only the title is very small, lets say 10, but then has a decreasing interval between 10 and 40 features, and



(A) The 'ASD' dataset



(B) The 'GEW' dataset

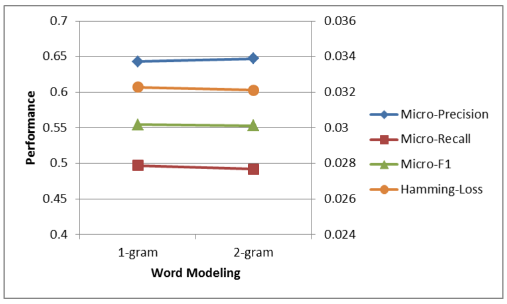
FIGURE 5.9: The interaction between the number of features and the text selection method on the distinct number of features in the decision tree model

then increases again when more features are used. So we recommend to investigate the number of features experiment with a wider range of possible numbers.

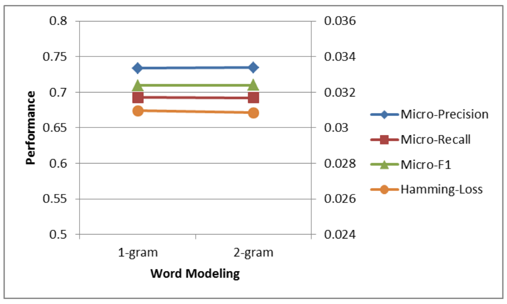
It would be interesting to find out the real optimal number of features for when using only the title. It looks like the performance is still increasing when more than 160 features are used.

5.3.1.6 Word modeling

Figures 5.10a and 5.10b display a graph of the average performance for 1-gram and up to 2-gram *word modeling*. For the 'ASD' dataset experiments using all the text have been excluded when calculating the average because we excluded the combination of using up to 2-grams and all the text from the factorial experiment.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.10: The average performances of experiments with on the x-axis the word modeling method

We can see there is not a big improvement in performance in any measure, when using up to 2-grams over 1-grams. Especially in the 'GEW' dataset the performances are almost identical. In the 'ASD' dataset we see the *Micro-Precision* slightly increasing when using 2-grams at a cost of *Micro-Recall*. Table 5.6a and 5.6b show the performance values.

Table 5.6: The averages measures per Word Modeling variable (*The experiments with TextSelection:All are excluded from the calculation for the 'ASD' dataset.)

	Word Modeling	
Measure	**1-gram**	**2-gram**
Micro-Precision	.643	**.647**
Micro-Recall	**.497**	.492
Micro-F1	**.554**	.553
AtLeast1Good	**.747**	.741
NonePredictedAtAll	**.149**	.155
Hamming-Loss	.0323	**.0321**

(a) The 'ASD' dataset

	Word Modeling	
Measure	**1-gram**	**2-gram**
Micro-Precision	.734	**.734**
Micro-Recall	**.692**	.692
Micro-F1	.709	**.710**
AtLeast1Good	**.789**	.789
NonePredictedAtAll	**.120**	.121
Hamming-Loss	.0310	**.0309**

(b) The 'GEW' dataset

We performed an ANOVA on the main effect and obtained *p* values of 0.941 and 0.902 for the 'ASD' and 'GEW' dataset respectively. This means that the main effect is not statistically significant.

Discussion

The key result from our investigation is that including 2-grams in the feature space does not improve the classification performance. However it does drastically increase the computational complexity, especially when a large amount of text is used. This is discussed in section 5.3.1.1 on 'Feature Explosion'.

The reason why using up to 2-grams is not really beneficial is probably because when the 1-grams, where the 2-gram is made of are also available, the 2-gram has almost nothing to contribute. There is only a small difference between a model that uses two 1-grams and a model that uses one 2-gram that consists of those 1-grams. The difference is that the latter has the constraint that the terms have to be in that specific order, which might be more accurate sometimes. But in other cases it might be to restrictive.

To investigate the 2-grams that end up in the selected features and eventually the generated DT, we performed some analysis. Figure 5.11 shows the ratio of 1-grams and 2-grams in the generated DTs. The number of 2-grams is not that high but we think that it is still high enough to make the difference.

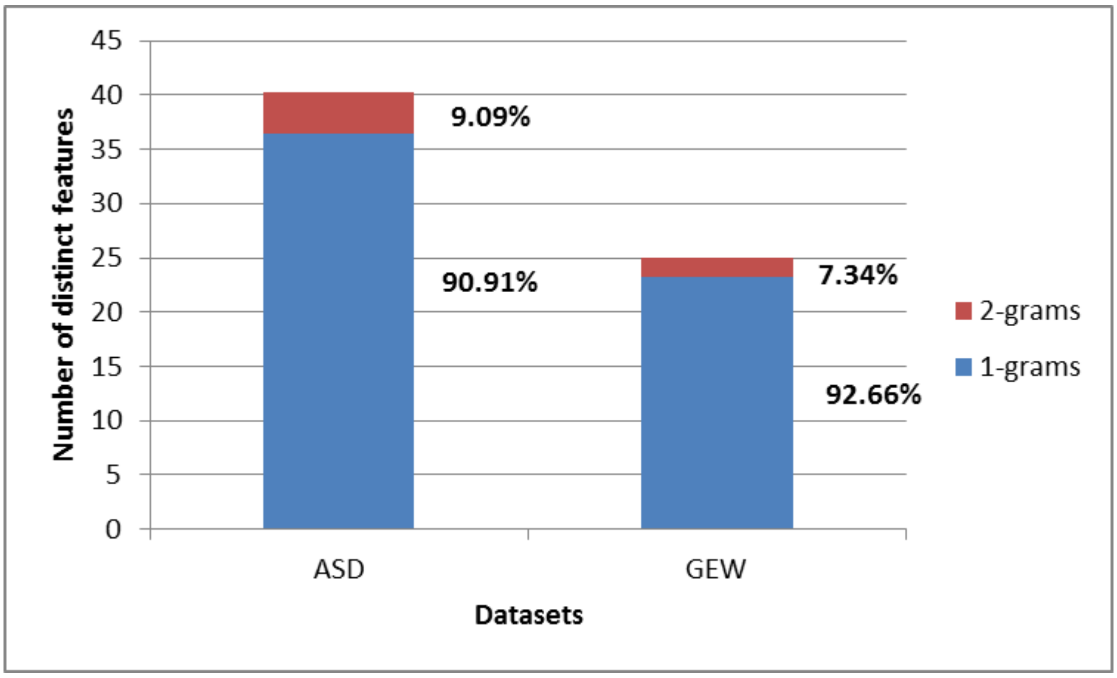


FIGURE 5.11: The number of distinct 1-grams and distinct 2-grams in generated decision trees using 2-grams

To further investigate we analyzed the features that were selected by *Chi Squared* feature selection in various experiments that use 2-grams. We analyzed the features and classified three types of 2-grams, based on whether the 1-grams that make up this 2-gram also are being selected. Take for example the 2-gram 'explosive device', it's 1-gram components are 'explosive' and 'device'. We looked at all features that were selected in various experiments and determined for each 2-gram whether it's 1-gram components were also selected. We identified three classes of 2-grams; *2-gram_0*, *2-gram_1* and *2-gram_2*, where the x in *2-gram_x* stands for how many of it's 1-gram components are also selected by *Chi Squared* feature selection. So if for example 'explosive device' is selected and 'device' is also selected, but 'explosive' not, than we would classify 'explosive device' as a *2-gram_1*.

Figure **??** shows the average ratio's of this type of 2-grams and 1-grams that end up being selected either when selecting 40, 80 or 160 features in different experiments.

We can see from the figure that from the 2-grams that are being selected only a very small number has no 1-gram components that could be selected to replace it. This means that for almost every 2-gram an equivalent model can be build that uses it's 1-gram components instead. The only advantage the 2-gram now has, is that the words have to occur adjacent to each other in the order in which they are specified by the 2-gram. When using it's component 1-grams instead, the 1-grams can occur in any order,

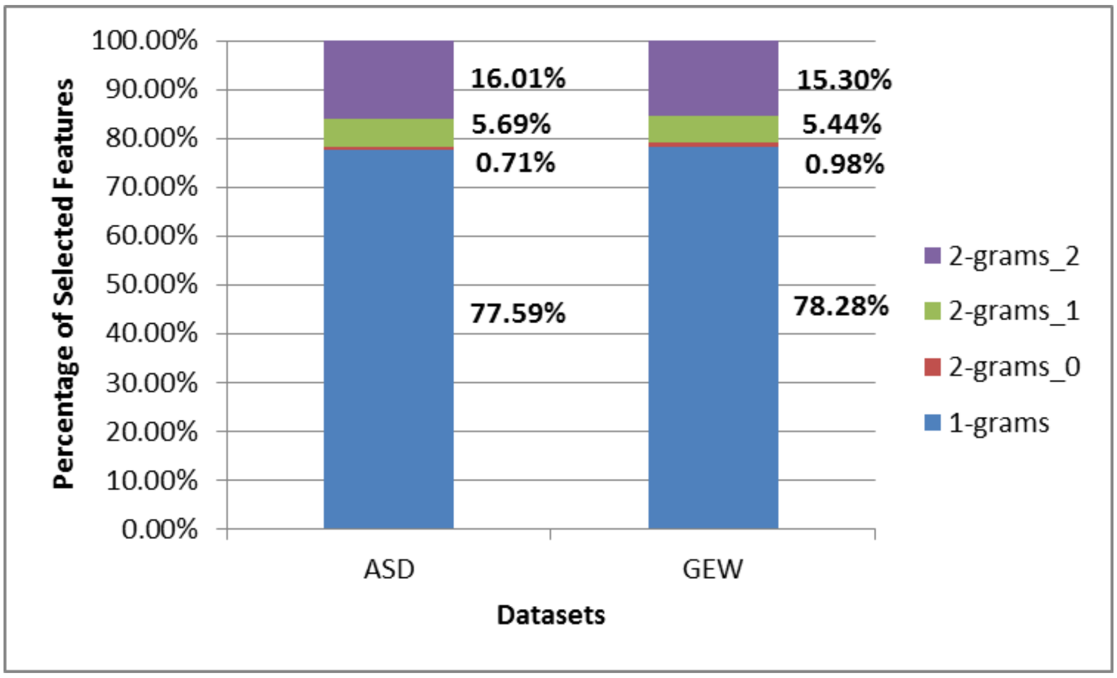


FIGURE 5.12: The number of 1-grams and different types of 2-grams in the selected features

and also anywhere in the text. A nice property of using 2-grams is that the size of the DT is a little bit smaller and easier to interpret.

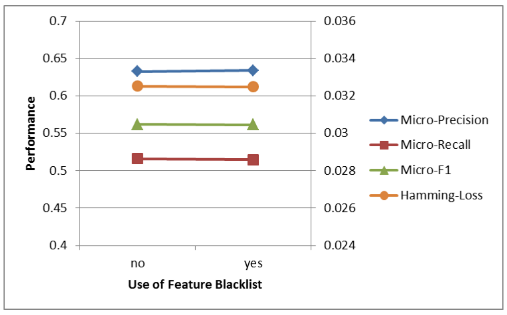
Bekkerman [50] supports our finding and argues that the number of highly descriptive 2-grams that can improve the performance is low compared to the 'junk' 2-grams that are found. Especially compared to the number of 1-grams. Other research have found mixed results [1, 4, 29, 30].

5.3.1.7 Custom blacklist

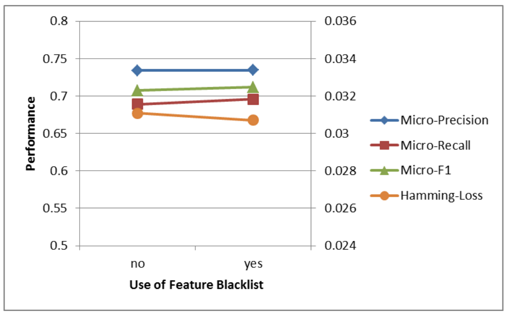
The *blacklist* variable does not have much effect on the performance. Figure 5.17a and 5.17b show the averages result of using and not using the blacklist. We can see a slight improvement of almost all performance measures when using the blacklist for the 'GEW' dataset.

The average values of the performances are displayed in table 5.7a and 5.7b for the 'ASD' and 'GEW' dataset respectively.

We performed an ANOVA on the main effect of using the blacklist and obtained a p value of 0.971 and 0.485 for the 'ASD' and 'GEW' dataset respectively. This indicates that the main effect is not statistically significant.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.13: The average performances of experiments with on the x-axis the usage of the blacklist

TABLE 5.7: The average measures for either using or not using the blacklist

	Use Blacklist	
Measure	**no**	**yes**
Micro-Precision	.633	**.634**
Micro-Recall	**.516**	.515
Micro-F1	**.562**	.561
AtLeast1Good	**.768**	.767
NonePredictedAtAll	**.129**	.131
Hamming-Loss	.0325	**.0325**

(A) The 'ASD' dataset

	Use Blacklist	
Measure	**no**	**yes**
Micro-Precision	.734	**.734**
Micro-Recall	.689	**.696**
Micro-F1	.707	**.712**
AtLeast1Good	.785	**.793**
NonePredictedAtAll	.124	**.117**
Hamming-Loss	.0311	**.0307**

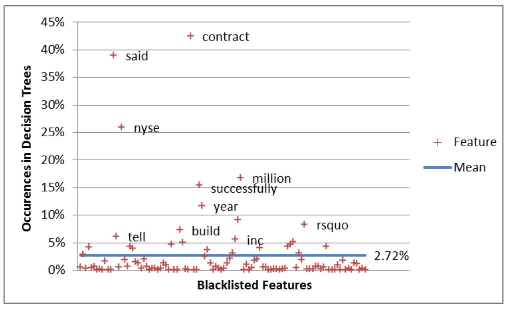
(B) The 'GEW' dataset

Discussion

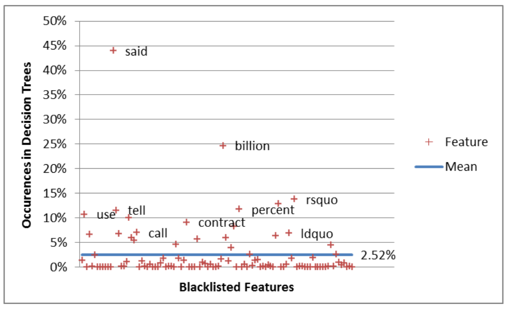
The key result of our finding is that the use of a feature blacklist can have a small improvement on the performance, depending on the dataset. The improvement is mainly in *Micro-Recall*.

The idea behind this blacklist, is that it filters out features that would model noise. We want to know whether the blacklisted features also get selected to split on in the DT, or only by the *Chi Squared* feature selection. In other words, do the blacklisted features end up in the DT if we wouldn't filter them? If the DT would filter them out, then we wouldn't see any improvement in performance. We investigated how many blacklisted features occur in the DTs generated which didn't use the blacklist. In figure 5.14 you can see in how many DT models the blacklisted features end up when not using the blacklist.

We can see that although many blacklisted features do not end up in the model or only in a small percentage of models, there are a few features that occur in a large part of the models. A blacklisted feature occurs on average in 2.72% and 2.52% of the generated DTs in the 'ASD' and 'GEW' dataset respectively. You can see that the term 'said' occurs in 40% to 45% of the models, depending on the dataset. Domain experts indicated that it could be due to the source 'AFP' of which many news comes from, that



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.14: The occurrences of blacklisted features in models when no blacklist is used

often uses this style of writing text. There are 1,525 (5,65%) articles from 'AFP' in the 'ASD' dataset and 2585 (41,29%) articles from 'AFP' in the 'GEW' dataset. Also the 'rsquo' and 'ldquo' would come from this source, which is an incorrectly stored opening and ending quote. It's quite amazing how these words end up with such a high *Chi Squared* score, but apparently they are a good predictor for certain categories. By using the blacklist we can prevent these features to end up in the model, but the question is whether it helps the classification. It would be a better idea to clean the data source, so that those encoding artifacts won't show up.

We have seen that blacklisting features can make a difference, as some of them occur in quite some models and a feature occurs on average in around 2.5% of the models. However perhaps there is no significant difference because the other features that get selected instead, have around the same predictive power. So this means, the features do have predictive power but not as high that we can see the performance decrease when ignoring the features.

5.3.2 Title and full-text weighting

In this experiment we looked at the effect of applying weights to text originating from the title and from the full-text. The full-text is all the text from a news article, except from the title. Recall that this experiment is done using the selected settings from the factorial experiment as explained in section 5.3.1.2.

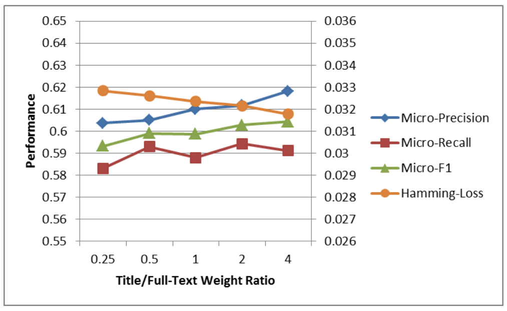
Due to a problem with our chosen data-mining software, we had to use a different stemming algorithm than in the other experiments, in order for the attribute weighting to work correctly. We normally use the *Wordnet* [48] stemming algorithm but for this experiment we had to use the *Snowball* stemming algorithm, as described in [51]. It seems that the *Snowball* stemming algorithm results in poorer performance. This is why the results of having equals weights can differ from the results obtained from the selected setting from the factorial experiment. For the 'ASD' dataset there is a slightly decrease in performance, a drop of 0.005 in *Micro-F1*. For the 'GEW' dataset there is not a significant difference.

Figures 5.15a and 5.15b show the performances of the classifier using different *title/full-text weight ratios* for the 'ASD' and 'GEW' dataset respectively. We can see that this variable has quite some impact on the performance measures.

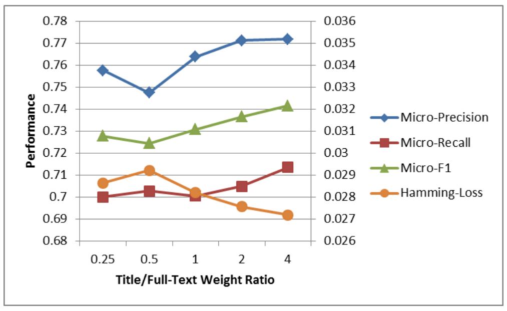
In both datasets a *title/full-text weight ratios* of four, results in the best *Micro-F1* performance. The *Hamming-Loss* indicates the same result. Interesting is that using a *title/full-text weight ratio* of 0.5 increases the *Micro-Recall* in both datasets. The *Micro-Recall* drops again when using a ratio of 0.25.

The 'GEW' dataset seems to benefit more from this variable than the 'ASD' dataset. The average performances are displayed in table 5.8a and 5.8b for the 'ASD' and 'GEW' dataset respectively.

Because the *title/full-text weight ratios* of four gives the highest *Micro-F1* value in both datasets, we will use this ratio in experiment II.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.15: The average performances of experiments with on the x-axis the title/full-text weighting ratios

5.3.2.1 Discussion

This experiment has some similarities with the *Text Selection* variable in the factorial experiment. There we looked at which part of the full-text we need to include, and if we even need to include it aside from the title. Here we assign weights to the different parts of the news article, in this case the title and the words occurring in the first 500 characters of the full-text.

Recall that to apply the weights, the calculation of the *TF-IDF* score is adjusted. When counting the term occurrences, the weight associated with the attribute where the term originates from is first applied.

Table 5.8: The averages measures per title/full-text weight ratio

	Title/Full-text weight ratio				
Measure	**0.25**	**0.5**	**1**	**2**	**4**
Micro-Precision	.604	.605	.610	.612	**.618**
Micro-Recall	.583	.593	.588	**.594**	.591
Micro-F1	.593	.599	.599	.603	**.604**
AtLeast1Good	.847	.854	.850	**.855**	.854
NonePredictedAtAll	.060	**.055**	.060	.056	.060
Hamming-Loss	.0328	.0326	.0323	.0321	**.0318**

(a) The 'ASD' dataset

	Title/Full-text weight ratio				
Measure	**0.25**	**0.5**	**1**	**2**	**4**
Micro-Precision	.758	.747	.764	.771	**.772**
Micro-Recall	.700	.703	.701	.705	**.714**
Micro-F1	.728	.724	.731	.737	**.742**
AtLeast1Good	.799	.803	.802	.808	**.814**
NonePredictedAtAll	.115	**.109**	.114	.112	.111
Hamming-Loss	.0286	.0292	.0282	.0276	**.0272**

(b) The 'GEW' dataset

The key result is that having a *title/full-text weight ratios* of four gives the best performance in both datasets. This supports our idea that the title is more important or has more descriptive words than the full-text.

We know from the *Text Selection* experiment that when training a classifier which uses only the title, it has a very high *Micro-Precision* but low *Micro-Recall*. Meaning that it can accurately classify a part of the positive examples.

The reason why a higher weight on the title than the full-text improves the performance can be because of the following. Consider news articles which have terms in the title that are discriminating for one or more categories, in the case of a higher weight on the title, they will probably be more likely to be classified with those categories than otherwise. On the other hand, news articles that contain discriminating terms for one or more categories that occur in the full-text and not in the title will be probably less likely to be classified with those categories. Apparently this improves the performance, considering the fact that the title can be very precise.

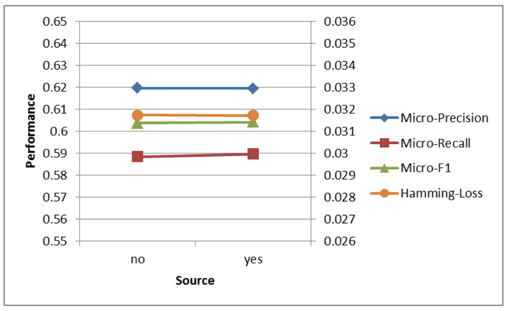
Because the *title/full-text weight ratios* of four is the highest value we experimented with, and this value also gives the best performance, it immediately shows the limitations of our experiment. It would be very interesting to see the results of further increasing this value above four.

5.3.3 Including the source attribute

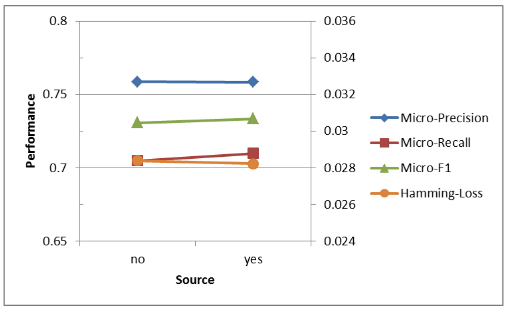
This experiment tested the effect of including the source of a news article as nominal attribute available for the learning algorithm. Please note that the 'ASD' dataset has 17% missing values for the *source* attribute and the 'GEW' dataset has only a couple (0%) missing values. The DT learning algorithm handles these missing values by considering all missing sources as being one source.

Figures 5.16a and 5.16b show the performances of the classifier either including or excluding the source attribute for the 'ASD' and 'GEW' dataset respectively. The corresponding performance values are displayed in table 5.9a and 5.9b for the 'ASD' and 'GEW' dataset respectively.

We can see that for the 'ASD' dataset, including the source attribute has almost no impact on the performance results. We observe a small change in *Micro-Precision* and *Micro-Recall* but an almost identical *Micro-F1* value. For the 'GEW' dataset the results are slightly different, here a small improvement on the *Micro-F1* value can be seen. Which is because of a small increase in *Micro-Recall* and a smaller decrease in *Micro-Precision*. However the improvement is very small.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.16: The performances of experiments with on the x-axis the inclusion of the source attribute

TABLE 5.9: The performances of either including or excluding the source attribute

	Include source	
Measure	**no**	**yes**
Micro-Precision	**.620**	.619
Micro-Recall	.588	**.590**
Micro-F1	.604	**.604**
AtLeast1Good	.847	**.848**
NonePredictedAtAll	.064	**.063**
Hamming-Loss	.0317	**.0317**

(A) The 'ASD' dataset

	Include source	
Measure	**no**	**yes**
Micro-Precision	**.759**	.758
Micro-Recall	.705	**.710**
Micro-F1	.731	**.733**
AtLeast1Good	.807	**.812**
NonePredictedAtAll	.107	**.102**
Hamming-Loss	.0284	**.0282**

(B) The 'GEW' dataset

5.3.3.1 Discussion

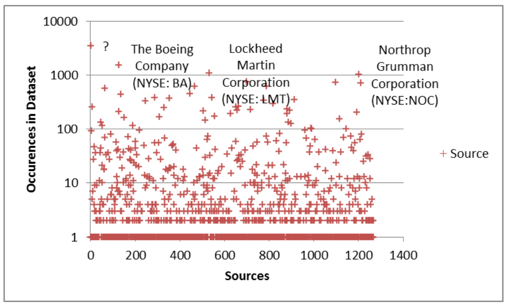
In this experiment we have observed that including the source attribute has no effect on the *Micro-F1* value for the 'ASD' dataset and gives a very small improvement of the *Micro-F1* value for the 'GEW' dataset. Domain experts have indicated that the source attribute could be very helpful in classification, as a source only publishes in a subset of the categories available.

Some more insight in the source attribute
To get some more insight into the source attribute, let us repeat some statistics from chapter 3 in table 5.10. The source attribute has many different values, each value only occurring in very few examples. In the 'GEW' dataset there is something peculiar, the single source 'AFP' contributes around 41% of the news from the 'GEW' dataset, which is a tremendous number of articles. On average a source occurs 17 times in the 'ASD' dataset and 5 times in the 'GEW' dataset. The 'ASD' dataset has 1,276 different sources and the 'GEW' dataset 1,196.

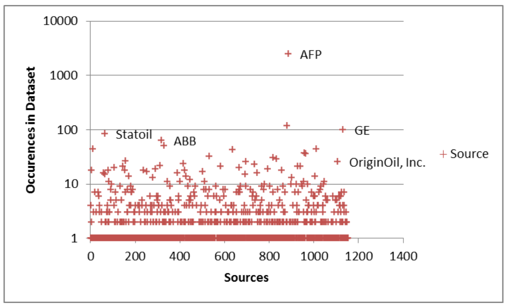
Table 5.10: Some statistics of the source attribute in both datasets

Dataset	**Missings**		**MAX**	**AVG**	**Distinct**	**Description**
ASD	4,481	(17%)	1,525	17	1,276	The news source
GEW	14	(0%)	2,585	5	1,196	The news source

We created a scatter plot of the number of news articles a source has contributed per dataset, which is shown in figure 5.17. Due to the logarithmic scale it may look like the number of articles the sources contribute are closer to each other than it would look like on a normal scale. However, only around 5% of the sources in the 'GEW' dataset occur in more than ten examples. Table 5.11 and 5.12 show data supporting the graph and some new measures, for the 'ASD' and 'GEW' dataset respectively. The column 'Examples' show different partitions of the data, where the partition is based on the number of examples in the dataset where the source occurs. For example, the most sources occur only once in the dataset. Sources that occur ten times or less make up around 84 percent and 95 percent of the sources in the 'ASD' and 'GEW' dataset respectively. The column *categories*, shows the average number of different categories a news article is labeled with per source. The 'ASD' dataset has in total 53 categories, and the 'GEW' dataset 22. So if a source has 30 distinct categories in the 'ASD' dataset, it still means that it has never published in 23 categories, which can be useful. For each category and each source also the probability is calculated that the source will publish an article labeled with that category. The category that has the highest probability is taken as the *max probability* for each source, which is then averaged out over all sources in that partition.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.17: A scatter plot of the number of news articles a source has contributed to the dataset, the y axis is logarithmic

TABLE 5.11: The number of sources and other statistics for sources that occur in a specific number of examples in the 'ASD' dataset

Examples	Sources	% of all Sources	Categories	Max Probability
1	700	55.25%	2.40	1
2 to 10	369	29.12%	5.10	0.789
11 to 100	154	12.15%	15.25	0.669
101 to 1,000	40	3.16%	34.48	0.562
1,001 to 10,000	4	0.32%	50.5	0.314

TABLE 5.12: The number of sources and other statistics for sources that occur in a specific number of examples in the 'GEW' dataset.

Examples	Sources	% of all Sources	Categories	Max Probability
1	728	63.14%	1.14	1
2 to 10	372	32.26%	1.96	0.856
11 to 100	50	4.34%	4.96	0.774
100 to 1,000	2	0.17%	19	0.322
1,001 to 10,000	1	0.09%	23	0.523

Sources that occur very few times are not reliable to use for predicting the categories of a news article. Since it may be completely by chance that a specific source is associated with a specific category. The more examples a source has, the more certain we are about which categories it can predict. In addition a source that occurs only once will be of no use when using 10-fold validation. This is because either the source occurs in the training set or in the test set, but not both. This means that when testing the classifier, the source of an example will not be found in the model, and will be regarded as a missing value.

Unfortunately we can see from the tables that for sources that occur frequently the *distinct categories* they publish in increases and the *max probability* decreases.

With the sources that occur in $1,001$ to $10,000$ documents is something special. For the 'ASD' dataset the missing values (4,481 in total) are counted here as one source, which will probably account for half of the examples in that partition. This probably causes the *distinct categories* average to be a bit higher and the *max probability* to be a bit lower than when left out.

For the 'GEW' dataset the 'AFP' source, which contains around 41% of the data, is the only source in the partition of sources that occur in $1,001$ to $10,000$ documents. We can see it publishes in all available categories, but also has a high *max probability.* Which means it publishes 52.2 percent of the time in a specific category. With this information we know that it will be very hard to predict a category based on the source alone. Furthermore it will be very hard for the source attribute to be helpful as this multinomial value, because there are too many values that are very rare. The improvement in performance for the 'GEW' dataset is probably due to the high number of examples that have the source 'AFP'.

Decision trees and multinomial values

In section 2.6 DTs are briefly explained, as well as how they split on nominal values. The 'minimal leaf size' parameter is also briefly discussed there.

We figured out our DT induction algorithm uses the C4.5 way of splitting nominal attributes, which means that it creates a branch for every value. If the split would be made at the root node, then it could mean for the 'GEW' dataset that around 1,200 branches are constructed with a lot of child nodes containing just one example. You can imagine that now the data is so fragmented that further splitting on terms is of little use. Of course splitting on the source attribute can also occur deeper in the tree, where possibly less branches are constructed, but still the data gets very fragmented. It could be a good thing if every source publishes in the same categories every time. However this is not the case.

We noticed that because of a setting in our DT algorithm, called 'minimal leaf size', which was set to two, almost no splits on the *source* attribute were made. This parameter determines the minimum number of examples allowed for a leaf node at the DT. When the algorithm is determining whether to split on an attribute and one of the generated child nodes would contain less than this specified 'minimal leaf size', the split will not be considered.

Therefore, when 'minimal leaf size' is set to two, only a split on the source attribute can be considered when all the values in that partition of the data have more than one example. For example, when first splitting on terms, a partition of the data can occur where only a few sources are left, which may all have more than one example in that partition. We analyzed the DTs and noticed this is indeed the case in the 'GEW' dataset with the 'AFP' source and in some cases in the 'ASD' dataset. Apparently the splits on 'AFP' do help in the 'GEW' dataset, because the performance of using the source is better than not using the source.

We tested changing the 'minimal leaf size' to one, which results in splits on all values of the source attribute, often at root level, which results in too fragmented data. Also when not using the source, setting the 'minimal leaf size' to one decreases the performance. We also tested the performance of a DT which uses the source as only feature. The performances are displayed in table 5.13 and table 5.14 for the 'ASD' and 'GEW' dataset respectively.

Perhaps using a different method of splitting nominal values is better suited for a nominal attribute with many distinct values. For example, the CART algorithm creates only two branches, and groups different values together. However, because then still sources that occur only one time in the dataset are used for predicting, we still doubt if it would make an improvement.

TABLE 5.13: The results of the 'Including the source attribute' experiment on the 'ASD' dataset.

Experiment	**SourceIncluded**	**MinimalLeafSize**	**Micro-Precision**	**Micro-Recall**	**Micro-F1**	**AtLeast1Good**	**NonePredictedAtAll**	**HammingLoss**
Default	no	2	.620	.588	.604	.847	.064	.0317
Default	yes	2	.619	**.590**	**.604**	**.848**	**.063**	**.0317**
Minimal Leaf Size	no	1	.603	.576	.589	.836	.065	.0330
Minimal Leaf Size	yes	1	.549	.555	.552	.801	.076	.0370
Only source	yes	1	**.638**	.032	.061	.054	.936	.0405

TABLE 5.14: The results of the 'including the source attribute' experiment on the 'GEW' dataset

Experiment	**SourceIncluded**	**MinimalLeafSize**	**Micro-Precision**	**Micro-Recall**	**Micro-F1**	**AtLeast1Good**	**NonePredictedAtAll**	**HammingLoss**
Default	no	2	.759	.705	.731	.807	.107	.0284
Default	yes	2	.758	**.710**	**.733**	**.812**	**.102**	**.0282**
Minimal Leaf Size	no	1	.735	.686	.710	.787	.117	.0307
Minimal Leaf Size	yes	1	.735	.597	.659	.680	.224	.0338
Only Source	yes	1	**.840**	.190	.310	.225	.742	.0463

In order to make use of the source attribute, with our current DT induction algorithm, we have to translate it to one or more features that the DT can efficiently use. We think there is a way of translating the source attribute to a feature that can help the DT. However DTs learned with the source as only feature have very poor performance. In the next section we will explain a method which translates the source attribute to different features, which might help the classifier.

5.3.4 Improving the source attribute experiment

Because the source attribute as a multinomial value did not work well in our DT induction algorithm, we decided to come up with an alternative way of using it. We know from the previous section that rare sources probably have lower predictive power than sources that occur more frequently in the data. Furthermore each source tends to publish in many distinct categories. We tried to combine these two properties and generated a new feature for each category c and source s, called the $\mathit{SmoothedSourceProbability}(s, c)$, which replaces the original source attribute. *Smoothing* is a technique in statistics that tries to filter out noise or other short term variations, to reveal important patterns in the data. In our case we attempt to remove the short term variations that is caused by sources that don't occur frequently in the dataset. Each classifier for a specific category c then gets this $\mathit{SmoothedSourceProbability}(c)$ feature instead of the original source attribute. This feature is a weighted combination of the $\mathit{SourceProbability}(s, c)$ and the $\mathit{priorProbability}(c)$. The $\mathit{SourceProbability}(s, c)$ is the estimated conditional probability that an article belongs to c given that it's published by source s. The $\mathit{priorProbability}(c)$ is the estimated probability that an arbitrary example belongs to category c, without knowing it's source. We will explain both more formally next.

Source probabilities Given a source s and a category c, the $\mathit{SourceProbability}(s, c)$ is the estimated conditional probability that s will publish a news article belonging to category c. The probabilities are calculated with the training data to give an estimate for previously unseen examples.

Definition 5.1. SourceProbability(c,s) : SP(c,s) = $\hat{P}(category = c|source = s)$

We could use only the *SourceProbability* as a feature, but then sources that occur in only one news article have a probability of one for each category that news article is labeled with. To give more weight to a source that occurs more frequently in the dataset, we decided to *smooth* the *sourceProbability* by the *PriorProbability.* The *PriorProbability* for a specific category c is the probability that an arbitrary example will belong to c, without that any information about the source of the article is known. More formally this is defined as:

Definition 5.2. PriorProbability(c) : PP(c) = $\hat{P}(category = c)$

The idea is to give very rare sources a probability that is close to the *PriorProbability*, and sources that have more examples a probability that is closer to their actual *SourceProbability*. We constructed a weighted combination of both the *Prior Probability* and the *SourceProbability*, called the *SmoothedSourceProbability*. Let α be the *PriorProbabilityWeight* and let β be the *SourceProbabilityWeight*. Then the *SmoothedSourceProbability* is defined as follows:

Definition 5.3. SmoothedSourceProbability(c,s) : SSP(c,s) = $\dfrac{\alpha.PP(c) + \beta.SP(c,s)}{\alpha + \beta}$

We take β to be equal to the number of examples with the source s in the training data Tr, $\beta = |Tr[source = s]|$. When a source in the test set is unknown or is missing, we just take the prior probability.

We constructed a small experiment to test different values of α to find out which value is the best suited for each dataset. We decided to test for the following values for α: 0, 2, 5, 10 and 50. We will refer to these settings as **SmoothedSourceProbability:0**, **SmoothedSourceProbability:2**, **SmoothedSourceProbability:5**, **SmoothedSourceProbability:10** and **SmoothedSourceProbability:50** respectively.

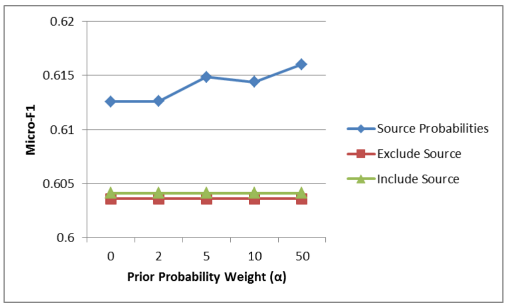
5.3.4.1 Results

We have tested the effect of transforming the source attribute to a vector of smoothed probabilities, one for each category. Each classifier for a specific category c now receives a *SmoothedSourceProbability* instead of the source attribute itself. We experimented with varying the *PriorProbabilityWeight* (α), which we will present here.

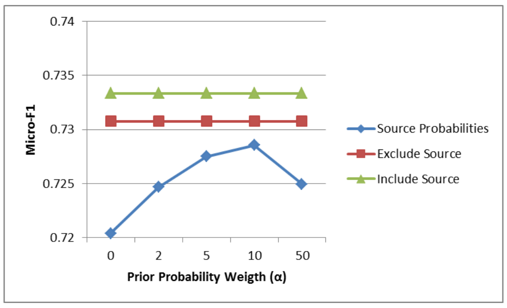
In figure 5.18a and 5.18b we present the *Micro-F1* performances of using source probabilities with different *prior probability weights* for the 'ASD' and 'GEW' dataset respectively. The graph also shows the performances of including the source attribute as a nominal value and not using the source at all. We scaled the y-axis differently than in previous experiments, because the differences in performance are very small.

Looking at the graphs we can see that transforming the source attribute to probabilities does have a positive effect on the classification performance for the 'ASD' dataset. However for the 'GEW' dataset, the performance decreases. Here, using no source attribute at all still performs better than using source probabilities, however using the source as a nominal attribute gives the best performance. Looking at the *PriorProbabilityWeight* (α) we can see that the best value differs for each dataset. It seems that for the 'GEW' dataset the best weight is ten. For the 'ASD' the best value is 50, or perhaps even higher. The performance seems to increase when α increases, until the best value is reached. We must note however that the performances of classifiers with different values for α do not differ much.

All performance measures for the *SmoothedSourceProbabilities* experiment are displayed in table 5.15a and 5.15b for the 'ASD' and 'GEW' dataset respectively.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.18: The performances of the *SmoothedSourceProbabilities* experiment with on the x-axis the *PriorProbabilityWeight* (α)

TABLE 5.15: The *SmoothedSourceProbability* measures per *PriorProbabilityWeight* (α) value

	PriorProbabilityWeight				
Measure	$\alpha = \mathbf{0}$	$\alpha = \mathbf{2}$	$\alpha = \mathbf{5}$	$\alpha = \mathbf{10}$	$\alpha = \mathbf{50}$
Micro-Precision	.6130	.6123	.6115	**.6167**	.6125
Micro-Recall	.6121	.6129	.6182	.6120	**.6195**
Micro-F1	.6126	.6126	.6148	.6144	**.6160**
AtLeast1Good	.8568	.8578	.8637	.8627	**.8670**
NonePredictedAtAll	.0536	.0527	.0466	.0492	**.0455**
Hamming-Loss	.03178	.03182	.03180	**.03154**	.03171

(A) The 'ASD' dataset

	Prior ProbabilityWeight				
Measure	$\alpha = \mathbf{0}$	$\alpha = \mathbf{2}$	$\alpha = \mathbf{5}$	$\alpha = \mathbf{10}$	$\alpha = \mathbf{50}$
Micro-Precision	.7405	.7433	.7513	**.7528**	.7490
Micro-Recall	.7014	**.7070**	.7051	.7058	.7024
Micro-F1	.7204	.7247	.7275	**.7285**	.7249
AtLeast1Good	.8008	**.8059**	.8023	.8028	.7985
NonePredictedAtAll	.0930	**.0909**	.1028	.1036	.1068
Hamming-Loss	.02976	.02936	.02887	**.02875**	.02913

(B) The 'GEW' dataset

5.3.4.2 Discussion

The key result to this experiment is that transforming the weight to source probabilities works for the 'ASD' dataset and doesn't for the 'GEW' dataset. In addition the effect of the *prior probability weight* is very small, but it's best value in both datasets is not close to zero. A value of 50 and 10 seems to work the best for the 'ASD' and 'GEW' dataset respectively.

The reason why the *SmoothedSourceProbabilities* does work on the 'ASD' and doesn't work for the 'GEW' dataset could be explained by various differences between the two datasets. For example, the source attribute in the 'GEW' dataset has only 52 sources that occur more than 10 times in the dataset, compared to the 'ASD' dataset which has 198 sources that occur that many times. The fact that the 'GEW' dataset has a single source, 'AFP', that occurs in around 41% of the examples can also be the reason why the *SmoothedSourceProbabilities* do not work well. Because this is such a large part of the data, it is perhaps better if a split could be made in the DT such that we obtain a node that contains only examples from 'AFP', instead of splitting on a value of the *SmoothedSourceProbability*.

We decided to give a weight to the *prior probability* such that sources that do not occur frequently in the dataset have a probability close to the prior probability. Sources that occur more frequent will have a probability close to their real source probability. The motive behind this was that sources that are very rare in the dataset don't have much predictive power, as opposed to sources that occur more frequently. The DT implicitly groups sources together, by splitting on a specific *SmoothedSourceProbability* value. The *PriorProbabilityWeight* plays a role in distributing the low-frequency sources among these groups. Apparently this weight helps the classifier, which could mean our motive was right.

5.3.5 Combining the results

We have come to the end of the first experiment, 'The feature representation of articles'. We have experimented with various representations and will give an overview of the performances so far. We will combine the winner from the *attribute weighting* experiment and the winner from the *including the source attribute* experiment. Because in both datasets this combination has a higher performance than each experiment in isolation, we call this setting the 'BestRepresentation'.

Table 5.16 and 5.17 give an overview of the different settings of all experiments. The default and selected settings from the *factorial experiment* are displayed as well as the best settings from the *attribute weighting* experiment and the *including the source attribute* experiment. The default experiment for the *factorial experiment* are the settings what we would have chosen as default values.

Recall that the attribute weighting did not work in combination with the 'WordNet' stemming algorithm, so we changed it to the 'Snowball' stemming algorithm for that experiment. This resulted in a small decrease of 0.005 in *Micro-F1* performance for the 'ASD' dataset and no significant difference for the 'GEW' dataset.

We will use the 'BestRepresentation' of both datasets for the next experiment, 'Exploiting the label structure'. This experiment will also uses the 'Snowball' stemming algorithm, because we also use attribute weights here.

Table 5.16: An overview of settings and performances of the 'The feature representation of articles' experiment on the 'ASD' dataset

Setting/Performance	**FactorialDefault**	**FactorialSelected**	**AttributeWeightsBest**	**SourceBest**	**BestRepresentation**
Features	80	80	80	80	80
Word Modeling	1-gram	1-gram	1-gram	1-gram	1-gram
BlackList	no	yes	yes	yes	yes
Text Selection	All	T+500c	T+500c	T+500c	T+500c
Title/Full-text weight ratio	1	1	4	1	4
Stemming Algorithm	WordNet	WordNet	SnowBall	WordNet	SnowBall
Source Included	no	no	no	yes	yes
Source Type	-	-	-	probability	probability
Micro-Precision	.584	.620	.618	.613	**.620**
Micro-Recall	.612	.588	.591	**.620**	.614
Micro-F1	.598	.604	.604	.616	**.617**
AtLeast1Good	.866	.847	.854	**.867**	.863
NonePredictedAtAll	.041	.064	.060	**.046**	.050
Hamming-Loss	.0338	.0317	.0318	.0317	**.0313**

Table 5.17: An overview of settings and performances of the 'The feature representation of articles' experiment on the 'GEW' dataset

Setting/Performance	**FactorialDefault**	**FactorialSelected**	**AttributeWeightsBest**	**SourceBest**	**BestRepresentation**
Features	80	40	40	40	40
Word Modeling	1-gram	2-gram	2-gram	2-gram	2-gram
BlackList	no	yes	yes	yes	yes
Text Selection	All	T+500c	T+500c	T+500c	T+500c
Title/Full-text weight ratio	1	1	4	1	4
Stemming Algorithm	WordNet	WordNet	Snowball	WordNet	Snowball
Source Included	no	no	no	yes	yes
Source type	-	-	-	nominal	nominal
Micro-Precision	.681	.759	**.772**	.758	.769
Micro-Recall	**.730**	.705	.714	.710	.721
Micro-F1	.704	.731	.742	.733	**.744**
AtLeast1Good	**.826**	.807	.814	.812	.820
NonePredictedAtAll	.072	.107	.111	**.102**	.103
Hamming-Loss	.0335	.0284	.0272	.0282	**.0271**

5.4 Exploiting the label structure

This experiment, as described in section 4.3, consists of two sub-experiments, namely *chaining* and *hierarchical top-down classification* (HTC). The first experiment will explore the *classifier chains* (CC) technique while varying the number of chaining features. The HTC experiment will also vary the number of features that is selected. Both experiments will use the 'BestRepresentation' setting from the previous experiment, assuming that it will also give the best results in combination with these techniques.

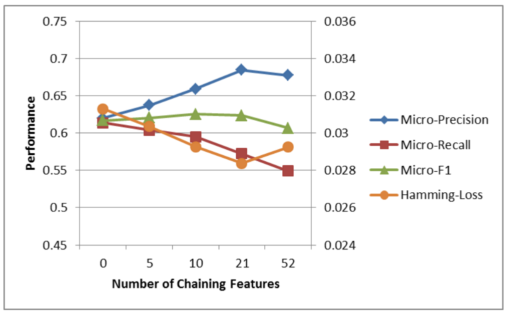
5.4.1 Chaining

We have tested the CC technique as described in section 4.3.2. In addition the number of chaining features that are selected for each classifier is varied by applying feature selection specially for these features.

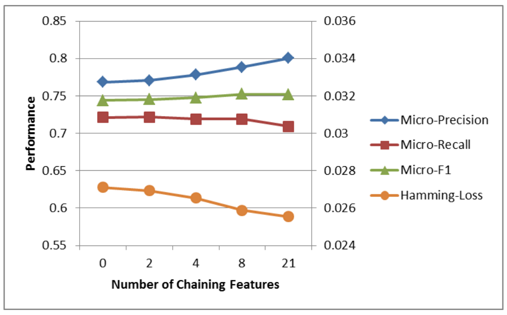
Figure 5.19a and 5.19b show the performances of these experiments for the 'ASD' and 'GEW' dataset respectively. We observe that CC has a positive effect on the performance of the classifiers. The best performance is achieved by not using all chaining features, but instead by using the top 10 and 8 chaining features for the 'ASD' and 'GEW' dataset respectively.

The number of chaining features clearly has a positive effect on the *Micro-Precision* and a negative effect on *Micro-Recall*. The effect is the strongest on the 'ASD' dataset. The *Hamming-Loss* measure seems to be similar to the *Micro-Precision* measure.

Table 5.18a and 5.18b show the performances for the 'ASD' and 'GEW' dataset respectively. In general the *Micro-F1* performance is slightly increased when using the right number of chaining features.



(A) The 'ASD' dataset



(B) The 'GEW' dataset

FIGURE 5.19: The performances of the chaining experiment with on the x-axis the number of chaining features

Table 5.18: The performances of the chaining experiment per number of chaining features

	Number of Chaining Features				
Measure	**0**	**5**	**10**	**21**	**52**
Micro-Precision	.620	.638	.659	**.685**	.678
Micro-Recall	**.614**	.604	.595	.572	.549
Micro-F1	.617	.620	**.625**	.623	.606
AtLeast1Good	**.863**	.854	.851	.834	.809
NonePredictedAtAll	**.050**	.062	.065	.077	.098
Hamming-Loss	.0313	.0304	.0293	**.0284**	.0292

(a) The 'ASD' dataset

	Number of Chaining Features				
Measure	**0**	**2**	**4**	**8**	**21**
Micro-Precision	.769	.771	.778	.789	**.800**
Micro-Recall	.721	**.722**	.719	.719	.709
Micro-F1	.744	.746	.748	**.752**	.752
AtLeast1Good	.820	**.821**	.819	.819	.809
NonePredictedAtAll	.103	.101	.100	**.098**	.100
Hamming-Loss	.0271	.0269	.0265	.0259	**.0256**

(b) The 'GEW' dataset

5.4.1.1 Discussion

The key results from the chaining experiment are that when using the right number of chaining features it does improve the *Micro-F1* performance. Furthermore it seems that the number of chaining features does have a positive effect on the *Micro-Precision* and a negative effect on the *Micro-Recall*.

We analyzed the generated DTs and noticed that the chaining features occur a lot. Table 5.19a and 5.19b show some statistics about the generated DT models. Chaining edges are the number of edges in the generated DT that indicate a split on a chaining feature. We can see that when using enough features, more than 30% of the edges in the DT are chaining edges. Another interesting thing is that when all chaining features are used, the total number of edges decreases considerably. This is a good thing, because it seems like the model becomes considerably less complex. However, including the prediction of another classifier as features could also be seen as including the whole DT of that classifier at that place in the tree. Thereby making the DT not less, but instead more complex.

The reason why the *Micro-F1* performance increases is that apparently the chaining features can help the DT in predicting the class memberships. The idea behind chaining is that we can exploit the correlations between categories. The reason why *Micro-Precision* increases and *Micro-Recall* decreases could be explained by the number of

Table 5.19: Information about the decision tree models per number of chaining features

ChainingFeatures	**Edges**	**ChainingEdges**	**%**
0	277.55	0	0 %
5	282.80	60.75	21.48%
10	249.53	86.97	34.85%
21	120.06	43.72	36.42%
52	76.40	27.05	35.41%

(a) The 'ASD' dataset

ChainingFeatures	**Edges**	**ChainingEdges**	**%**
0	86.34	0	0 %
2	90.93	9.45	10.39%
4	91.90	18.30	19.91%
8	92.55	31.29	33.81%
21	67.14	26.41	39.34%

(b) The 'GEW' dataset

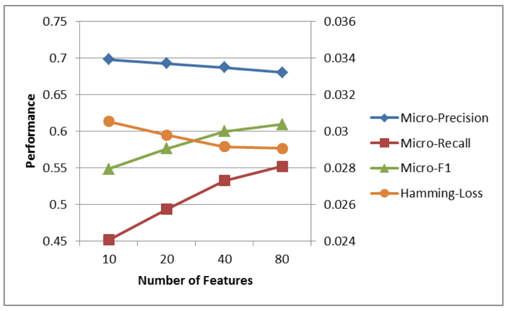
edges in the generated DT. We can see that the effect on *Micro-Precision* and *Micro-Recall* is the largest with the 'ASD' dataset. Looking at table 5.19 we also see the number of edges drop more considerably in the 'ASD' dataset.

Chaining was introduced by Read et al. [18] as a way to exploit correlation between class labels. In his experiments an ensemble of chaining classifiers(ECC) is created, each with a random order of the chain. In our experiments we constructed the order in advance, by using regression, see section 4.3.2. This method seems to work, as the performance in both datasets is improved. In addition we applied a separate feature selection mechanism for chaining features, which also has a clear effect on the performance and generated DT models.

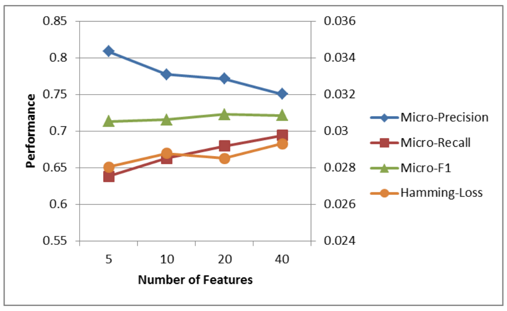
5.4.2 Hierarchical top-down classification

We experimented with HTC using the 'siblings' approach to select training-data for each node in the hierarchy. Based on research by Koller and Sahami [9] we decided to experiment with the number of features. The classification setup that does not use the hierarchical setup is called *flat classification* (FC). More information about the experiment can be found in section 4.3.3 and details about how the hierarchical category structure is constructed can be found in section 3.4.

Figures 5.20a and 5.20b show the results of this experiment for the 'ASD' and 'GEW' dataset respectively. We observe that for both datasets the best performing number of features is not far from the best performing number of features in FC. Moreover we can see that for the 'ASD' dataset the performance increases when more features are used and the best performing number of features is 80, as used in FC. For the ‘GEW’ dataset we observe the same behavior, except that the best performing number of features is 20, which is lower than in the FC setup. However we did not test this number of features for the ‘GEW’ dataset in the first experiment, so it might as well be equal to the best performing number of features.



(A) The ‘ASD’ dataset



(B) The ‘GEW’ dataset

FIGURE 5.20: The performances of the hierarchical top-down experiment with on the x-axis the number of features

To compare the results with the flat classification approach we included the ‘BestRepresentation’ experiment, denoted as ‘Flat’ in the results. These results are shown in table 5.20a and 5.20b for the ‘ASD’ and ‘GEW’ dataset respectively. Compared to the ‘Flat’ approach the performance of the ‘Hierarchical’ approach is considerably worse in both datasets. The *Micro-Precision* of the ‘Hierarchical’ approach is higher than the ‘Flat’ approach with the same number of features in the ‘ASD’ dataset and just a little bit lower in the ‘GEW’ dataset. However the *Micro-Recall* is considerably lower in all ‘Hierarchical’ experiments.

The performance of a HTC system depends heavily on the individual performance of the classifier in higher levels of the hierarchy. When classifiers at higher levels of the hierarchy predict the non-membership for their category, then the child classifiers, lower in the hierarchy will not be applied. This is called the *blocking-problem*. Therefore we are also interested in the performances of the classifiers that are not at the bottom of the hierarchy. In our case there is only one layer above the leaf classifiers (level 3), which we will refer to as level 2 classifiers. The performances of the level 2 classifiers are displayed in table 5.21a and 5.21b for the ‘ASD’ and ‘GEW’ dataset respectively.

5.4.2.1 Discussion

One of the results was that the best number of features for HTC is probably the same as the number of features needed in FC. The reason why our results are different from the research by Koller and Sahami [9] is probably because they didn’t use local feature

Table 5.20: The performances of the hierarchical top-down experiment per number of features

	Number of Features				
	Flat	**Hierarchical**			
Measure	**80**	**10**	**20**	**40**	**80**
Micro-Precision	.620	**.698**	.692	.687	.680
Micro-Recall	**.614**	.452	.493	.532	.552
Micro-F1	**.617**	.549	.576	.600	.609
AtLeast1Good	**.863**	.710	.759	.796	.820
NonePredictedAtAll	**.050**	.194	.150	.117	.095
Hamming-Loss	.0313	.0305	.0298	.0292	**.0291**

(a) The 'ASD' dataset

	Number of Features				
	Flat	**Hierarchical**			
Measure	**40**	**5**	**10**	**20**	**40**
Micro-Precision	.769	**.808**	.777	.771	.750
Micro-Recall	**.721**	.638	.663	.680	.694
Micro-F1	**.744**	.713	.716	.723	.721
AtLeast1Good	**.820**	.738	.763	.780	.790
NonePredictedAtAll	**.103**	.177	.138	.124	.115
Hamming-Loss	**.0271**	.0280	.0288	.0285	.0293

(b) The 'GEW' dataset

Table 5.21: The performances of the level-2 classifiers in the Hierarchical Top-Down experiment per number of features

	Number of Features			
Measure	**10**	**20**	**40**	**80**
Micro-Precision	.856	.855	**.872**	.867
Micro-Recall	.818	.868	.871	**.881**
Micro-F1	.837	.861	.871	**.874**

(a) The 'ASD' dataset

	Number of Features			
Measure	**5**	**10**	**20**	**40**
Micro-Precision	.876	.877	.884	**.892**
Micro-Recall	.850	.858	.871	**.880**
Micro-F1	.863	.867	.877	**.886**

(b) The 'GEW' dataset

selection for their FC approach. They used global feature selection instead, which means that the same features are selected for all categories/classifiers. As opposed to selecting different features per category/classifier. Furthermore they didn't use DTs in their experiment, but probabilistic classifiers.

Another key result of our experiment was that HTC performed worse than FC. In a survey on *Hierarchical Classification* (HC) by Silla and Freitas [10], HC outperformed

FC in many cases. There are however many approaches of HC, depending on whether the original problem is *multi-label* or *single-label*. Perhaps the *hierarchical structure* we created in this domain does not offer advantages over *flat classification* or is not large enough.

A possible problem in HTC is the *blocking problem*. This occurs when higher-level classifiers incorrectly classify a positive example, a *false negative*, such that the example will not be propagated down the hierarchy tree. The higher-level classifier blocks the positive example from being classified correctly. Our higher-level classifiers (level-2) got a *Micro-Recall* of around 88% in each dataset, which is quite accurate, see tables 5.21a and 5.21b. However still 12% of the positive examples gets blocked. By increasing the *Micro-Recall* of those level-2 classifiers we could probably increase the overall performance. However, increasing the *Micro-Recall* often goes hand-in-hand with decreasing *Micro-Precision*, which will be bad for the performance.

Although the results were not as expected, we note that the CPU-time for training a classifier drastically decreased when using HTC. This is because the size of the training data is much smaller for most classifiers.

5.4.3 The best performing setting

We have come to the end of the second experiment and found out that CC results in the best performance, with 10 and 8 *chaining features* for the 'ASD' and 'GEW' dataset respectively. With this setting the *Micro-F1* performances are **.625** and **.752** for the 'ASD' and 'GEW' dataset respectively. We always showed the *micro averaged* performance of all 'one-against-all' classifiers, because we want to measure the performance of the classification system as a whole. We would like to display the performances of all individual classifiers for once, just so you can get an idea how the individual categories perform. Table 5.22 and 5.23 show the performances for all individual classifiers for the 'ASD' and 'GEW' dataset respectively. We can observe that the individual performances differ a lot. Some categories even have a *F1* performance near zero. In general the more positive examples a category has, the better the *F1* performance. However there are some categories that do have a moderate number of positive examples and still have a really poor performance.

TABLE 5.22: The performances per category for the best performing setting for the 'ASD' dataset, sorted by the number of positive examples

Category	**Positive Examples**	**% of dataset**	**Precision**	**Recall**	**F1**
103050	5215	20.03%	.809	.888	.846
101078	4043	15.52%	.629	.719	.671
102030	3706	14.23%	**.903**	**.944**	**.923**
102050	3324	12.76%	.659	.754	.703
101058	3152	12.10%	.692	.686	.689
101048	2722	10.45%	.638	.610	.624
101020	1969	7.56%	.687	.708	.698
101024	1677	6.44%	.769	.729	.749
101030	1641	6.30%	.648	.562	.602
103034	1569	6.02%	.558	.370	.445
101014	1569	6.02%	.573	.361	.443
101060	1558	5.98%	.819	.721	.767
101054	1552	5.96%	.553	.422	.479
102060	1414	5.43%	.506	.322	.394
101028	1244	4.78%	.723	.750	.736
101070	1220	4.68%	.621	.807	.702
102034	1139	4.37%	.613	.678	.644
102010	1084	4.16%	.307	.057	.096
102054	1061	4.07%	.501	.322	.392
103040	1024	3.93%	.604	.457	.520
101068	976	3.75%	.648	.607	.627
101040	953	3.66%	.448	.341	.387
103020	809	3.11%	.683	.714	.698
101084	787	3.02%	.442	.375	.406
101044	775	2.98%	.408	.279	.331
102040	745	2.86%	.521	.670	.586
101074	685	2.63%	.647	.438	.522
101050	655	2.52%	.468	.220	.299
101080	650	2.50%	.631	.438	.517
103070	628	2.41%	.654	.817	.727
101064	571	2.19%	.278	.070	.112
103030	570	2.19%	.599	.467	.525
102090	538	2.07%	.389	.370	.379
102024	535	2.05%	.381	.234	.290
101034	516	1.98%	.557	.320	.406
102020	499	1.92%	.347	.066	.111
101038	467	1.79%	.757	.827	.790
102074	385	1.48%	.437	.281	.342
103010	331	1.27%	.537	.284	.372
101010	330	1.27%	.426	.427	.427
102014	324	1.24%	.250	.096	.138
103090	324	1.24%	.089	.012	.022
103014	309	1.19%	.644	.725	.682
101088	230	0.88%	.338	.300	.318
101018	224	0.86%	.533	.688	.600
102044	220	0.84%	.048	.009	.015
103044	195	0.75%	.120	.015	.027
103024	132	0.51%	.305	.220	.256
102070	123	0.47%	.115	.024	.040
102080	114	0.44%	.390	.263	.314
103060	77	0.30%	.227	.130	.165
102084	57	0.22%	.077	.035	.048
102064	42	0.16%	.273	.071	.113

Table 5.23: The performances per category for the best performing setting for the 'GEW' dataset, sorted by the number of positive examples

Category	**Positive Examples**	**% of dataset**	**Precision**	**Recall**	**F1**
201014	1820	30.51%	.846	.850	.848
202020	1329	22.28%	**.959**	.924	**.941**
201040	547	9.17%	.924	**.934**	.929
202010	499	8.37%	.910	.912	.911
202024	443	7.43%	.854	.716	.779
202064	309	5.18%	.492	.317	.386
201060	306	5.13%	.371	.173	.236
201010	300	5.03%	.471	.460	.465
202034	288	4.83%	.644	.604	.624
202050	187	3.13%	.599	.599	.599
201050	186	3.12%	.516	.355	.420
202044	178	2.98%	.590	.646	.617
202030	165	2.77%	.774	.745	.759
201034	142	2.38%	.519	.472	.494
201044	119	1.99%	.641	.765	.697
202040	68	1.14%	.154	.059	.085
201064	62	1.04%	.167	.048	.075
202014	61	1.02%	.918	.738	.818
202054	56	0.94%	NaN	.000	NaN
201020	45	0.75%	.304	.156	.206
201030	36	0.60%	.083	.028	.042
201024	27	0.45%	.188	.111	.140

Chapter 6

Conclusions

We have performed a case study on classifying news articles, collected by ASDMedia. They publish business-to-business news articles online for different markets. For our series of experiments we use two datasets namely 'ASD' and 'GEW'. The 'ASD' dataset contains news articles for the 'Aerospace and Defence' market, which is more *multi-label* and also around 4.3 times bigger than the 'GEW' dataset, which contains news articles for the 'Energy and Resources' market. Each news article can be labeled with multiple categories, thereby making it a *multi-label text classification* (MTC) problem. This makes it a more challenging problem than *single-label classification* (SC) problems.

The goal of this study is to find out if it's possible to construct a classification system that can classify these news articles with reasonable performance. For analytical purposes we restricted ourselves to classifiers which are easy to interpret by humans and therefore decided to use *Decision Trees* (DTs). We experimented with various settings and techniques to find out which setup for a classification system is the best suited for each dataset. We have done this by:

- **Finding the best feature representation of news articles**
  In this case study we investigated various feature representations of news articles. We experimented with varying the number of features that is needed, applying a feature blacklist and whether or not to include the source of the news article. Furthermore we looked at how the features are generated from the text by experimenting with the amount of text that is selected. We experimented with applying different weights to words originating from the title and words originating from the full-text. We also considered generating features of word-phrases of length two (2-grams) instead of single words (1-grams).

- **Exploiting the label structure**
  After having experimented with the feature representation of the news article, we selected the best representation and continued to look at two techniques that try to exploit the label structure of this classification problem; namely *classifier chains* (CC) and *hierarchical top-down classification* (HTC). The first method exploits an internal structure between the labels, namely the correlation between them. It creates a chain of classifiers and passes the result of each classifier earlier in the chain to the classifiers later in the chain. The second method makes use of the hierarchical structure of the categories, which is an external structure of the labels. It does so by learning classifiers for each node in the hierarchy, and then applies the classifiers in a top-down fashion.

6.1 Main findings

In this section we will discuss our main findings. First we discuss the findings in the 'feature representation of news articles' experiment followed by the findings in the 'exploiting the label structure' experiment.

6.1.1 Feature representation of news articles

Our research on the feature representation of news articles shows a statistically significant interaction between the amount of text that is selected and the number of features that is selected. In general the highest performance was achieved by selecting the title and the first paragraph of news articles. A classifier using only the title achieves a very high *Micro-Precision* but has a very low *Micro-Recall.* Furthermore, the best performing number of features is different for each dataset. Either increasing the number of features selected or increasing the amount of text selected results in a better *Micro-Recall* and a worse *Micro-Precision.*

We experimented with including 2-grams in the feature space, which are features that represent two words next to each other in the text. However, in combination with selecting a large amount of text and a large dataset this led to a 'feature explosion', which we experienced ourselves. Including 2-grams however, had no significant effect on the performance. We showed that for a high percentage of the 2-grams that are being selected, the single words (1-grams) that form the 2-gram are also being selected by *feature selection.* This means that they probably could be replaced by using both 1-grams in the DT, which explains why there is not much difference in the performances. The main difference between using two 1-grams in the DT and using one 2-gram is that with the latter the words have to occur in exactly that order and next to each other, which apparently doesn't make the difference in this domain.

We experimented with a custom feature blacklist made by domain-experts. This feature blacklist consists of features that shouldn't be discriminative for a category according to the domain-experts. The features in the blacklist are removed from the feature space. Although it did slightly improve the performance on the 'GEW' dataset, it had no effect on the 'ASD' dataset.

Next we found that giving a higher weight to terms originating from the title of a news article than words that occur in the full-text improves the classification performance. The words in the title are usually very descriptive, which could explain why this weighting mechanism does work.

Lastly we found out that including the source of a news article as a feature for our DT learning algorithm was not a trivial task. There are many different sources which occur with a very low frequency in the dataset, each source having news articles labeled with different categories. For one dataset, the 'GEW' dataset, including the source as a nominal (categorical) feature, improved the performance. However for the other dataset, the 'ASD' dataset, the performance decreased. The 'GEW' dataset has one single source which covered around 40% of the data. For the other dataset we came up with a method to translate the source to a probability vector. For each source s, and classifier for a specific category c, we calculated the probability that s would publish in category c. Furthermore we applied a penalty to sources that do not occur frequently in the data. Including these source probabilities and applying the penalty did improve the performance for the 'ASD' dataset.

6.1.2 Exploiting label structures

Our research on exploiting the label structure showed that a modified version of CC introduced by Read et al. [18] does improve the performance. This is a technique where all 'one-against-all' classifiers form a chain, and the results of the classifiers earlier in the chain are propagated to the classifiers later in the chain. Instead of training an ensemble of classifiers which all have a random order of the chain, we constructed one order by using *multiple linear regression.*

We tried to exploit the hierarchical structure of the categories, by performing HTC. This is a technique which trains a classifier for each node in the hierarchy. Then it starts at the top of the hierarchy and applies classifiers in a top-down fashion. When a classifier predicts the non-membership for its category, its children classifiers are not applied anymore. We found no improvement in performance by using this method, instead the performance decreased considerably. However, the CPU-time needed for this method also considerably decreased, because most classifiers do not use all the training-data. We found no decrease in the best performing number of features for this method as opposed to the default method, which is termed 'flat classification' (FC). Koller and Sahami [9] did find a decrease in the number of features needed when using HTC. However their FC setup differs from ours. We performed feature selection locally, that is for each category separately. They used a global feature selection approach, which means that for all categories the same features are selected.

6.1.3 Conclusion

We have shown that by using an optimized feature representation of articles and by applying the chaining technique to exploit correlations between labels, we achieve the best performance. There are still enough interesting topics to research that might improve the performance even more, which we will discuss in the next section.

First we want to come back to our research question. We have built a classification system by using machine learning and restricted ourselves to an algorithm that generates humanly interpretable models, namely DTs. The question remains whether our classifiers achieve reasonable performance. We intentionally kept vague what precisely reasonable performance is, because the system will be used in a semi-automatic environment. This means that the result of the system is checked by a human, before the choice is final.

There are two types of errors made by the classification system we can distinguish: false positives (FPs) and false negatives (FNs). When for an example a FP is predicted, this indicates that the example is labeled with a category it doesn't have. In this case it should be removed by manual intervention. The other case is a FN, now an example is missing a label it should have. Now manual intervention is needed to add the missing category to the example. Assuming both errors have the same cost, there is no preference between either a high recall or a high precision, so the *Micro-F1* value gives the best indication of how many intervention is required. At some point, it will cost more work to adjust the predictions than it saves work. However we are not sure at which *Micro-F1* value this point lies, but we think it will not be above a value of 0.5 for the *Micro-F1* value. The *Micro-F1* value for the best setting is **.625** and **.752** for the 'ASD' and 'GEW' dataset respectively. So we can conclude that the classification system achieves reasonable performance on both datasets, where the classification system performs considerably better on the 'GEW' dataset.

6.2 Future work

For future research and/or implementation we first will make a couple of remarks. After that we will share our thoughts organized per finding. First of all, we like to note that most experiments had a limited scope. For example, we varied the number of features for only three distinct values, namely 40, 80 and 160 features. For future research it would be advisable to extend this range and include more values. It is possible that the best value is not found yet.

6.2.1 Future implementation

- **Try out other classifiers**
  We advise to experiment with other classifiers, as the DT classifier may not be the best in the field of TC. The *Support Vector Machine* (SVM) is the current state-of-the-art classifier in this field, which would be interesting to investigate.
- **Collect more data for minority classes**
  There are quite a few minority classes which have very few supporting data. The classifiers trained for those classes perform very badly, we advise to collect more data for those categories.
- **Implement mandatory prediction**
  In both datasets there are still examples which are predicted to the empty set of classes. The best performing setting still has 0.065 and 0.098 of those empty set predictions for the 'ASD' and 'GEW' dataset respectively. Since all examples in the training data belong to at least one category, this can clearly be improved.
- **Optimize each setting per classifier**
  In our experiments we tried to find an optimal setting for a specific experiment, by using the same value for each binary classifier in the classification system. For example we found that 80 is the best performing number of features to use for the classification system that uses the 'ASD' dataset. However, this is the best number of features considering that each binary classifier uses the same number of features. For future research it would be interesting to find out the optimal settings per classifier instead. Perhaps some classifiers for categories with very few positive examples could have a higher performance when using more text, or perhaps more features. We suspect that finding the optimal setting per classifier would surely improve the overall performance.
- **Structure and clean the dataset**
  We noticed that the text in the datasets sometimes contained encoding artifacts. In

addition it was not possible to obtain the first paragraph from the full-text, therefore we used the words occurring in the first 500 characters of the full-text instead. It would be advisable to store the data of news articles in a more structured way such that for example headlines and paragraphs can be easily distinguished. This way a classifier can be learned which can exploit this structure. In addition cleaning the dataset from encoding artifacts may also help increase the performance as some characters are now not correctly interpreted.

6.2.2 Future research

- **Stepwise classification**
  We have seen that a classifier that uses only the title can be very precise, but has a low recall, as opposed to a classifier using the title and (a part of) the full-text. It would be interesting to investigate which part of the positive examples they both can classify correctly. More specifically: how large is the intersection between the positive examples correctly classified by both classifiers? Perhaps there is a benefit in training separate classifiers, one that uses only the title and one that uses the full-text and perhaps one that uses both. Then we could first apply the classifier that is the most precise and when it has a low confidence we could apply the second most precise classifier. This mimics the behavior of the domain-experts, as they also classify in a step-wise fashion, by first looking at the title, then the first paragraph and eventually the whole full-text.

- **Finding 2-grams that make the difference**
  We have shown that most of the 2-grams that get selected, also have their 1-gram components selected. They do not have much to contribute if their 1-gram components could also be selected instead. It would be interesting to see if applying a special feature selection for 2-grams, such that we only select distinctive 2-grams that do not have distinctive 1-gram components, could make the difference.

- **Grouping Sources together**
  By translating the multinomial source attribute to smoothed source probabilities we could increase the performance a little bit in the ‘ASD’ dataset. The DT algorithm we used (C4.5), implicitly grouped sources together based on their smoothed source probability. It does so by splitting the data on a particular value for this attribute. However there are many other ways to group sources together, which are interesting to explore. For example it would be interesting how the performance is when using the CART DT algorithm. This algorithm splits a multinomial value in two groups explicitly.

- **Hierarchical Chaining**
  We have tried two techniques to exploit the label structure, CC and HTC. The first did improve the performance and the latter didn't. It would be interesting to investigate how the combination of both methods performs. One could chain the classifiers in multiple ways. For example, all predictions from the classifiers in a certain level can be chained to the classifiers in a lower level. But also classifiers within a level can be chained. There are many possibilities here that are interesting to research.

- **Exploiting the structure of text** We have shown that using titles and the first paragraph of news articles gives a better performance than using only the title, or the title and all the full-text. By weighting the title higher than the first paragraph the performance increased even more. This shows that better results can be obtained if the structure of text is taken into account. Since the first and last sentence in a paragraph often are more important, it would be interesting to segment the full-text in paragraphs and sentences. Then we would expect that giving words from the first and last sentence of a paragraph a higher weight would improve the performance. In addition different weightings can be given to each paragraph.

Bibliography

[1] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[2] Enrique de Argaez. http://www.internetworldstats.com/, 2013. [Online; Accessed on 9 April 2013].

[3] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In Nick Cercone, Tsau Y. Lin, and Xindong Wu, editors, *Proceedings of ICDM-01, IEEE International Conference on Data Mining*, pages 521–528, San Jose, CA, 2001. IEEE Computer Society Press, Los Alamitos, US.

[4] Alessandro Moschitti and Roberto Basili. Complex linguistic features for text classification: A comprehensive study. In Sharon McDonald and John Tait, editors, *Proceedings of ECIR-04, 26th European Conference on Information Retrieval Research*, pages 181–196, Sunderland, UK, 2004. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 2997.

[5] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997.

[6] Ion Androutsopoulos, John Koutsias, Konstantinos V. Cb, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM Press, 2000.

[7] Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001. Available: `http://www.lsi.upc.es/~carreras/pub/boospam.ps`.

[8] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009.

[9] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *ICML-97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 435–443, San Francisco, CA, USA, 1997. Morgan Kaufmann.

[10] Carlos N. Silla, Jr. and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining Knowledge Discovery*, 22(1-2):31–72, January 2011.

[11] Tom M. Mitchell. The discipline of machine learning. Unpublished manuscript, Carnegie Mellon University., 2006.

[12] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach.* Pearson Education, Upper Saddle River, NJ, 2nd international edition, 2003.

[13] de Carvalho and A. A. Freitas. *A tutorial on multi-label classification techniques*, volume Foundations of Computational Intelligence Vol. 5 of *Studies in Computational Intelligence 205*, pages 177–195. Springer, September 2009.

[14] Wei Bi and James T. Kwok. Multilabel classification on tree- and DAG-structured hierarchies. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 17–24. Omnipress, 2011.

[15] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[16] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, volume 3056 of *Lecture Notes in Computer Science*, pages 22–30. Springer, 2004.

[17] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[18] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier Chains for Multi-label Classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 254–269, Berlin, Heidelberg, 2009. Springer-Verlag.

[19] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 279–286. Omnipress, 2010.

[20] Julio H. Zaragoza, Luis Enrique Sucar, Eduardo F. Morales, Concha Bielza, and Pedro Larrañaga. Bayesian chain classifiers for multidimensional classification. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2192–2197. IJCAI/AAAI, 2011.

[21] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer US, 2010.

[22] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[23] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 2(73): 185–214, August 2008.

[24] Feihong Wu, Jun Zhang, and Vasant Honavar. Learning classifiers using hierarchically structured class taxonomies. In Jean-Daniel Zucker and Lorenza Saitta, editors, *Abstraction, Reformulation and Approximation*, volume 3607 of *Lecture Notes in Computer Science*, pages 902–902. Springer Berlin / Heidelberg, 2005.

[25] Carlos Nascimento Silla Jr. and Alex Alves Freitas. Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In *SMC*, pages 3499–3504. IEEE, 2009.

[26] Giorgio Valentini. True path rule hierarchical ensembles for genome-wide gene function prediction. *j-TCBB*, 8(3):832–847, May/June 2011.

[27] Tiziano Fagni and Fabrizio Sebastiani. Selecting negative examples for hierarchical text classification: An experimental comparison. *JASIST*, 61(11):2256–2265, 2010.

[28] G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[29] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text

categorization. In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US, 2001.

[30] Johannes Fürnkranz. A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, 3(1998):1–10, 1998.

[31] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[32] Tadashi Nomoto and Yuji Matsumoto. Exploiting text structure for topic identification. In *Workshop On Very Large Corpora*, 1996.

[33] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248, July 1991.

[34] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proc. 11th International Conference on Machine Learning*, pages 121–129. Morgan Kaufmann, 1994.

[35] Monica Rogati and Yiming Yang. High-performing feature selection for text classification. In Konstantinos Kalpakis, Nazli Goharian, and David Grossmann, editors, *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM-02)*, pages 659–661, New York, November 4–9 2002. ACM Press.

[36] Noam Slonim and Naftali Tishby. The power of word clusters for text classification. In *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*, Darmstadt, DE, 2001.

[37] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. Published in the “Lecture Notes in Computer Science” series, number 1398.

[38] Pei-Lei Tu and Jen-Yao Chung. A new decision-tree classification algorithm for machine learning. In *ICTAI*, pages 370–377, 1992.

[39] J. Ross Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[40] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[41] Ron Kohavi. A study of cross-validation and Bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1143. Morgan Kaufmann, 1995.

[42] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III*, ECML PKDD'11, pages 145–158, Berlin, Heidelberg, 2011. Springer-Verlag.

[43] R. E. Schapire and Y. Freund. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[44] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 37–50, New York, NY, USA, 1992. ACM.

[45] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, 1979.

[46] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[47] Kate McCarthy, Bibi Zabar, and Gary Weiss. Does cost-sensitive learning beat sampling for classifying rare classes? In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 69–77. ACM, 2005.

[48] Princeton University. About wordnet. `http://wordnet.princeton.edu`. Online; Accessed on 14 February 2013.

[49] M.H. DeGroot and M.J. Schervish. *Probability and Statistics.* Addison-Wesley series in statistics. Addison-Wesley, 2002.

[50] Ron Bekkerman and James Allan. Using bigrams in text categorization. *Department of Computer Science, University of Massachusetts, Amherst*, 2004.

[51] Martin F. Porter. Snowball: A language for stemming algorithms. Published online, October 2001. URL `http://snowball.tartarus.org/texts/introduction.html`. Online; Accessed on 14 February 2013.