Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELOR THESIS

Recognition of Company Mergers Using Interactive Labeling and Machine Learning Methods

Anne Lorenz

MIN Faculty
Department of Informatics

Degree course: Software Systems Development
Student number: 6434073

Examiners: Dr. Julian Kunkel, Eugen Betke
Supervisors: Dr. Julian Kunkel, Doris Birkefeld

Submission: Hamburg, 2019-05-08

Abstract

Being informed about current business events is essential for decision makers in a company. The aim of this thesis is to develop a strategy for recognizing company mergers in news articles by applying common and experimental Machine Learning methods. For this text classification problem, an interactive human-computer labeling technique is explored in order to manually label an unclassified data set in a more efficient way. Using class probabilities based on the Naive Bayes algorithm, the iterative approach accelerates the data labeling process by propagating labels through the data set. Through experimental research on this practical application problem, it is found that the proposed labeling technique is eight times faster than conventional labeling, because the number of articles to be labeled can be reduced. On the resulting data set, a common Support Vector Machines model achieves a Recall score of 86.9% and a Precision score of 86.1%. The presented incremental method that is simple to implement is not only suitable for text classification problems, but universal for all kinds of large unclassified data sets, as, e.g., in image classification and speech recognition.

Acknowledgements

I would like to thank my supervisor Dr. Julian Kunkel very much for the intensive, dedicated support, his many ideas, and for challenging me through his high ambitions.

Equally, I would like to thank my supervisor Doris Birkefeld, who has been motivating me and providing me with practical advice and support in all matters.

My thanks also go to my sister Dr. Eva Lorenz, who helped me with the translation into English and other questions regarding scientific work.

Contents

1. Introduction

This chapter illustrates the motivation for this thesis in Section 1.1 by explaining its context and giving practical examples in order to emphasize the utility of this work. The goals of this study are presented in Section 1.2 structuring the overall problem into several subtasks. Finally, Section 1.3 gives an outline of this work by describing the contents of the individual chapters.

1.1. Motivation

Mergers and acquisitions are amongst the most extensive transformations a business company can go through. When it comes to a merger, new constellations of corporations may lead to changing business processes. As this can create new opportunities for other service providers, it is of high interest for organizations to be kept informed of these transactions.

Reading the latest business news daily may be an option for managers to keep track, but it may happen that a merger is missed. Since this approach is furthermore time-consuming and potentially error-prone, it is a candidate for automatization. Current Machine Learning techniques can be used to solve this Natural Language Processing problem. As with all text classification problems, a labeled data set is required to make predictions for future news articles. A data set of classified business news articles can help the computer decide whether a text is about a company merger or not.

For a variety of Machine Learning problems, a labeled data set is needed at the beginning to train a prediction model. In general, the larger the training data, i.e., the labeled data set is, the better the prediction.

However, labeled instances are often difficult to obtain. It may also be time-consuming to produce them and, thus, expensive, since they require the efforts of an experienced person, e.g., if long texts have to be screened. There are service providers, such as *Amazon Mechanical Turk*[1], where requesters can have done simple tasks for a fee. But when it comes to more demanding tasks or sensitive data, the user probably wants to keep control of it. Therefore, it is a fundamental interest to accelerate the process of data labeling. A more effective method that labels a large amount of data in a short time can help to better classify future news articles and is of interest for general tasks in the field of supervised Machine Learning.

The idea is to develop a concept for interactive labeling that is able to adapt to changing requirements (in our case to adjust to the errors discovered) during the process in a similar way to online algorithms and thereby continuously improve itself.

[1] https://www.mturk.com

1.2. Goals

The goal of this thesis is to develop a Machine Learning model that recognizes company mergers in business news articles and to develop and investigate the suitability of an interactive human-computer data labeling strategy for this task.

This task consists of the following subtasks:

1. Providing and preparing an unlabeled data set
2. Pre-processing the text
3. Investigating the suitability of different labeling techniques
4. Developing a classification model

These subtasks are described as follows: First, an unlabeled data set of business news articles has to be obtained. The data should be as representative and complete as possible, which is why general business news over a time period of twelve months[2] are considered.

With the help of Natural Language Processing methods, the texts then have to be pre-processed for further proceedings.

On the basis of this practical example, the concept of interactive data labeling will be investigated using several experiments. By applying principles of human-computer interaction and Semi-Supervised Machine Learning, the aim is to optimize this process compared to strictly manual labeling. Our goal is to create a relatively large data set in order to improve the predictions.

To finally recognize company mergers, a classification model is developed that can distinguish news articles according to whether an article is about a merger or not. As the Accuracy of the model is essential, a comprehensive set of alternative models is explored and compared.

The follow-up question about the two merger partners and the exact date of completion of a merger (*"Which two companies merge and when?"*) can be solved by examining positively classified articles with Named Entity Recognition algorithms. The automatization of the identification of the details of a merger is not addressed in this thesis.

1.3. Outline

Chapter 2 provides background information about company mergers, Natural Language Processing and Machine Learning, that is essential for this work. We also explain the principles of interdisciplinary concepts that underlie our idea of interactive labeling. Chapter 3 presents the state of the art and related work to the topic. In Chapter 4, the design of the entire process and the corresponding evaluation is described. The underlying data set is explored more closely in Chapter 5 using visualizations. Chapter 6 shows some examples of the practical implementation, especially how the interactive part was implemented. In Chapter 7,

[2] The time period of the data set for the conventional labeling study is only one month.

the labeling techniques are evaluated in detail and a comprehensive evaluation of alternative Machine Learning models is conducted. This thesis concludes with a discussion summarizing all findings and commenting on future perspectives in Chapter 8.

In the next chapter, the background to the domain and the used tools and algorithms is provided.

2. Background

In this chapter, the related scientific fundamentals to the main topics of this work are introduced. In Section 2.1, background information on the topic of mergers and acquisitions and on the prerequisites for classifying news articles are given. Section 2.2 introduces interdisciplinary approaches that are related to the interactive labeling method. Section 2.3 provides an overview of the most important text analysis methods which are used for the data preprocessing. Important Machine Learning basics are described in Section 2.4. In Section 2.5, the Machine Learning algorithms used for classification are briefly explained. The last section also gives information about possible tuning options for algorithms and about the metrics used to evaluate the results.

2.1. Company Mergers

This section briefly introduces the basic terms from the field of company mergers and acquisitions.

Mergers and Acquisitions (M&A) is a generic term used to describe a range of possible corporate transactions. According to [BA07, 11], these include:

- *Mergers*, i.e., corporate mergers under the Transformation of Companies Act and asset transfers.
- *Acquisitions and Divestitures*, both as purchase or sale of shares in companies (*this is called Share Deal*) and individual assets (*Asset Deal*).
- From strategic alliances and *Joint Ventures* to outsourcing agreements for strategically relevant functions.

The following definitions in this section are taken from [Eng17, 4]:

In M&A transactions, the vendor is often referred to as the **seller** and the acquirer as the **buyer**. Other parties involved may be the object of purchase itself (in the Share Deal) and, under certain circumstances, the management or any guarantors.

A **Share Deal** is the acquisition of all or part of the shares in a company that operates a business. Through the share deal, all rights and obligations and thus all assets and liabilities of the acquired company are indirectly transferred to the buyer.

A **Joint Venture** is when two or more companies merge to form a joint venture company. This is usually done by way of a share deal by acquiring shares in an existing company or by founding a new company for this purpose.

Acquisition of control means the acquisition of more than 50 % of the shares of a company in the share deal outside listed companies. The term also includes the acquisition of less than 50 % of the shares of a company in the share deal in conjunction with contractually guaranteed reservations of consent or veto rights for the buyer (e.g., in a joint venture).

In the **Due Diligence** examination, the prospective buyer wants to examine the company to be acquired (the company in the share deal or the assets in the asset deal) as closely as possible.

Signing is the signing or notarization of the purchase contract. Whether or not a notarial certification is required depends on the object of purchase.

Closing refers to the stage at which the transaction is completed. Signing and closing often fall apart because a series of steps can only take place after signing: Payment of the sale price, transfer of the books of account and the actual control, etc. (the so-called closing conditions).

2.2. Interdisciplinary Concepts

The concept of the labeling process is based on several interdisciplinary approaches that are briefly explained in this section.

2.2.1. Human-Computer Interaction

A **human-computer** dialog consists of **interactions** between the user and the computer to achieve a specific goal. An interaction step consists of a single input for the dialog. The input and output interface includes all information representations and inputs. It thus determines the possible interaction steps [Hei04, 18-19].

2.2.2. Visual Analytics

Visual Analytics is an interdisciplinary approach that combines the advantages of human and automated analysis. It is used to examine large, complex data sets.

When it comes to the recognition of known patterns, the computer can make relatively clear decisions in shorter time and human working time is not absolutely necessary. But when we consider unexpected correlations, more diverse perception is needed as well as creativity or general knowledge in order to decide - as in our case - whether a merger is actually taking place or not. In that case, the human can make the better and more informed decision.

One goal of Visual Analytics is to combine these different skills as best as possible and find a better division of tasks. It helps the analyst to evaluate and correctly intermediate results faster and to improve methods and models [KPW14, 171-176].

2.3. Text Analysis

This section explains the basics of common text processing concepts that are used in this project.

2.3.1. Natural Language Processing (NLP)

NLP is a field of computer science that focuses on computers being able to understand language in a *natural* way, just like humans do. Typically this refers to tasks such as understanding the sentiment of text, speech recognition or text classification [Bey18, 1].

2.3.2. Pre-processing

In this part, the pre-processing steps Tokenization, the concept of ignoring Stop Words, and Stemming are explained. All these steps are necessary to prepare text for a valid input for Machine Learning algorithms.

Tokenization

Tokenization means segmenting a sentence into units of the word level [MM16], for example, the sentence

```
On Monday, Github and Microsoft announced their merger.
```

is fragmented into the tokens:

```
['On', 'Monday', 'Github', 'and', 'Microsoft', 'announced',
'their', 'merger'].
```

Ignoring Stop Words

Each language contains a variety of linking words, such as *one, like, the, these,* etc.. Although these words are important for humans, they are often not useful or even disturbing for text analysis. These often short words are called **Stop Words** and should be filtered out, so that only the important words are left [MM16]. For example, the sentence

```
Switzerland's Sunrise Communications Group has confirmed it is
    in talks over a potential acquisition of broadband provider
    Liberty Global's Swiss business UPC Schweiz.
```

contains the stop words

```
['has', 'it', 'is', 'in', 'over', 'a', 'of'].
```

The choice of Stop Words depends on the specific application and is implemented variously in practice.

Stemming

The process of reducing a word to its stem, also called root, is called **Stemming** [MM16]. It is about grouping similar words to be able to use them as synonyms for the same features in algorithms. It also helps to analyze sentences by tokenization, which we use for NER explained in Section 2.3.5.

Example:

```
German Economy Minister Peter Altmaier said on Tuesday that he
    did not favor getting ministerial approval for deals such as
    the proposal to merge Siemens and Alstom's rail businesses to
     better compete in Europe and abroad.
```

After applying Stemming the result is:

```
['german', 'economi', 'minist', 'peter', 'altmaier', 'said', 'on
    ', 'tuesday', 'that', 'he', 'did', 'not', 'favor', 'get', '
    ministeri', 'approv', 'for', 'deal', 'such', 'as', 'the', '
    propos', 'to', 'merg', 'siemen', 'and', 'alstom', 's', 'rail
    ', 'busi', 'to', 'better', 'compet', 'in', 'europ', 'and', '
    abroad']
```

2.3.3. Bag-of-Words Model (BOW)

Categorical data such as text or words must first be converted to numerical data before we can pass them to learning algorithms for further processing. The **BOW** model [Mur12] is a simple approach to describe text documents. Each document is represented by a binary *feature vector*, which records the number of occurrences of each word. The length of all vectors corresponds to the number of different terms in all documents. Every entry x_{ij} describes the number of occurrences of terms j in document i.

Absolute and Relative Word Frequency

Depending on the task, either the **absolute word frequency** (the absolute number of occurrences of one specific term in a document) or the **relative word frequency** (the absolute number of occurrences of one specific term in a document divided by the total number of occurrences of all terms in all documents) is being used. The absolute number of occurrences may thus range from 0 to infinity, while the relative frequency represents a probability and can only take values between 0 and 1.

Document Term Matrix

When applying the BOW model to a text corpus, the summary of all feature vectors is called the **document term matrix** or also *feature matrix*.

The following example uses three text documents; *doc1*, *doc2*, and *doc3*.

```
doc1: "The sun is shining."
doc2: "The weather is sweet."
doc3: "The sun is shining and the weather is sweet."
```

Using absolute term frequencies, the corresponding document term matrix is shown in Table 2.1. To calculate the relative term frequencies, the number of occurrences of each word is divided by the number of words in all documents (17 words in total). The corresponding document term matrix is shown in Table 2.2.

	the	sun	is	shining	weather	sweet	and
doc1	1	1	1	1	0	0	0
doc2	1	0	1	0	1	1	0
doc3	2	1	2	1	1	1	1

Table 2.1.: Document Term Matrix, Absolute term frequencies

	the	sun	is	shining	weather	sweet	and
doc1	0.059	0.059	0.059	0.059	0	0	0
doc2	0.059	0	0.059	0	0.059	0.059	0
doc3	0.118	0.059	0.118	0.059	0.059	0.059	0.059

Table 2.2.: Document Term Matrix, Relative term frequencies

2.3.4. N-Gram

Related text fragments or words are called **N-grams** [Ras17, 241]. The text fragments created in the BOW model by decomposing the text are called *unigrams* (n=1), because each token of the vocabulary represents a single word. The *bigram* (n=2) representation of a text always considers two sequential words. In this work, the unigram model is mainly used.

2.3.5. Named Entity Recognition (NER)

NER [FGM05] labels sequences of words in a text which are terms, such as person and company names, or dates and locations.

Each word (and punctuation mark) is assigned a label, for example

```
On Monday, Github and Microsoft announced their merger.
```

is mapped as follows:

```
[('On', 'O'), ('Monday', 'O'), (',', 'O'),
('Github', 'ORGANIZATION'), ('and', 'O'),
('Microsoft', 'ORGANIZATION'), ('announced', 'O'),
('their', 'O'), ('merger', 'O'), ('.', 'O')]
```

It is possible to use a pre-trained model or your own name training set, depending on how specific the search is. For this project, we use a pre-trained model and perform the customizations necessary for our problem only afterwards.

2.4. Machine Learning

In the following, we introduce the Machine Learning basics that are essential to understand the proceeding of the next chapters.

2.4.1. Supervised, Semi-Supervised Learning

If a data set shows the mapping of instances to a certain class using labels, **Supervised Learning** [Mar09, 7-9] can be applied. The algorithm learns based on a training set and generalizes to respond correctly to all possible future inputs.

Semi-Supervised Learning [Zhu05] is a special form of classification. Traditional classifiers only use labeled data (feature / label pairs) for training. Meanwhile unlabeled data may be relatively easy to collect, but there have been few approaches to use them. Semi-Supervised Learning addresses this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers. More precisely, already labeled data can be used to label more of the unlabeled data. This is repeated until all data is labeled and only a supervised learning problem remains. The advantage of this approach is, that it requires less human effort than strictly Supervised Learning.

2.4.2. Binary, Multi-Class Classification

An algorithm learns a mapping from each input to a specific class. If there are exactly two possible classes, this is called **binary classification**. If there are more than two classes, this is called **multi-class classification** [Mur12, 3]. In this work, we are using both concepts.

2.4.3. Balanced, Imbalanced Data Sets

If there is an equal number of instances of each class in a data set, it is called a **balanced data set**. Otherwise, the data set is called **imbalanced**. Dealing with imbalanced data sets leads to several challenges in building and evaluating the prediction model. By using *stratified sampling* (see Section 2.4.4) the imbalance of the individual classes can be corrected though. However, this may lead to the situation that only a small subset of the available data can be used. As only a very small proportion of the examined business news items consists of articles on the subject of mergers, the data set used in this work is imbalanced.

2.4.4. Training, Testing, Validation Set

In order to build a Machine Learning model for supervised learning, a labeled data set is needed first. The initial data set is then divided into a training, testing and (optionally) a validation set as illustrated in Figure 2.1.

Figure 2.1.: Training, validation and testing set

With the help of the **training data**, the model is fitted. The **testing data** must not be used in training or for manual optimization of the Hyper-Parameters. They are used to test the finished model against unknown data with which it was not trained. This allows an unbiased evaluation of a final model that fits the training data set.

To be able to check whether the quality of the results is improving during the training and optimization process, another part of the data can be reserved as **validation data** [Mar09, 66-67]. This allows an unbiased valuation of a model that fits the training data set while manually tuning the Hyper-Parameters of the model. However, the evaluation becomes more biased as the knowledge about the validation data set influences the model configuration.

Using a validation dataset also reduces the amount of data available for training and may make the training process vulnerable to random selection of validation data.

After evaluation, all data can finally be used to learn the final classifier.

There are several ways to divide a data set into training, test and (optionally) validation set. Below we briefly present some common practical techniques.

Test-Train-Split

This is the simplest division variant. The data is divided into a training and a testing set. The proportion of training to testing data is not specified, but the convention is to split the data set into 75:25 for a very large amount of data and to split it into 60:40 for a smaller one [Mar09, 67].

Shuffle Split

This random permutation Cross-Validation generates indices to split data into training and testing sets. In contrast to other Cross-Validation strategies, random splits do not guarantee that all folds will be different, although this is still very likely for sizable data sets [Sci18b].

Stratified Split

This split method uses stratified folds, which is especially important for unbalanced data sets [Sci18c]. The folds are derived by preserving the percentage of samples for each class.

To validate a model, there are different validation methods we briefly explain below.

(K-fold) Cross-Validation

This popular method for estimating the generalization error can be divided into **Two-Fold** ($k = 2$) and **K-Fold Cross-Validation**. The Two-Fold approach is also referred to as the *Holdout Method*, where the data set is divided once into training and test data. As mentioned before, the former is used to train the model, the latter is used to evaluate the model performance.

To improve generalization, the K-Fold Cross-Validation divides the training data set into k subsets (without replacement). Then $k - 1$ subsets are used to train the model and the last subset is used for testing. This procedure is repeated k times so that we can obtain k models and k estimates of performance and calculate the average from them. In practice, it is common to use five to ten folds [Mar09, 178-181]. In my implementation, I mainly use Ten-Fold Cross-Validation.

2.5. Classification Algorithms

This section briefly introduces the Machine Learning algorithms that are applied to the classification problem.

2.5.1. Naive Bayes

The **Naive Bayes** classifier [Zha04] is a simple classifier, that is popular in NLP. It is one of the most efficient and effective inductive learning algorithms for Machine Learning and Data Mining.

The simplest approach is to assume that every pair of features is conditionally independent given the class label. That is why the classifier is called *naive*.

Given the class variable y and dependent feature vector $x = (x_1, \ldots, x_n)$ with the number of features n, Bayes' theorem states the following relationship:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption that for all i

$$P(x_1|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

this relationship can be simplified to

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, the following classification rule can be used:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{n} P(x_i|y)$$

and *Maximum A Posteriori (MAP)* can be used to estimate $P(y)$ and $P(x_i|y)$; where $P(y)$ is then the relative frequency of class y in the training set.

Multinomial Naive Bayes

This variant of Naive Bayes algorithm for multinomially distributed data that is also often used in text classification. The distribution is parametrized by vectors $\Theta_y = (\Theta_{y1}, \ldots, \Theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and Θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y. The parameter Θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\Theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

The smoothing prior $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations [Sci18a].

2.5.2. Support Vector Machine (SVM)

The **SVM** [Mar09, 119] algorithm is one of the most popular algorithms in Machine Learning. It can be used for classification and regression and performs well on reasonably sized data sets. The main idea is to modify the data by changing its representation.

Using binary classified training data, a model is created that assigns new samples to one category or another, making them a non-probabilistic binary linear classifier. An SVM model is a representation of the samples as points in space that are mapped in such a way that the examples of the individual classes are separated by as large a distance as possible, called *margin*. New samples are then mapped into the same space and are predicted to belong to a class based on which side of the gap they fall on. SVMs can efficiently perform non-linear classification using a *kernel trick* by implicitly mapping their inputs into high-dimensional feature spaces.

2.5.3. Tuning Options

The parameters of the algorithms presented above and also the model fitting process can partly be adapted in order to achieve better results. In this subsection, we introduce some methods to improve the algorithms' performance.

Hyper-Parameters

Parameters of a learning algorithm that can be modified and optimized to improve performance are called **Hyper-Parameters**. The process of selecting appropriate values for the adjustable parameters of a given classification task is known as *Model Selection* [Ras17, 33, 81].

Feature Selection

Assuming that each instance of a training set is represented by a vector of d features, **Feature Selection** means to learn a predictor that only relies on k with $k << d$ features. Predictors that use only a small subset of features require a smaller memory footprint and can be applied faster. Finally, constraining the hypothesis class to use a small subset of features can reduce its Error Rate and thus prevent Overfitting. Ideally, all subsets of k out of d features must be tried in order to choose the subset, which leads to the best performing predictor. In practice, as such a search is too exhaustive, the established methods cannot guarantee finding the best subset, but they often work reasonably well [SSBD14, 310].

2.5.4. Evaluation Metrics

Metrics are used to measure the quality of a classification and help us to select the best possible classification algorithm, taking into account our individual criteria. They are built from a Confusion Matrix which records correctly and incorrectly recognized examples for each class. Table 2.3 presents a Confusion Matrix for binary classification, where `tp` are *true positive*, `fp` *false positive*, `fn` *false negative*, and `tn` *true negative* [SJS06].

Class / Recognized as	positive	negative
positive	true positive (tp)	false negative (fn)
negative	false positive (fp)	true negative (tn)

Table 2.3.: Confusion Matrix for Binary Classification

Accuracy

The most used empirical measure is **Accuracy** [SJS06] which is calculated as follows:

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn}$$

One should keep in mind, however, that Accuracy can be very misleading with imbalanced data sets. Therefore we focus on the metrics Recall and Precision in this work.

Precision, Recall, F1-score

Within a set of classes, there is a class of special interest, usually the positive labeled objects. Other classes either remain unchanged (multi-class classification) or are combined into one (binary classification). The measures of choice calculated on the positive class are:

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn} = Sensitivity$$

and

$$F1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

also known as f-measure. All three measures distinguish the correct classification of labels within different classes.

Recall is a function of *true positives* and *false negatives* (*"How many relevant items are selected?"*), while **Precision** is a function of *true positives* and *false positives* (*"How many selected items are relevant?"*). The **F1-score** combines the two metrics Recall and Precision.

Error Rate

The **Error Rate** [Ras17, 195] or also called *estimation error* is the number of incorrectly classified examples per total number of classified examples and takes a value between 0 and 1, where 0 denotes absence of Error and 1 denotes the maximum Error Rate.

$$Error\ Rate = \frac{fp + fn}{fp + fn + tp + tn}$$

Resubstitution Error

When training and testing using the same data set, the number of incorrectly classified examples is called **Resubstitution Error**. It is used to show whether the data used is ambivalent or incorrect, or whether an inappropriate method was used.

Variance, Bias, Overfit, Underfit

Variance is a measure of the consistency (or variability) of the model prediction for a particular object when we train the model more than once, for example with different subsets of the training data set. This allows us to see whether the model is sensitive to the randomness of the training data.

The **Bias**, on the other hand, indicates how much the predictions deviate from the correct values if we apply the model several times to different training data sets. It is a measure of the *Systematic Error* that is not due to randomness.

Sometimes, the model works well as long as training data is used because it memorizes the features of the training data. Unknown data (testing data), on the other hand, can hardly be handled because generalization is not successful.

If a model suffers from such a problem called **Overfit**, one also speaks of a large *Variance*, which is often caused by the fact that there are too many parameters that lead to a model that is too complex for the underlying data.

Similarly, a model may suffer from **Underfit** (large *Bias*, high tendency to make certain decisions) if it is not complex enough to recognize the patterns in the training data and therefore also fails to work properly when applied to unknown data [Ras17, 81-82].

Summary:
In this chapter, the difference between company mergers and other transactions was explained. The reader is now familiar with the interdisciplinary approaches that are referred to in this work, Human Computer Interaction and Visual Analytics. There was also a short introduction to text processing and Machine Learning, which helps to get deeper into the topic. In Chapter 3, the State of the Art in the field of Semi-Supervised Machine Learning and other related work is presented.

3. State of the Art and Related Work

This chapter presents the State of the Art in Semi-Supervised Machine Learning and similar approaches that are related to this work. As an overview of current research, algorithms that are able to improve themselves are discussed in Section 3.1, then the functionality of weakly supervised algorithms is explained in Section 3.2 and lastly, the concept of algorithms that use decision boundaries is introduced in Section 3.3.

A number of scientists are researching optimized usage of class labels in Machine Learning which illustrates the relevance of this subject. All studies presented are dealing with the problem that large data sets are difficult to completely be labeled manually and they all aim at improving the classifier. However, these works differ in focus and execution and can be categorized into:

a) Self-improving Algorithms

b) Weakly Supervised Algorithms

c) Active Learning Algorithms

The basics of these approaches and how they differ from our work is explained below.

3.1. Self-Improving Algorithms

These types of algorithms improve their performance with the help of internal calculations and perform iteratively until an optimal result has been achieved. There is no human intervention intended during the process.

3.1.1. Expectation-Maximization

Nigam, McCallum, Thrun and Mitchell propose a text classification from labeled and unlabeled documents using Expectation-Maximization and a Naive Bayes classifier [NMTM00]. It shows, that the Accuracy of learned text classifiers can be improved by augmenting a small number of labeled training documents with a large pool of unlabeled documents. The algorithm first trains a classifier using the available labeled documents, and probabilistically labels the unlabeled documents. It then trains a new classifier using the labels for all documents, and iterates to convergence.

Differentiation from our work: The idea of reducing the effort for manual labeling by using the class probabilities of the Naive Bayes algorithm is similar to our motivation. The procedure differs from ours by the fact that no further instances are labeled by the user after the first step. The algorithm operates independently during the following iterations instead.

3.2. Weakly Supervised Algorithms

Weakly Supervised Learning algorithms automatically use the unlabeled data in addition to the labeled data to improve the classifier. No human intervention is intended during the process.

3.2.1. Label Propagation

Zhu and Ghahrarami [ZG02] investigate the use of unlabeled data to better classify labeled data. They propose a simple iterative algorithm, called Label Propagation, to propagate labels through the data set along high density areas defined by unlabeled data. A constructed graph, where each document is represented by a node, represents similarities among documents with its edge weights.

Pawar, Ramrakhiyani, Hingmire and Palshikar [PRHP16] also propose an algorithm for weakly supervised text classification based on Zhu's and Ghahrarami's Label Propagation algorithm. Additionally, they discover underlying topics using *Latent Dirichlet Allocation (LDA)* and enrich the document graph by including the topics in the form of additional nodes. The edge weights between a topic and a text document represent the level of *affinity* between them. Their approach does not require document level labeling, instead it expects manual labels only for topic nodes. This minimizes the level of supervision needed as only a few topics are observed to be enough for achieving sufficiently high Accuracy. The Label Propagation algorithm is employed on this enriched graph to propagate labels among the nodes. The approach combines the advantages of Label Propagation (through document-document similarities) and Topic Modeling (for minimal but smart supervision). They demonstrate the effectiveness of this approach on various data sets and compare it with other weakly supervised text classification approaches.

3.2.2. Cluster Assumption

In their paper, Zhou, Bousquet, Lal, Weston and Schölkopf [ZBL+04] present a solution, that is similar to the basic Label Propagation algorithm. Also they consider the general problem of learning from labeled and unlabeled data. The principle is based on the assumption of consistency, which means that similar feature vectors are highly likely to have the same label. Their method provides experimental results on a number of classification problems and shows the effective use of unlabeled data.

Differentiation from our work: Their concept focuses on the usefulness of unlabeled data in model building. The goal is not to label the unlabeled instances, but only to improve the

classifier.

3.3. Active Learning Algorithms

Active Machine Learning algorithms are a special supervised Machine Learning technique. They are used when large numbers of unlabeled samples are available and getting labels for them is costly (e.g., requiring consulting a human expert).

The Active Learning process takes as input a set of labeled examples, as well as a larger set of unlabeled examples, and produces a classifier and a relatively small set of newly labeled data. The overall goal is to create a classifier as good as possible, without having to supply the learner with more data than necessary. The learning process aims at keeping the human labeling effort to a minimum, only asking for advice where the training utility of the result of such a query is high. Some Active Learning algorithms focus on refining the decision boundary through exploring new regions which the current hypothesis classifies incorrectly.

3.3.1. Pre-Selection for Manual Labeling using Decision Boundaries

Osugi, Kun and Scott [OKS05] propose an Active Learning algorithm that balances such exploration with refining of the decision boundary by dynamically adjusting the probability to explore at each step. Their experimental results demonstrate improved performance on data sets that require extensive exploration while remaining competitive on data sets that do not.

Fredrik Olsson's report [Ols09] is a literature survey of Active Learning from the perspective of NLP. He describes Active Learning as a supervised Machine Learning technique where the algorithm controls the data used for learning. A human with extensive knowledge of the respective domain is then asked about the classes of instances for which the model makes unreliable predictions so far. The Active Learning process takes a series of labeled samples and a larger number of unlabeled samples as input and generates a classifier and a relatively small amount of newly labeled data. The overall goal is to create a classifier as good as possible without providing more data than necessary. The learning process aims to reduce the human annotation effort to a minimum and only asks for human advice if the training benefit of the result of such a query is high.

Tong and Koller [TK01] concentrate on the Machine Learning algorithm for support vector machines. Instead of using a randomly selected training set of labeled data, the learner has access to a pool of unlabeled instances and can request the labels for a number of them. They introduce a new algorithm for performing active learning with support vector machines, i.e., an algorithm for choosing which instances to request next.

Differentiation from our work: These concepts use decision boundaries to pre-select instances to be labeled by the user, which is a similar approach we use in this work. However, they do not involve human-computer interaction, which distinguishes these projects from our work.

Summary:
The research presented in this chapter is concerned with how to improve a classifier using unlabeled data. However, unlabeled instances are not actually labeled during the process. Instead, they are used for internal calculations to improve the classifier. The aspect of human-computer interaction also plays an important role in our work, in which the user is involved during the entire process. This concept distinguishes our research from the others. In the next chapter the design of this project is explained in detail.

4. Design

In this chapter, the design of this project is presented by describing the individual processing steps to recognize mergers in news articles. In Section 4.1, a generic strategy is presented which contains the most important steps to solve the problem. Then two different concepts are distinguished. We present a conventional concept in Section 4.2 and an interactive concept in Section 4.3 and explain these in detail. The two examined variants of data labeling and processing are compared to each other discussing strengths and weaknesses.

4.1. Overview

At this point, an overview is given of how the entire project is structured. To detect company mergers in news articles, a number of steps are necessary, as shown in Figure 4.1.

Figure 4.1.: Data processing steps

The individual steps consist of:

- Data Download: After identifying an accessible source for business news articles, an appropriate data set must be downloaded.
- Data Cleaning: The downloaded data must be cleaned in order to remove disruptive information.
- Data Selection: An appropriate subset of all cleaned data must be selected.
- Data Labeling: Each instance of the data record needs to be assigned to a class. First, an experiment with the conventional labeling approach is performed evaluating its advantages and disadvantages. Secondly, the ideas of *Visual Analytics* and *Human Computer Interaction* are transferred to the topic of data labeling and an innovative strategy will be developed. We perform an experiment with the interactive labeling approach in order to compare both methods regarding time effort and number of instances that have to be labeled manually.
- Data Pre-Processing: In order to make the texts of the individual articles usable for machine learning algorithms, they must be prepared appropriately.

- Model Selection: A Machine Learning Model is trained on the labeled data set to make class predictions for future news articles.
- Recognition of Mergers: The goal of this work is to recognize articles that deal with the merger of two companies.

The description of the data processing is divided into two separate sections; Section 4.2 for the conventional and Section 4.3 for the interactive labeling method.

4.2. Conventional Method

This section describes how the conventional data processing method is applied to the data set downloaded in Section 4.2.2. After an overview in the following section, the individual steps are described in more detail.

4.2.1. Data Processing Pipeline

Figure 4.2[1] shows all individual steps of data processing when applying conventional manual labeling to the data set. This processing method may already be known to the reader as the typical procedure when it comes to classification problems. It can be described as *top-down* or *waterfall* method, since the individual steps are executed one after another.

After data downloading and data cleaning, a comprehensive data set for model building is assembled by selecting specific instances. Then, all of these instances are labeled at once in a single step, which is the most significant characteristic of the conventional data processing variant. Having pre-processed the textual part of the data, a Machine Learning model can be fitted. Optionally, at this point, the wrongly predicted instances are examined and revised. Next, the model can be improved using new hypotheses from the newly obtained knowledge during model selection. This last step of testing and tuning is repeated until the best possible result is achieved. In the final step, the identified articles about company mergers are presented to the reader.

In the following sections, the individual processing steps of conventional data processing are described in detail.

4.2.2. Data Download

To start the process, appropriate data must be identified and downloaded. As a source for news articles, RSS feeds from established business news agencies such as *Reuters* or *Bloomberg* are considered. However, when crawling RSS feeds, it is only possible to retrieve the current news and not the news from the past. Since we want to analyze news retrospectively, the data set is obtained from the provider *webhose.io*[2]. It offers access to English news articles

[1] For a better understanding, processing steps which are identical or similar in both methods are colored blue in the diagram. The steps that are different in each method are colored red. Optional steps are indicated by a dashed line.

[2] `https://webhose.io`

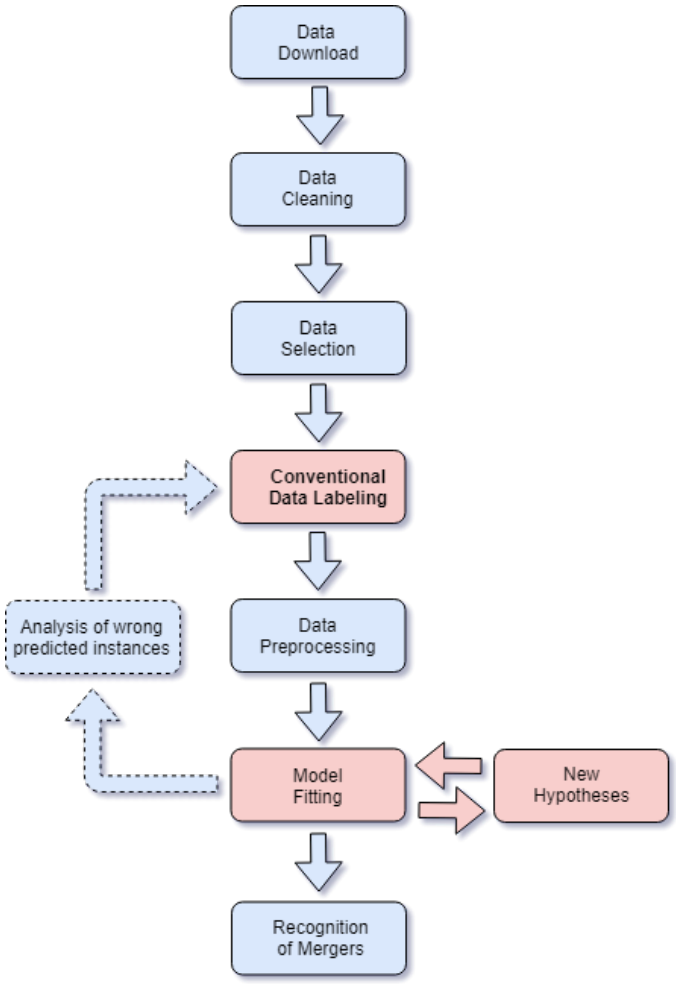

Figure 4.2.: Data processing pipeline for conventional labeling

from sections like *Financial News*, *Finance* and *Business* at affordable fees compared to news agencies' offers. As only reliable sources are of interest, the request is limited to the websites of the news agencies *Reuters, Bloomberg, Financial Times, CNN, The Economist* and *The Guardian*. With this provider, news of a time period of the last 30 days can be downloaded for free as a stream of JSON files via an API, which is sufficient to execute the conventional labeling strategy. In order to make a pre-selection and increase the ratio of news about mergers, the request includes only articles that contain at least one of the keywords `merger`, `acquisition`, `take over`, `deal`, `transaction` or `buy` in the header. Furthermore, the request is restricted to original contributions in order to exclude user comments and the like[3]. For further processing, the JSON files' relevant information is stored as a CSV file.

4.2.3. Data Cleaning

At this point, it must be ensured that all news articles are standardized, data errors are removed or corrected, and no duplicates or incomplete data are contained.

4.2.4. Data Selection

Since the query is already restricted by filtering settings (see Section 4.2.2), the complete downloaded data set is selected for further proceedings.

The resulting CSV file consists of the five columns:

Timestamp	Title	Text	SiteSection	Label

- **Timestamp:** The thread's publishing date and time in the format YYYY-MM-DDThh:mm (GMT+3).
- **Title:** The news article's headline.
- **Text:** The news article's plain text.
- **SiteSection:** The link to the section of the site where the thread was created.
- **Label:** The article's class label is entered here later.

The columns *Title* and *Text* contain the main data, whereas the remaining attributes contain the meta data.

4.2.5. Conventional Data Labeling

As the focus of this thesis is on corporate mergers (see Section 2.1), binary classification is used as follows: An article is defined as *positive* (also called hit article, assigned to Class 1), if it deals with a corporate merger of two companies and if it will definitely take place or has recently taken place. We are not interested in mergers in the past, nor in negotiations, planning or speculation about corporate mergers. For these special cases and all other unrelated news we mark them as Class 0, what corresponds *negative*. The two classes are:

[3] See `https://docs.webhose.io/docs/filters-reference` for more filter settings.

Class 0: other/unrelated news

Class 1 (merger): merger of companies A and B

The CSV file is then exported to *Microsoft Excel* so that the relevant text columns are clearly displayed next to the label column. Now the procedure is as follows: If it is already clear from the title of an article that it is an irrelevant article, the article is labeled directly with 0 without reading the full text. If we assume a hit article, the entire article text is read and afterwards the correct label inserted. During the labeling procedure, the time needed is measured for the evaluation.

4.2.6. Data Pre-Processing

In order to use the news articles for Machine Learning algorithms, we must first prepare the cleaned data appropriately. The aim is to transform the texts as document term matrices so that classification algorithms can be applied.

For pre-processing, there's not just one right solution. It rather depends on the individual case how it should be implemented. We are guided by the implementation of *scikit learn's CountVectorizer*, as it marks an unofficial standard in the field of NLP and is widely used. According to our problem, we commit to the following sub-steps:

- **Removing punctuation marks:** Punctuation marks are replaced with white spaces.
- **Tokenization:** Each news article is split into a list of single words.
- **Ignoring numbers:** All numbers in the news article are left out, as they are irrelevant to our classification problem.
- **Ignoring company names:** We leave all company names in the news articles (see Section 7.3), as they can lead to wrong decisions during training.
- **Transforming words to lower case:** Every word is transformed to lower case.
- **Word Stemming:** Every word is reduced to its word stem.
- **Ignoring Stop Words:** We filter out common words, so that they do not interfere with our calculations.

4.2.7. Model Fitting

Once the entire data set has been labeled, a Machine Learning model is built using K-Fold Cross-Validation. With stratified sampling, the stratified folds are selected by preserving the percentage of samples for each class.

Now, a suitable algorithm is searched for and Hyper-Parameters are optimized in several steps. As classification algorithms, *scikit learn*'s Gaussian Naive Bayes and SVM are appropriate for this kind of NLP problems. When using Naive Bayes, we assume that the probability of

a word is independent of its position within the document; thus, for example, the probability of seeing the word *merger* in the first sentence of an article is the same as seeing it in any other position. Since a headline is actually more important than the last sentence, for example, one could include this observation in our model. As a human, we do this unconsciously by assigning greater importance to the headline, but our calculation model currently does not take this into account.

As soon as the best model is identified, the process is completed.

4.2.8. Creating New Hypotheses

New hypotheses in this model are limited to the step of model fitting. The different parameters and variants of the algorithms are tested as described above in order to find the best algorithm and parameter combination.

4.2.9. Analysis of Wrongly Predicted Instances

To investigate the incorrectly predicted articles, the Resubstitution Error is calculated. All manually labeled data is used as training data set and the resulting model is applied to exactly the same data set (as testing data set). This method allows to identify ambivalent data and see if the user mistakenly labeled an instance with the wrong class. Then, if necessary, these examined labels are corrected. The Resubstitution Error is also an indication of how well the model already works (but it is subject to Overfitting).

In general, however, this only affects a very small part of the overall data. Besides, this step is often neglected in practice.

4.2.10. Recognition of Company Mergers

The articles predicted by the classification algorithm as Class 1 (merger) are those in which a merger of two companies is reported. This is the last step in our pipeline. In future work, the merger partners and the time of contract conclusion could be extracted from these articles with the help of NER.

4.3. Interactive Method

In the following, the interactive labeling method is presented in the context of data processing. We explain to what extent this method differs from the conventional one and discuss its strengths and weaknesses.

4.3.1. Data Processing Pipeline

In data processing with interactive labeling, the first three steps of the pipeline are similar to those described in Figure 4.2.

However, after downloading, cleaning and selecting the data, this variant differs significantly from the previously described concept. In contrary to labeling all articles at once, the labeling

process is done incrementally in separate iterations. As shown in Figure 4.3, the steps of each iteration are: Labeling of a determined number of instances, pre-processing of the data, model fitting, analysis of wrongly predicted instances and/or creating new hypotheses.

The last step is identical to the previous pipeline: As soon as the best model has been found, it can be used to identify the searched articles about company mergers. All individual processing steps are described in more detail below.

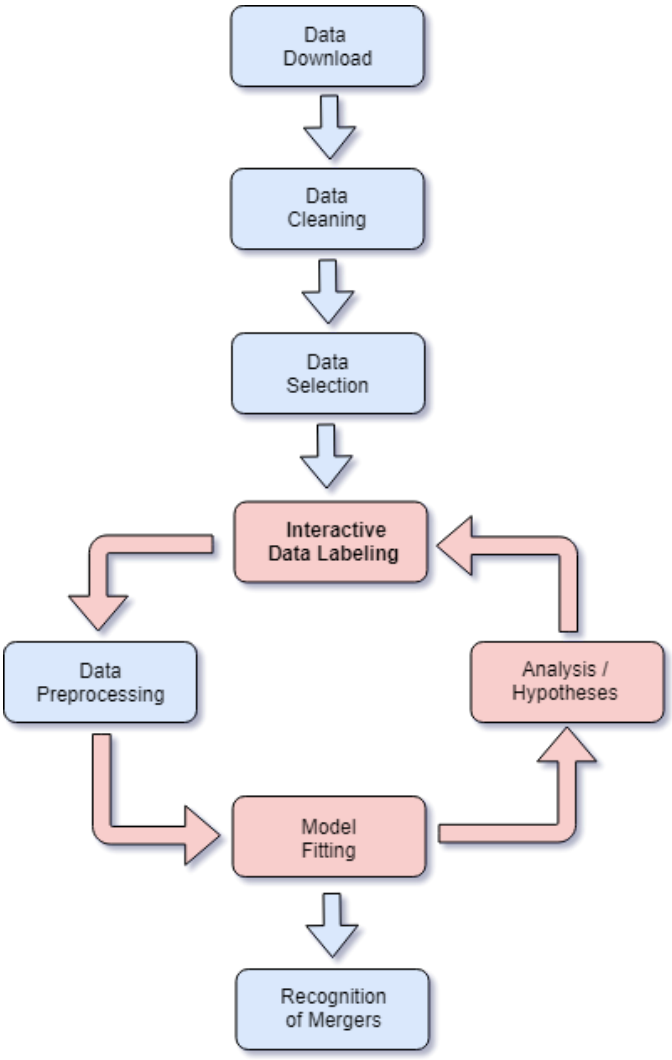

Figure 4.3.: Data processing pipeline for interactive labeling

4.3.2. Data Download

To investigate the interactive labeling strategy, a larger data set than before is needed to perform a number of iterations, why a period of twelve months of the year 2017 is considered.

This time, the requested sources are extended to *Reuters, Bloomberg, Financial Times, CNN* and *The Economist* to get a more diverse result. The request is sent to the *webhose.io* historical archive.[4] The data is then downloaded via a link; each news article as a single JSON file. The exact query can be found in Section 6.3.1.

4.3.3. Data Cleaning

Data Filtering: As *webhose.io* is a secondary source for news articles and only crawls the news feeds itself, it may occur that some RSS feeds are not parsed correctly or an article is tagged with a wrong topic as *site categories*. Downloaded files may also contain blog entries, user comments, videos, graphical content and other spam, or even news in other languages than English, which all should be filtered out.

Removing Duplicates: Articles that are only quoting or copying an original article, as well as duplicate articles with exactly the same headline should be removed from the data set.

4.3.4. Data Selection

After filtering out all the irrelevant data as described in the last section, the data is stored in multiple CSV files, one for each month, with a different number of articles in each case. Because the initial working data set's articles should be evenly distributed throughout the year, the same number of articles from each month is selected. The resulting CSV file consists of the following eight columns:

Uuid	Title	Text	Site	SiteSection	Url	Timestamp	Label

- **Uuid:** Universally unique identifier, representing the article's thread.
- **Title:** The news article's headline.
- **Text:** The article's plain text.
- **Site:** The top level domain of the article's site.
- **SiteSection:** The link to the section of the site where the thread was created.
- **Url:** The link to the top of the article's thread.
- **Timestamp:** The thread's publishing date and time in the format YYYY-MM-DDThh:mm (GMT+3).
- **Label:** The article's class label is entered here later.

The columns *Title* and *Text* contain the main data, whereas the remaining attributes denote the meta data.

[4] The requested data was downloaded in September 2018 with JSON as file format. Access to the historical archive is obtainable for a fee.

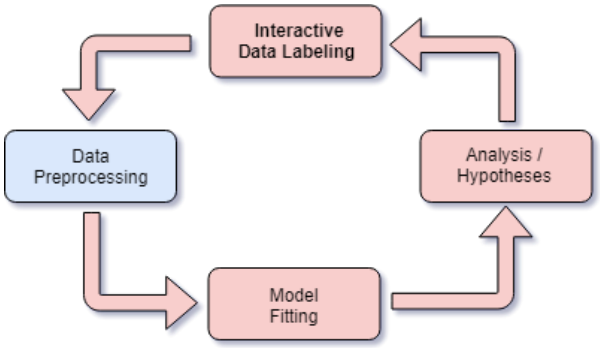

Figure 4.4.: Labeling iterations

4.3.5. Interactive Data Labeling

Data labeling is the core of the process and makes the difference between the two presented methods. This time, the ideas of *Visual Analytics* and *Human Computer Interaction*, described in Section 2.2, are used as a hybrid approach to the labeling problem. The computer should automatically label those news articles which correspond to already known patterns and therefore can be clearly categorized. The human only should examine those news articles whose patterns are not known yet and which therefore require human evaluation.

The time saved can be used to label a larger data set than with conventional labeling. A larger labeled data set, in turn, can improve the classification model and produce better predictions.

To annotate important information during the labeling process, the file's structure is extended by the four additional columns:

Index	Round	Probability	Estimated

- **Index:** This column makes it easier to address the individual instances of news articles.
- **Round:** Here the iteration number in which the label was entered manually is specified.
- **Probability:** After applying the classification algorithm, the estimated probability of the most likely class of each article is saved in this column.
- **Estimated:** The class estimated by the algorithm in the current round is annotated here.

The distinguished iteration part of the interactive labeling pipeline is shown in Figure 4.4.

Each iteration proceeds as follows: The loop starts by human labeling of a certain number of instances.

This time, we use a multi-class model since it is able to distinguish more detailed information about different types of news texts. This allows us to differentiate between irrelevant articles, e.g., about staff turnover, and those related to mergers, e.g., dealing with merger negotiations.

In order to apply multi-class classification, at least the following three classes are necessary:

Class 0: other/unrelated news

Class 1 (merger): merger of companies A and B

Class 2: news about deals/mergers/etc.

More classes are possible.

The advantages of using fewer classes are that the user has to remember less information, the assignment to a class is faster, and that in the case of an unbalanced data set, fewer classes have to be taken into account for stratified sampling. Whereas the use of many classes could possibly improve the quality of the prediction, because the individual articles can be assigned more precisely. Which exact number of classes is most useful in this context is evaluated in Section 7.4.2.

After the step of manual labeling, the data is pre-processed in order to build a classification model. As classifier, we use the Multinomial Naive Bayes algorithm by default. This algorithm is itself based on probabilities, so these are considered as useful. The manually labeled articles are used as training data, the remaining unlabeled articles are used as testing data. Regarding the testing data, the estimated class for each article and the estimated class probability are stored in columns *Estimated* and *Probability*. Now, the model investigated so that the findings can be included in the next round and the process improved. At this point, the Resubstitution Error should also be calculated in order to detect any incorrectly labeled instances.

From the second iteration on, the model's label estimations are checked by the user. Which instances should be examined exactly, is evaluated experimentally with different concepts:

a) The articles that could not be clearly assigned should be checked first, i.e., those articles which have a relatively low estimated probability for their class.

b) Articles with a high estimated class probability are checked first.

c) Separate binary models - one per class - estimate the unlabeled articles. Where inconsistencies can be observed, the proposed labels are checked.

After manually validating a suitable number of instances, a new model with all so far labeled instances is built and so forth. This is repeated until a sufficient number of labels is obtained and the model gives a reasonable estimate for the unlabeled instances.

4.3.6. Data Pre-Processing

Pre-processing is implemented in the same way as with the conventional method described in Section 4.2.6.

4.3.7. Model Fitting

In contrary to the conventional approach, a model is fitted after each iteration which is intended to estimate the unlabeled articles. Since we are in the state of labeling and therefore only have limited time available for the model fitting step, we will limit ourselves to the algorithms Multinomial Naive Bayes and SVM.

In order to find the best possible model based on the classified data after completing the labeling process, more variants are applied and evaluated for Recall and Precision. Regarding the final model, the best possible Recall score is identified, as it is important not to miss any news about mergers. For this purpose, the algorithms Multinomial Naive Bayes and SVM are investigated more closely.

Naive Bayes

As parameters, we use the default settings: Prior probabilities of the classes are adjusted according to the data (`priors = None`) and the portion of the largest variance of all features that is added to variances for calculation stability is set to the minimum (`var_smoothing = 1e-9`).

SVM

A grid search is executed to find the best parameters. We use the default parameters for the SVC implementation but use the settings `C=[1.0, 0.1]`, `kernel=['linear']` and `gamma=[0.00001, 0.0001]`.

Furthermore, the intention is to find out if CountVectorizer, my own BOW implementation (which is implemented similar to CountVectorizer) or Word2Vec is best suited as numeric representation of the texts. In addition, it will be evaluated whether the classification model using bigrams produces better results than using unigrams.

4.3.8. Creating New Hypotheses

In contrast to the conventional pipeline, the step of creating new hypotheses is placed in a wider cycle as Figure 4.4 demonstrates. It is not only during model fitting that new hypotheses can be formulated and verified. New knowledge can already be incorporated during labeling and thus have a greater influence on the overall process.

4.3.9. Analysis of Wrongly Predicted Instances

The recognition of wrongly predicted instances is also essential in this design and therefore firmly embedded in the cycle. It allows errors to be detected earlier and thus the continuous improvement of the model.

4.3.10. Recognition of Company Mergers

The step of recognizing articles about company mergers is done in the same way as the conventional method described in Section 4.2.10.

Summary:
This chapter explained the two approaches in theory and highlights the differences. The influence of the wider loop within the pipeline on the overall process was shown and that the classification model can improve continuously through the incremental technique. In Chapter 5 we explore the data set and illustrate important aspects with the help of visualizations.

5. Data Exploration

This chapter explores the textual corpus of news articles and illustrates the findings using visualizations. Understanding the composition of the corpus is relevant to chose the appropriate data processing steps and to understand the obtained results of the evaluation. Section 5.1 examines the text regarding the news sources of the data set, the news articles' length, the most common words and other attributes of the corpus. In Section 5.2, NER is applied to the textual corpus to extract information about the mentioning of company names in the news articles.

5.1. General Exploration of Textual Corpus

After data selection (see Section 4.3.4), a data set of **10,000** news articles with a total size of 27 MB remains, containing 833[1] articles of each month in 2017. The textual corpus consists of the news articles' headlines and plain texts, if not specified otherwise. For the analysis, the unigram model is used.

As highlighted in Table 5.1, the main source for news articles in the data set is *reuters.com*. The two reasons for this are that *webhose.io* does not have equal access to the requested sources for legal purposes and, the news from other sources than *reuters.com* have been parsed with lower quality. How this imbalance affects our work will be discussed in Section 7.2.

News Agency	Domain	Percentage
Reuters	reuters.com	94%
The Guardian	theguardian.com	3%
The Economist	economist.com	2%
Bloomberg	bloomberg.com	< 1%
CNN	cnn.com	< 1%
Financial Times	ft.com	< 1%

Table 5.1.: News sources of the articles in the data set

Before pre-processing, the total number of words in the textual corpus is 4,098,730. After pre-processing, when word duplicates are removed and Stop Words are filtered out, the document term matrix of the entire data set has 41,403 features.

Excluding headlines, the average length of the news articles examined is 2,462 characters. The distribution of the article length in the data set is shown in Figure 5.1.

[1] We select 834 from every third month: (8 * 833) + (4 * 834) = 10,000.

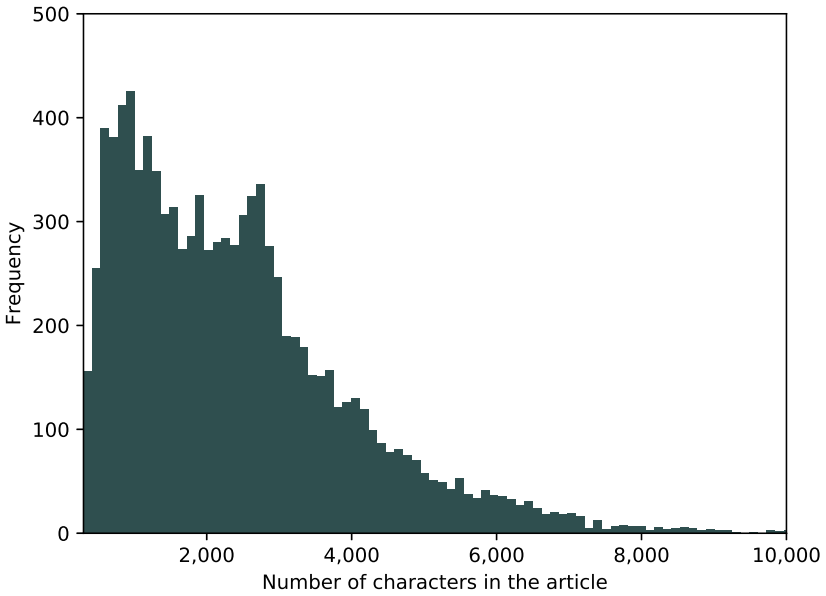

Figure 5.1.: Article length (Histogram)

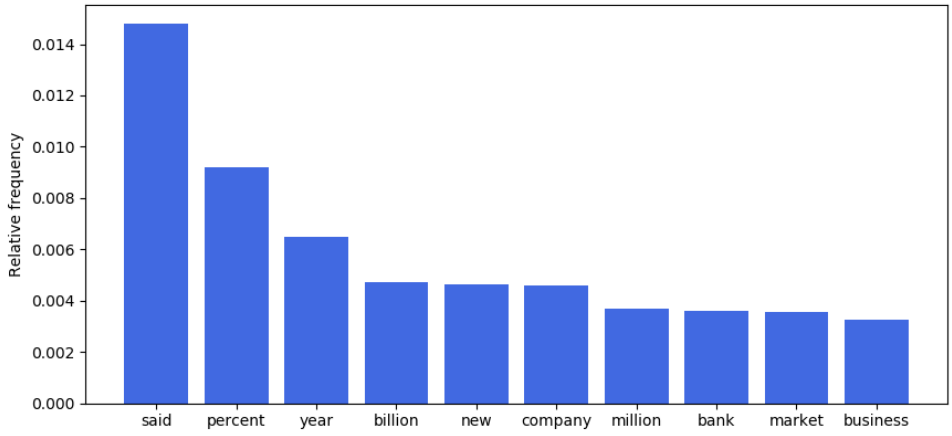

Figure 5.2.: Ten most frequent words in the data set (Bar Chart)

The ten most common words in the data set are `said`, `percent`, `year`, `billion`, `new`, `company`, `million`, `bank`, `market`, `business`, as shown in Figure 5.2.

Considering the text of articles about mergers, the ten most common words are `said`, `percent`, `year`, `billion`, `company`, `new`, `million`, `market`, `last`, `business` as shown

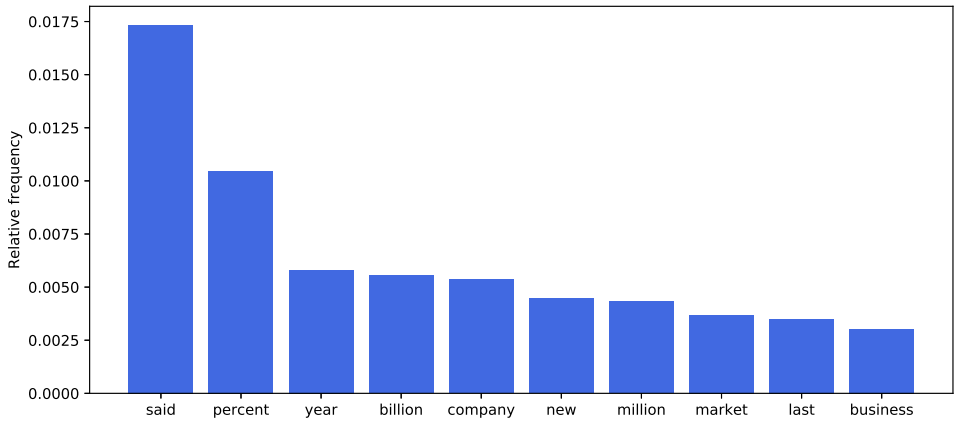

Figure 5.3.: Ten most frequent words in the articles about mergers (Bar Chart)

in Figure 5.3. Regarding the most common words, it can be seen, that this subset differs only marginally from the total amount of articles.

As the word `said` is not a specific term to business news or even news about mergers, it is not particularly relevant for classification. This leads to the question whether or not the unigram model is suitable for our problem at all. Therefore, the bigram model is also considered during evaluation in Chapter 7.

The Word Cloud in Figure 5.3 visualizes the number of occurrences of the 200 most common words in the data set. The calculation is based on the BOW model without Stemming with a total number of features of 57,339.

5.2. Exploration using NER

In the textual corpus, 17,312 different company names are mentioned in total. The average number of different company names mentioned per news article is five. This is due to the fact that an article deals not only with, e.g., the two merger partners, but also with competitors or inspecting state organizations. The importance of the individual mentions does not come out here, although this question is not addressed in this thesis.

Figure 5.5 presents the number of different company names per article as a histogram.

Figure 5.6[2] illustrates that the majority of company names are only included once in the data set. But there are also 478 companies that occur more frequently, which shows that big companies seem to dominate the reporting.

The most frequent company names in the textual corpus are `Apple` (272 articles), `Volkswagen` (224 articles), `Facebook` (218 articles), `Google` (209 articles) and `Boeing` (194 articles).

[2]Since no special aspects are visible from $x > 14$ anymore, the diagram's x-axis is cut off at $x = 14$.

Figure 5.4.: Most frequent words in the data set (Word Cloud)

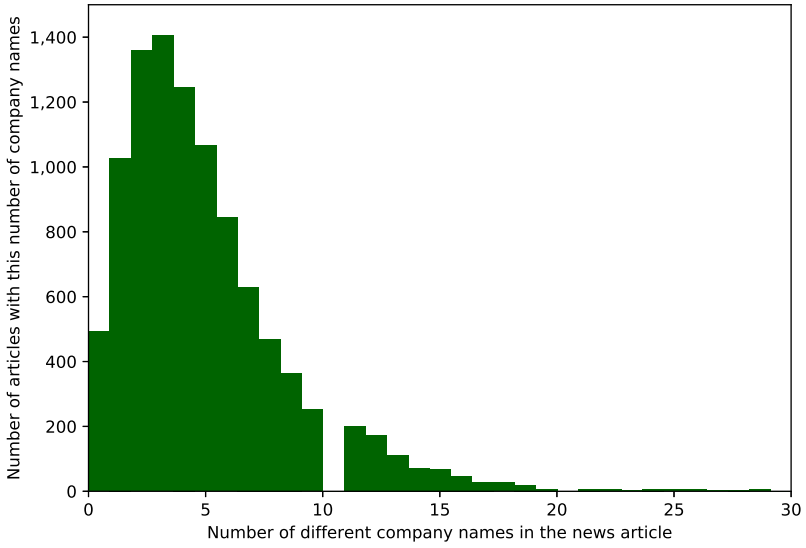

Figure 5.5.: Number of company names per news article (Histogram)

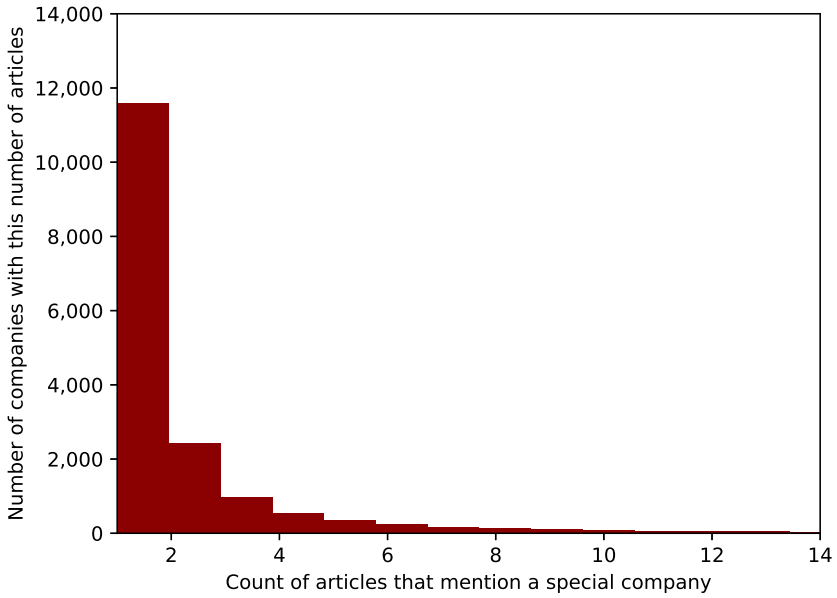

Figure 5.6.: Company Names Distribution (Histogram)

Summary:
This chapter provides important characteristics and features of our textual corpus in order to better interpret results of the evaluation later on. We have seen that the main source in the data set is Reuters, that the most common word is 'said', and that big companies dominate the reporting. The next chapter gives an overview of the practical implementation of this project.

6. Implementation

This chapter provides background knowledge about the used modules in Section 6.1 and used applications in Section 6.2. In Section 6.3, extracts of the most relevant parts of my implementation are shown and briefly explained.

6.1. Python Modules

The project is implemented in Python 3. The used modules are presented in the following.

gensim

Gensim [ŘS10] is a collection of Python scripts by Radim Řehůřek that were originally used to create a short list of the most similar articles to a particular article (gensim = "generate similar"). It utilizes *Latent Semantic Methods* to realize unsupervised semantic modeling from plain text. The gensim's Doc2Vec implementation was used, which is based on Word2Vec, to inspect an alternative to the BOW model for the numerical representation of a news article's text.

matplotlib

Matplotlib [Hun07] is a 2D graphics package used in Python for application development, interactive scripting, and publication-quality image generation across user interfaces and operating systems. The package *matplotlib* was used for visualizations such as histograms and bar charts.

nltk

NLTK (Natural Language Toolkit) [BLK09] is a platform for creating Python programs to work with natural language data. It provides user-friendly interfaces to over 50 corporate and lexical resources, as well as a range of word processing libraries. The package *nltk* was used in this project for Stop Words, Tokenization, Stemming and Tagging.

numpy

NumPy [Num05] is a basic package for scientific computing with Python. Among others, it contains an n-dimensional array object, broadcast functions, linear algebra and random number functions. The package *numpy* was used for array objects.

pandas

Pandas (Python Data Analysis Library) [Pan08] is an open source library that provides data structures and data analysis tools for Python. Of this module, the set of labeled array data structures (*Series* and *DataFrame*) was used as well as the Input/Output tools for loading tabular data from files (CSV).

sklearn

Scikit-learn [PVG+11] are tools for data mining and data analysis that are built on *NumPy*, *SciPy*, and *matplotlib*. The *sklearn* implementation for classification algorithms, split methods, pre-processing and the document term matrix (*CountVectorizer*) was used, as it allows a faster calculation than my own BOW implementation.

webhoseio

webhose.io [Web99] offers on-demand access to web data feeds. Each web data feed is optimized to provide coverage of a specific content domain such as news, blogs, online discussions. Filters can be used to specify the search. The historical data archive dates back to 2008 and is available for a fee. The *webhoseio* module was used in this work in order to download the data set.

wordcloud

A Word Cloud is a visual representation of text data that displays a list of words and is useful for quickly perceiving the most common terms of a textual corpus. The importance of each is shown with font size or color. The *wordcloud* project [A. 18] is a simple Python implementation that can be easily modified. It has the advantage of filling all available areas, and enabling the use of masks.

6.2. Jupyter Notebook

Project Jupyter [Pro14] is a open source project designed to support interactive data science and scientific computing in numerous programming languages. The *Jupyter Notebook* extends the approach of interactive computing and provides a web-based application that is capable of capturing the entire computing process: code development, documentation and execution, as well as communication of results. The *Jupyter Notebook* was used for interactive coding, labeling, visualization and documentation.

6.3. Own Implementation

Here are briefly introduced the relevant parts of this project's included implementation. The complete implementation can be found on `https://github.com/AnneLorenz/IntLab`.

6.3.1. Downloading the Data

Data Set for Conventional Labeling

Through *webhose.io*, news of the last 30 days can be downloaded for free. The desired news data can be received as a stream of JSON files via an API.

We only use articles that contain at least one of the keywords `merger`, `acquisition`, `take over`, `deal`, `transaction` or buy in the header. Furthermore, we restrict the request to original contributions (`is_first: true`) in order to exclude user comments and the like.[1]

Since 30 days seem sufficient to test the conventional strategy, we download the data through our Python script `Requester.py`, that contains the following query:

```
thread.title:(merger OR merges OR merge OR merged OR acquisition
↪ OR "take over" OR "take-over" OR takeover OR deal OR
↪ transaction OR buy)

is_first: true

site_type: news

site: reuters.com

language: english
```

This request leads to a data set of 1,497 news articles from different *Reuters* RSS feeds dating from the period of one month in 2018[2].

Among others, all JSON files contain the following information:

```
{
      "thread": {
        "uuid": "a931e8221a6a55fac4badd5c6992d0a525ca3e83",
        "url": "https://www.reuters.com/article/us-github-m-a-mic
        ↪ rosoft-eu/eu-antitrust-ruling-on-microsoft-buy-of-git
        ↪ hub-due-by-october-19-idUSKCN1LX114",
        "site": "reuters.com",
        "site_section":
        ↪ "http://feeds.reuters.com/reuters/financialsNews",
        "section_title": "Reuters | Financial News"
        "published": "2018-09-17T20:00:00.000+03:00"
```

[1] More filter settings can be found on `https://docs.webhose.io/docs/filters-reference`.

[2] The time period is May 25 - June 25 2018, retrieved on June 25 2018.

```
        "site_type": "news",
        "spam_score": 0.0,
      },
      "title": "EU antitrust ruling on Microsoft buy of GitHub due by October 19",
      "text": "BRUSSELS (Reuters)-EU antitrust regulators will decide by Oct. 19 whether to clear U.S. software giant Microsoft's $7.5 billion dollar acquisition of privately held coding website GitHub. Microsoft, which wants to acquire the firm to reinforce its cloud computing business against rival Amazon, requested European Union approval for the deal last Friday, a filing on the European Commission website showed on Monday. The EU competition enforcer can either give the green light with or without demanding concessions, or it can open a full-scale investigation if it has serious concerns. GitHub, the world's largest code host with more than 28 million developers using its platform, is Microsoft's largest takeover since the company bought LinkedIn for $26 billion in 2016. Microsoft Chief Executive Satya Nadella has tried to assuage users' worries that GitHub might favor Microsoft products over competitors after the deal, saying GitHub would continue to be an open platform that works with all the public clouds. Reporting by Foo Yun Chee; Editing by Edmund Blair",
      "language": "english",
      "crawled": "2018-09-18T01:52:42.035+03:00"
}
```


Data Set for Interactive Labeling

This time, news articles of the twelve months in 2017 are requested. The query is expanded to reliable sources such as *Reuters*, *Bloomberg*, *Financial Times*, *CNN* and *The Economist* to get a more diverse result. The request is sent to the *webhose.io* historical archive.[3]

[3] The requested data was downloaded in September 2018 with JSON as file format. Access to the historical archive is obtainable for a fee.

```
site: (reuters.com OR ft.com OR cnn.com OR economist.com OR
 ↪  bloomberg.com OR theguardian.com)

site_category: (financial_news OR finance OR business)

timeframe: january 2017 - december 2017
```

Every news article is saved in a single file. In total 1,478,508 files (4.69 GiB) are downloaded.

6.3.2. Jupyter Notebook for Interactive Labeling

The interactive labeling part is realized with the help of a Jupyter Notebook, as it comfortably supports interaction with the user combining a manual with an automated labeling technique.

An important requirement regarding the implementation is that, at any time, it must be possible to interrupt the process without data loss, as no user is able to label hundreds or even thousands of instances at once. To improve the Notebook's usability, user instructions are written in upper-case.

The Notebook is divided into three parts:

Part I (Data preparation): The data set of 10,000 business news articles is imported from the CSV file as *DataFrame* object and prepared for the further process.

Then the iteration part starts.

Part II (Manual checking of estimated labels): A specified number of article labels is checked manually, in our case 100 articles. In order to avoid imbalance, the number of occurrences of a particular company name in all checked articles is limited to three, using a created dictionary of all article indices (*keys*) with a list of mentioned organizations (*values*). The algorithm's label estimation of the last iteration is displayed and can be revised by the user using an *Ipywidget* slider if the estimate was wrong. Finally, all entered labels as well as the current iteration number are saved.

Part III (Model building and automated labeling): The classification algorithm is applied to estimate the unclassified news articles' labels. It returns a vector 'class_probs' $(K_1, K_2, ..., K_n)$ per sample with the probabilities K_i per class i. Estimated class labels are stored for the next iteration.

The following code section implements the user-computer interaction in *Part II*. Figure 6.1 shows the resulting Output at run time.

```
def show_next(index):
    ''' displays an article's title and text and creates an
    interactive slider to modify the label manually
    '''
    print('News article no. {}:'.format(index))
    print()
    print('HEADLINE:')
    print(df.loc[df['Index'] == index, 'Title'])
    print()
    print('TEXT:')
    print(df.loc[df['Index'] == index, 'Text'])

    def f(x):
        # save user input
        df.loc[df['Index'] == index, 'Label'] = x
        df.loc[df['Index'] == index, 'Round'] = m

    # create slider widget for labels
    interact(f, x = widgets.IntSlider(min=-1, max=2, step=1,
        value=df.loc[df['Index'] == index, 'Estimated']))

    print('0: Unrelated news, 1: Merger,')
    print('2: Other news related to deals etc.')
    print('_____________________________________________________')
    print()

# list of article indices that will be presented next
label_next = []

# generate new list of article indices for labeling
batchsize = 100
label_next = pick_random_articles(batchsize)

for index in label_next:
    show_next(index)
```

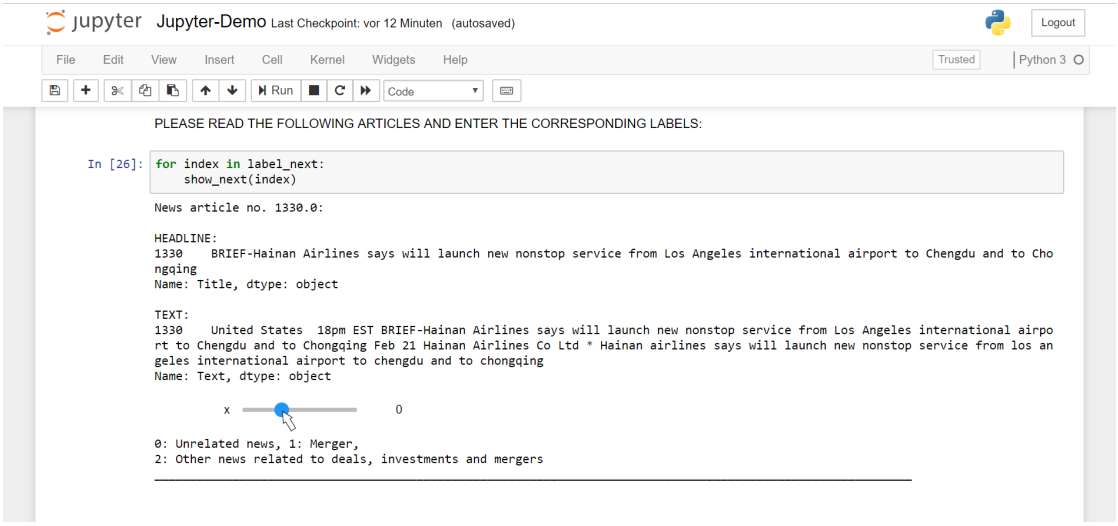

Figure 6.1.: Demo of the Jupyter Notebook labeling part

Summary:
This chapter briefly outlined the practical implementation of the interactive labeling method. It was shown how the user-computer interaction was implemented and which requirements were to be considered. In the next chapter, the methods used in this thesis are evaluated.

7. Evaluation

In this chapter, the suitability of the proposed methodology and implementation is evaluated. Section 7.1 provides an overview of the evaluation methodology. In Section 7.2 the used data set is evaluated, before discussing the use of NER to recognize company names in Section 7.3. Presenting several experimental studies in Section 7.4, the theoretical concepts on conventional and interactive labeling explained in Chapter 4 will be put into practice. Several approaches are executed and evaluated regarding achieved results, usability and time effort. Then, various options for the interactive labeling strategy are implemented and evaluated. In Section 7.5, we examine which instances are suitable to be presented to the user. Furthermore, we investigate different classification algorithms in Section 7.6 and select the most suitable one for our task. After all preliminary studies, Section 7.7 applies the developed concept of the optimized labeling strategy in practice. With the help of the gained knowledge, a forecast is made for the so far unlabeled data in Section 7.8.

7.1. Overview

First the data used and the application of NER are analyzed.

Then we evaluate the developed labeling strategy using several practical studies. These are proceeded as follows: To begin, the conventional labeling as described in Section 4.2 is realized and then evaluated.

Next, the interactive process is executed iteratively. This means that the news articles are not all labeled manually at once, but step by step in several iterations. When evaluating different labeling strategies for an unlabeled data set, the focus is on the various aspects of possible improvements to a manual labeling process. To explore the possibilities, we apply concepts step by step in an experimental way and investigate their impact. The specific data processing steps are performed as described in Section 4.3. Regardless of the success of the different methods, all experiments are documented and discussed, since they might be successfully applied to other tasks. Unless otherwise indicated, we apply Multinomial Naive Bayes[1] classification during the interactive process using the unigram model and removing Stop Words, numbers, and company names. For the analysis of the results, we concentrate on the reporting of the metrics Recall and Precision and the presentation of Confusion Matrices, especially for Class 1 (mergers). For each experiment, we record the observations and draw conclusions.

Three different variants of model building are used for the analysis:

[1] The Multinomial Naive Bayes implementation of *scikit learn* with the Hyper-Parameters `alpha=1.0e-10, fit_prior=False, class_prior=None` is used here.

a) Evaluation using the **Resubstitution Error** (referred to as ***'RE'***): Training and testing data set are identical. The RE is calculated to evaluate whether the model is consistent or ambivalent.

b) Evaluation using the **Hold-Out** method (referred to as ***'HO'***): Only a subset of labeled data is used for training, a different subset or all remaining data is used for testing and computing the validation metrics.

c) Evaluation using the **K-Fold Cross-Validation** (referred to as ***'KF'***): The data set is randomly divided into k groups. In each of the k runs, one of the groups is used for testing, the other groups for training, until all parts are used exactly once for testing.

We further consider how many classes should be used for multi-class labeling, for which we examine the content of the texts in more detail. To distinguish relevant news articles about mergers from irrelevant ones, at least two classes are required. How many additional classes are useful in differentiating the content of the texts will be covered during the discussion.

Furthermore, the question which instances should be shown to the user for verification is discussed. Working with class probabilities or model generation per class are considered here.

Then, the best possible classification algorithm is to be identified to be used during the interactive process. In the calculation, however, the context of interactive labeling is not taken into account. The algorithms Naive Bayes and SVM, also combined with Word2Vec, are tested by applying regular model fitting.

To verify the gained knowledge, a last iteration is performed with the optimized strategy. Finally, a prediction of the entire data set's structure is made to estimate the total duration of labeling.

7.2. Data used

For the study on conventional labeling, the data set of 1,497 instances described in Section 4.2.4 is used. For all studies concerning interactive labeling, the data set of 10,000 instances described in Section 4.3.4 is used, since it offers more possibilities due to its larger scope. As described in Section 5.1, news articles from the source Reuters are mainly used, because these are parsed by *webhose.io* with the best quality. Articles from other sources mainly had to be filtered out due to incorrect parsing and resulting low quality.

Conclusions: The quality of the downloaded data set can be described as rather low. As many news texts were parsed incorrectly and could only be used with a lot of time-consuming extra filtering, we were forced to remove them from the data set. Retrospectively, we should have used a more suitable source than *webhose.io*.

For a better generalization, we might want to include other data sources. Particularly, as it may be problematic to apply a model trained with our data in practice to news articles from other sources. Our model may then not be able to generalize, but the strategy can be kept for alternative sources. However, the selection of the data source does not affect the individual steps of our pipeline.

7.3. Recognition of Company Names

For data exploration and pre-processing, we examine the news texts for company names using *Stanford NER*. The knowledge gained as well as occurring problems are described below.

7.3.1. Quality of Company Recognition

When examining the texts for company names, several issues are noticed.

Observations:

- The algorithm does not differentiate by type of organization. For example, ministries, universities, or commissions are also tagged as 'ORGANIZATION'. It is not possible to distinguish them from the business companies in which we are interested.
- Company names also appear in an abbreviated version and are recognized as different companies. For example, *Volkswagen* and *VW*, *Amazon* and *Amazon.com*, *Ford*, *Ford Motor*, and *Ford Motor Company* are recognized as different organizations.
- Company names are sometimes used with and without their legal form (e.g., *Corp., Inc., LLC, GmbH* or *AG*) and are thereby recognized as different organizations.
- Some unusual company names are not recognized as *'ORGANIZATION'* at all, e.g., *Moneysupermarket.com*.
- If a location is placed before a company name, this is recognized as part of the name, e.g., *Germanys Commerzbank*. Also, if two companies follow directly one after the other, they are put together incorrectly, like in *Facebook and Alphabet*.
- Missing white spaces between two sentences, which are due to incorrect parsing by *webhose.io*, lead to the fact that a nonexistent company name is put together, like *United States.General Motors*. Unfortunately, all articles were parsed without spaces after each sentence (see Section 7.2), why this problem occurs frequently.

Conclusions: Compared to other implementations, e.g., to the *Natural Language Toolkit (nltk)*[NLT17] Tokenization, the *Stanford NER* is a quite reasonable method for recognizing organizations. Despite its weaknesses and limitations, it still provides a workable starting point for our analyses. It is also possible to train and adapt the Stanford NER for special individual tasks.

In order to separate corporate legal forms from the actual company names, we restrict ourselves to the most frequently used abbreviations, even if not all could be taken into account due to the high number of different international abbreviations.

7.3.2. Restricting the Number of Articles per Company Name

In Chapter 5, it was shown that large companies are particularly frequently reported on. During the interactive labeling, the number of articles in which a particular company is mentioned is limited, in order not to distort the analysis. When filtering out the entity 'ORGANIZATION' with the help of NER, we make a number of observations that raise further questions.

Observations and potential issues for the analysis

- In some cases, country names are incorrectly tagged as 'ORGANIZATION'.

  Questions: What happens when a country name like `France` is limited to three mentions? Could we miss a French merger?

- Bank names are correctly tagged as 'ORGANIZATION'. Especially when it comes to financing deals of all kinds, banks are often involved as lenders. Even if an article mainly deals with the merger of two companies and the bank is only mentioned as a sideline, all mentions are treated equally.

  Questions: What effects does it have if the number of bank names is also limited? Do we always get the crucial article about the conclusion of the contract? Could we miss a merger of two banks?

- A similar problem applies to state organizations such as government offices or commissions that are indirectly involved in the merger process, e.g., *Department of Justice, Antitrust Division or Federal Trade Commision*.

  Question: Does it have a disadvantage if government organizations occur very frequently?

- News agency names are also tagged as 'ORGANIZATION'.

  Questions: What effects does it have if the mentions of *Reuters*, e.g., are also limited? What if we filter out them? Couldn't a news agency ever merge?

- Before a merger actually takes place, there are reports on intentions, discussions, negotiations, approvals, etc. in the forefield. So all in all, there are a lot of articles in the news reporting about the complete development of a merger, but maybe only one of them about the completion of the merger.

  Question: Will we get any hit articles at all if we limit the occurences of company names?

- It is conceivable that a large company, such as, e.g., *Amazon*, is involved in several mergers at the same time.

  Question: Should these different reportings also be limited to three in total?

Conclusions: In order to use a fairly distributed data set for model selection, we limit the number of articles used to three per organization during the studies described in the following sections. The names of news agencies, banks, countries and state organizations are filtered out, keeping in mind possible side effects. Also, the filtering for keywords like *bank, banca, banque, ministry, commission, department, commitee*, etc. does not ensure completeness.

With regard to the follow-up problem – the recognition of the two merger partners – the question arises as to how the two merger partners should be distinguished from other actors in the text. If there are identified more than two different 'ORGANIZATION' objects in a Class 1 (merger) article, it is not obvious which two are the merger partners.

7.4. Labeling Studies

In the following, the concepts developed for data labeling are realized and evaluated. On the basis of intermediate results, conclusions for an optimal labeling strategy will be drawn.

7.4.1. Practical Study on Conventional Labeling

The CSV file containing the data set is first imported into the application *Microsoft Excel*. The procedure with conventional labeling is as follows: The headline is read first. If the headline already indicates the class of the article, the label is entered immediately in the column *Label*. In case the user is in doubt, sentence by sentence is read until the class can be assigned.

Observations:

- **Time Effort:** About 50 to 100 articles were labeled per hour, depending on the choice of words and length of the articles. In total, it takes about 15 to 30 hours to label 1,497 articles, which corresponds to four days in the worst case. Assuming the user wants to manually label a data set of 10,000 articles, it would take up to four weeks.

- As no special tools are required with conventional labeling, the user can start immediately without major preparations.

- The fact that a label was assigned based on the headline in some cases may be a problem, if something important is mentioned in the last sentence, which would affect the class label.[2]

- Depending on the readers domain expertise, some articles are still difficult to classify. Here are some unclear patterns that could be observed in various articles leading to follow up questions:

[2] There is a journalistic convention regarding writing news articles, to mention important things first and details and marginal remarks last. This also applies to the articles examined for this work.

- *'Company A acquires more than 50% of the shares of company B.'*

  Question: How should share sales be handled? Even if it is not a real merger, it actually means a change of ownership.

- *'Company X will buy/intends to buy company Y.'*

  Question: Will the merger definitely take place? On which circumstances does it depend?

- *'Last year company X and company Y merged. Now company A wants to invest more in renewable energies.'*

  Question: Only an incidental remark deals with a merger in the past. The main topic of the article is about something else.

Conclusions: These arising questions led to the idea of using multi-class labeling, which was finally implemented in the interactive labeling method.[3]

For a data set sized as the one considered in this work, the conventional method was acceptable. Labeling more data in this manner, however, would not be reasonable for the user.

7.4.2. Practical Study on Naive Interactive Labeling

This part of the study is a first attempt to approach the best possible interactive labeling strategy. As it will turn out later, it is not optimal yet and is therefore called **'naive'**.

We structure the process into multiple iterations. In each iteration, a small subset of new articles is labeled as described in Section 4.3.5. Then the classification algorithm is applied to the already labeled data as training data in order to make an estimation for the still unknown labels.

As classification algorithm, *Scikit learn*'s Multinomial Naive Bayes is used. Since many words will have zero counts at the beginning, we will perform a Laplace Smoothing with low alpha=1.0e-10. We set the parameters `fit_prior=False` and `class_prior=None` in order to avoid that the number of labels which have already occurred has an influence on the estimate. Only the structure of the feature vectors should be decisive for the assignment of a class.

Scikit learn's interface provides access to the probabilities for the individual classes via the method *predict_proba(X)*, which returns probability estimates for the test vector *X*. The influence of the class probabilities on the quality of the prediction will be examined later in Section 7.5. The more labeled data there already is, the better the prediction can be for the unknown labels. For this reason, the estimation should improve during the process, which will be examined later. In each iteration, a (random) selection of articles with already proposed labels is presented to the user, in order to be checked and revised by the user. Once the labels have been confirmed or changed, the process continues.

[3] Some of these questions still apply to interactive labeling, even when using multi-class classification. They are therefore not repeated in the following.

Regarding the labeling study, a first pre-study was performed, where the articles were classified into six different classes. A second pre-study follows, in which the number of classes was reduced to three. Then, findings from the pre-studies were used to conduct the main study using three classes.

Pre-Study I

Multi-class classification was started with six different classes. 800 articles are labeled using:

Class 0: other/unrelated news

Class 1 (merger): merger of companies A and B

Class 2: merger pending or in talks or to be approved

Class 3: merger rejected or aborted or denied

Class 4: share deal or asset deal or acquisition

Class 5: merger as incidental remark (not main topic or not current)

Observations:

- **Time Effort:** On average, $1\frac{1}{2}$ working hours were needed for 100 articles.
- Although the predictions were surprisingly good[4], the process of deciding which label was the most appropriate takes a long time if there is a difficult decision between two similar classes.

Conclusions: Fewer classes should be used to speed up the user's decision making process.

Pre-Study II

Now the classes 2 to 5 are merged into one. Thus, the remaining classes are:

Class 0: other/unrelated news

Class 1 (merger): merger of companies A and B

Class 2: news about deals/mergers/etc.

In this way, the next 300 articles are labeled to study the procedure.

[4] In some iterations there were only 4% of wrongly predicted labels.

Observations:

- The manual assignment is now much faster.
- In case of doubt, unambiguous articles were always sorted into the less valuable class[5]. Since a high Recall value is desirable, as no merger articles should be lost, this might not be a clever strategy.
- Reportings about a merger that is to be completed, but not yet official often contain the subjunctives `would` or `should`.
- If a merger has been canceled again, the words `not` and `no` have an impact on the class.

Conclusions: It is important, that these articles above can be distinguished from the articles about confirmed mergers, that we want to detect. These important keywords observed priorly will be removed from our list of stop words.

In order to avoid false negatives, articles should be assigned to the more valuable class.

Main Study

After identifying occurring problems, we perform ten iterations with an improved strategy.

The used classes are:

Class 0: other/unrelated news

Class 1 (merger): merger of companies A and B

Class 2: news about deals/mergers/etc.

During the first iteration, the algorithm is not yet able to make an estimate (as there is no data to learn from), so there is no proposed class yet why the user has to consider all the labels himself. In each subsequent iteration, 100 new articles are presented, which the user has to check for correctness.

Iteration 0: 100 randomly[6] selected articles are labeled manually. In the end, 84 articles are labeled as Class 0, three articles are labeled as Class 1 and 13 articles are labeled as Class 2. A model is then built: The 100 manually labeled articles are the training set, the remaining 9,900 articles are the "testing" set[7].

[5] The ranking is Class 0 < Class 2 < Class 1 (merger).

[6] With the help of NER, we make sure that no company name is mentioned more than three times in our labeled data set.

[7] We have no ground truth for the class labels, yet.

Predicted \Actual	0	1	2	Sum(Predicted)
0	79	4	17	100
1	0	0	0	0
2	0	0	0	0
Sum(Actual)	79	4	17	100

Table 7.1.: Iteration 1, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	79	0	21	0	79.0 %	100.0 %	79.0 %
1	0	96	0	4	0.0 %	0.0 %	96.0 %
2	0	83	0	17	0.0 %	0.0 %	83.0 %
Average					26.3 %	33.3 %	86.0 %

Table 7.2.: Iteration 1, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	82	4	12	98
1	0	0	0	0
2	0	1	1	2
Sum(Actual)	82	5	13	100

Table 7.3.: Iteration 2, Naive Interactive Labeling, Confusion Matrix

Iteration 1: In order to check the selection of articles, we let the algorithm calculate the probabilities for the individual class affiliations. Each article in the testing data is then assigned a class by the algorithm along with a certain probability. We store this class and its probability per article. This procedure is also described in Section 7.5.3. Of those articles which were estimated by the algorithm with a probability of greater than 0.99 for their class, 100 articles are randomly presented to and checked by the user and revised if necessary. With the help of NER, we make sure that no company name is mentioned more than three times in our labeled data set. This is to make the data set more balanced. The metrics of this iteration are presented in Table 7.1 and Table 7.2. It describes the correctness of the proposed labels and can be described as *HO*. The column *'Predicted'* contains the label proposed by the algorithm, the row *'Actual'* contains the manually checked/revised label by the user.

Iteration 2: The procedure for the next iterations is always the same as in iteration 1. Only those articles are considered which were estimated by the algorithm with a probability higher than 0.99 for a class. Again, 100 articles are randomly checked and revised if necessary. The metrics of this iteration are presented in Table 7.3 and Table 7.4.

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	82	2	16	0	83.7 %	100.0 %	84.0 %
1	0	95	0	5	0.0 %	0.0 %	95.0 %
2	1	86	1	12	50.0 %	7.7 %	87.0 %
Average					44.6 %	35.9 %	88.7 %

Table 7.4.: Iteration 2, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	82	4	12	98
1	0	0	0	0
2	0	0	2	2
Sum(Actual)	82	4	14	100

Table 7.5.: Iteration 3, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	82	2	16	0	83.7 %	100.0 %	84.0 %
1	0	96	0	4	0.0 %	0.0 %	96.0 %
2	2	86	0	12	100.0 %	14.3 %	88.0 %
Average					61.2 %	38.1 %	89.3 %

Table 7.6.: Iteration 3, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	80	4	13	97
1	0	0	0	0
2	1	0	2	3
Sum(Actual)	81	4	15	100

Table 7.7.: Iteration 4, Naive Interactive Labeling, Confusion Matrix

Iteration 3 - 9: The procedure of the previous iterations is repeated. The related metrics are presented in Table 7.5 to Table 7.18. For each individual iteration, we first display a Confusion Matrix showing which labels were suggested by the algorithm and how articles were manually labeled by the user. Then we present the corresponding metric table, which contains Precision, Recall and Accuracy for each class. An overview of iterations 0 to 9 can be found in Figure 7.1.

Detecting incorrect labels through *RE*: After iteration 9, *RE* is applied to all 1,000 labeled articles. Two articles that have been incorrectly labeled by the user are detected. After manual correction of the two labels, the *RE* is zero.

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	80	2	17	1	82.5 %	98.8 %	82.0 %
1	0	96	0	4	0.0 %	0.0 %	96.0 %
2	2	84	1	13	66.7 %	13.3 %	86.0 %
Average					49.7 %	37.4 %	88.0 %

Table 7.8.: Iteration 4, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	88	2	10	100
1	0	0	0	0
2	0	0	0	0
Sum(Actual)	88	2	10	100

Table 7.9.: Iteration 5, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	88	0	12	0	88.0 %	100.0 %	88.0 %
1	0	98	0	2	0.0 %	0.0 %	98.0 %
2	0	90	0	10	0.0 %	0.0 %	90.0 %
Average					29.3 %	33.3 %	92.0 %

Table 7.10.: Iteration 5, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	88	1	9	98
1	0	0	0	0
2	0	0	2	2
Sum(Actual)	88	1	11	100

Table 7.11.: Iteration 6, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	88	2	10	0	89.8 %	100.0 %	90.0 %
1	0	99	0	1	0.0 %	0.0 %	99.0 %
2	2	89	0	9	100.0 %	18.2 %	91.0 %
Average					63.3 %	39.4 %	93.3 %

Table 7.12.: Iteration 6, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	76	6	14	96
1	0	0	0	0
2	0	0	4	4
Sum(Actual)	76	6	18	100

Table 7.13.: Iteration 7, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	76	4	20	0	79.2 %	100.0 %	80.0 %
1	0	94	0	6	0.0 %	0.0 %	94.0 %
2	4	82	0	14	100.0 %	22.2 %	86.0 %
Average					59.7 %	40.7 %	86.7 %

Table 7.14.: Iteration 7, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	79	6	12	97
1	0	1	0	1
2	0	1	1	2
Sum(Actual)	79	8	13	100

Table 7.15.: Iteration 8, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	79	3	18	0	81.4 %	100.0 %	82.0 %
1	1	92	0	7	100.0 %	12.5 %	93.0 %
2	1	86	1	12	50.0 %	7.7 %	87.0 %
Average					77.2 %	40.1 %	87.3 %

Table 7.16.: Iteration 8, Naive Interactive Labeling, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	80	2	14	96
1	0	0	1	1
2	0	0	3	3
Sum(Actual)	80	2	18	100

Table 7.17.: Iteration 9, Naive Interactive Labeling, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	80	4	16	0	83.3 %	100.0 %	84.0 %
1	0	97	1	2	0.0 %	0.0 %	97.0 %
2	3	82	0	15	100.0 %	16.7 %	85.0 %
Average					61.1 %	38.9 %	88.7 %

Table 7.18.: Iteration 9, Naive Interactive Labeling, Metrics

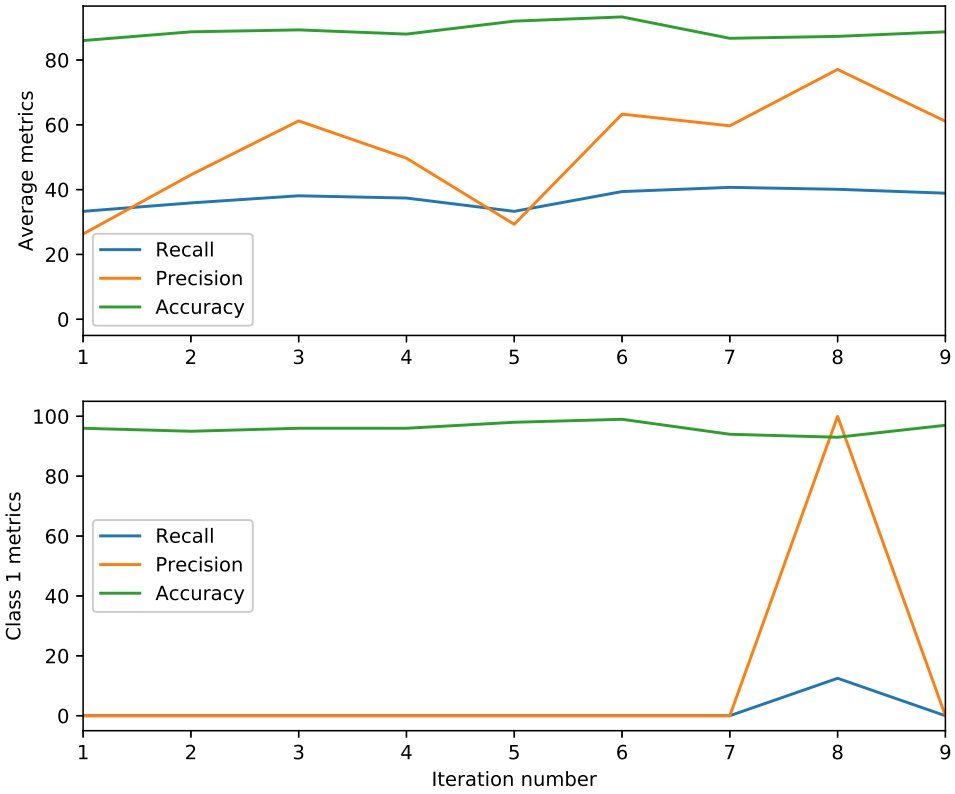

Figure 7.1.: Metrics during iterations 0 to 9 (Naive Interactive Labeling)

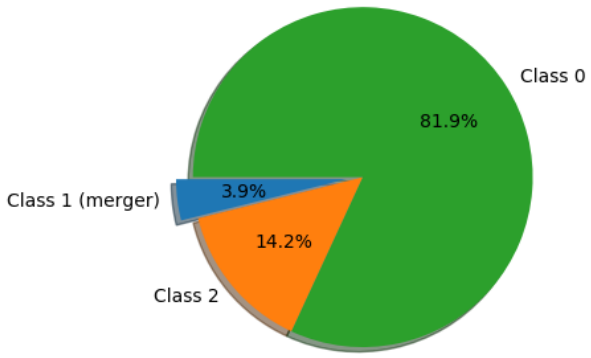

Figure 7.2.: Number of labels per class after iteration 9 (Pie Chart)

Observations:

- **Model improvement over iterations 0 to 9:** We fit a model after each iteration from all the so far labeled data to measure possible improvement of the overall naive model. The number of used instances thus increases from 100 to 200, ..., 1,000 instances during iteration 0 to 9. Table 7.19 shows that the model's Recall value slightly decreases, whereas the model's Precision value increases slightly from iteration to iteration. Generally, there is a minor improvement from iteration 0 to 9.

- The data set is imbalanced, because most articles are non-mergers. Figure 7.2 illustrates the total number of labels per class after iteration 0 - 9.

- During iterations 0 to 9, the entire quantity of previously labeled instances was always used as training data for our estimation model, although we are dealing with a strongly imbalanced data set. It is found that the predictor focuses on Class 0. Class 1 (merger) is almost ignored. The model does not predict an instance of Class 1 before iteration 8.

Conclusions: Using the entire quantity of previously labeled instances is actually inappropriate. It should be investigated whether the predictions would have been better, if we had applied stratified sampling to the training set.

It is also recommended to select the instances presented to the user for review more carefully. The first idea is to label about the same number of articles per iteration of all three classes. The second idea is that the user should concentrate on the articles that are difficult to classify and leave the articles out that are easy to classify for the algorithm. Which these are exactly, is examined in Section 7.5.

Iteration	Recall (Average)	Precision (Average)	F1-Score (Average)
0	85.3 %	73.7 %	79.0 %
1	81.5 %	66.5 %	73.3 %
2	82.4 %	70.0 %	75.7 %
3	83.1 %	74.7 %	78.4 %
4	83.5 %	76.7 %	79.9 %
5	84.4 %	80.1 %	82.1 %
6	85.5 %	80.0 %	82.5 %
7	84.3 %	80.2 %	82.1 %
8	84.1 %	80.7 %	82.2 %
9	84.0 %	78.7 %	81.2 %

Table 7.19.: Improvement of naive model over iterations 0 to 9 (*KF*)

7.4.3. Reinvestigating Model Improvement

When building a model during the iterations, the number of elements per class was unbalanced since most samples were of Class 0. As a result, most of the testing data is also predicted as unrelated news. Since most articles actually belong to Class 0, this in turn leads to artificially good results.

To achieve objective results, a stratified sampled model is rebuild after each iteration from all the so far labeled data. Then it is checked whether the next iteration's 100 labels could have been better predicted with this strategy.

In the following, the first and last iteration are examined in detail by calculating *RE* and *HO* for various training and testing data set combinations. The analyses of iterations 2 to 8 are presented in the appendix (see Appendix A.1). We refer to the individual analyses j of an iteration i with *No.*$i.j$ in order to be able to differentiate them better.

Iteration 0: After iteration 0, 84 instances are labeled as Class 0, three as Class 1 (merger), and 13 instances as Class 2. As a stratified sampled training data set, three instances of each class (nine in total) are selected to train the model.

- *RE (No. 0.0)*: The nine selected samples of iteration 0 are used as training and also testing data. The corresponding *RE* is zero.

Iteration 1: After iteration 1, the next 100 articles are labeled. 79 instances of Class 0, four instances of Class 1 (merger), and 17 instances of Class 2 are added to the classified data set.

The stratified sampled training data set is extended by four instances of each class (12 in addition) to a total sum of 21 instances.

- *RE (No. 1.0)*: Selecting the 12 samples after iteration 1 as training and testing data set, the *RE* is zero.

Predicted \Actual	0	1	2	Sum(Predicted)
0	104	0	7	111
1	34	5	3	42
2	25	2	20	47
Sum(Actual)	163	7	30	200

Table 7.20.: *HO (No. 1.1)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	104	30	7	59	93.7 %	63.8 %	67.0 %
1	5	156	37	2	11.9 %	71.4 %	80.5 %
2	20	143	27	10	42.6 %	66.7 %	81.5 %
Average					49.4 %	67.3 %	76.3 %

Table 7.21.: *HO (No. 1.1)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	125	0	11	136
1	29	7	6	42
2	9	0	13	22
Sum(Actual)	163	7	30	200

Table 7.22.: *HO (No. 1.3)*, Confusion Matrix

- *HO (No. 1.1)*: Using the 12 samples that are selected after iteration 1 as training data set and all 200 checked elements of iteration 0 to 1 as testing data set, the result is as shown in Table 7.20.

- *RE (No. 1.2)*: Using the 21 selected samples of iterations 0 to 1 as training and testing data set, there is no *RE*.

- *HO (No. 1.3)*: Taking the 21 selected samples of iterations 0 to 1 as training data set and all 200 checked elements of iteration 0 to 1 as testing data set, the resulting tables are Table 7.22 and Table 7.23.

- *HO (No. 1.4)*: Using 9 selected samples of iteration 0 as training data set and all 200 checked elements of iteration 0 to 1 as testing data set, the results are displayed in Table 7.24 and Table 7.25.

- *HO (No. 1.5)*: Using nine selected samples of iterations 0 as training data set and all 100 checked elements of iteration 1 as testing data set, the results are displayed in Table 7.26 and Table 7.27.

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	125	26	11	38	91.9 %	76.7 %	75.5 %
1	7	158	35	0	16.7 %	100.0 %	82.5 %
2	13	161	9	17	59.1 %	43.3 %	87.0 %
Average					55.9 %	73.3 %	81.7 %

Table 7.23.: *HO (No. 1.3)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	82	1	12	95
1	68	6	10	84
2	13	0	8	21
Sum(Actual)	163	7	30	200

Table 7.24.: *HO (No. 1.4)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	82	24	13	81	86.3 %	50.3 %	53.0 %
1	6	115	78	1	7.1 %	85.7 %	60.5 %
2	8	157	13	22	38.1 %	26.7 %	82.5 %
Average					43.9 %	54.2 %	65.3 %

Table 7.25.: *HO (No. 1.4)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	37	1	9	47
1	36	3	6	45
2	6	0	2	8
Sum(Actual)	79	4	17	100

Table 7.26.: *HO (No. 1.5)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	37	11	10	42	78.7 %	46.8 %	48.0 %
1	3	54	42	1	6.7 %	75.0 %	57.0 %
2	2	77	6	15	25.0 %	11.8 %	79.0 %
Average					36.8 %	44.5 %	61.3 %

Table 7.27.: *HO (No. 1.5)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	0	1	2	3
1	4	3	2	9
2	0	0	0	0
Sum(Actual)	4	4	4	12

Table 7.28.: *HO (No. 1.6)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	0	5	3	4	0.0 %	0.0 %	41.7 %
1	3	2	6	1	33.3 %	75.0 %	41.7 %
2	0	8	0	4	0.0 %	0.0 %	66.7 %
Average					11.1 %	25.0 %	50.0 %

Table 7.29.: *HO (No. 1.6)*, Metrics

- *HO (No. 1.6)*: Next, the nine selected samples of iteration 0 are used as training data set and the 12 selected samples of iteration 1 are used as testing data set. The resulting metrics are Table 7.28 and Table 7.29.

Iteration 2 - 8: The details of iterations 2 to 8 can be found in the appendix (see Appendix A.1).

Iteration 9: The procedure here is the same as in the previous iterations. The stratified sampled model of the last iteration is applied to the 100 presented articles of the current iteration. We now extend the stratified sampled training data by two instances per class to 117 samples in total.

- *RE (No. 9.0)*: Using the selected samples of iteration 9 as training and testing data set, the calculated *RE* is zero.

- *HO (No. 9.1)*: Taking the selected samples of iteration 9 as training data set and all 1,000 checked elements of all iterations 0 to 9 as testing data set, the resulting tables are Table 7.30 and Table 7.31.

- *RE (No. 9.2)*: Using the selected samples of iterations 0 to 9 as training and testing data set, the *RE* is zero.

- *HO (No. 9.3)*: Using the selected samples of iterations 0 to 8 as training data set and all 1,000 checked elements of iterations 0 to 9 as testing data set, the resulting tables are Table 7.32 and Table 7.33.

Predicted \Actual	0	1	2	Sum(Predicted)
0	280	4	36	320
1	39	5	13	57
2	500	30	93	623
Sum(Actual)	819	39	142	1000

Table 7.30.: *HO (No. 9.1)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	280	141	40	539	87.5 %	34.2 %	42.1 %
1	5	909	52	34	8.8 %	12.8 %	91.4 %
2	93	328	530	49	14.9 %	65.5 %	42.1 %
Average					37.1 %	37.5 %	58.5 %

Table 7.31.: *HO (No. 9.1)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	675	0	35	710
1	100	38	53	191
2	44	1	54	99
Sum(Actual)	819	39	142	1000

Table 7.32.: *HO (No. 9.3)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	675	146	35	144	95.1 %	82.4 %	82.1 %
1	38	808	153	1	19.9 %	97.4 %	84.6 %
2	54	813	45	88	54.6 %	38.0 %	86.7 %
Average					56.5 %	72.6 %	84.5 %

Table 7.33.: *HO (No. 9.3)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	68	0	6	74
1	8	1	11	20
2	4	1	1	6
Sum(Actual)	80	2	18	100

Table 7.34.: *HO (No. 9.4)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	68	14	6	12	91.9 %	85.0 %	82.0 %
1	1	79	19	1	5.0 %	50.0 %	80.0 %
2	1	77	5	17	16.7 %	5.6 %	78.0 %
Average					37.9 %	46.9 %	80.0 %

Table 7.35.: *HO (No. 9.4)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	0	0	1	1
1	0	1	1	2
2	2	1	0	3
Sum(Actual)	2	2	2	6

Table 7.36.: *HO (No. 9.5)*, Confusion Matrix

- *HO (No. 9.4)*: To see what would have happened if we had estimated the articles from this iteration with the stratified sampling model, the following analysis is executed: All selected samples of iterations 0 to 8 are used as training data set and the current 100 articles of iteration 9 as testing data set. The results are displayed in Table 7.34 and Table 7.35.

- *HO (No. 9.5)*: The selected samples of iterations 0 to 8 are used as training data set, while the selected samples of iteration 9 are used as testing data set. The results are shown in Table 7.36 and Table 7.37.

- *HO (No. 9.6)*: When using the 117 selected samples of iterations 0 to 9 as training data set and all 883 remaining labeled samples of iteration 0 to 9 as testing data set, the resulting metrics are shown in Table 7.38 and Table 7.39.

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	0	3	1	2	0.0 %	0.0 %	50.0 %
1	1	3	1	1	50.0 %	50.0 %	66.7 %
2	0	1	3	2	0.0 %	0.0 %	16.7 %
Average					16.7 %	16.7 %	44.5 %

Table 7.37.: *HO (No. 9.5)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	648	0	33	681
1	90	0	51	141
2	42	0	19	61
Sum(Actual)	780	0	103	883

Table 7.38.: *HO (No. 9.6)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	648	70	33	132	95.2 %	83.1 %	81.3 %
1	0	742	141	0	0.0 %	0.0 %	84.0 %
2	19	738	42	84	31.2 %	18.5 %	85.7 %
Average					42.1 %	33.8 %	83.7 %

Table 7.39.: *HO (No. 9.6)*, Metrics

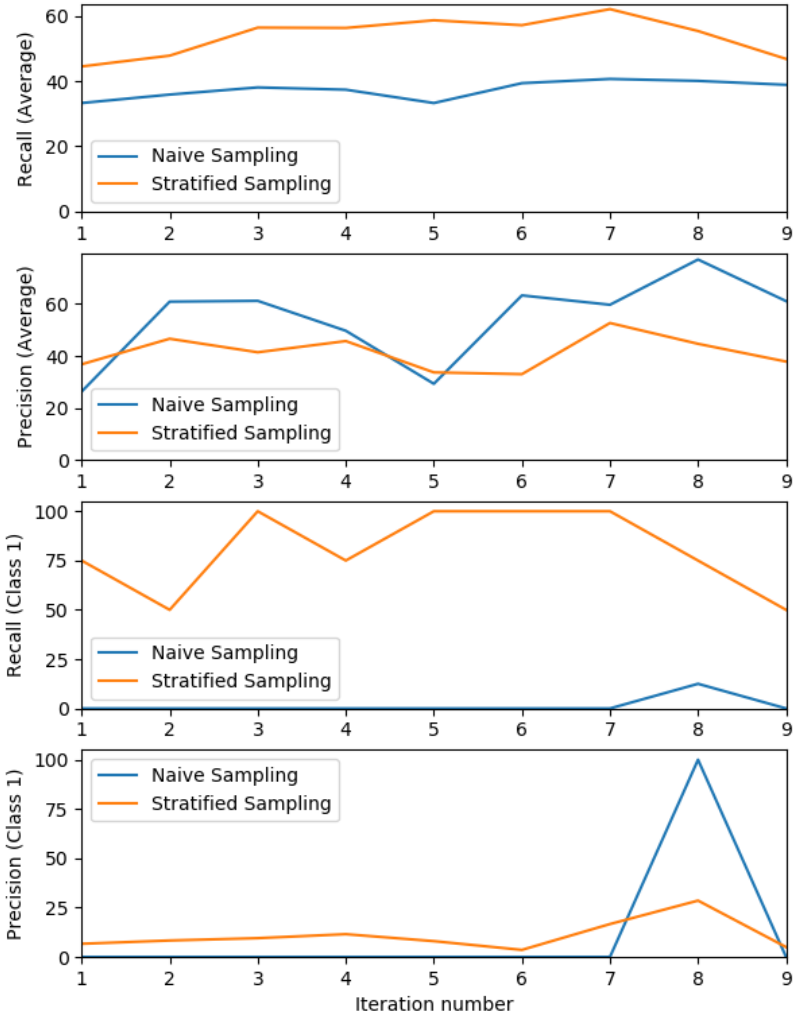

Figure 7.3.: Metrics over iterations 0 to 9

Observations:

- **Time Effort:** With interactive labeling, it is possible to classify up to 200 articles in one hour. Manual labeling can be done fast with the prototype developed in *Jupyter Notebook*, because moving the slider with the mouse is faster and more comfortable than typing a number in *Excel*. With labels that are already suggested correctly, most of the time is saved because no user input is necessary at all. However, some time is needed per iteration to apply the algorithm to update the labels.

  In sum, about ten hours were needed to label 1,500 articles, which corresponds to a working time of one or two working days. To label 10,000 articles, the user would need approximately one working week. This can be further optimized by using an improved interactive labeling strategy, as suggested in Section 7.7.

- Figure 7.3 shows that the model's Recall scores improve significantly through stratified sampling, especially for Class 1 (merger). Precision improved on average through stratified sampling, as can be seen from the graph.

Conclusions: Stratified sampling is the preferred method for an optimal labeling strategy as it significantly saves time and is more comfortable compared to the conventional method. High Recall scores for Class 1 (merger) are encouraging. Still, the quality of the predictor is not yet satisfactory. This could be due to the fact that the training data set is very small and consists of only 117 instances in the last iteration.

7.5. Finding the Best Instances for User Verification

This section examines which strategy is best suited for presenting instances to the user. First, a method using three separate binary models as predictor is compared to the multi-class model. As an alternative, a strategy using class probabilities (generated by the classification algorithm) as decision boundaries is considered.

7.5.1. Three-Model-Approach – Classification using Binary-Models

So far, a single overall model is used to classify the articles. We now compare three separate, independent classification models to it. Each of these binary models refers to one of the three label classes, what is also called *One-vs-All* strategy. The idea is: By simultaneously applying these three models to our data, we want to find out whether it provides information on disambiguities. If the three individual models conflict, that could also be used to select suitable instances for user verification. Table 7.40 illustrates how such a detection could work in practice using an example with dummy data. Possible ambiguities are marked in yellow.

Each of the three models is created according to the following scheme:
For a Class X, a corresponding binary Model X is trained. In order to take stratified sampling into account, half the training data is of Class X, half the training data is in equal parts of

Article Index	Model 0	Model 1	Model 2	Case
0	1.0	1.0	0.0	ambiguous
1	1.0	0.0	0.0	unambiguous
2	0.0	1.0	0.0	unambiguous
3	1.0	0.0	0.0	unambiguous
4	1.0	1.0	1.0	ambiguous
5	0.0	1.0	0.0	unambiguous
6	1.0	0.0	0.0	unambiguous
7	1.0	0.0	1.0	ambiguous
8	1.0	0.0	0.0	unambiguous
9	0.0	0.0	0.0	ambiguous

Table 7.40.: Demo: Detecting ambiguous cases with the Three-Model-Approach

the other classes Y and Z (as 'Class $\overline{X}$'). The model will then be used to predict the articles of Class X.

Of the 1,000 labeled articles of iterations 0 to 9, there are in total

- 819 samples of Class 0,
- 39 samples of Class 1 (merger) and
- 142 samples of Class 2

To apply Ten-Fold Cross-Validation, we split the data into ten stratified folds, each of them containing 82 instances of Class 0, four instances of Class 1 (merger), and 14 instances of Class 2. Then ten validation rounds are performed. Each time one fold is reserved as testing set, the remaining nine folds are used for training.

In order to be able to compare the quality of these binary models with that of a multi-class model, we built a Multinomial Naive Bayes model according to the same pattern.

- **Model 0** is trained with the following instances: 35 instances of Class 0 (as 'Class 0'), 18 instances of Class 1 (merger), and 18 instances of Class 2 (as 'Class $\overline{0}$'). Thus a training data set comprising 71 instances is obtained.
- **Model 1** is trained with the following instances: 35 samples of Class 1 (as 'Class 1'), 18 samples of Class 0, and 18 samples of Class 2 (as 'Class $\overline{1}$'). Thus a total number of 71 instances is obtained.
- **Model 2** is trained with the following instances: 35 samples of Class 2 (as 'Class 2'), 18 samples of Class 0, and 18 samples of Class 1 (merger) (as 'Class $\overline{2}$'). Thus one receives a training data set of 71 instances.
- **Model MNB** is trained with the following instances: 24 samples of Class 0, 24 samples of Class 1 (merger), and 24 samples of Class 2. Thus one receives a training data set of

	Rec (Min)	Rec (Max)	Rec (Av)	Prec (Min)	Prec (Max)	Prec (Av)
Class 0	0.0 %	50.0 %	33.3 %	0.0 %	3.2 %	1.9 %
Class $\overline{0}$	6.1 %	29.3 %	19.2 %	71.4 %	91.3 %	84.0 %
Average	10.1 %	37.8 %	26.2 %	37.0 %	47.2 %	42.9 %

Table 7.41.: Metrics of Model 0, Naive Bayes (*KF*)

	Rec (Min)	Rec (Max)	Rec (Av)	Prec (Min)	Prec (Max)	Prec (Av)
Class 1	25.0 %	100.0 %	72.5 %	4.3 %	11.1 %	9.1 %
Class $\overline{1}$	57.3 %	73.2 %	66.8 %	95.2 %	100.0 %	98.1 %
Average	49.1 %	84.0 %	69.6 %	49.8 %	55.9 %	53.6 %

Table 7.42.: Metrics of Model 1, Naive Bayes (*KF*)

	Rec (Min)	Rec (Max)	Rec (Av)	Prec (Min)	Prec (Max)	Prec (Av)
Class 2	0.0 %	75.0 %	36.7 %	0.0 %	15.0 %	6.6 %
Class $\overline{2}$	65.9 %	85.2 %	73.5 %	93.5 %	98.6 %	96.0 %
Average	35.4 %	77.1 %	55.1 %	46.8 %	56.7 %	57.8 %

Table 7.43.: Metrics of Model 2, Naive Bayes (*KF*)

	Rec (Min)	Rec (Max)	Rec (Av)	Prec (Min)	Prec (Max)	Prec (Av)
Class 0	86.3 %	84.2 %	73.4 %	94.0 %	100.0 %	97.4 %
Class 1 (merger)	50.0 %	100.0 %	80.0 %	4.6 %	18.2 %	12.9 %
Class 2	7.1 %	42.9 %	29.6 %	9.1 %	50.0 %	31.1 %
Average	36.3 %	71.6 %	61.0 %	35.9 %	53.86 %	47.1 %

Table 7.44.: Metrics of Multi-Class Model, Multinomial Naive Bayes (*KF*)

72 instances. With a number of only 24 instances per class and a total training data set of 72 instances, we want to ensure that the multi-class model has no advantage over the binary models, which are trained with only 71 instances.

7.5.2. Comparing Three-Model-Approach to Multi-Class Model

First, we examine how well these three individual models perform on the already labeled data, more precisely, on the 1,000 articles up to iteration 9. The results of these classifications are shown in Table 7.41, Table 7.42, Table 7.43, and Table 7.44.

Observations: Comparing the tables, it can be seen that Precision and Recall scores, especially for Class 1 (merger), are higher with multi-class classification than with the Three-Model-Approach.

Conclusions: Unexpectedly, the multi-class model works better, although Model 1 was trained with more instances of Class 1 (merger). As the Recall score for Class 1 (merger) is important for our task, the multi-class model is better suited to classify the articles and should therefore be used for the optimal interactive labeling strategy presented in Section 7.7.

Regardless of the metrics, this strategy can still be useful to present the instances to the user. Investigating this is a suitable topic for future work (see Section 8.2). Also, the Three-Model-Approach is a method of applying algorithms that work without class probabilities and offers a viable alternative to the multi-class classifier.

7.5.3. Study: Using Class Probabilities with Cutt-Off (Decision Boundaries)

During iterations 0 to 9, only those articles which were assigned to their class with a high probability by the algorithm were checked by the user. We now examine whether the model improves by letting the user check those articles that were assigned to their class with a low probability instead. The idea behind this choice is that these articles may be particularly difficult to classify and therefore be more prone to incorrect estimates – thus, the model could learn the most by classifying them correctly.

First we verify if there is a correlation between the estimated probability for a certain class and the Accuracy of the prediction. Therefore, we analyze whether there are more wrongly predicted articles in the probability range from 0.3 to 0.9 as a percentage of the total quantity. We apply the stratified sampled Multinomial Naive Bayes model obtained in iteration 9 again, and estimate the class affiliations and corresponding probabilities of all labeled articles from iterations 0 to 9. Table 7.45 shows the results of this investigation.

Prediction \Probability	$0.3 < p <= 0.9$	$0.9 < p < 1.0$	$p = 1.0$
correct	37.7 %	63.4 %	85.4 %
incorrect	62.3 %	36.6 %	14.6 %

Table 7.45.: Correlation between Probability and Error Rate

Observations: The table shows that instances predicted with a low probability for their class were more often incorrectly estimated than instances predicted with a high probability for their class.

Conclusions: The observed values confirm that there is a correlation between the estimated probability and the Error Rate. These results are in line with our theory that the algorithm has more difficulties with low probabilities for the class assignment and the user should rather be involved.

This consideration will be put into practice during eight more iterations:

Iterations 10 to 14 investigate the articles with low class probabilities. The results, observations and conclusions can be found in Appendix A.2 and Appendix A.3.

Iterations 15 to 17 investigate the articles with high class probabilities. The results, observations and conclusions can be found in Appendix A.4.

7.6. Finding the Best Classification Model

In this section, the context of interactive labeling is ignored to create the best possible multi-class model using the so far labeled data set. For this purpose, common algorithms with changing Hyper-Parameters are applied and compared to each other.

The procedure for each calculation is:

Ten-Fold Cross-Validation is applied to the labeled data set using stratified sampling. Starting point are the 1,082 labeled articles after iteration 11. Afterwards, for the ten created models the minimum, maximum, and average score of Recall and Precision are listed.

7.6.1. Multinomial Naive Bayes Algorithm

Setting Hyper-Parameters *alpha* to *1.0e-10*, *fit_prior* to *False* and *class_prior* to *None*, we use *scikit learn's CountVectorizer* for text conversion. The resulting metrics are listed in Table 7.46.

	Recall	Precision
Min	78.7 %	72.2 %
Max	84.4 %	81.5 %
Average	81.7 %	77.7 %

Table 7.46.: Multinomial Naive Bayes algorithm, CountVectorizer (*KF*)

Using my own BOW implementation instead of *CountVectorizer* (e.g. company names are filtered out), yields slightly better results, as can be seen in Table 7.47.

	Recall	Precision
Min	81.7 %	76.7 %
Max	86.2 %	84.4 %
Average	84.1 %	81.0 %

Table 7.47.: Multinomial Naive Bayes algorithm, BOW (*KF*)

Replacing the unigram model with the bigram model (*range = (1,2)*), results in the values displayed in Table 7.48. Replacing it with the bigram model (*range = (2,2)*), the values in Table 7.49 result. Although bigrams can make a significant difference, e.g. "they do **not merge**", the bigram model achieves a poorer result. These results are therefore unexpected, but are not investigated further in this work.

	Recall	Precision
Min	78.7 %	61.9 %
Max	82.6 %	80.6 %
Average	80.9 %	75.6 %

Table 7.48.: Multinomial Naive Bayes algorithm, Bigram (1,2) (*KF*)

	Recall	Precision
Min	77.8 %	61.9 %
Max	82.6 %	80.0 %
Average	80.4 %	74.1 %

Table 7.49.: Multinomial Naive Bayes algorithm, Bigram (2,2), (*KF*)

7.6.2. SVM

Among the various SVM implementations of the *scikit learn* module, LinearSVC combined with *CountVectorizer* and default settings offers the best results as listed in Table 7.50.

	Recall	Precision
Min	83.3 %	80.8 %
Max	91.7 %	91.8 %
Average	86.9 %	86.1 %

Table 7.50.: SVM (LinearSVC) algorithm (*KF*)

7.6.3. Word2Vec

Replacing *CountVectorizer* with the *gensim Word2Vec* implementation, Multinomial Naive Bayes algorithm generates the results shown in Table 7.51. Using *Word2Vec*, LinearSVC algorithm generates the metrics in Table 7.52.

	Recall	Precision
Min	71.3 %	80.1 %
Max	87.0 %	89.2 %
Average	76.1 %	84.4 %

Table 7.51.: Word2Vec, Multinomial Naive Bayes algorithm (*KF*)

	Recall	Precision
Min	80.7 %	76.8 %
Max	86.0 %	85.8 %
Average	83.2 %	80.5 %

Table 7.52.: Word2Vec, SVM algorithm (*KF*)

Observations:

- The Recall and Precision scores of all classifiers applied are similar and lie between approximately 76 % and 86 %.
- The bigram model produces poorer results than the unigram model. *Word2Vec* achieves higher Precision scores than *CountVectorizer*, but lower scores for Recall. As Recall is more important regarding the recognition of company mergers, this does not bring us any advantages.
- Overall, the SVM algorithm (especially LinearSVC) achieves best results on our labeled data set among the tested algorithms.

Conclusions: We have found that the best results for Recall and Precision are achieved with the SVM classifier with a very small advantage. However, SVMs do not directly provide probability estimates, these are calculated using an expensive Five-Fold Cross-Validation. But, Multinomial Naive Bayes also achieves very good results. Since Naive Bayes' class probabilities are much more reliable, we will concentrate on the Multinomial Naive Bayes classifier in the further labeling process.

7.7. Applying the Optimized Interactive Labeling Strategy

From the knowledge gained, we draw the conclusion of how the optimal interactive labeling process must look like. As our goal is to create the best possible model with as little manual labeling as possible, we concentrate on articles that are highly likely to be Class 1 (merger). Since the searched articles statistically make up only 4.6 % of all articles in the data set, they represent the bottleneck in stratified sampling. By significantly increasing these class instances, the total training data set increases and the model can thus be improved.

Iteration 18: After the previous iteration, there are 8,588 unlabeled instances. The training data set consists of 104 instances of each class, so 312 instances in total. With the help of Multinomial Naive Bayes classifier, the classes for all unlabeled instances are estimated. Class 0 is predicted 7,060 times, Class 1 (merger) is predicted 481 times, and Class 2 is predicted 1,047 times. Of the 481 articles estimated as Class 1 (merger), 396 have a class probability of exactly 1.0. Of these, 100 instances are randomly selected to be presented to the user. The results of iteration 18 are displayed in Table 7.53 and Table 7.54.

Predicted \Actual	0	1	2	Sum(Predicted)
0	0	0	0	0
1	18	33	49	100
2	0	0	0	0
Sum(Actual)	18	33	49	100

Table 7.53.: Iteration 18, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	0	82	0	18	0.0 %	0.0 %	82.0 %
1	33	0	67	0	33.0 %	100.0 %	33.0 %
2	0	51	0	49	0.0 %	0.0 %	51.0 %
Average					11.0 %	33.3 %	55.3 %

Table 7.54.: Iteration 18, Metrics

Observations:

- The quality of the prediction has already significantly improved compared to the previous iterations. Almost all texts examined have a merger reference or are about a deal. In the false positive articles, which had to be corrected to Class 2, it is often about an aborted or still running merger. With such articles it is sometimes difficult even for the user to distinguish them from the relevant ones.

- With a 33% hit rate for Class 1 (merger), this strategy is the best we have examined in this thesis. Compared to the statistical frequency of Class 1 (merger) instances (3.9%), we were able to increase the hit rate by a factor of eight.

Conclusion: The strategy presented here is an appropriate approach to achieve the best possible result with as few manual labels as possible. The labeling process is controlled by the user so that the stratified sampling training set grows evenly and the model can improve from iteration to iteration. The presented concept can also be adapted individually. For example, it is conceivable to include a keyword search or restriction of the word count of an article in order to get as many hits as possible of a certain class.

7.8. Conclusion for the Entire Data Set

The knowledge gained in the last sections is now applied to the entire data set of 10,000 news articles. The basis for the predictions is the LinearSVC model from Section 7.6, which was identified as the best model. As these are estimations only, they may differ significantly from reality.

Class	Samples	Percent
0	5,543	62.2 %
1	1,183	13.3 %
2	2,192	24.6 %
Sum (Total)	8,918	100.0 %

Table 7.55.: LinearSVC prediction for the unclassified articles (*HO*)

Class	Samples	Percent
0	847	78.3 %
1	50	4.6 %
2	185	17.1 %
Sum (Total)	1,082	100.0 %

Table 7.56.: Actual number of labels per class

7.8.1. Prognoses for the Remaining Unlabeled Data

The remaining unlabeled instances after iteration 9 (8,918 articles) are estimated as shown in Table 7.55.

To determine if that is plausible, the actual number of labels of the already classified data in Table 7.56 is counted as a comparison.

Observations: Looking at the actual number of labels per class (see Table 7.56) and transferring this to the remaining unlabeled data, label 0 should occur 6,983 times, label 1 should occur 410 times and label 2 should occur 1,525 times. The SVM algorithm estimates, however, 5,543, 1,183, and 2,192 instances for classes 0, 1, and 2 (see Table 7.55), which marks a significant discrepancy.

Conclusions: The observed discrepancy could be an indication that our model is not yet accurate enough and that the number of labeled data is not yet sufficient. The prognoses on the remaining unclassified data could possibly be improved by a larger data set.

7.8.2. Perspective of Users

Based on the best model identified in Section 7.6, we analyze the metrics regarding Class 1 (merger) as shown in Table 7.57. Scaled up to the unknown data set, this corresponds to the distribution in Table 7.58.

The fact, that the previously made predictions do not include any false negatives, is already satisfying, since the user does not want to miss any articles about mergers. However, there is quite a large number of false positives. The user would have to check all articles that are predicted as Class 1 (merger) to get an accurate assignment for articles about mergers. All true positives (410 instances) and all false positives (749 instances) add up to 1,159 articles

Metric	Samples	Percent
true positives	50	4.6 %
true negatives	941	87.0 %
false positives	91	8.4 %
false negatives	0	0.0 %
Sum (Total)	1,082	100.0 %

Table 7.57.: Accuracy for Class 1 (merger)

Metric	Samples	Percent
true positives	410	4.6 %
true negatives	7,759	87.0 %
false positives	749	8.4 %
false negatives	0	0.0 %
Sum (Total)	8,918	100.0 %

Table 7.58.: Estimated Accuracy for Class 1 (merger) based on Table 7.57

in total, which corresponds to 13.0 % of the entire data set (see the orange marked rows in Table 7.58). Of these, approximately 64.6 % would have been incorrectly labeled, so that only 35.4 % would actually belong to Class 1 (merger).

If the user labels 100 articles per hour, this corresponds to a working time of 12 hours. Whereas, if the user would label the articles of all classes, he would get about 2 $\frac{1}{2}$ weeks for all remaining 8,918 articles. We have therefore achieved a time saving of 87.0 %.

With the optimized labeling strategy presented in Section 7.7, it is possible to increase the hit rate for Class 1 (merger) instances by a factor of eight if news articles with the highest class probability are examined first. This means that a balanced model, i.e., using stratified sampling, can be built eight times faster than with conventional labeling.

7.8.3. Perspective of Management

By applying the methods described in this thesis, company mergers in news articles can be recognized much faster than if every single article of a news feed is read by a human reader. For the manager of a company this means that in the long run, a considerable amount of working time and, therefore, costs can be saved. However, the cost-effectiveness depends on the expected gain for each identified merger, if the gain is very high (e.g., say 1,000's of €), the manual labeling of articles is beneficial as missing a single article reduces the opportunities – in some cases, even having multiple labels from experts per article might be cost-effective.

A special feature of the developed method is that the model is already trained while the user is still labeling. This can be compared to a pre-trained model, which normally has to be purchased externally, assuming that a model suitable for the task exists at all.

Even without iterations, labeling in general can be done by starting with the articles that are "easy to categorize" (which have a high class probability) and then working towards the

articles which have a low class probability.

The hit rate for Class 1 (merger) of 33.0 % achieved with the presented strategy is eight times higher than the 3.9% hit rate with conventional labeling. In this way, the user is able to quickly get many instances of the interesting class (in our case Class 1 (merger)), which allows to rapidly build as large a stratified set as possible for a good classifier.

The labeling process can be terminated as soon as the user is satisfied with the quality of the model or as soon as the estimates mostly match the manual labels, so that the user does not have to label the complete data set here either.

Summary:

In this chapter, we first examined the company names recognition in more detail discovering which aspects require special attention. We have evaluated the two different labeling techniques by executing several practical studies. It has been found that conventional manual labeling is particularly suitable for small data sets. Regarding interactive labeling, the appropriate number of classes used was discussed, as well as the importance of stratified sampling was determined. Strategies for selecting the articles to be presented to the user have been researched. We found out that it is useful to consider the class probabilities of the Naive Bayes algorithm focusing on articles that have been assigned to their class with a low probability. A strength of interactive labeling is that large data sets can be labeled selectively and thus faster. More precisely, the data does not have to be completely labeled by the user, as part of the "simple" work is done by the algorithm. Through a final practical study, we conduct the optimized interactive labeling strategy and discuss our observations. Last but not least, we make a forecast for the entire data set and find that our strategy can save the user 87.0 % of the time compared to conventional labeling. In the last chapter, we summarize the essential aspects of this thesis and make suggestions for further work.

8. Summary

This last chapter summarizes our work and draws conclusions from the gained knowledge in Section 8.1. Lastly, we make suggestions for future work in Section 8.2.

8.1. Summary and Conclusions

In this thesis, we developed a strategy for recognizing company mergers in news articles focusing on the creation of a labeled data set.

In Chapter 1, we explained the benefit for companies to be informed about current corporate mergers and how we intend to solve this problem. In addition, the idea behind the interactive labeling was presented underlining its value for the user. In Chapter 2, background information about company mergers, human-computer interaction, NER, text classification, and the used metrics was provided. Chapter 3 presented other work related to our strategy of data labeling. In Chapter 4, we defined how the problem of detecting mergers in news articles is solved and how the interactive labeling strategy is investigated in comparison to a conventional labeling strategy. The used data set was explored in Chapter 5, as it is particularly important for further evaluation to have as much information as possible about the data set. We found out, e.g., that big companies are particularly frequently reported on and that on average five different company names are mentioned in one article. Chapter 6 presented an excerpt of the developed interactive labeling tool demonstrating how to use it. In Chapter 7, difficulties regarding company recognition using NER were analyzed. Also, both labeling techniques were evaluated in detail.

We found out, that conventional labeling is particularly suitable for small data sets, as no special tools are required and it is possible to start directly without major preparations. For large data sets, however, it is rather time-consuming and inefficient.

The presented method of interactive labeling is a convenient alternative to conventional data labeling when it comes to large data sets. Because the number of articles to be labeled can be reduced, the user can save up to 87.0 % of time, why we consider this approach to be promising. The time used for the realization of interactivity is compensated the faster and larger the data set to be labeled is.

A special feature of interactive labeling is that the user can (to a certain extent) control the articles to be labeled and the tendency in which the model develops.

Furthermore, interactive labeling allows to concentrate on articles that are difficult to classify during the labeling process, thus creating a reasonable division of work between human and machine. Alternatively, the user can concentrate on the articles with a high class probability and thus achieve a higher hit rate of desired articles.

The innovative approach of using labeling iterations allows the Machine Learning model to

be continuously improved. On the labeled data set, a common SVM model achieves a Recall score of 86.9 % and a Precision score of 86.1 %.

Our demonstration of numerous studies on labeling methods encourages users to apply this approach themselves. It is simple to implement and users should experiment with the presented concepts, depending on the individual task. The presented incremental method is not only suitable for text classification problems, but universal for all kinds of large unclassified data sets.

Last but not least, the psychological aspect of human-computer interaction plays an important role. Regarding the user experience, it can be perceived as exciting to see if the algorithm's predictions are correct. Also, the model's improvement from iteration to iteration spurs the user to perform more iterations which leads to a growing labeled data set. In total, the experience of using the developed interactive labeling tool is much more interesting and diversified as manual labeling and can therefore increase the user's work motivation.

8.2. Future Work

This work offers various points for further work, as we encountered numerous challenges during our analysis.

Concerning interactive labeling, the proposed optimized strategy should be applied from the beginning over numerous iterations in order to better evaluate the method. Regarding the selection of unambiguous class label predictions in Section 7.5, the Three-Model-Approach would still have to be examined in detail.

The problem of making predictions about news articles could also be solved with an *Artificial Neural Network (ANN)*. Neural Networks have become increasingly popular in the last few years as they can yield better results than the classic Machine Learning methods. In our case, a *Feedforward Neural Network (FFN)* or a *Recurrent Neural Network (RNN)*, which can word order take into account, could bring the desired results.

Difficulties have also been identified in recognizing company names (see Section 7.3), which could not all be solved. Regarding this problem, better results may be achieved with the help of a *ANN*. Murthy and Bhattacharyya have already discussed a *Deep Learning* solution which promises good results. [MB16].

Bibliography

[A. 18] A. Mueller. WordCloud Documentation. `https://amueller.github.io/word_cloud`, 2018. Accessed on 2018-10-29.

[BA07] Ulrich Balz and Olaf Arlinghaus. *Praxisbuch Mergers & Acquisitions. Von der strategischen Überlegung zur erfolgreichen Integration*. mi-Fachverlag, 2007.

[Bey18] Taweh II Beysolow. *Applied Natural Language Processing with Python*. Apress Media LLC, 2018.

[BLK09] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[Eng17] Clemens Engelhardt. *Mergers & Acquisitions*. Springer Gabler, 2017.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics. *ACL*, pages 363–370, 2005.

[Hei04] Andreas M. Heinecke. *Mensch-Computer-Interaktion*. Hanser Fachbuchverlag, 2004.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[KPW14] Jörn Kohlhammer, Dirk U. Proff, and Andreas Wiener. *Visual Business Analytics : Effektiver Zugang zu Daten und Informationen*. dpunkt.verlag, 2014.

[Mar09] Stephen Marsland. *Machine Learning: an algorithmic perspective*. Chapman & Hall/CRC, 2009.

[Mat02] Matplotlib. Python Plotting. `https://matplotlib.org`, 2002. Accessed on 2018-11-13.

[MB16] V. Mudra Murthy and Pushpak Bhattacharyya. A Deep Learning Solution to Named Entity Recognition. In *Computational Linguistics and Intelligent Text Processing, Part II*, pages 427–439. Alexander Gelbukh, 2016.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MM16] John Paul Mueller and Luca Massaron. *Data Science mit Python*. John Wiley & Sons Inc., 2016.

[Mur12] Kevin P. Murphy. *Machine Learning: a probabilistic perspective*. Massachusetts Institute of Technology, 2012.

[NLT17] NLTK Project. Natural Language Toolkit. `https://www.nltk.org`, 2017. Accessed on 2018-10-29.

[NMTM00] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2):103–134, May 2000.

[Num05] NumPy. `https://www.numpy.org`, 2005. Accessed on 2018-11-13.

[OKS05] Thomas Osugi, Deng Kun, and Stephen Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Fifth IEEE International Conference on Data Mining*, 2005.

[Ols09] Fredrik Olsson. A literature survey of active machine learning in the context of natural language processing. Technical Report SICS T2009:06, Swedish Institute of Computer Science, 2009.

[Pan08] Pandas: Python Data Analysis Library. `https://pandas.pydata.org`, 2008. Accessed on 2018-11-13.

[PRHP16] Sachin Pawar, Nitin Ramrakhiyani, Swapnil Hingmire, and Girish K. Palshikar. Topics and Label Propagation: Best of Both Worlds for Weakly Supervised Text Classification. In *Computational Linguistics and Intelligent Text Processing, Part I*, pages 446–459. Alexander Gelbukh, 2016.

[Pro14] Project Jupyter. `https://jupyter.org`, 2014. Accessed on 2018-11-13.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Ras17] Sebastian Raschka. *Machine Learning mit Python*. mitp Verlags GmbH & Co. KG, 2017.

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[Sci18a] Scikit Learn. Multinomial Naive Bayes. `https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes`, 2018. Accessed on 2018-12-12.

[Sci18b] Scikit Learn. Shuffle Split. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html`, 2018. Accessed on 2018-11-23.

[Sci18c] Scikit Learn. StratifiedKFold. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`, 2018. Accessed on 2018-11-23.

[SJS06] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. In *AI 2006: Advances in Artificial Intelligence.*, pages 1015–1021. Sattar A. and Kang B., 2006.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[The06] The Stanford Natural Language Processing Group. `https://nlp.stanford.edu/software/CRF-NER.shtml`, 2006. Accessed on 2018-10-29.

[TK01] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 45-66, 2001.

[Web99] Webhose.io. Documentation Output Reference. `https://docs.webhose.io/docs/output-reference`, 1999. Accessed on 2018-10-19.

[ZBL+04] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, 2004.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie-Mellon University, 2002.

[Zha04] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

[Zhu05] Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Appendices

A. Additional Evaluation

A.1. Iterations 2 - 8 (Reinvestigating Model Improvement)

In addition to Section 7.4.3, iterations 2 to 8 are examined more closely here.

Iteration 2: After iteration 2, another 100 instances are labeled. Of 82 samples of Class 0, five samples of Class 1 (merger) and 13 samples of Class 2, we add 15 samples (five per class) to the stratified sample. The overall stratified sampled model now consists of 36 instances.

To reconstruct what would have happened if we had estimated the articles of this iteration with our stratified sampling model, several analyses are performed.

- *HO (No. 2)*: Now, the 21 selected samples of iterations 0 to 1 are used as training data set and all 100 presented elements of iteration 2 as testing data set. The resulting metrics are shown in Table A.1 and Table A.2.

Iteration 3: After iteration 3, another 100 instances are labeled. Of 82 samples of Class 0, four samples of Class 1 (merger) and 14 samples of Class 2, we add 12 (four per class) samples to the training data. The overall stratified sampled model now consists of 48 instances.

The next calculation shows what would have happened, if we had estimated the articles from this iteration with our stratified sampling model.

Predicted \Actual	0	1	2	Sum(Predicted)
0	65	2	4	71
1	14	2	8	24
2	3	0	2	5
Sum(Actual)	82	4	14	100

Table A.1.: *HO (No. 2)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	65	12	6	17	91.6 %	79.3 %	77.0 %
1	2	74	22	2	8.3 %	50.0 %	76.0 %
2	2	83	3	12	40.0 %	14.3 %	85.0 %
Average					46.6 %	47.9 %	79.3 %

Table A.2.: *HO (No. 2)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	51	0	1	52
1	26	4	12	42
2	5	0	1	6
Sum(Actual)	82	4	14	100

Table A.3.: *HO (No. 3)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	51	17	1	31	98.1 %	62.2 %	68.0 %
1	4	58	38	0	9.5 %	100.0 %	62.0 %
2	1	81	5	13	16.7 %	7.1 %	82.0 %
Average					41.4 %	56.5 %	70.7 %

Table A.4.: *HO (No. 3)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	60	1	4	65
1	15	3	8	26
2	6	0	3	9
Sum(Actual)	81	4	15	100

Table A.5.: *HO (No. 4)*, Confusion Matrix

- *HO (No. 3)*: The 36 selected samples of iterations 0 to 2 are used as training data set, all 100 elements of iteration 3 as testing data. The results can be seen in Table A.3 and Table A.4.

Iteration 4: In iteration 4, 81 samples of Class 0, four samples of Class 1 (merger) and 15 samples of Class 2 are labeled. From these, five of each class (15 in total) are randomly selected for the stratified sampled data set. The total number of training samples is now 60.

- *HO (No. 4)*: As training data set, all 48 selected samples of iterations 0 to 3 are used. As testing data set, the 100 elements of iteration 4 are used. Table A.5 and Table A.6 show the resulting metrics.

Iteration 5: The procedure here is the same as in the previous iterations. The stratified sampled model of the last iteration is applied to the 100 presented articles of the current iteration. Afterwards, the stratified sampled training data set is extended by the equal number of instances per class.

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	60	14	5	21	92.3 %	74.1 %	74.0 %
1	3	73	23	1	11.5 %	75.0 %	76.0 %
2	3	79	6	12	33.3 %	20.0 %	82.0 %
Average					45.7 %	56.4 %	77.3 %

Table A.6.: *HO (No. 4)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	67	0	5	72
1	18	2	5	25
2	3	0	0	3
Sum(Actual)	88	2	10	100

Table A.7.: *HO (No. 5)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	67	7	5	21	93.1 %	76.1 %	74.0 %
1	2	75	23	0	8.0 %	100.0 %	% 77.0
2	0	87	3	10	0.0 %	0.0 %	87.0 %
Average					33.7 %	58.7 %	79.3 %

Table A.8.: *HO (No. 5)*, Metrics

- *HO (No. 5)*: The selected subset of iterations 0 to 4 is used as training data set, all 100 elements of iteration 5 are used as testing data set. Table A.7 and Table A.8 show the resulting metrics.

Iteration 6: The procedure here is the same as in the previous iterations. The stratified sampled model of the last iteration is applied to the 100 presented articles of the current iteration. Then, the stratified sampled training data set is extended by the same number of instances per class. Table A.9 and Table A.10 show the resulting metrics.

- *HO (No. 6)*: We use the selected samples of iterations 0 to 5 as training data set and all 100 elements of iteration 6 as testing data set.

Iteration 7: The procedure here is the same as in the previous iterations. The stratified sampled model of the last iteration is applied to the 100 presented articles of the current iteration. Then, the stratified sampled training data set is extended by the same number of instances per class.

Predicted \Actual	0	1	2	Sum(Predicted)
0	63	0	3	66
1	19	1	8	28
2	6	0	0	6
Sum(Actual)	88	1	11	100

Table A.9.: *HO (No. 6)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	63	9	3	25	95.5 %	71.6 %	72.0 %
1	1	72	27	0	3.6 %	100.0 %	73.0 %
2	0	83	6	11	0.0 %	0.0 %	83.0 %
Average					33.0 %	57.2 %	76.0 %

Table A.10.: *HO (No. 6)*, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	53	0	5	58
1	20	6	10	36
2	3	0	3	6
Sum(Actual)	76	6	18	100

Table A.11.: *HO (No. 7)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	53	19	5	23	91.4 %	69.7 %	72.0 %
1	6	64	30	0	16.7 %	100.0 %	70.0 %
2	3	79	3	15	50.0 %	16.7 %	82.0 %
Average					52.7 %	62.1 %	74.7 %

Table A.12.: *HO (No. 7)*, Metrics

- *HO (No. 7)*: Using the selected samples of iterations 0 to 6 as training data set and all 100 checked elements of iteration 7 as testing data set, the resulting tables are Table A.11 and Table A.12.

Iteration 8: The procedure here is the same as in the previous iterations. The stratified sampled model of the last iteration is applied to the 100 presented articles of the current iteration. Then, the stratified sampled training data set is extended by the same number of instances per class.

- *HO (No. 8)*: Using the selected samples of iterations 0 to 7 as training data set and

Predicted \Actual	0	1	2	Sum(Predicted)
0	66	1	4	71
1	7	6	8	21
2	6	1	1	8
Sum(Actual)	79	8	13	100

Table A.13.: *HO (No. 8)*, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	66	16	5	13	93.0 %	83.5 %	82.0 %
1	6	77	15	2	28.6 %	75.0 %	83.0 %
2	1	80	7	12	12.5 %	7.7 %	81.0 %
Average					44.7 %	55.4 %	82.0 %

Table A.14.: *HO (No. 8)*, Metrics

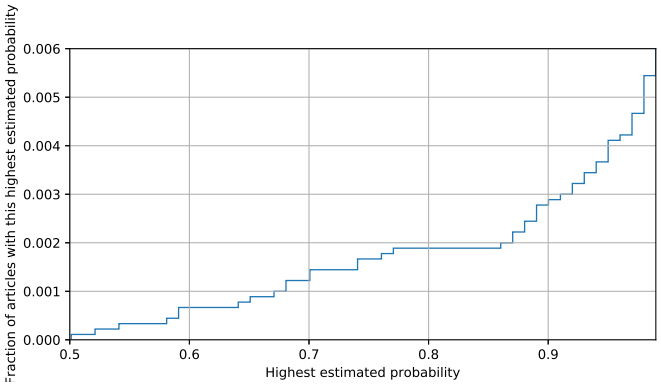

Figure A.1.: Estimation after labeling iteration 9

all 100 checked elements of iteration 8 as testing data set, the resulting tables are Table A.13 and Table A.14.

A.2. Iterations 10 - 11

To find out at what probability threshold the articles should be examined next, we plot a probability distribution in Figure A.1. It shows how many of the articles were assigned to their class with the respective probability. Based on this diagram, the user can then decide on a split point depending on how many items are to be checked in the next iteration. As decision boundary for the next iteration, a probability of 0.99 is chosen.

Predicted \Actual	0	1	2	Sum(Predicted)
0	9	4	12	25
1	3	1	1	5
2	4	1	14	19
Sum(Actual)	16	6	27	49

Table A.15.: Iteration 10, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	9	17	16	7	36.0 %	56.3 %	53.0 %
1	1	39	4	5	20.0 %	16.7 %	82.0 %
2	14	17	5	13	73.7 %	51.9 %	63.0 %
Average					43.2 %	41.6 %	66.0 %

Table A.16.: Iteration 10, Metrics

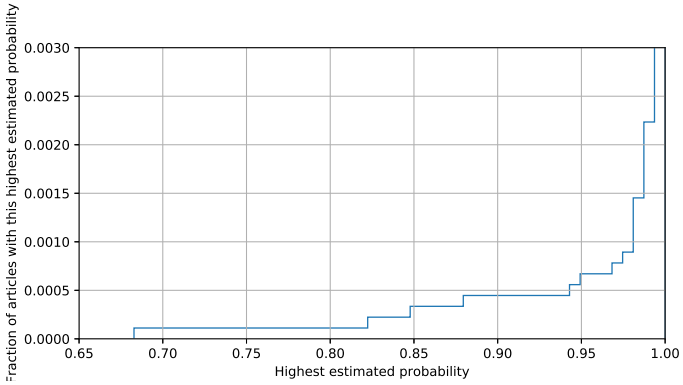

Figure A.2.: Estimation after labeling iteration 10

Iteration 10: All 49 articles of the remaining 9,000 unlabeled articles that are estimated with a probability below the decision boundary of 0.99, are presented to the user. The resulting Confusion Matrix (after correction by the user) and the corresponding metrics are listed in Table A.15 and Table A.16.

After iteration 10, we plot the current probability distribution displayed in Figure A.2. As can be seen from the diagram, the intermediate values have thinned out, which is an indicator that the classifier can now differentiate better.

Iteration 11: As decision boundary, a probability of 0.999 was chosen this time. 33 articles of the remaining 8,951 unlabeled articles are estimated with a probability below 0.999, which are presented to the user. The correctness of the proposed labels can be assessed using Table A.17 and Table A.18.

Predicted \Actual	0	1	2	Sum(Predicted)
0	9	1	7	17
1	1	1	0	2
2	2	3	9	14
Sum_Actual	12	5	16	33

Table A.17.: Iteration 11, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	9	13	8	3	52.9 %	75.0 %	67.0 %
1	1	27	1	4	50.0 %	20.0 %	85.0 %
2	9	12	5	7	64.3 %	56.3 %	64.0 %
Average					55.7 %	50.4 %	72.0 %

Table A.18.: Iteration 11, Metrics

Observations:

- For the user as well, the articles during iterations 10 and 11 are particularly difficult to classify, as they often lie between two classes in terms of content.
- We discover that these articles are remarkably short. Sometimes they consist of only two or three sentences.
- Since these articles are more often wrongly predicted (due to their relatively low probability) than those with a higher probability, the metrics Precision and Recall initially decrease.

Conclusions: As already mentioned, only articles that are particularly difficult to classify have been added in the last iterations. After several repetitions, after the most difficult articles are processed, the model might even ameliorate.

An alternative to the selection of articles described in the previous would be to focus on instances predicted as Class 1 (merger) and Class 2 during labeling, so that the quantity of all classes grows equally. Thereby, stratified sampling could be applied in the best possible way after each iteration. However, the difficulty here is, that instances with such low probabilities actually often turn out to be a different class than what they were estimated for. This means that it is only possible to influence whether the classes increase evenly to a limited extent.

A.3. Iterations 12 - 14

We start with the already labeled articles (1,082 instances after interations 0 to 11) and use these to train a stratified SVM model. After predicting all 8,918 unlabeled instances, the estimated class label and the class probability are annotated.

Predicted \Actual	0	1	2	Sum(Predicted)
0	2	0	2	4
1	5	0	1	6
2	0	0	0	0
Sum(Actual)	7	0	3	10

Table A.19.: Iteration 12, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	2	1	2	5	50.0 %	28.6 %	30.0 %
1	0	4	6	0	0.0 %	0.0 %	40.0 %
2	0	7	0	3	0.0 %	0.0 %	70.0 %
Average					16.7 %	9.5 %	46.7 %

Table A.20.: Iteration 12, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	1	0	2	3
1	2	1	1	4
2	2	0	1	3
Sum(Actual)	5	1	4	10

Table A.21.: Iteration 13, Confusion Matrix

Appling the concept of Section 7.5.3, ten articles with the lowest probabilities are presented to the user and revised manually.

This procedure is executed with three further iterations 12, 13 and 14. After each iteration, *RE* is calculated to detect errors during manual labeling.

Iteration 12: In order to evaluate how accurate the estimated labels match the actual labels in this iteration, the corresponding Confusion Matrix is listed in Table A.19 and the metrics are presented in Table A.20. The *RE* is zero.

Iteration 13: We evaluate how well the estimated labels match the actual labels in this iteration. The results are listed in Table A.21 and Table A.22. The *RE* is zero.

Iteration 14: Again, we evaluate how accurate the predictions for this iteration are. The results are listed in Table A.23 and Table A.24. The *RE* is zero.

Observations:

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	1	3	2	4	33.3 %	20.0 %	40.0 %
1	1	6	3	0	25.0 %	100.0 %	70.0 %
2	1	4	2	3	33.3 %	25.0 %	50.0 %
Average					30.6 %	48.3 %	53.3 %

Table A.22.: Iteration 13, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	2	0	1	3
1	2	0	0	2
2	3	1	1	5
Sum(Actual)	7	1	2	10

Table A.23.: Iteration 14, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	2	2	1	5	66.7 %	28.6 %	40.0 %
1	0	7	2	1	0.0 %	0.0 %	70.0 %
2	1	4	4	1	20.0 %	50.0 %	50.0 %
Average					28.9 %	26.2 %	53.3 %

Table A.24.: Iteration 14, Metrics

Predicted \Actual	0	1	2	Sum(Predicted)
0	0	0	0	0
1	60	17	23	100
2	0	0	0	0
Sum(Actual)	60	17	23	100

Table A.25.: Iteration 15, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	0	40	0	60	0.0 %	0.0 %	40.0 %
1	17	0	83	0	17.0 %	100.0 %	17.0 %
2	0	77	0	23	0.0 %	0.0 %	77.0 %
Average					5.7 %	33.3 %	44.67 %

Table A.26.: Iteration 15, Metrics

- Metrics have considerably decreased from iteration 11 to 14. The model's predictions' poor performance for the tested samples is possibly due to the fact that the selected articles consisted of those that were assigned to their class with a low probability and this in turn indicates some uncertainty.
- Especially for Class 1 (merger) the predictions are weak; in iteration 12 and 14 Recall and Precision equal zero.

Conclusions: In order to evaluate the proposed optimised method even more precisely, many more iterations would actually have to be performed. Only then, it is possible to finally conclude how effective the proposed method actually is. This should be a focus of future work, as proposed in Section 8.2.

A.4. Iterations 15 - 17

This section's focus is on the investigation of articles that have been estimated by the classifier with a high probability as Class 1 (merger).

Iteration 15: At the beginning of this iteration there are still 8,888 news articles without label. Since SVM has been identified as the best classifier in Section 7.6, its class probabilities are now inspected. For this, SVM is applied to the unlabeled data annotating the class probabilities. Now the 100 instances with the highest probability are manually checked, in this case with a class probability $>= 0.4646$.

Observations: The examined texts are exclusively short news, partly consisting of only one sentence or even only headwords.

Predicted \Actual	0	1	2	Sum(Predicted)
0	0	0	0	0
1	76	13	11	100
2	0	0	0	0
Sum(Actual)	76	13	11	100

Table A.27.: Iteration 16, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	0	24	0	76	0.0 %	0.0 %	24.0 %
1	13	0	87	0	13.0 %	100.0 %	13.0 %
2	0	89	0	11	0.0 %	0.0 %	89.0 %
Average					4.33 %	33.33 %	42.0 %

Table A.28.: Iteration 16, Metrics

Conclusion: Since the classification cannot work properly with extremely short texts, we limit the selection to texts with at least 700 characters in the next iteration.

Iteration 16: At the beginning of this iteration, there are still 8,788 news articles without a class label. Again, the SVM algorithm is applied to the unlabeled data by using a stratified sampled training data set and then annotating the class probabilities. Considering all articles whose class is predicted as Class 1 (merger), we limit the selection to articles that contain at least 500 characters in the full text and have a high class probability of at least 0.47003. These 100 instances are manually checked and corrected if necessary. The results are presented in Table A.27 and Table A.28.

Observations: The selection still consists of many short news that have little to do with mergers in terms of content.

Conclusion: Since the classification of very short articles works insufficiently, the limit of the number of characters has to be increased even further.

Iteration 17: There are 8,688 news articles without a class label after iteration 16. Again, the SVM algorithm is applied to the unlabeled data by using a stratified sampled training data set and then annotating the class probabilities. Class 0 is predicted 6,066 times, Class 1 (merger) is predicted 1,349 times, and Class 2 is predicted 1,273 times. Considering all articles whose class is predicted as Class 1 (merger), we limit the selection to articles that contain at least 700 characters in the full text and have a high class probability of at least 0.41064. These 100 instances are manually checked and corrected if necessary. The results are presented in Table A.29 and Table A.30.

Predicted \Actual	0	1	2	Sum(Predicted)
0	0	0	0	0
1	58	22	20	
2	0	0	0	
Sum(Actual)	58	22	20	100

Table A.29.: Iteration 17, Confusion Matrix

Class \Metric	TP	TN	FP	FN	Precision	Recall	Accuracy
0	0	42	0	58	0.0 %	0.0 %	42.0 %
1	22	0	78	0	22.0 %	100.0 %	22.0 %
2	0	80	0	20	0.0 %	0.0 %	80.0 %
Average					7.3 %	33.3 %	48.0 %

Table A.30.: Iteration 17, Metrics

Observations: After removing short news from the selection, the classification works much better than in the previous iterations. There are even many Class 2 articles that are related to mergers.

Conclusion: This iteration represents the best variant of the methods examined so far, in terms of the hit rate of Class 1 (merger) articles. However, the result is not yet sufficient, which is why the Multinomial Naive Bayes algorithm is applied in the last iteration. As it turns out, the last iteration 18 delivers the best results and is therefore presented in section Section 7.7.