Which Financial News to Trust?
Financial News Article Author Assessment using Sentiment Analysis and Named Entity Linking in a Streaming Environment

Master Thesis

Boy Adrianus Jacobus Raaijmakers BSc

Madrid, July 2019
This thesis is submitted to the ETSI Informáticos at Universidad Politécnica de Madrid in partial fulfillment of the requirements for the degree of EIT Digital Master in Data Science.

Master Thesis
EIT Digital Master in Data Science

Thesis Title: Which Financial News to Trust? Financial News Article Author Assessment using Sentiment Analysis and Named Entity Linking in a Streaming Environment

July 2019

Author: Boy Adrianus Jacobus Raaijmakers BSc
Universidad Politécnica de Madrid

UPM Supervisor: Alejandro Rodríguez González
Associate Professor

Deloitte Supervisor: Ehsan Mehdad
PhD

Computer Languages and Systems and Software Engineering
E.T.S de Ingenieros Informáticos
Universidad Politécnica de Madrid

Financial Advisory
Deloitte TTL
The Netherlands

ETSİ Informáticos
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte (Madrid)
Spain
Abstract

The challenge tackled in this thesis is combining the methods of sentiment analysis and named entity linking in a data stream environment applied to the task of assessing the expertise quality of financial news article authors. To do this, existing streaming methodology in both fields is reviewed and their quality in this context is assessed. Using these insights, a novel stream-based sentiment model and two novel named entity recognition (NER) post-processing methods are introduced to improve the current state-of-the-art.

The work reviews currently available lexicon methodology for sentiment analysis on unlabeled data and will formulate a novel unsupervised method for lexicon-based sentiment analysis in data streams. The best-performing current sentiment lexicon on the validation data is SentiWordNet 3 achieving an accuracy of 0.616 and F1-score of 0.730. A novel adaptive unsupervised sentiment analysis (AdaUSA) is introduced. AdaUSA updates a baseline lexicon under a data stream by learning and forgetting information in either a tumbled or slid window paradigm. Using the optimal configuration of AdaUSA, the experiments show an accuracy of 0.666 and F1-score of 0.757 on the validation data, thus denoting an increase in performance of 8.1% compared to the SentiWordNet 3 baseline.

Additionally, the work reviews current methodology in named entity recognition and named entity linking. Using our custom tagging algorithm, Stanford CoreNLP gives the best performing method with a classification accuracy of 0.487 and F1-score of 0.250. Combining this model with the novel post-processing $\text{ADom}$-selection and $\text{RDom}$-selection methods presented in this thesis, the accuracy can be improved to at most 0.741 and 0.812 respectively. Finally, the tagged target companies are shown to be correctly linked to their corresponding semantic web entities using DBPedia Spotlight with an accuracy of at most 0.776.

The insights in both tasks are combined into a stream-based architecture. The environments of Apache Storm and Apache Flink were implemented for sentiment analysis and named entity linking and the computational complexity is compared. Apache Flink showed the fastest article throughput on the test system using the best performing models on the validation data with an average processing time of 436.22 seconds per 1000 articles. The system with the highest performance on the validation data is hence an Apache Flink environment using AdaUSA with the SentiWordNet 3 lexicon as baseline for sentiment analysis, Stanford CoreNLP using $\text{RDom}$-selection for named entity recognition and DBPedia Spotlight for named entity linking.

Deploying the full architecture against the validation set of authors, the work finds significant differences in author quality per industry and over time. Especially in assessing the sentiment of news articles, the results shows a group of six well performing authors and four poorly performing authors. Assessing the impact the authors in the validation dataset have on the stock prices of the companies they write about, the study finds that the poorest performing author has a score of 0.223, while the best performing author has a score of 0.601.
Acknowledgements

First and foremost, I would like to thank my girlfriend Lisa for all the love and support she offered, not only throughout the course of the thesis execution, but also in the full length of my master programme and beyond.

Next, I would like to show gratitude for the fun and useful coaching of both my university supervisor Alejandro Rodríguez González and my Deloitte supervisor Ehsan Mehdad. With their combination of academic knowledge and business mindset, the work was not only strengthened, but served as a foundations to business solutions to be developed in the future. Also, during the full length of the project, they were vital by providing feedback and asking critical questions to steer the work in the right direction. Surely, the energy they provided served as the basis for the new techniques introduced in this work.

A third thanks goes out to my dear colleagues from my team at Deloitte. The Data & Reporting Advisory team was (and is) a very helpful source of energy and fruit for thought during the project. Additionally, the team provided me with the necessary distraction during my working days such that I would not get caught up in the project work. From different client projects that I contributed to, to designing websites, challenging other’s solutions and, of course, the numerous games of foosball, ping pong and squash, it all contributed to fulfillment of my final project.

Next, I would like to thank my fellow board members of DB Libertas, fraternity members of Drago and JV Orion for the support and interests during the thesis. They offered me the chance of distraction from study and work by fun activities, weekends off and numerous drinks we had together.

Lastly, I would like to thank everyone else that was involved in any way with my process throughout the project and during my studies as a whole.

Thank you!
Contents

1 Introduction 11
   1.1 Deloitte Nederland .................................... 12
   1.2 Thesis background .................................... 12
   1.3 Research contributions .................................. 13

2 Literature Review 17
   2.1 Machine learning basics: binary classification ................. 17
      2.1.1 Loss functions ................................... 17
      2.1.2 Evaluation metrics ................................ 18
   2.2 Multi-class classification .................................. 18
      2.2.1 Evaluation metrics ................................ 19
   2.3 Natural language processing .................................. 19
      2.3.1 Information retrieval ................................ 19
      2.3.2 Deep learning .................................... 20
   2.4 Sentiment analysis .................................... 21
      2.4.1 Polarity detection .................................. 21
      2.4.2 Hourglass of emotions ............................... 22
      2.4.3 Rule-based methods ................................ 22
      2.4.4 Lexicon-based methods ............................... 23
      2.4.5 Machine learning-based methods ....................... 25
      2.4.6 Hybrid methods .................................... 25
      2.4.7 Multi-modal methods ................................ 26
      2.4.8 Multi-lingual methods ................................ 26
   2.5 Semantic web modelling .................................. 26
      2.5.1 Resource description framework (RDF) .................... 27
      2.5.2 SPARQL ...................................... 27
   2.6 Named entity recognition .................................. 28
      2.6.1 Entity linking ................................... 28
   2.7 Stream mining and online learning ............................ 29
      2.7.1 Concept drift .................................... 30
      2.7.2 Adaptive models .................................... 30
      2.7.3 Stream data windows ................................ 31
      2.7.4 Evaluation metrics in stream mining ...................... 32
      2.7.5 Online deep learning ................................ 33
      2.7.6 Hybrid approaches ................................ 33
   2.8 Machine learning in the financial domain ...................... 33
      2.8.1 Natural language processing in financial forecasting ......... 34
## CONTENTS

3 Thesis Statement 37
3.1 Research questions ...................................... 37
3.2 Scope of the thesis ..................................... 38
3.2.1 Data stream simulation ............................. 38
3.2.2 System development ............................... 38
3.2.3 System evaluation ................................ 39

4 Data Gathering and Processing 41
4.1 Data gathering ....................................... 41
4.1.1 News articles ................................... 41
4.1.2 Stock information ................................. 42
4.1.3 Open linked data ................................. 42
4.2 Data pre-processing .................................... 43
4.2.1 Stop-word removal ................................ 43
4.2.2 Stemming and lemmatization .......................... 43
4.2.3 Numerical representation ............................ 44
4.2.4 Other techniques ................................. 44
4.3 Data labelling ....................................... 45
4.3.1 Manual labelling ................................. 45
4.3.2 Gathering labeled data .............................. 46
4.3.3 Noisy-labelling .................................. 46
4.3.4 Lexicon comparison ............................... 46
4.3.5 Used labeling strategy .............................. 46

5 Current Methodology 49
5.1 Natural language processing .............................. 49
5.1.1 Sentiment lexicons ................................ 49
5.1.2 Java frameworks ................................. 50
5.1.3 Python frameworks ............................... 51
5.1.4 Other languages ................................. 52
5.2 Pre-processing block ................................... 53
5.3 Sentiment model ..................................... 53
5.4 Open linked data (SPARQL) ............................... 54
5.5 Named entity linking ................................... 54
5.6 Stock compare block ................................... 55
5.7 Execution environment ................................ 57

6 Adaptive Unsupervised Sentiment Analysis 59
6.1 Lexicon adaptation based on context .......................... 59
6.1.1 SentiCircle ..................................... 60
6.1.2 Adaptation rules ................................ 62
6.1.3 Batch-based adaptation algorithm ....................... 62
6.2 Data stream environment ................................ 63
6.2.1 Windowed TDOC ................................ 64
6.2.2 Learning behavior ................................ 64
6.2.3 Forgetting behavior ................................ 66
6.2.4 Final algorithm .................................. 67
## CONTENTS

7 Named Entity Post-Processing for Financial Information .......... 71
   7.1 Company tagging ........................................ 71
   7.2 Dominant company selection ............................. 72
      7.2.1 Absolute dominance (ADom) .......................... 72
      7.2.2 Relative dominance (RDom) ........................... 73
   7.3 NER post-processing algorithm ............................ 73

8 Experimental Setup ............................................ 77
   8.1 Experimental setup ....................................... 77
      8.1.1 Test system description ............................. 77
      8.1.2 Pre-processing design ............................... 79
      8.1.3 Sentiment analysis design ............................ 79
      8.1.4 Named entity linking design .......................... 79
      8.1.5 Full setup architecture .............................. 80
      8.1.6 Execution environment ............................... 80
   8.2 Validation data ........................................... 82
   8.3 Test cases ................................................ 82
      8.3.1 Prediction performance measurement .................... 83
      8.3.2 Computational performance measurement ................. 84
   8.4 Financial author assessment ............................... 85
      8.4.1 Stock ticker retrieval on DBPedia ...................... 85
      8.4.2 Stock ticker retrieval on Wikidata ..................... 86
      8.4.3 Stock comparison to sentiment .......................... 87

9 Results and Evaluation ......................................... 89
   9.1 Experiment results ................................ ....... 89
      9.1.1 Sentiment analysis .................................... 89
      9.1.2 Adaptive sentiment model .............................. 92
      9.1.3 Named entity recognition ............................. 93
      9.1.4 Named entity linking ................................. 94
      9.1.5 Full architecture ..................................... 95
      9.1.6 Computational performance results ....................... 95
   9.2 Financial author assessment ............................... 96
      9.2.1 Author analysis results ............................... 96
      9.2.2 Author quality assessment ............................ 97
   9.3 Research question answers ................................ 98
      9.3.1 Sentiment analysis .................................... 98
      9.3.2 Named entity recognition and linking .................... 99
      9.3.3 Full architecture ...................................... 99

10 Conclusion ..................................................... 103
   10.1 Thesis conclusion ......................................... 103
   10.2 Discussion ................................................ 104
   10.3 Main contributions ................................ ....... 106
   10.4 Thesis limitations ........................................ 106
   10.5 Future research ........................................... 107
      10.5.1 Sentiment analysis .................................... 107
      10.5.2 Named entity linking .................................. 108
      10.5.3 Financial news article author assessment ............... 108

A Detailed Methodology Description ................................ 118
C Detailed Validation Results

C.1 Sentiment models
C.1.1 SentiWordNet 3
C.1.2 SentiStrength
C.1.3 SenticNet
C.1.4 Densify
C.1.5 Sentiment140
C.1.6 NRC Hashtag
C.1.7 Textblob

C.2 AdaUSA results
C.2.1 AdaUSA tumbled learning and no forgetting
C.2.2 AdaUSA tumbled learning and tumbled forgetting
C.2.3 AdaUSA tumbled learning and slid forgetting
C.2.4 AdaUSA slid learning and no forgetting
C.2.5 AdaUSA slid learning and tumbled forgetting
C.2.6 AdaUSA slid learning and slid forgetting

C.3 Named entity recognition models
C.3.1 Spacy NER
C.3.2 Apache OpenNLP
C.3.3 Stanford CoreNLP
C.3.4 Illinois CogComp NLP
# List of Figures

1.1 Deloitte’s structure with the location of the D&RA team. ............................................. 12

2.1 Number of articles written on sentiment analysis over time. ................................. 21

2.2 3D Model of the Hourglass of Emotion as defined by Cambria et.al. The $y$-axis represent a weighted scoring model ranging from very positive (one) to neutral (zero) to very negative (minus one) [13]. ......................................................... 23

2.3 Overview of sentiment analysis techniques as surveyed by [73]. ......................... 24

2.4 Typical batch- and stream data mining overview. .................................................... 29

2.5 Example of sudden and gradual concept drift in which the mean of the observations is measured [10]. .......................................................... 30

2.6 Number of articles written on online learning over time [50]. ................................ 31

2.7 Different window types visualized. ........................................................................... 32

2.8 Articles on financial analysis using textual data mining over time [50]. .................. 34

5.1 Overview of the system architecture. ................................................................. 49

5.2 Overview of the pre-processing module using various techniques. ....................... 53

5.3 Overview of sentiment models in batch and online setting. .................................... 54

5.4 Overview of the linking module using NER, NEL and SPARQL. ......................... 55

5.5 Overview of the stock module. ............................................................................... 56

6.1 Lexicon adaptation structure used by Saif et al [92]. ............................................. 60

6.2 Visualization of the different quadrants on the SentiCircle. ............................... 61

8.1 Overview of the overall test architecture. ............................................................... 78

8.2 Overview of the test system requirements. ............................................................... 78

8.3 Overview of test architecture in Apache Storm and Apache Flink. ....................... 81

8.4 Overview of input data structure of the ‘golden standard’ dataset. ....................... 82

8.5 Proportion of positive articles over time. ............................................................... 83

9.1 Windowed metrics for SentiWordNet 3. ................................................................. 90

9.2 QQ-plot of distribution of proportion of positive articles and windowed accuracy of SWN3 along with the line $y = x$. ...................................................... 91

9.3 QQ-plot of distribution of proportion of positive articles and windowed accuracy of AdaUSA with slid learning and tumbled forgetting along with the line $y = x$. ...................................................... 92

9.4 Windowed accuracy of Stanford CoreNLP over time. .......................................... 94
List of Tables

2.1 Naming of different classification outcomes. ........................................ 18
4.1 Extracted article features from LexisNexis [49]. .................................. 41
5.1 Sentiment lexicon comparison. ............................................................ 50
5.2 Java NLP methodology comparison. ..................................................... 51
5.3 Python NLP methodology comparison. ................................................ 52
5.4 Different available methodologies for pre-processing. .......................... 53
5.5 Comparison of different SPARQL libraries for Python and Java. .......... 54
5.6 Comparison of different NEL libraries for Python and Java. ................ 55
5.7 Comparison of streaming environments for Python and Java. ............... 57
6.1 Adaptation rules as proposed by Saif et al [92]. ................................... 62
8.1 Test case replication per considered method. ....................................... 84
8.2 All behavioral parameter settings for the AdaUSA test cases. ............... 84
9.1 Performance metrics of different sentiment models. ............................ 90
9.2 Correlation with respect to the proportion of positive articles. .............. 91
9.3 Results for different configurations of the AdaUSA method. .................. 93
9.4 Performance metrics of different named entity recognition models. ........ 93
9.5 Results for ADom-selection and RDom-selection in combination with Stanford CoreNLP. ................................................................. 94
9.6 Proportion of correctly linked entities with respect to the all classified companies and all correctly classified companies. ................................. 95
9.7 Average computational performance over 1000 articles in seconds. ........ 96
9.8 Results of sentiment analysis and named entity recognition grouped by author. 97
9.9 Results of linking and assessment grouped by author. .......................... 97
A.1 Version details of the used methods of this thesis. ............................. 118
A.2 Detailed descriptions of the used sentiment models of this thesis. .......... 119
A.3 Detailed descriptions of the used NER and NEL methods and streaming environments of this thesis. ......................................................... 120
B.1 Detailed author information from the validation dataset. ....................... 121
List of Algorithms

1  Target selection in named entity recognition. ........................................ 56
2  Lexicon adaptation for batched collection of documents. ..................... 63
3  Tumbled Learning for lexicon adaptation. ......................................... 65
4  Slid Learning for lexicon adaptation. ............................................... 65
5  Tumbled Forgetting for lexicon adaptation. ..................................... 66
6  Slid Forgetting for lexicon adaptation. ........................................... 67
7  AdaUSA algorithm for adaptive sentiment analysis. ............................ 68
8  ADom-selection in named entity recognition. .................................... 73
9  RDom-selection in named entity recognition. .................................... 74
Chapter 1

Introduction

Traditional machine learning (ML) solutions analyze data in a so called ‘batch mode’ which assumes all data to be stored, bounded and accessible at will. Despite these approaches being very effective, it might not always be the suitable way to go. Some ML tasks require immediate action upon observation of new pieces of data. An example of such tasks is found in stock trading in which fast decisions are key to making profit. As an alternative, one can consider methodology from data stream mining to gain immediate and adaptive insights on data as it is generated.

Additionally, in stock trading, prices are known to be significantly influenced by newspaper articles concerning the specific related company. Nevertheless, there is no guarantee about the quality of such articles. Furthermore, these news corpora consist of unstructured, latently formulated information which is challenging to analyze automatically. Known techniques to approach this problem are found in field of natural language processing (NLP). Applying these techniques to a specific textual corpus and combining the results will yield information on what action to take, thus providing means of quality validation.

The challenge tackled in this thesis is combining the methods of sentiment analysis and named entity linking in a data stream environment applied to the task of assessing the expertise quality of financial news article authors. The aim will be to analyze news articles ‘on-the-fly’, classifying their trustworthiness (or quality) and, at the same time, adapt the model to this new observation. The classification of articles (i.e. their sentiment score) are compared to the actual changes in stock prices of the company targeted by the article, thus giving information on whether an author’s prediction was correct. The combination of classify and adapt will enable a model to account for (sudden) changing circumstances, thus creating the possibility of indefinite model deployment. Additionally, the online setting of the solution creates scalability opportunities as one can add many different news platforms (i.e. authors) as data sources and there are possibilities for distributed (scale-out) deployment of the models.
1.1 Deloitte Nederland

The thesis is created under the company brand of Deloitte Nederland. Deloitte Nederland is part of the Deloitte Touche Tohmatsu Limited. Deloitte Nederland employs approximately 7,000 people, whereas the global brand has more than 275,000 employees. Deloitte is one of the leading professional service companies worldwide. In 2019, Deloitte is listed the third strongest brand in the world in the Global 500 of the World Economic Forum 2019 [43]. Among three others, Deloitte belongs to the Big Four companies in professional services and provides a broad range thereof to numerous customers [24].

Deloitte’s mission statement: “Make an impact that matters.”

The work is written within the Data and Reporting Advisory team in the Financial Advisory group of the Risk Advisory department. A visual representation of this structure is found in Figure 1.1. The D&RA team aims at aiding their customers with reporting tasks within the financial domain. The expertise of this team will contribute towards a thorough understanding of the financial domain.

1.2 Thesis background

Previous research on using financial news articles to describe events on industry markets have shown promising results. However, these methods mainly made use of batch-based machine learning models to analyze the textual information. A major flaw in batch-based solutions is their lack of adaptability to unpredictable situations. Adapting such a model requires retraining it, costing several hours of time and resources. With an increasing pace at which financial information is being generated, these batch-based solutions fail in keeping a high performance over time.

Furthermore, in natural language processing, Sentiment Analysis (SA) is growing in popularity. SA is used to identify the level of emotion found in a certain document. This level can be measured using a binary true (positive) or false (negative) but can also be denoted by more complex models of which the Hourglass of Emotions is the most recent example. Typical tasks aim at classifying the sentiment given some textual corpus of data. For this,
1.3. RESEARCH CONTRIBUTIONS

either a lexicon (i.e. a wordlist with words and their level of emotion) can be constructed or a machine learning classifier can be trained. Most recent studies have shown that both methods, as well as a combination thereof, perform significantly well, showing an accuracy of 95% on industry-specific datasets. Having a rapidly (exponentially) increasing number of articles written on the topic, sentiment analysis is one of the traditional, yet popular tasks within natural language processing in machine learning.

Besides sentiment analysis, a rapidly growing field in data mining is that of semantic web modelling. In a traditional fashion, data on the world wide web is stored as separate entries, not being directly linked or related to each other. However, with a growing number of data sources, the absence of structure is a growing challenge to data scientists. This problem is solved using semantic web modelling. The principle denotes a collection of standards to link entities in both intra- and inter-network open data sources. By linking these sources, relevant information can be retrieved more efficiently, even though it might originate from different (physical) sources. The increasing popularity for open linked data is found in many fields, of which the most interesting is the willingness of several local and national governments to make their data publicly available for semantic web developers to link to their own sources.

Lastly, the velocity at which data on the world wide web, as well as the semantic web, is generated is ever growing. Traditional machine learning models are struggling with keeping up with this speed and hence a need for new forms of data mining methods arises. A relatively new field of methods aims at analyzing data as a stream flowing by over time. In data stream mining, the dataset is not pre-constructed; data will be observed over time, potentially in the future. Furthermore, data is not available at will but rather can be looked at at most once after which the next element is processed. Due to these two aspects, stream mining can handle large amounts of information at a high speed, hence solving the problem encountered with batch-based methods.

The task tackled in this work is that of describing the impact of financial news article authors. Specifically financial articles are chosen as they are known to have a significant and instant effect on industry markets, as is stated by the efficient market hypothesis (EMH). Assessing the quality of these authors will be done using the techniques of sentiment analysis and semantic web modelling in a stream mining context.

1.3 Research contributions

This thesis rises from the combination of three fields of expertise described in Section 1.2. More specifically, it aims at combining the fast pace at which financial information is created, the power of sentiment analysis constructed over the years and the expanding open linked knowledge available on the semantic web. This combination will result in an architecture of real-time document analysis, linking and target description.

To the author’s knowledge, this combination of stream classification and linking in the financial domain has not been performed before and hence poses an open challenge in literature of natural language processing and stream mining. Additionally, the challenge of stream mining applied to financial news articles is posed as an open challenge by previous analysts who recently surveyed the topic [116]. The proposed architecture in this thesis can be adapted to many different practical use cases in which a high-speed stream of unstructured (i.e. textual) information should be analyzed.
To construct a feasible and relevant thesis statement, a thorough literature review of the considered data mining tasks will be given next. Once the knowledge on the different tasks is reviewed, the thesis statement, including requirements, specifications, purpose and proposed methods will be defined in Chapter 3.
Chapter 2

Literature Review

To identify the exact gap in literature filled by this work, first, a thorough understanding of the current state-of-the-art is required. The expertise on the current literature can be used in the remainder of the thesis to ensure an optimal result. The study on current knowledge on the different aspects of the thesis topic will be given in this chapter. After having the knowledge on the separate sub-tasks, an overview of combining the different parts in the financial domain will be presented.

2.1 Machine learning basics: binary classification

To develop a proper understanding on the tasks this thesis aims to tackle, it is good to have an understanding of the fundamental tasks of machine learning. The task of binary classification is a task in which some input $x$ from domain $D$ is given as input and a function $f : D \to \{0, 1\}$ is used to classify $x$ as either zero (first class) or one (second class) [76]. The two targeted classes vary per domain and machine learning task. Examples of binary classification are found in polarity detection (explained in Section 2.4.1), disease detection and spam detection.

2.1.1 Loss functions

One of the convenient properties of binary classification is the assessibility of the loss generated by a model. Loss is given by a numerical value defining the number of misses (i.e. wrong classifications) made. In binary classification, a miss is made whenever the predicted outcome has one class, whereas the true outcome is of the other. Keeping this in mind, one can define the different loss functions for binary classification. For this, the loss functions $L$ are defined using a dataset $X$ consisting of $N$ data points. Each data point $x_i \in X$ has a true label $y_i$ and a predicted label $f(x_i) = \hat{y}_i$. The following are examples of common loss functions for binary classification [95]:

- **Squared loss**: The number of misses divided by the number of observations in total, i.e.
  \[
  L_{\text{square}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2
  \]

- **Cross-entropy loss**: also known as log-loss and is most commonly used in a logistic regression task. The loss function emerges from the taking the logarithm of the maximum likelihood estimator.
  \[
  L_{\text{ce}} = \frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) - (1 - y_i) \cdot \log(1 - \hat{y}_i)
  \]
2.2. MULTI-CLASS CLASSIFICATION

- **Hinge loss**: Mostly used in support vector machines, this loss function takes the classes -1 and 1 as potential outcomes and translates it to a positive number.

\[ L_{hinge} = \frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - y_i \cdot \hat{y}_i\} \]

2.1.2 Evaluation metrics

Besides the convenient use of loss functions, binary classification has a benefit in the ease of performance assessment. Depending on the context of the classification task, misclassification of labels might have different impacts. For example, in spam detection, there is a much higher loss if a valid email is classified as spam. To assess these context specific metrics, distinctions of the different outcomes are introduced. These outcomes are found in Table 2.1 [76].

<table>
<thead>
<tr>
<th>Input label</th>
<th>Predicted</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Naming of different classification outcomes.

From these different outcomes, one can define the following metrics to assess the quality of a model:

- **Precision**: What portion of the positively classified inputs is truly positive?

\[ \text{Precision} = \frac{TP}{TP + FP} \]

- **Recall**: What portion of the positive input labels are classified as positive?

\[ \text{Recall} = \frac{TP}{TP + FN} \]

- **Accuracy**: What portion of the input labels is correctly classified?

\[ \text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \]

- **F1-score**: Metric to measure the weight of precision and recall against one another.

\[ F1-score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \]

2.2 Multi-class classification

Although binary classification solves many machine learning problems, it might not be suitable for all models as the limitation of two output classes might not suffice. The classification problem can be generalized to a multi-class variant, allowing an arbitrary large number of output labels. These problems aim to find a function \( f : D \rightarrow T \) mapping input features from some domain \( D \) to an output label space \( T \) in which all possible output labels are found. Note that one can now choose the label space \( T \) to contain any suitable labels for the problem the classification task should solve.
2.2.1 Evaluation metrics

In binary classification, evaluation metrics were constructed making use of the four outcomes a model can attain: true and false positive and true and false negative. Multi-class classification models do not directly offer similar notions, so the evaluation metrics should be altered to suit the output space $T$.

The used generalization of the metrics are given by a piece-wise version thereof for every element in the label space. Let $t \in T$ and treat the classification problem as a binary one, having output classes $t$ and not $t$. Now, using this problem, one can calculate precision $pre_t$, recall $rec_t$, accuracy $acc_t$ and F1-score $F_1$, using the binary metrics defined in Section 2.1.2. Doing this for every element of $T$, the overall metrics are defined by the average of metrics over all elements.

- **Precision:**
  \[
  \text{Precision}_{\text{multi}} = \frac{1}{|T|} \sum_{t \in T} pre_t
  \]

- **Recall:**
  \[
  \text{Recall}_{\text{multi}} = \frac{1}{|T|} \sum_{t \in T} rec_t
  \]

- **Accuracy:**
  \[
  \text{Accuracy}_{\text{multi}} = \frac{1}{|T|} \sum_{t \in T} acc_t
  \]

- **F1-score:**
  \[
  \text{F1-score}_{\text{multi}} = \frac{1}{|T|} \sum_{t \in T} F_1_t
  \]

Having these metrics at hand, one can assess the performance of both binary and multi-class classification models. Note, in essence, these metrics work on batches of data, assuming that many observations are classified and assessed at the same time. When considering a streaming model, one needs to take into account that these metrics will change on every classified observation. Further investigation into this adaptive behavior will be elaborated on in Section 5.

2.3 Natural language processing

Natural language processing (NLP) denotes the task of analyzing textual, unstructured data in an automatic manner. NLP is seen as the intersection between computer science and linguistics and has a broad application portfolio in a wide range of fields and industries [21][82]. Examples of applications are, among a wide spectrum, found in medical decision support [28] and financial forecast support systems [89][85].

2.3.1 Information retrieval

One of the sub-domains of NLP is the field of information retrieval. Information retrieval (IR) denotes the task of finding documents from a large collection satisfying some input user need or purpose [69]. Within IR, there are numerous tasks at hand, such as querying documents, ranking page relevance and identifying document distances [18]. Serving as the fundamental task in NLP, it gave rise to different relevant metrics, of which one will be defined next.
2.3. NATURAL LANGUAGE PROCESSING

TF-IDF

One of the metrics within IR which is widely known is TF-IDF. Term frequency-inverse document frequency (TF-IDF) is a scoring model for assessing relevance some input word [69] given some collection of documents. As one might induce from the name, this metric combines two scoring models, namely intra- and inter document word scoring, term frequency and document frequency respectively.

TF-IDF takes a collection of document (corpora of words) as input. Per document, the words are counted (term frequency). Then, the occurrence of a certain word in all documents is counted (document frequency). The score is obtained by taking the inverse of the document frequency over the number of document and multiplying it with the term frequency. Formally, let $D$ be a collection of documents and let $x_i$ be word $x$ in document $i$ be the word of interest. The TF-IDF score of $x_i$ is given by:

$$\text{TF-IDF}(x_i) = \text{TF}(x_i) \cdot \text{IDF}(x_i) = \sum_{w \in i} I\{x_i = w\} \cdot \frac{|D|}{\sum_{d \in D} I\{x_i \in d\}}$$

Note, there are other options for the functions TF and IDF which, for example, normalize or bound the scoring.

Note how TF-IDF scoring can be used to translate corpora of text into numerical representations. This representation can be used in, for example, a machine learning model to translate the text into some useful information.

Additionally, note the necessity of availability of all the whole collection of documents when applying TF-IDF. If one is not able to access one of the documents (because it is not loaded or not available yet), he/she is not able to calculate the full IDF score. Alternatively, one can use an incremental IDF score, which adapts to new documents and calculates the scores ‘on-the-fly’. This approach has been used in web crawling environments in which pages are visited one-by-one and not all information is readily available [102].

2.3.2 Deep learning

A special field in NLP is the use of deep learning strategies for data analysis. Deep learning denotes using artificial neural networks for data transformation and processing to solve complex tasks [82]. In particular, the use of recurrent neural networks is common in tasks concerning textual information. These types of networks look at a sentence as a sequence of words. A recurrent neural network analyzes a word by looking at the target itself and its neighbours (i.e. words before and after it). By using the correct parameter configurations, neural networks can be used to analyze text and solve numerous NLP problems with very high accuracy [89][82].

Note, although the power of deep learning approaches is proven to be very high, neural networks tend to be computationally heavy to construct (i.e. training the networks costs a lot of time and resources). Now, considering an online learning or streaming environment as studied in this thesis, the high latency of training such models will pose challenges for the high velocity at which data might arrive. More detail on this challenge is given in Section 2.7.5.
2.4 Sentiment analysis

Sentiment analysis (SA) concerns the task of assigning emotion to words, sentences or corpora of text. The task is one of the sub-problems within NLP in which textual, unstructured data is translated to structured entities in some sentimental spectrum. SA is used in social network analysis, customer review reflection and many other fields in which the information of the sentiment of a piece of text defines a significant role in the analysis process [86][8]. The relevance of the topic is high as the number of articles is increasing exponentially over time, as is seen in Figure 2.1 [50].

![Figure 2.1: Number of articles written on sentiment analysis over time.](image)

This section will review the current state of sentiment analysis techniques. This review is split in different parts to get a clear overview of the different methods used in literature. First, the most fundamental task of polarity detection (i.e. positive or negative sentiment) is presented to get basic understanding of how sentiment analysis can be applied. After this, the more complex (and recent) hourglass model for sentiment classification is analyzed. Knowing the goals of analyzing sentiment, an overview is given of the two principle methods to achieve these tasks: lexicon-based approaches and machine learning-based approaches.

2.4.1 Polarity detection

Polarity detection is one of the fundamental tasks in sentiment analysis in which a model is created having two (or three) output classes, given by positive, negative and, if needed, neutral sentiments [57]. Using the polarity of a piece of text, one is able to assess the intention of a piece of text in more detail.

A polarity detection model can be described by a function $f : D \rightarrow \{P, N\}$ mapping some domain $D$ of input features to either a positive or a negative class, such that, for $x \in D$, $f(x) = P$ if $x$ holds a positive sentiment and $f(x) = N$ is $x$ holds negative sentiment. An example of such function is $f = g \circ I$, with $g : \mathbb{R}^n \rightarrow [0, 1]$ a linear regression function with weights $w \in \mathbb{R}^n$ and $I : [0, 1] \rightarrow \{P, N\}$ an indicator function with threshold 0.5, given by
2.4. SENTIMENT ANALYSIS

\[ f(x) = I(g(x)) = I(w \cdot x) = \begin{cases} P & \text{if } I(x) > 0.5 \\ N & \text{if } I(x) \leq 0.5 \end{cases} \]

in which \( x \) is some numerical representation of the input text (e.g. the TF-IDF vector). In words, this function translates a numerical representation of an input vector \( x \) to some score between 0 and 1 (e.g. the probability a sentiment is positive). If the regression model tends more towards 1, the sentiment is classified as positive. Otherwise, the sentiment is negative.

Although a seemingly simple model, it is still popular in literature as the quality is very assessable [47][86]. Especially the two-class polarity model can be seen as a binary classification task, thus opening the possibility of using the metrics defined in Section 2.1.

2.4.2 Hourglass of emotions

Expanding the view of polarity in sentiment, the Hourglass of Emotion model is introduced [13]. This hourglass model defines a more broad view on the classifications of emotions into different partitions. A 3D representation of this hourglass is found in Figure 2.2. The hourglass is to be read by means of the different quadrants, which are each others opposites. The three dimensions in the model give rise to metrics in which one can assess the orientation (positive or negative), type (what kind of emotion) and severity (how strong is the emotion). Applications of this model are found in the wordnet created by the same authors [14] and frameworks using this wordnet [4].

Next, the different approaches known for sentiment analysis will be deepened out. As is surveyed by [73], a division of methods in two parts is common: lexicon-based and machine learning-based methods. A visualization of the distinction made for techniques is found in Figure 2.3. These methods denote the current state-of-the-art for sentiment analysis. However, to develop a full understanding of the task, the fundamental method of rule-based classification is described first.

2.4.3 Rule-based methods

The first methods developed for sentiment analysis are found in rule-based methods. These methods make use of a pre-determined rule-set to classify the sentiment of a piece of text. The set can be enriched over time, adding more rules. An example of a rule-based method given next. Consider the following sentence for which the polarity needs to be classified:

\textit{The lovely princess liked eating the bad apple.} —Positive

Now assume one wants to build a rule to classify this sentence. A key word giving the sentiment in this case is ‘loved’. Hence a good rule for this collection would be:

\textbf{If sentence contains} \textit{liked} \textbf{then output} \textit{Positive}.

This simple system would classify this one sentence correctly. However, if another sentence is added, the behavior would perhaps be too simple, so new rule should be added.

Despite that rule-based systems seem to provide a powerful method for sentiment analysis, it brings challenges. One of such challenges is scalability of the rule-set. In rule-based systems, edge-cases to a model can easily be fixed by adding a new rule. However, when considering a large collection of text, one can imagine the number of rules growing rapidly to a very complex system.
Although these challenges may be present, making use of rule-based systems is still popular among analysts. For example, in the hybrid approaches mentioned in Section 2.4.6, one of the sub-models used is a rule-based one. Hence, rule-based methods denote a simple, yet powerful method in sentiment analysis.

### 2.4.4 Lexicon-based methods

The lexicon-based approach of sentiment analysis denotes the task of giving a score of emotion (polarity or more complex annotation) in a certain document using the semantic orientation of words in that document [101]. The semantic orientation of a word is a measure of subjectivity and opinion in text, capturing a polarity and strength (degree to which the word is positive or negative) towards a topic, person or idea [81]. In lexicon-based approaches, this orientation is used as an assumption of words having a prior polarity, i.e. their polarity is independent of the context in which they appear. Semantic orientation is a topic well-studied in the field of psychology and linguistics and is nowadays used in constructing lexicons for domain specific tasks [44].

To develop better understanding on how lexicon-based sentiment analysis works, the example from the previous section will be reviewed again from a new perspective. Recall the target sentence to be:

*The lovely princess liked eating the bad apple.*
Assume one wants to use a lexicon to classify the sentiment of this sentence, with an output label of either positive (+1) or negative (-1). Consider the following example lexicon consisting of words and their corresponding sentiment scores:

<table>
<thead>
<tr>
<th>word</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>+1.00</td>
</tr>
<tr>
<td>bad</td>
<td>-1.00</td>
</tr>
<tr>
<td>lovely</td>
<td>+0.75</td>
</tr>
<tr>
<td>liked</td>
<td>+0.66</td>
</tr>
<tr>
<td>hated</td>
<td>-0.66</td>
</tr>
</tbody>
</table>

In lexicon-based sentiment analysis, one is interested in the words that appear in both the sentence and lexicon, hence the words *lovely* (+0.75), *loved* (+0.66) and *bad* (-1.00). The sentiment of the sentence is determined by taking the sum of the scores of overlapping words. Therefore, the sentiment is given by

\[ 0.75 + 0.66 - 1.00 = 0.41 \]

Finally, because this score is positive, the article is considered to be positive and hence the classified label is +1. In practice, many studies have been written on the construction and active deployment of sentiment lexicons. Examples of such are the WordNet lexicon introduced in 2002 [57], which is later on refined to SentiWordNet 1.0 [33] in 2006 (author note: no 2.0) and 3.0 in 2010 [7] and SenticNet [14].

The main power of lexicon-based sentiment analysis comes in two-fold: transparency and efficiency [86]. Transparency is found in the construction of the dictionaries. As dictionaries, in essence, can be constructed completely by hand, a constructor is able to incorporate domain specific jargon rather than generic emotional annotations. Note, however, that lexicons can also be generated automatically, creating context-aware dictionaries. An example of such automatically created lexicons is the earlier mentioned SenticNet which makes use of deep learning methods [14].
2.4. SENTIMENT ANALYSIS

The latter power of lexicons is efficiency. Essentially, a lexicon is a word list containing just the necessary terms with their ’meaning’ (i.e. semantic orientation score). The task left for a model is hence doing a lookup on in the lexicon and aggregate the results of the textual body. This lookup is usually fast (logarithmic time over the number of elements in the dictionary).

In a stream mining context, an important feature of lexicons is adaptiveness of word weighting. Over time, the load of certain words might change, words evolve (i.e. new forms or representations) or new words are introduced. This behavior over time should be accounted for. Examples of existing methods in this field previously proposed online sentiment analysis using (heuristically) labeled observations [6] or making use of existing lexicons (such as SentiWordNet 3 [7] or Thelwall-Lexicon [92].

Apart from these, different methods exist on expanding an existing lexicon to a context-dependent one. These architectures start by looking at a renowned general purpose lexicon and iteratively (per observation) expand it to fit the needs of the context. This approach is applied in expanding the Thelwall-Lexicon [92] and in an unsupervised manner using a manually constructed lexicon [58].

2.4.5 Machine learning-based methods

Besides making use of lexicons, more recent methods make use of machine learning-based approaches to assess sentiment scores. As is shown in Figure 2.3, many ML techniques are constructed in a supervised manner, making use of many different classifiers [73]. Practical examples of machine learning-based sentiment analysis are found in automatic processing of customer reviews using bayesian networks [20] or analysis of social media during natural disasters, such as hurricanes [68].

In a streaming fashion, different ML methods for sentiment analysis are known. These methods make use of existing classifiers for online learning. More information on these methods is found in Section 2.7.

2.4.6 Hybrid methods

Instead of choosing either a lexicon or a machine learning model for sentiment analysis, a common approach is a hybrid one in which, most typically, an ensemble or combination of the different methods is used. In such an ensemble, one is classifying sentiment using different methods (e.g. using a lexicon and an ML model) and outputs the result constructed from all methods. The way the results of the sub-models are transformed in the final result may vary in different settings. Examples of transformations are taking the (weighted) average or maximal score over all results.

Examples of hybrid methods in sentiment analysis are found in brand monitoring using Twitter data, in which both a fixed lexicon and machine learning methods are used to assess customer satisfaction [37], the combination of three models (rule-, lexicon- and ML-based) methods [8] and an hybrid approach to assessing sentiment in paper reviews [60]. These studies all show better performance using hybrid approaches over using a single model.
2.5. SEMANTIC WEB MODELLING

Although showing promising results, hybrid models will typically be less efficient in terms of running time since the methods require the construction of multiple models. When a new observation is classified, the methods require classification by all sub-models after which these results are aggregated. Obviously, this will yield a linear increase of running time over the number of models considered in the ensemble. Especially when considering a streaming context, a high processing time as such could have a very negative impact on the overall model performance.

2.4.7 Multi-modal methods

A special case of hybrid sentiment analysis is one in which the ensemble makes use of multiple sources of data, potentially having a different information representation [99]. This approach is known as multi-modal. In natural language processing, an example of a multi-modal approach is, in addition to considering textual corpora, using speech data to aid the analysis of a certain topic.

Applications of multi-modal approaches are found in the enrichment of rule-based textual analysis by images analyzed by a neural network model [12]. Furthermore, as is surveyed in [99], multi-modal approaches are used in the analysis of TV-programs by automatically summarizing content and identification of politically persuasive content from multiple sources.

2.4.8 Multi-lingual methods

Especially in a more international view on sentiment analysis, one might consider involving multiple languages into the model. Multi-lingual models are typically not linked to a specific piece of textual information but rather focus on the entity or event described therein [27]. This entity is analyzed using many different text corpora in different languages, thus providing multiple views on the topic. Examples of such methods from literature are found in the analysis of radicalisation using YouTube transcriptions and video comments [9].

For the research proposed in this thesis, the multi-lingual aspect of sentiment analysis will be deemed irrelevant as authors tend to write articles in a single language. Looking at the aim of the thesis, one might consider incorporating more languages when authors write in a multi-lingual manner (e.g. they write for different news papers or a paper offers articles in multiple languages).

2.5 Semantic web modelling

With an ever growing number of information sources available on the World Wide Web, the increase in complexity gave rise to many challenges in web-based online analytics. In its traditional form, the world wide web consists of numerous unrelated sources making use of different data formats, representations and languages. The main idea of the semantic web is using these data sources, together with a common unified understanding of the represented data. This understanding is formed by linking entities in this common understanding together by their corresponding relationships.
2.5. SEMANTIC WEB MODELLING

2.5.1 Resource description framework (RDF)

The representation most often used for semantic web information is that of the resource description framework (RDF) [15]. RDF is a standard proposed by W3C [111] for exposing information on the semantic web. The standard makes use of the following structural representation of data:

 subject - predicate - object

The interpretation of this representation is that a resource (subject) is linked to a value or other resource (object) by a certain relation (predicate). An example of such relations is:

 John - hasAddress - Calle Gran Vía 3, Madrid, Spain

In essence, one can store any triple of these entities in the RDF framework. However, using only the textual information might lead to ambiguity. For example, one datastore might use the predicate "hasAddress" while the other uses "has-address". To uniquely define these objects, subjects and predicates, RDF makes use of URI’s. These URI’s link semantic entities to their corresponding origin and purpose. Again, considering the address example, one will see

 http://www.w3.org/2006/vcard/ns#hasAddress

This new notion of the address predicate is less ambiguous and can be universally applied. In RDF, groups of common entities can be documented into a common agreement on its semantic representation, which is called an ontology [108]. An example of the ontology holding the address predicate seen earlier is the Vcard ontology, provided by W3C in 2006 [110].

Reconsidering the rolling example of John’s address, one can describe the triple in the following way:

 http://example.com/John http://www.w3.org/2006/vcard/ns#hasAddress "Calle Gran Vía"

In the example, John is now an online resource described by its URI on example.com, having a commonly understood address with value Calle Gran Vía which is a String object.

Note, RDF only provides the framework for semantic knowledge representation. To use the framework, multiple data structures are available, such as XML (RDF/XML), turtle, JSON-LD, etc. [109]

Making use of RDF representation, one can construct a graph of linked information. This graph is known as a knowledge graph in semantic web modelling. The knowledge graph can be traversed in order to find information related to specific entity of interest. To traverse and search for information on this graph, however, a suitable query language is needed.

2.5.2 SPARQL

To make efficient use of the semantic web information, a new query language is developed in 2008 [3]. The language SPARQL made convenient use of the triple data structure of RDF and gave rise to the possibility of linked data manipulation and querying. SPARQL has a structure similar to SQL.

SPARQL works by matching graph patterns with the necessary information. These graph patterns are given by the subjects and objects as vertices and the predicate as directed edge from subject to object. SPARQL uses triple patterns to match a subject, predicate and object and retrieve the information needed by a user.
2.6. NAMED ENTITY RECOGNITION

Queries are constructed by providing known and unknown variables of a triple pattern. The unknown variables, prefixed with a "?", describe a pattern to which a triple should adhere. The known variables provide constraints on this pattern. The known variables can be seen as filters of the matching patterns. For example, to query of address of John, the following SPARQL query can be used:

```sparql
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
PREFIX example: <http://example.com/>
SELECT ?address
WHERE {
  example:John vcard:hasAddress ?address
}
```

The `WHERE` clause in the query describes the pattern which should be matched. The known variables are the resource subject `John` from `example.com` and the `vcard` predicate `hasAddress`. The unknown variable if the object of the triple, which, in this example, is the String holding the address. Using the `SELECT` clause, one can extract this unknown variable from all matching triples in the queried datastore.

### 2.6 Named entity recognition

Another task in natural language processing relevant for the work in this thesis is that of named entity recognition (NER). NER concerns the task of tagging words with a specific value or purpose in a sentence [70]. For instance, consider the following sentence:

*John went to Carrefour to do his groceries.*

The task of NER is to identify the entities of this sentence. One might interested in the persons and companies in the text. The task of an NER-algorithm is to tag them in the sentence. Ideally, this would result in the following output: Examples of tasks solved by NER are found in automatically processing regulatory documentation [78], decision support in different domains [85][28], automatic analysis of doctors notes and many other examples [80]. Many readily available models exist in different programming languages. The ones used in the remainder of this thesis will be defined in Section 5.1.

#### 2.6.1 Entity linking

A sub-field in named entity recognition is the task of named entity linking (NEL). In addition to NER, NEL concerns linking the entity found in the text to an open data source on the web [42]. Examples of such sources are Wikipedia [41], Wikidata [40], DBpedia [22], etc. Creating a link to these platforms helps in disambiguous tagging of text. An example is finding the difference between an apple (fruit) and Apple (the company). Linking entities to open linked data sources helps in developing the underlying meaning of a sentence.

Referring back to the example of John doing his groceries, the extended task of NEL is linking the entity Carrefour to some known open linked entity such that one can understand the subject better. A prospected result of an NEL tagger would be:
2.7 Stream mining and online learning

Current trends in technology result in data being generated in larger amounts and in a much higher velocity. Because of this trend, analysis of data sources in so-called batch mode grows less tractable in a growing number of cases anymore, as data behavior might change fast and storage will become expensive. A new field in data analysis is the technique of looking at data on-the-fly and adapting a model to newly observed behavior. A visual representation of the difference between batch- and stream data mining is found in Figure 2.4.

Online models look at data points as a sequence of information in which every point can only be viewed once and in order [46]. The main benefit of online learning is given by not having to store batches of data anymore. As data is flying by and the model looks at an observation only once, data does not have to be stored persistently. Because of this, data sources for online learning models are scalable without having a need for more storage resources. Unfortunately however, the reduction of storage need comes at a cost. As observations can only be viewed once and classification (and adaptation) need to be done fast, online learning models typically provide less accuracy.

The remainder of this section will review the state of the current knowledge by first introducing the main problem online learning tackles, after which techniques and frameworks are explained.
2.7. STREAM MINING AND ONLINE LEARNING

2.7.1 Concept drift

A common problem solved by online learning techniques is that of concept drift. Concept drift means that, over time, the behavior of a data-stream will change because the analyzed concept is dependent on some unknown, hidden context [114]. This change is either sudden or could happen in a gradual manner.

A visualization of a simplified representation of concept drift is given in Figure 2.5. When concept drift occurs, batch-based methods in machine learning will lose performance over time, whereas the task of online learning is to adapt to this new behavior. Concept drift occurs in many different situations in which data in considered over a larger period of time.

Examples of occurrences of concept drift are found in weather forecasts (in which the seasons influence temperatures), stock prices (linked to the efficient market hypothesis mentioned in Section 2.8) or customer buying preferences [106]. These specific examples all show gradual concept drift in which there is not one specific point in time at which a shift is seen. The challenges in these examples are solved by making use of adapting online algorithms.

2.7.2 Adaptive models

The most common solution in online learning is the adaptation of models to new observations of data. Adapting, in this sense, has two main tasks: learn and forget. Learning means incorporating the information contained in new observations into the model. Forgetting is the task of not taking old observations into account. Forgetting is especially important when concept drift occurred because the old behavior needs to be forgotten as soon as new behavior is observed.

Examples of adaptive models in practice are used to keep track of the presidential elections in the USA in 2012 [112]. This study analyzed Twitter data subjecting the elections, classifying the sentiment thereof and thereby predicting who will become the new president. Another study developed an online method to identify topics of real-time data flows using Latent Dirchlet Allocation (LDA). Lastly, new methods for optimizations in an online fashion are being developed. The usual gradient descent algorithm to optimize parameters of a model has been extended to an adaptive variant which is known as Adagrad [30]. Besides these examples, the research into online learning is becoming a more and more popular field, as is seen in the number of articles written on the topic, seen in Figure 2.6.
2.7. STREAM MINING AND ONLINE LEARNING

As is seen in Figure 2.4, stream data mining, in essence, only makes use of observations once and after that they are discarded. In practice, one might be interested in looking at patterns in a certain sequence of observations and hence should maintain a memory of some sort. For example, one might consider analyzing a highway in which cameras register the whereabouts of cars driving. If you would like to know if a car is speeding in ten consecutive camera registrations, one need to collect all information and analyze all of it at once.

In order to do this, stream mining algorithms make use of windows. A window is a time- or count-based snapshot of observations. Time-based windows have a size $s$, open at a timestamp $t$ and close at time $t + s$. Count-based windows of size $n$ count the number observations and close after $n$ elements are counted. Once a window is closed, the observations therein are collected and can be analyzed together. Using varying window sizes, one can enlarge or shrink the memory of observations considered.

Common window types are tumbling, sliding and global windows. Visualizations of the behavior of these windows is found in Figure 2.7. Tumbling windows will collect observations and, after closing, discard all previous observations for the new window. Sliding windows will collect observations and, once closed, will slide with a certain number to collect new elements. In sliding windows, observations might be used in different windows and hence analyzed multiple times. Global windows will collect every observation that is observed. Global windows are mostly combined with custom triggers to close the window. For example, once a stopped car is registered, analyze its full trajectory up and until that point.

Windowing is a common technique in stream mining algorithms as it provides means to cluster the information flying by and create chunks or batches of data which can be analyzed. The popularity of the technique is, among others, found in data stream frameworks such as Apache Flink [16], Microsoft Azure Stream Analytics [72] and SQLStream [100][71].
2.7.4 Evaluation metrics in stream mining

The evaluation metrics defined in Section 2.1.2 apply to a batch of information. In order to calculate any of the metrics, one would first need the outcome of all validation set entries. Although giving useful insights of a model performance, in stream mining one is more interested in performance over time. Therefore, there is a need for adapted metrics in order to show the behavior over time. For this, the sliding window described in Section 2.7.3 is particularly useful. The sliding window over the validation observations can be used to calculate the performance metrics over time.

Formally, let $W$ be the articles in the current window. Now, similar to the previous definition of the metrics, one can look at $TP_W$ (true positives in $W$), $FP_W$ (false positives in $W$), $TN_W$ (false negatives), $FN_W$ (false negatives) and define the evaluation metrics by:

- **Precision:**
  \[ Precision_W = \frac{TP_W}{TP_W + FP_W} \]

- **Recall:**
  \[ Recall_W = \frac{TP_W}{TP_W + FN_W} \]

- **Accuracy:**
  \[ Accuracy_W = \frac{TP_W + TN_W}{TP_W + FP_W + TN_W + FN_W} \]

- **F1-score:**
  \[ F1-score_W = 2 \cdot \frac{Precision_W \cdot Recall_W}{Precision_W + Recall_W} \]

Reporting the values for these metrics over time, one will get insights into the model performance on different points of deployment.
2.7.5 Online deep learning

Although having a relatively high complexity, minor research has been done towards online deep learning [90]. Traditionally, the training algorithms and parameter optimizers of artificial neural networks are determined by viewing the whole dataset numerous times. Doing this task in an online setting, thus viewing each observation once, is a challenging task. The current research into this topic constructed a method using only a single layer of neurons (also known as a ‘shallow learning’ model). Although the research opens possibilities for deep online learning, it is deemed too simple for the tasks performed in this thesis and hence will not be handled any further.

2.7.6 Hybrid approaches

Instead of concerning a pure online environment, one can construct a windowed approach to the adaptation of the model. In this approach, one slices the stream into (potentially overlapping) batches of data on which the model is trained. This trained model is used to classify data until the next batch of data is analyzed and the model is adapted to the new observations.

Hybrid online learning can be seen as an adaptation to the second structure overview given in Figure 2.4. The overview explains that after each new observation, there is a combination of predict and adapt. This combination is still present in a hybrid online learning environment, yet one only re-trains the model after a set of chosen size of observations is received. Once a model is adapted, the new batch is predicted with this model, after which is used to, once again, adapt it.

2.8 Machine learning in the financial domain

Machine learning in financial analysis is popular topic in research. A rapidly increasing number of studies has been conducted into financial forecasting. The number of articles written over time on the topic is shown in Figure 2.8. Especially in the past few years, an exponential increase of articles can be seen, hence the topic is deemed relevant to data scientists.

In its most essential form, the flow of stock price can be seen as a time series. Investigation into this series is a popular topic and has been studied very well over the past years [116]. Despite this exhaustive research, previous literature empirically proved the inability of predicting stock prices solely using information on the history thereof. This phenomenon, known as the efficient market hypothesis (EMH) [67], posed a challenge to analysis of stock prices as a classical time series. The hypothesis states that a company’s stock price is directly and efficiently affected by (reported) events of any kind having an impact on the image or operations of the company. News articles written on such events are empirically shown to have a high impact on stock prices and hence are acknowledging the hypothesis [67].

To fill this information gap, many studies shifted towards incorporating the use of external information sources to enrich the knowledge on stock prices, thus providing better predictive models. Natural language processing for financial forecasting is used in analysis of textual data, mostly originating from either Twitter or newspaper articles. These tweets or articles improve the performance of financial decision-making [2].
2.8. MACHINE LEARNING IN THE FINANCIAL DOMAIN

Figure 2.8: Articles on financial analysis using textual data mining over time [50].

2.8.1 Natural language processing in financial forecasting

As natural language processing techniques become more popular among data scientists, their impact on financial analysis grows as well. In line with the EMH, stating that the market gets significantly and quickly influenced by the news articles written thereon, NLP approaches target unstructured textual data as enrichment to their performance.

This trend is not a novel innovation in the field, as more heuristic methods for such analysis were already used more than twenty years ago. Rule-based analysis was used on financial corpora as enrichment to an existing time series analysis [115]. Performance did not show promising results and, over time, the techniques to use the combination of data sources improved significantly. Considering machine learning-based techniques, support vector machines were used to achieve the same predictions, yet with higher accuracy [93]. More recently, this method has been enhanced with embedding the textual data into TF-IDF vectors [31].

Besides traditional machine learning, the rise of the deep learning methodology opened new ways of financial analysis [116]. Many different approaches have been proposed for the use of textual information to predict stock prices of companies, currency exchanges [56] as well as gross domestic products [105]. The first of such articles makes use of simple neural networks with varying structure [11], whereas newer studies make use of more complex convolutional neural networks [29], recurrent neural networks [48] or combinations thereof [17][89][119].

More related to the proposed work in this thesis is the analysis of stock prices using semantic web-based data solutions. The work in [85] enriches the performance of existing financial decision support systems with open linked data adhering the domain specific ontology.

Besides these approaches, other methods are proposed to tackle somewhat similar problems. For example, rather than predictive analysis, Stock Sonar handles descriptive analysis, automatically assessing the structure of a stock making use of sentiment analysis methodology on batches of collected financial news articles on a daily basis [35].
Chapter 3

Thesis Statement

Having the knowledge of the state-of-the-art in sentiment analysis, named entity linking, stream mining and their applications in the financial domain given in Chapter 2, it is now possible to define the main tasks performed in the thesis. Along with the tasks, the requirements and specifications of the final system will be given. The described tasks will be guided by data presented in Section 4. The task of the thesis will be specified in terms of different research questions and a narrowing scope. These two main parts of the thesis statement will be defined next.

3.1 Research questions

The thesis will describe financial news analysis in a stream mining context. The work will assess the quality of news article authors in the financial domain. To analyze the articles written by these authors, two main techniques are used: sentiment analysis and named entity linking. Sentiment analysis, as is specified in the state-of-the-art, is generally denoted by two main lines of study, lexicon-based and machine learning-based methods. Because of its low latency and transparency, the main method studied in this thesis is a lexicon-based one. The following questions will be investigated for this specific task:

Q1 With what performance can the sentiment of financial news articles be classified using sentiment lexicon methodology?

Q1.1 With what predictive performance do different existing pre-trained lexicons perform on financial news articles over time?

Q1.2 Which of these existing methods on sentiment analysis performs best?

Q1.3 Does a stream lexicon adaptation method achieve higher accuracy than batch-based ones for sentiment analysis over time?

Together with the sentiment of an article, the targeted company and industry need to be identified. For this, named entity recognition and linking tasks elaborated in Section 2.6 will be utilized. These techniques will be used to extract the target company from an article. After this, the found company can be linked to a semantic entity on an open data platform, as described in Section 2.5. For this, the thesis aims at answering the following questions:

Q2.1 With what accuracy can a company targeted by a news article be recognized from the text of a news article?

Q2.2 What portion of the identified companies can be correctly linked to their corresponding semantic web entity using named entity linking?
Finally, having the information on the industry and sentiment of the article, the final result of the architecture can be constructed, thus describing the expertise and impact of article authors. For this final task, the following questions will be investigated:

**Q3.1** Which streaming environment will process financial news articles with the lowest computational complexity?

**Q3.2** With what accuracy can the field of expertise of financial news article authors be described given the sentiment of their article and companies they write about?

### 3.2 Scope of the thesis

The main scope of the thesis is developing a novel solution in mining unstructured data streams of financial information using sentiment analysis and named entity linking. In order to achieve this, the scope of the work is narrowed down to the specific task described in Section 1. This scope will be defined in terms of the used data, system requirements and corresponding system evaluation.

Additionally, natural language processing has a high dependency on which languages are considered. First, one can consider that most resources on the sentiment analysis and named entity recognition task are available and have the most quality in English. Also, the source of data (i.e. news papers) are most popular in English (e.g. Financial Times [39]). Therefore, the scope of the work will be narrowed to the English language.

#### 3.2.1 Data stream simulation

The thesis concerns data as a stream of information which arrives at a specific point in time. For testing purposes, this stream will be simulated. This simulation is achieved by using a stored batch of news articles and letting them pass by the system one by one based on their historic timestamp. In practice, this assumption would not hold since data would then be analyzed whenever it is observed. However, to assess the quality of the system, one needs to prepare and label the data upfront. More information on the used validation data can be found in Chapter 4.

#### 3.2.2 System development

The main scope of the thesis is to adapt existing methods for stream sentiment analysis and named entity linking towards the financial domain. The different building blocks of the thesis (as will be described in Chapter 5) are assumed to be behaving independently of one another. An implication of this assumption is that the optimal solutions to the sub-tasks (i.e. sentiment analysis and named entity linking) will also yield optimal results in the full system.

The main purpose of the thesis is to develop a system that performs the core descriptive analysis in line with the main task of financial news article author assessment. The final product of the thesis is hence a streaming environment presenting results of at least the used validation dataset. Additionally, a web interface presenting the individual results of the different building blocks is constructed.
3.2.3 System evaluation

The system will be evaluated using the metrics described in Section 2.1.2 and Section 2.2. There are four parts in which evaluation will be performed:

- **Sentiment analysis**: The sentiment analysis methodology presented in this thesis will be tested and the performance will be assessed. The results of this evaluation will serve as the basis to the answers of research questions Q1.1, Q1.2, Q1.3 and Q1.

- **Named Entity Linking**: The named entity recognition and named entity linking methodology presented in the work will be tested and its performance will be assessed. The result of this evaluation will serve as the basis to the answers of research questions Q2.1 and Q2.2.

- **Full system**: The sentiment analysis and named entity linking methodology will be combined in a stream data mining environment for real-time news article assessment. The computational performance of this complete system will be assessed. This result will serve as the basis to answer research question Q3.1.

- **Assess news article authors**: This part will focus on the main task tackled in this thesis. The impact of financial news article authors will be assessed by retrieving the stock prices of news articles and see whether they align with the sentiment of the article. The result of this part will serve as the basis to answer research questions Q3.2.

The full system evaluation will be done using a logical AND of the results of the first two evaluations. That is, if a model correctly classifies both sentiment- and NEL-entity, it is assumed to generate correct results for the full system. More specifications on the evaluation methods are found in Chapter 8.
Chapter 4

Data Gathering and Processing

One of the most important elements in a data science related study is the data used in the analysis. This data drives the findings and relevance of the study and hence needs to be taken into account from the start of a project onward. This section will explain the different sources of data considered in the work. Additionally, some basic techniques for data pre-processing are elaborated.

4.1 Data gathering

4.1.1 News articles

The used data will originate from newspaper articles in the financial domain. These articles are gathered using the academic platform LexisNexis Academic [49]. This platform provides an extensive archive of many different newspapers in a variety of languages, having a time frame between 1980 - 2019 (or during the establishment period of the newspaper between these years). The platform provides querying methods using author, keyword, newspaper, etc. This archive will be considered as the main source of information during the thesis.

Data provided by the platform consists of the textual corpus of the article as well as various pieces of metadata. The available metadata might vary per article. The features, if available, provided by the platform are found in Table 4.1. The bold-faced features are the ones necessary for execution of the thesis research and are hence obligatory to each used article.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>A textual timestamp at which the article is published.</td>
</tr>
<tr>
<td>Title/Headline</td>
<td>The title of the article.</td>
</tr>
<tr>
<td>Body</td>
<td>The full text corpus of the article.</td>
</tr>
<tr>
<td>News platform</td>
<td>The newspaper or website on which the article is published.</td>
</tr>
<tr>
<td>Author</td>
<td>The author of the article.</td>
</tr>
<tr>
<td>Section</td>
<td>The section of the newspaper or website the article appeared in.</td>
</tr>
<tr>
<td>Wordcount</td>
<td>The number of words in the article.</td>
</tr>
<tr>
<td>Language</td>
<td>The language in which the article is written.</td>
</tr>
<tr>
<td>Publication-type</td>
<td>Medium with which the article is published (paper, website, etc.)</td>
</tr>
<tr>
<td>Subject</td>
<td>Topic classification added by Nexis.</td>
</tr>
<tr>
<td>Notes</td>
<td>Additional annotations regarding the topic or author of the article.</td>
</tr>
</tbody>
</table>

Table 4.1: Extracted article features from LexisNexis [49].
During data gathering, one can take into account two dimensions: The number of news sources considered and the number of target authors considered. During the analysis, scaling of the handled data can be achieved by expanding is either of these dimensions (i.e. look for either more newspapers or more authors). Which of these scaling options will be chosen during analysis is left undecided for now, yet both are possible because of the large archive maintained by the LexisNexis.

As a starting point for the work in the thesis, a fixed list ten of authors will be considered. These authors are chosen because they are experts in a particular field, together covering a broad spectrum of industries and are frequent publishers of financial articles. In practice, this fixed list can be scaled in either of the two dimensions mentioned earlier in this section. Using only a small number of target authors provides the possibility to watch the behavior of the algorithm closely and identify improvements in a timely manner.

To fulfill the essence of the proposed thesis topic (i.e. online learning), the data downloaded from the archive will be handled as a stream. This means that, although all data will be available and stored on disc upfront, articles are handled one-by-one in an ordered, time-based manner. In the initial phases of the analysis, the number of authors considered is limited and hence one can imagine that the streaming aspect of the data is weak. However, once the used input data of the study is scaled, the information will become more variational, having a higher velocity, thus creating the need for the streaming environment.

### 4.1.2 Stock information

Stock information will be used to validate the prediction made by news articles in the validation dataset. The financial data will be handled in a daily manner, making use of the opening and closing times of stocks. The information will be retrieved from the platform Quandl [51] for Java and Yahoo Finance [52] in Python. Quandl is an open financial platform from which users can retrieve the historical information of company stocks from the start of their listing onward. Yahoo Finance provides similar functionality, providing users means to retrieve historical stock prices. Both platforms provide the following properties:

- **Date**: The day of the retrieved stock information.
- **Open**: The opening price of the stock.
- **Close**: The closing price of the stock.
- **Low**: The lowest price reached by the stock on that day.
- **High**: The highest price reached by the stock on that day.
- **Volume**: The number of bonds from that stock traded that day.

The essential information used in the analysis are the date, opening and closing price. This information can be transformed to a two-value time series which is used later on in the work. More information on this transformation is found in Section 5.6.

### 4.1.3 Open linked data

To make use of open information on companies of interest, semantic web modelling will be used. This data will be collected from known linked data sources. It could be the case that a specific data platform does not hold all necessary data. Therefore, multiple open platforms will be considered to ensure full coverage of companies. The following sources will be considered:
4.2 Data pre-processing

In order to tackle an NLP task, one should start by taking suitable methods to represent and work with the input corpora. The handled data is unstructured (i.e. text), hence very noisy and challenging to interpret for a machine learning model. To make the analysis more accurate and tractable, several data pre-processing steps are usually performed. The most common tasks for text pre-processing will be elaborated next.

4.2.1 Stop-word removal

Handling textual data means that the information adheres to grammatical and syntactical rules given by the language the text is written in. One of the negative implications of this property is the presence of noise in the data. Noise can be present in a variety of forms. One of these is the presence of stop-words. Stop-words are defined by words that are used commonly in text without holding any relevant information. Examples of stop-words are:

*the, a, an, to, and, in, each, etc.*

Because of their high frequency, these words will significantly influence the learning process of NLP models. Therefore, it is better to remove them during the pre-processing of the data. As the removal of stop-words is a common task in NLP, there are numerous libraries available for the task. Furthermore, most common toolboxes for NLP (see Chapter 5 for examples) contain functionality for stop-word removal.

4.2.2 Stemming and lemmatization

Another common challenge in natural language problems is word-morphologies. Because of semantic- and grammar rules, words might be morphed to different forms, without loosing their definition or essence. Examples of such words are:

*to be, am, is, are, were, was*

Note that all of these words are morphed forms of the verb ‘to be’, all having the same meaning (except for their tense). Stemming is the process of reducing the number of morphed words. This reduction is done by transforming the morphed words back into their general form. In the example, all words could be transformed their essential verb ‘to be’.

• **DBPedia** [22]: Started by the university of Berlin, DBPedia has grown to be one of the largest and most extensive source of open linked data.

• **WikiData** [40]: Serving as a main backbone for platforms like Wikipedia, Wikidata is a platform for open data hosted by the Wikimedia Group. Wikidata mainly holds metatags for information represented on Wikipedia.

• **OpenCorporates** [66]: OpenCorporates is an open linked data platform explicitly created to link information of corporations. The data store of OpenCorporates links with multiple open government portals to provide a user a complete overview of a target company.

All of the above mentioned platform make use of an API for semantic querying. These API’s can be used to execute SPARQL queries to retrieve linked information. More details on this process is provided in Section 5.4.
Lemmatization is a special form of stemming, which aims at reducing morphologies by removing suffixes from words. For regular verbs in English, this process stems words from past to present tense by removing the suffix -ed or the simple past singular third person -s. Examples:

\[
\text{worked, works, work} \quad \text{and} \quad \text{played, plays, play}
\]

Both stemming and lemmatization reduce the number of distinct words in a text corpus and hence provides means for better analysis. In practice, as was the case with stop-word removal, most common toolboxes for natural languages processing provide stemming and lemmatization functionality.

### 4.2.3 Numerical representation

Many text classification models in NLP require the input information (i.e. text) to be numerical. To achieve this, the words in the input corpora should somehow be transformed into a suitable numerical representation. To do this, several methods exist, all having their own pros and cons that come with it. The two main distinguished methods are Bag of Words and word embeddings.

The Bag of Words (BoW) representation transforms the textual information in a matrix representation. The columns of the matrix represent all distinct words present in a corpus and the rows represent the different documents in the dataset. For each document, the words therein are counted and the count is stored on the document row-word column in the matrix. Although easy to deploy, this method has two main limitations.

The first is loss of contextual information. The sentences "I did not do that although I am good at it" and "I did do that although I am not good at it" have the same representation but have different meaning. The second limitation is the columnar dimension of the BoW matrix. A relatively small dataset of 5000 articles can already hold more than 200,000 distinct words in it. This high sparsity will not be beneficial for model performance.

A different transformation are word embeddings. Word embeddings are a technique of transforming words into vector representation. By defining a vector space to represent words, it is possible to define distance and similarity between words. A classical example of the power of word embeddings is the following:

\[
\text{King} - \text{Male} + \text{Female} = \text{Queen}
\]

A special variant of word embeddings is the word2vec model introduced by Google in 2013 [63]. Word2vec is a ensemble of models to transform words into a vector representation. It utilizes neural networks to train the transformation on a corpus [74].

Note, word embeddings are particularly powerful when a considerate amount of data (i.e. more than 10,000 entries) is available. As this thesis handles less data, the method will not result in higher predictice performance compared to the bag of words model.

### 4.2.4 Other techniques

Many other techniques for data pre-processing exist. Specifically for lexicon-based sentiment analysis and named entity recognition in a practical context, there are a three techniques worth mentioning. The first concerns noise generated during the collection of data in the form of malformed characters. Upon sending bytes through a system or over the internet, it might happen that information is lost, hence meaning that the piece of text will have a malformed function.
In practice, this mostly means that certain characters from the news article are not parseable and hence cannot be used for analysis. The pre-processing step solving this problem will remove these words from the corpus, thus not considering them during the further processing.

The second technique is the removal of punctuation from the input corpus. Both named entity recognition and lexicon-based sentiment analysis as used in this work do not account for sentence-based word meaning and therefore punctuation is irrelevant. This removal will result in the news article being one long list of words, similar to a bag of words structure described earlier.

The last technique concerns transformation of numerical characters to their written numbers. For example, the sentence 'The dog saw 3 cats' will be transformed to 'The dog saw three cats'. This transformation is particularly relevant for sentiment analysis and different lexicons assign sentiment scores to certain numbers (e.g. thirteen might be seen as an unlucky number or hundred as positive when used as a percentage).

Note, this section aimed at providing an overview of the different techniques. Not all methodology described here is relevant to the research (e.g. the numerical representations will not be used). A definite list of relevant pre-processing steps is given in Section 5.2.

### 4.3 Data labelling

The online fashion in which the algorithm is constructed poses challenges for the supervised data mining. Especially sentiment analysis will make use of labels to validate a lexicon and potentially to adapt it to new observations. To validate a lexicon, the true label of an observation is needed. Published news articles typically do not contain such information and hence a strategy for labelling the articles is needed. In literature, several different approaches are known for this. These approaches will be elaborated next.

#### 4.3.1 Manual labelling

The most straightforward method to labelling the textual data is using manual labelling. Manual, in this sense, means that per article a human expert (i.e. who is able to identify sentiment) is needed to manually annotate the true polarity of an article. This approach will yield the most accurate results. However, it is also the most labour intensive strategy.

Note, in an online learning context, this approach is not feasible since it diminishes the instant analysis and adaptation of the algorithm. Although a human expert could label new articles quickly, letting him do so one article at a time for an indefinite period of time is intractable. Nevertheless, throughout the learning process, this approach can be used on a benchmarking dataset. One will be able to compare the results of one of the below mentioned approaches to the manual (true) annotations and see where key differences are found.

A possible solution to the high labour is delayed labelling. In this situation, there is a significant time-gap between receiving (and classifying) an observation and the arrival of its label. In this situation, it could be the case that multiple observations are already classified before they obtain a label and the model can be adapted. Adaptations to regular online models are described by [55].
4.3. DATA LABELLING

4.3.2 Gathering labeled data

Some studies concerning supervised sentiment analysis consider only labeled data sources as input. A classical example is the analysis of sentiment in customer reviews. The labels are retrieved from the rating (e.g. number of stars, performance figures, etc.) given by a user and the text is classified accordingly. To the authors knowledge, no such reviewing platforms or types of sources are available in the financial domain, hence this approach is not feasible for the case handled in the thesis.

4.3.3 Noisy-labelling

The noisy-labelling approach assumes the polarity of a piece of text to be directly linked to a fixed set of rules specified beforehand. It is called noisy because the rules used for initial labelling tend to be simplistic and hence create noise on the correctness of the labels. The rules consist of a word-list containing positive and negative words. Also, they contain rules for negation and other known semantic challenges for sentiment analysis. Although seemingly controversial, this approach is known for providing significant performances in previous online analysis tasks [83].

4.3.4 Lexicon comparison

The last approach used in literature is the labelling of data using a well-studied lexicon. Note, in this way, the baseline lexicon is assumed to give the true label. This approach is used in many adaptive lexicon-based approaches to validate the used model [5]. Known lexicon approaches are SentiWordNet [7], SenticNet [14] or other existing lexicons.

4.3.5 Used labeling strategy

The work in this thesis aims at achieving real-time insights with an increasing performance over time. The labeling strategy of the validation data will adhere to this need. The first consideration is the unavailability of labeled data. As the considered task looks at news articles in two different perspectives (i.e. sentiment analysis and entity linking), this data is not available. Furthermore, considering the importance of behavior over time, the presence of noisy labels and lexicon comparison are not acceptable as this does not yield stable and reliable results. Because of these considerations, the validation data will be labeled manually.
Chapter 5

Current Methodology

To adequately study the proposed research questions, suitable methodology needs to be defined. The first part of this methodology is the used data, which is already explained in Chapter 4. Besides collecting suitable data, the correct techniques and frameworks for the analysis task need to be identified. An overview of the proposed system is found in Figure 5.1.

![Figure 5.1: Overview of the system architecture.](image)

This section will list the available methodology in different programming languages for the system. The methodology for the grey blocks will be implemented. Additionally, a suitable stream execution environment will be identified. The overviews per building block given in this chapter will list pros and cons per method. A more elaborate list of used methodology with detailed descriptions and source information is given in Appendix A.

5.1 Natural language processing

5.1.1 Sentiment lexicons

The first NLP related methodology to be considered is the availability of lexicons for sentiment analysis. These lexicons are typically stored in a generic data structure which can be used in any preferred programming language. Although many different sentiment lexicons exist with numerous applications in different domains, this thesis will limit itself to the ones that achieved highest performance on news articles in previous literature [87]. The considered lexicons are found in Table 5.1.
5.1. NATURAL LANGUAGE PROCESSING

<table>
<thead>
<tr>
<th>Name</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet 3 [7]</td>
<td>- Makes distinction between verbs, adverbs, etc.</td>
<td>- General purpose (potential low performance)</td>
</tr>
<tr>
<td>SentiStrength [104]</td>
<td>- Lexicon is mostly created by manual annotation</td>
<td>- Annotation is done using highly subjective texts[87]</td>
</tr>
<tr>
<td></td>
<td>- Plug-and-play implementation</td>
<td></td>
</tr>
<tr>
<td>SenticNet [14]</td>
<td>- Reduces dimensionality (faster classification)</td>
<td>- Trained on small text (tweets).</td>
</tr>
<tr>
<td></td>
<td>- Trained using deep learning</td>
<td></td>
</tr>
<tr>
<td>Densify</td>
<td>- Trained on large corpora (long texts)</td>
<td>- Small training dataset</td>
</tr>
<tr>
<td></td>
<td>- Trained on news articles</td>
<td></td>
</tr>
<tr>
<td>Sentiment140 [38]</td>
<td>- Best performing method on news articles [87]</td>
<td>- Trained on small text (tweets)</td>
</tr>
<tr>
<td>NRC Hashtag [75]</td>
<td>- Well-performing on news articles [87]</td>
<td>- Trained on small text (tweets)</td>
</tr>
</tbody>
</table>

Table 5.1: Sentiment lexicon comparison.

5.1.2 Java frameworks

Oracle Java is one of the biggest programming languages widely adapted by industry. The typed language is used in a wide number of applications and industries [25]. In terms of NLP, implementations of functionality are found in open-source libraries across multiple sources. This section aims at identifying general pros and cons of the language, after which the specific NLP libraries will be reviewed.

- **Pros:** Java is known for the creation of durable and maintainable applications. This property makes it very suitable for the stream context of the thesis. Moreover, Java makes use of its own Java Virtual Machine (JVM) for data processing. Lastly, Java provides functionality for distributed processing out of the box.

- **Cons:** Writing code in typed languages is somewhat slower than in untyped languages. Also, the naming conventions used in Java give rise to overhead due to long class naming (especially when using third party libraries). Lastly, The JVM can be seen as a limitation as it does not provide full power over the machine a program is run on.

Note, the pros of the language outperform the cons and hence Java is a suitable language for the task. Next, it is important to identify the possibilities of Natural Language Processing the language has. For this, an overview of the methods is found in Table 5.2. As is clear from the overview, there is a large number of possibilities for NLP toolboxes. However, none of them provide the means for data stream mining. Therefore, some custom created method for the lexicon-based methods should be created if Java would be used for the thesis.
5.1. Natural Language Processing

<table>
<thead>
<tr>
<th>Name</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache OpenNLP [96]</td>
<td>- Customizable models</td>
<td>- Proved to be outperformed by Stanford CoreNLP [77]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Only batch training of sentiment models</td>
</tr>
<tr>
<td>Stanford CoreNLP [70]</td>
<td>- Outperforms Apache OpenNLP [77]</td>
<td>- Only batch training of sentiment models</td>
</tr>
<tr>
<td></td>
<td>- Built-in Company entity recognition</td>
<td></td>
</tr>
<tr>
<td>CogComp NLP [61]</td>
<td>- Provides an all-in-one tool</td>
<td>- Only batch training of sentiment models</td>
</tr>
<tr>
<td></td>
<td>- Outperforms the other methods [61]</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Java NLP methodology comparison.

5.1.3 Python frameworks

Python is another well-known programming language, especially in the machine learning community. The untyped language is designed to be readable and is mainly used for fast prototyping [26]. Again, the use of the language for the scope of the thesis will be identified by first giving the general pros and cons thereof. After that, the specific libraries for NLP are investigated.

- **Pros**: As explained, Python is known for fast prototyping. Its untyped structure gives rise to readable and interpretable code. Furthermore, it is the common standard for machine learning-based tooling.

- **Cons**: The language is less suitable for production (long term) deployment. Also, the language is more prone to errors due to its untyped syntax. Lastly, Python has the limitation that it does not provide functionality for distributed programming out of the box. This functionality can, however, be achieved by a third party library, such as Apache Spark [118].

As Python is the most common language for machine learning solutions, the number of available data processing libraries is vast. An overview of the ones providing NLP functionality is found in Table 5.3. From this table, one can see that many libraries can be used to solve a part of the (pre-) processing steps in the proposed architecture. Together, the libraries would pose a suitable solution for the whole system.
5.1. NATURAL LANGUAGE PROCESSING

<table>
<thead>
<tr>
<th>Name</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeautifulSoup [88]</td>
<td>- HTML parsing- and cleaning tools</td>
<td>- Does not provide ML tools</td>
</tr>
<tr>
<td>NLTK [64]</td>
<td>- Most commonly used library for NLP tasks</td>
<td>- Only batch learning methods</td>
</tr>
<tr>
<td>- Has both lexicon- and ML-based tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKLearn [84]</td>
<td>- Most common machine learning toolkit in Python</td>
<td>- Does not provide NLP tools</td>
</tr>
<tr>
<td>spaCy [1]</td>
<td>- Multi-lingual models</td>
<td>- Does not provide sentiment analysis</td>
</tr>
<tr>
<td>TextBlob [65]</td>
<td>- Pre-trained sentiment analysis</td>
<td>- Does not offer NER</td>
</tr>
</tbody>
</table>

Table 5.3: Python NLP methodology comparison.

5.1.4 Other languages

Besides the two well-studied language for doing natural language processing, alternatives exist in other popular programming languages. A few examples are given next.

- **C++**: For NLP in C++, two main commercial approaches are available: Intelleixer [94] and Apache Aurora [16]. The main benefit of C++ is the possibility to create well-maintainable production-ready solutions. Also, the language and run-time environment are known to provide much more power to programmer as more fundamental operations are possible. Nevertheless, the low availability of (open-source) NLP solutions is not beneficial to the work performed in this thesis.

- **Matlab**: Matlab is known in the academic world for having many functionality for modeling physical structures and other complex computational techniques. However, previous research also shows the power of the language in the analysis of financial news articles [105]. As Matlab is mostly known for simulation purposes and is not designed for long time deployment, it will not be considered in the study.

- **R**: Statistical software R is known for its statistical operability with numerical datasets. Additionally, some natural language processing libraries are provided, such as TidyText [34]. Nevertheless, R is not particularly known for it speed and efficiency in a non-numerical context as tackled in this thesis. Lastly, R does not provide extensive functionality to deploy applications on the long run. Therefore, it is not deemed useful for the sake of this thesis.

Although these frameworks perform tasks similar to the ones handled by Java and Python, the support for these languages in the field of data science is far from exhaustive or not existing at all. Therefore, for the selection of the used methodology, these will not be considered as feasible options.
5.2 Pre-processing block

The pre-processing block will focus on cleaning the text malformed and irrelevant information. The block will make use of the steps for data pre-processing described in Section 4.2. Four of the described pre-processing steps are relevant for analysis done in this thesis: noisy character removal, punctuation removal, stopword removal and number to textual number transformation. After these pre-processing steps, the cleaned text will contain only relevant words without noisy characters. An overview of the complete block is given in Figure 5.2.

![Figure 5.2: Overview of the pre-processing module using various techniques.](image)

The implementation of these different steps can make use of different methodologies. An overview of the possibilities per step is given in Table 5.4. For stopword removal and number transformations, multiple methods are needed because of the different languages considered in the thesis.

<table>
<thead>
<tr>
<th>Task</th>
<th>Method</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noisy character</td>
<td>Unicode encode</td>
<td>Java/Python</td>
<td>Native encoding libraries for unicode in both programming languages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Punctuation</td>
<td>Regular Expressions</td>
<td>Java/Python</td>
<td>Searches for punctuation and removes it.</td>
</tr>
<tr>
<td>Stopword removal</td>
<td>NLTK [64]</td>
<td>Python</td>
<td>Contains a preset list of stopwords. Customer list of stopwords.</td>
</tr>
<tr>
<td></td>
<td>Fixed input list</td>
<td>Java/Python</td>
<td></td>
</tr>
<tr>
<td>Number transformation</td>
<td>Inflect [32]</td>
<td>Python</td>
<td>Contains a preset list of number transformations. Customer number transformations.</td>
</tr>
<tr>
<td></td>
<td>Rule-based</td>
<td>Python/Java</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Different available methodologies for pre-processing.

Besides the listed four pre-processing steps, Section 4.2 list three other techniques: stemming, lemmatization and numerical representations. Stemming and lemmatization are not considered suitable techniques because, especially in lexicon-based sentiment analysis, different word representation tend to have different meaning and thus potentially different sentiment scores. Numerical representations are not considered in the work because both lexicons and named entity recognition take the words of a text as input rather than numerical representation thereof.
5.3 Sentiment model

As most of the methodology for natural language processing only support batch-based methods, there is a need for custom creation of adaptive lexicon methods. These methods are based on the literature described in Section 2.4.4. The difference between a traditional batch-based and online classification is visualized in Figure 5.3. For the batch-based baseline that will be investigated in this thesis, the methodology from Table 5.1 will be used. There is no readily available solution implementing the second adaptive unsupervised. The methodology for this structure is introduced in this thesis in Chapter 6.

![Figure 5.3: Overview of sentiment models in batch and online setting.](image)

5.4 Open linked data (SPARQL)

Methods to use SPARQL are given in Table 5.5. These different libraries have very similar functionality and all provide the functionality to link the open platform described in Section 4.1.3. In particular, Apache Jena is known for its efficiency and modularity in use [98].

<table>
<thead>
<tr>
<th>Name</th>
<th>Language</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Jena [98]</td>
<td>Java</td>
<td>- Fast query processing</td>
<td>- Own result structure</td>
</tr>
<tr>
<td>SPARQLWrapper [36]</td>
<td>Python</td>
<td>- Easily integrable</td>
<td>- Does only provide query processing</td>
</tr>
<tr>
<td>RDFLib [103]</td>
<td>Python</td>
<td>- Basic SPARQL functionality</td>
<td>- No further functionality</td>
</tr>
</tbody>
</table>

Table 5.5: Comparison of different SPARQL libraries for Python and Java.

5.5 Named entity linking

As an alternative to NER and SPARQL linking, the task of retrieving the company, industry and other meta-information can be done immediately using available Named Entity Linking methods. These methods identify objects in a piece of text and directly link them to a similar open entity found in the Semantic Web. The most suitable methods for this task are found in Table 5.6.

The difference between these methods in the system is found in Figure 5.4. Note, the output of both models is/are the identifier(s) of the linked company. From these identifiers, the additional information can be retrieved. For this, a final SPARQL query would be needed.
5.6 Stock compare block

In order to use the financial stock information described in Section 4.1.2, it needs to match the structure of the sentiment analysis block. For this, the module shown in Figure 5.5 will be used. As the sentiment analysis is done in a binary fashion (positive and negative), this behavior is requested for the stock module as well. The complete comparison is split into the transformation function and comparison function.

The transformation function will transform the opening and closing price of a stock on a particular day into either 1 for an increased value and -1 for a decreased value. Formally, let \( d \) be an arbitrary day on which the input article is published. Now, let \( o_d, c_d \in \mathbb{R} \) be the opening and closing price on day \( d \) for the requested stock. The transformation function \( f: \mathbb{R} \times \mathbb{R} \rightarrow \{-1, 1\} \) is now defined as follows:

\[
f(o_d, c_d) = \begin{cases} 
-1 & \text{if } o_d > c_d \\
1 & \text{if } o_d \leq c_d 
\end{cases}
\]
### Algorithm 1 Target selection in named entity recognition.

**Input:** String `article`, Procedure `Tokenizer`, Procedure `NER`

**Output:** String `targetCompany`

```python
1: procedure SELECT_TARGET
2:     companyCounts ← dict()
3:     articleWords ← Tokenize(article)
4:     articleTags ← NER(articleWords)
5:     for i in range(length(articleWords)) do
6:         if articleTags[i] == "ORG" then
7:             company ← articleWords[i]
8:             if company in companyCounts.keys() then
9:                 companyCounts[company] ← companyCounts[company] + 1
10:            else
11:                companyCounts[company] ← 1
12:     targetCompany ← argmax \( c \in \text{companyCounts.keys()} \) companyCounts[c]
13: return targetCompany
```

Figure 5.5: Overview of the stock module.
The comparison function takes the classified polarity from the sentiment module and the transformed stock information as input and outputs the correctness of the news article. Formally, let \( p, s \in \{-1, 1\} \) be the classified polarity and stock information respectively. Now, let \( g : \{-1, 1\}^2 \to \{0, 1\} \) be the comparison function, given by

\[
g(p, s) = \begin{cases} 
0 & \text{if } p \neq s \\
1 & \text{if } p = s 
\end{cases}
\]

For the stock compare block, no existing library will be used. The functionality will be implemented in the desired programming language of the chosen execution environment.

5.7 Execution environment

Lastly, a suitable streaming environment has to be selected to deploy the modules and run the system, as shown in Figure 5.1. For such an environment, the options from Table 5.7 are most suitable. Note, Apache Spark is originally designed for distributed processing rather than streaming. In practice, Spark is mainly used for fast batch processing rather than streaming. Additionally, Apache UIMA is known to have a very complex structure, making it challenging to use. Lastly, Apache Aurora is constructed to make real-time interaction between different systems possible. As the thesis essentially constructs a single architecture consisting of smaller blocks without complex interaction, this environment is considered too complex for the work and will not be used.

For the task handled in this thesis, Apache Storm and Flink are the most suitable frameworks as they provide a graph-structured flow of data which can be shaped to match the architecture presented in Section 5.1. More information on both frameworks can be found in Appendix A.

<table>
<thead>
<tr>
<th>Name</th>
<th>Language</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aurora</td>
<td>Java/Python</td>
<td>- Provides full power of an online algorithm</td>
<td>- Does not provide programming API’s</td>
</tr>
<tr>
<td>Flink</td>
<td>Java/Python</td>
<td>- Known for long-term deployment</td>
<td>- Only a fixed set of internal functions</td>
</tr>
<tr>
<td>Spark</td>
<td>Java/Python</td>
<td>- Provides fast distributed processing</td>
<td>- Not designed for streaming (streaming is added later)</td>
</tr>
<tr>
<td>Storm</td>
<td>Java/Python</td>
<td>- Designed for handling stream</td>
<td>- Complex structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Distributed processing possible</td>
<td></td>
</tr>
<tr>
<td>UIMA</td>
<td>Java</td>
<td>- Designed for unstructured data</td>
<td>- Complex structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Not designed for streaming</td>
</tr>
</tbody>
</table>

Table 5.7: Comparison of streaming environments for Python and Java.
Chapter 6

Adaptive Unsupervised Sentiment Analysis

This chapter will introduce a novel solution for lexicon adaptation in a streaming environment. The method is based on the lexicon expansion approach published by Saif et al [92]. The previously proposed method assumes the sentiment of a word to be correlated with that of the context it is used in. With this assumption, the method updates a lexicon by looking at prior and contextual sentiments of words in a document collection. This procedure will be described first, after which it will be adapted to a streaming environment.

6.1 Lexicon adaptation based on context

The previously proposed method of Saif et al in 2017 [92] describes a method to update any existing sentiment lexicon to a domain-specific one making use of prior and contextual sentiments of each distinct word in a collection of texts. The method is build on the hypothesis of Statistical Semantics [113]. More specifically, it assume:

\[ \text{Words that are used and occur in the same contexts tend to purport similar meanings.} \]

The procedure will adapt sentiment scores in some baseline lexicon by looking at the contexts of words. This contextual sentiment is compared to some adaptation thresholds and once one is met, the prior sentiment score of the word is adapted. The algorithm takes a pre-existing lexicon and a collection of textual documents as input and outputs a domain-specific adapted lexicon tailored to the input documents.

An overview of the updating process described by Saif et al is found in Figure 6.1. The contextual words co-occurring with "Great" in any of these documents are collected and stored in the context vector. Using the SentiCircle model (described next) for sentiment analysis, the contextual sentiment of "Great" is determined and the lexicon is adapted according to pre-specified adaptation rules.
6.1. LEXICON ADAPTATION BASED ON CONTEXT

6.1.1 SentiCircle

SentiCircle, a sentiment model introduced by Saif et al in 2014 [91], provides a method for extracting and assessing contextual sentiment of a word given a certain collection of documents. The goal of the method is to determine the semantics of a term $m$ using the relations of $m$ with all its context words (i.e. all words that occur with $m$ in the same context). The contextual terms are represented by points on a geometric circle. Finally, the baseline input lexicon is updated according the contextual sentiment scores. Intuitively, the procedure follows the following steps given by [91]:

1. Given are a collection of documents and some term $m$ of which one wants to compute the contextual sentiment.

2. Compute the individual relation between term $m$ and a contextual term $c$ using the $TDOC$ metric (defined later).

3. Compute the prior sentiment of a the contextual term $c$ using the baseline lexicon.

4. Use the $TDOC$ and prior sentiment of $c$ to place it on a SentiCircle.

5. Repeat steps 2-4 for all contextual terms of term $m$.

6. Calculate the geometric median of all points on the SentiCircle.

7. Determine the contextual sentiment of term $m$.

Formally, let $D$ be the collection of input textual corpora, $T$ be the set of all distinct words in $D$ and let $m \in T$ be the target word of which one wants to find the contextual sentiment. Furthermore, let $C \subseteq T$ be the collection of contextual words occurring at least once with $m$ in any corpus in $D$. For each $c \in C$, one is interested in its individual relation to $m$ and the prior sentiment in the baseline lexicon.

The individual relation is determined by the term degree of correlation ($TDOC$) of contextual term $c$ to $m$. This metric is inspired by the TF-IDF metric found in Section 2.3.1 and is defined by

$$TDOC(c, m) = f(c, m) \cdot \log \frac{|T|}{|T_c|}$$
where \( f(c, m) \) is the number of corpora in \( D \) where \( c \) and \( m \) co-occur and \( T_c \subseteq T \) the total number of distinct words that co-occur with \( c \) in \( D \). The prior sentiment of \( c \) is determined by its value in the baseline lexicon. If a term is not present in the lexicon, its prior sentiment is equal to 0.

Using these, a SentiCircle can be constructed. A SentiCircle is a circle centered around the origin on which the contextual sentiment scores can be projected. The coordinates of each contextual term \( c \) and target term \( m \) are determined using the polar coordinates

\[

t = TDOC(c, m) \quad \theta = Prior\_Sentiment(c) \cdot \pi
\]

Transforming these to geometric coordinates, one will find the points on the SentiCircle using the following definition

\[
\begin{align*}
  x &= r \cdot \cos(\theta) \\
  y &= r \cdot \sin(\theta)
\end{align*}
\]

Once the coordinates of all contextual points in \( C \) are determined, the geometric median will give the coordinates of the contextual sentiment vector of \( w \). The final contextual sentiment score of the target word is given by the \( y \)-coordinate of the sentiment vector.

The meaning of representing words on a circle is to make use of trigonometric properties of the polar coordinates. The way the coordinates are structured, four quadrants on the circle can be distinguishes. These quadrants are visualized in Figure 6.2. Depending on the quadrant a word is placed in, one can assess how strong the contextual sentiment of that word is.

![Figure 6.2: Visualization of the different quadrants on the SentiCircle.](image-url)
6.1. LEXICON ADAPTATION BASED ON CONTEXT

### Update Rule (Same Orientation)

**Antecedents**

\((|\text{contextual}_i| > |\text{prior}_i|) \land (|\text{contextual}_i| > \rho)\)

**Consequent**

\[ \text{prior}_i := \begin{cases} 
\text{prior}_i + \alpha & \text{if } \text{prior}_i > 0 \\
\text{prior}_i - \alpha & \text{if } \text{prior}_i < 0 
\end{cases} \]

### Update Rule (Different Orientation)

**Antecedents**

\((|\text{contextual}_i| > \rho) \land (|\text{prior}_i| \leq \rho)\)

**Consequent**

\[ \text{prior}_i := \begin{cases} 
\alpha & \text{if } \text{prior}_i < 0 \\
-\alpha & \text{if } \text{prior}_i > 0 
\end{cases} \]

**Antecedents**

\((|\text{contextual}_i| > \rho) \land (|\text{prior}_i| > \rho)\)

**Consequent**

\[ \text{prior}_i := \begin{cases} 
\text{prior}_i - \alpha & \text{if } \text{prior}_i > 0 \\
\text{prior}_i + \alpha & \text{if } \text{prior}_i < 0 
\end{cases} \]

### Expanding Rule

**Antecedents**

\(\text{term}(i) \notin \mathcal{L}\)

**Consequent**

\(\text{AddTerm}(i, \mathcal{L})\) and \((\text{prior}_i := \text{contextual}_i)\)

Table 6.1: Adaptation rules as proposed by Saif et al [92].

6.1.2 Adaptation rules

Once the contextual and prior sentiment are known, the lexicon can be adapted. For this, a suitable pre-defined ruleset is used to check whether the prior and contextual sentiment scores deviate significantly, and whether an adaptation is needed. There are two types of adaptation rules used:

- **Updating Rules**: These rules take an existing term from the lexicon and update the prior sentiment score.

- **Expanding rules**: These rules concern a term that do not yet exist in the lexicon and will be added according to their contextual score.

The rules make use of hyper-parameters \(\rho\) and \(\alpha\). \(\rho\) is used to assess whether an update to a rule is needed. \(\alpha\) provides the magnitude of the update. The different rules used by Saif et al. are found in Table 6.1. The terms \(\rho\) and \(\alpha\) in the table are defined as above. \(\text{prior}_i\) and \(\text{contextual}_i\) denote the current sentiment scores of the word itself and its context respectively. The lexicon \(\mathcal{L}\) is given by the current (not yet updated) lexicon and the function \(\text{AddTerm}\) will add a new term \((i)\) with its contextual sentiment to the lexicon.

Using the updating rules in the table, the previous paper showed an increase of 2% to 3% on domain specific corpora with respect to previous general purpose lexicons [92]. This result was achieved by deploying only the updating rules. From the experiments in their work, the expanding rules did not show improved results over the used baseline.

6.1.3 Batch-based adaptation algorithm

The full algorithm for batch-based lexicon adaptation is found in Algorithm 2. Note, this algorithm makes use of sub-procedures \(\text{CalculateTDOC}\), \(\text{GetGeometricMedian}\) and \(\text{UpdateRules}\). These procedures are adhering the explanations given above and are left abstract as implementation is trivial. Additionally, the algorithm will maintain a change memory to monitor which sentiment scores are changed by procedure.
6.2 DATA STREAM ENVIRONMENT

The algorithm takes a collection of documents, baseline lexicon, hyper-parameters $\rho$ and $\alpha$ and the above described procedures as input. First, the algorithm will create a lexicon object to be updated in line 2. Line 3 will initialize a change memory to keep track of what changes during adaptation. In line 4-8, all distinct words in the collection are collected in the list $W$. Lines 9-18 go through all words in $W$ and adapt the lexicon accordingly.

First, the $TDOC$ scores for a word given the collection is calculated. Then, per contextual word, the location on the SentiCircle is calculated. Next, the geometric median of these words is determined and the updated sentiment is determined according to the adaptation rules. Finally, lines 19-20 record the changes in sentiment scores in the change memory and adapted lexicon. After processing all the distinct words in the collection, the resulting adapted lexicon and change memory are given as output.

Algorithm 2 Lexicon adaptation for batched collection of documents.

**Input:** List<String> collection, Map<String, Double> lexicon, Double $\rho$, Double $\alpha$

**Output:** Map<String, Double> newLexicon, Map<String, Double> changeMemory

1: procedure LEXICON ADAPTATION
2: newLexicon ← lexicon
3: changeMemory ← Map<String, Double>
4: $W$ ← array()
5: for text in collection do
6:  for word in text do
7:   if word not in $W$ then
8:     $W$.append(word)
9:   contextualSentiments ← List<Double, Double>
10:  tdoc ← CalculateTDOC(word, collection)
11:  for (word, r) in tdoc do
12:    if lexicon.keyExists(word) then
13:      $\theta$ ← lexicon[word] · $\pi$
14:      $x$ ← $r \cdot \cos(\theta)$
15:      $y$ ← $r \cdot \sin(\theta)$
16:      contextualSentiments.append((x, y))
17:    contextualSentiment ← GetGeometricMedian(contextualSentiments)
18:    newSentiment ← AdaptationRules(word, lexicon, contextualSentiment, $\rho$, $\alpha$)
19:    changeMemory[word] ← newSentiment − lexicon[word]
20:   newLexicon[word] ← newSentiment
21: return newLexicon, changeMemory

6.2 Data stream environment

The adaptation method described in the previous sections aims at lexicon adaptation in a batch manner. That is, the algorithm is designed to update a lexicon once using a fixed collection of corpora after which it can be used in a domain specific area. To adapt the method to a streaming context, several adjustments need to be made. The first is a slight adjustment to the $TDOC$ metric defined before. This metric will be adjusted to fit the streaming context.
6.2. DATA STREAM ENVIRONMENT

Secondly, especially looking at longer periods of algorithm deployment, one needs to consider the fact that changes over time need to be identified and accounted for by the method. Additionally, once changes from the past are not relevant anymore, one needs to forget them. Both of these steps are to be taken into account when adjusting the current adaptation technique to a streaming context. The windowing techniques described in Section 2.7.3 are adopted to fit the need of the final algorithm.

A major choice in the windowing behavior of the algorithm is a pick in either time-based or count-based windows. Although both would yield feasible configurations for the final algorithm, the choice is made to only construct the algorithms for count-based windows. The different adjustments will be described in detail next, after which the final algorithm is presented.

6.2.1 Windowed TDOC

The definition of the current TDOC metric assumes the presence of a fixed collection of documents to calculate the scores on. In stream data mining, this fixed collection is not readily available and hence the method cannot be used as is. Streaming adaptation is achieved by modifying the function to a windowed variant, which considers chunks (a window) of the stream seen so far to calculate contextual sentiments.

Formally, let \( W \) be a windowed collection of documents. Now, let the windowed term degree of correlation over window \( W \) be defined by

\[
r_W = TDOC_W(c, m) = f_W(c, m) \cdot \log \frac{|T_W|}{|T_{W,c}|}
\]

where \( f_W(c, m) \) is the number of corpora in \( W \) where \( c \) and \( m \) co-occur, \( T_W \) is the collection of all distinct words in \( W \) and \( T_{W,c} \subseteq T_W \) the total number of distinct words that co-occur with \( c \) in \( W \).

Using this adapted \( TDOC_W \) metric, the contextual sentiments given a window \( W \) and target word \( m \) can be determined as explained in Section 6.1.1.

6.2.2 Learning behavior

A vital part of having a successful stream mining algorithm is adaptation of the model over time. For this, the model needs to learn from newly observed articles. Learning behavior in the proposed method is achieved using the windowing techniques described in Section 2.7.3. Both a tumbled learning technique, as well as a slid learning technique will be introduced.

**Tumbled learning**

Tumbled learning means that one will learn from each partition of articles that is observed. The behavior is analogous to processing a tumbling window in a stream mining context. In a tumbling window paradigm, every document gets considered in the learning process once, after which it is discarded.
The proposed stream-based algorithm for tumbled learning is found in Algorithm 3. This algorithm takes an article, a previously build memory, learning window size, lexicon and updating procedure as input. The article will be added to the memory (window). Next, the size of the memory is checked. If it is equal to the window size, the updating procedure is executed (i.e. the window is closed). After the lexicon is updated, the memory is deleted (a new window is opened). This updating procedure has the adapted lexicon and new memory as output.

Algorithm 3  
*Tumbled Learning* for lexicon adaptation.

| Input: String *article*, List<String> *memory*, Integer *learningWindow*, Map<String, Double> *lexicon*, Double *ρ*, Double *α*  
| Output: List<String> *newMemory*, Map<String, Double> *newLexicon*, Map<String, Double> *changeMemory* |

1: **procedure** TUMBLED LEARNING  
2:  
3:  
4:  
5:  
6:  

Note, the algorithm is built to process a single article per procedure call. This design choice is made because of the streaming context. Also note, the output of the procedure, i.e. the memory and lexicon, for a certain article will serve as the input for the next observed article.

**Slid learning**

Alternative to tumbled learning, one can consider a sliding window paradigm for the learning behavior. Slid learning means that for every observed article, the lexicon will be updated and the least recent (oldest) article in the window is removed. Note, every article is processed in multiple windows and hence is taken into account during multiple adaptations.

The proposed procedure for slid learning is found in Algorithm 4. This algorithm is similar to Algorithm 3 for tumbled learning. The difference between the two procedures is found in line 5. Whereas tumbled learning empties the whole memory, slid learning only removes the first (oldest) article from memory.

Algorithm 4  
*Slid Learning* for lexicon adaptation.

| Input: String *article*, List<String> *memory*, Integer *learningWindow*, Map<String, Double> *lexicon*, Double *ρ*, Double *α*  
| Output: List<String> *newMemory*, Map<String, Double> *newLexicon*, Map<String, Double> *changeMemory* |

1: **procedure** SLID LEARNING  
2:  
3:  
4:  
5:  
6:  

65
6.2. DATA STREAM ENVIRONMENT

6.2.3 Forgetting behavior

Besides learning from newly observed articles, it might also be relevant to forget information which was learned previously in the stream. The proposed forgetting behavior is again denoted by the tumbling and sliding window paradigms explained in Section 2.7.3. Both procedures will be introduced next.

Tumbled forgetting

Whereas tumbled learning means that one learns from a partition of articles, tumbled forgetting means that one forgets all information learned by the previous partition. This procedure will maintain a baseline lexicon and resets the adapted one once a pre-specified number of articles is observed. After this reset, the lexicon will again be updated using the learning procedures.

The procedure for tumbled forgetting is found in Algorithm 5. The algorithm takes as input the number of articles observed after the latest reset, the forgetting window size, the current lexicon and the baseline lexicon. The procedure will check whether enough articles are observed to close the forgetting window. If the forgetting window is closed, the lexicon is reset to its baseline and a new window is opened (i.e. the number of observed article is set to zero).

Algorithm 5 Tumbled Forgetting for lexicon adaptation.

Input: Integer articleCount, Integer forgettingWindow, Map<String, Double> lexicon, Map<String, Double> originalLexicon

Output: Integer newArticleCount, Map<String, Double> newLexicon

1: procedure TUMBLEDFORGETTING
2:   articleCount ← articleCount + 1
3:   if articleCount == forgettingWindow then
4:     lexicon ← originalLexicon
5:     articleCount ← 0
6:   return (articleCount, lexicon)

Again note, this procedure is executed per observed article, as was the case in the learning procedures. Also, the output of a call to this procedure will again be used as input when for the next call once a new article is observed.

Slid forgetting

Similar to slid learning, the sliding window paradigm can be used in the forgetting behavior of the algorithm. For this, a history of lexicon updates is maintained. Once an entry of this history is past a certain forgetting threshold, it is reverted (i.e. forgotten).

The procedure describing slid forgetting more formally is shown in Algorithm 6. Per observed article, the algorithm will check whether there was a change which is not within the window reach anymore. If this is the case, the change per word in that iteration is reverted. Note, for this procedure to work, an alteration to the adaptation procedure is needed which records changes of the lexicon in the memory.
6.2. DATA STREAM ENVIRONMENT

Algorithm 6 Slid Forgetting for lexicon adaptation.

| Input: Integer forgettingWindow, Map<String, Double> lexicon, |
| List<Map<String, Double>> changeMemory |
| Output: List<Map<String, Double>> newChangeMemory, |
| Map<String, Double> newLexicon |

1: procedure SLIDFORGETTING
2: if |changeMemory| == forgettingWindow then
3: toBeReverted ← changeMemory[0]
4: for (word, change) in toBeReverted do
5: lexicon[word] ← lexicon[word] − change
6: changeMemory.delete(0)
7: return (changeMemory, lexicon)

6.2.4 Final algorithm

Knowing all the components of the stream algorithm, it is now possible to formulate the final algorithm for adaptive sentiment analysis. This algorithm will incorporate the different learning and forgetting features to classify the sentiment per article found in a stream. The algorithm will be designed to run indefinitely, consuming articles from an input stream and pushing sentiment scores to an output stream. Internally, it will maintain all memories defined for the different sub-procedures. Before running the algorithm, the learning and forgetting behavior are chosen, together with the window sizes for both procedures.

The full algorithm is found in Algorithm 7. The algorithm takes as input a stream of news articles, the window sizes and behaviors for both learning and forgetting tasks, an initial lexicon and procedures for tokenizing an article, calculating its sentiment and updating the lexicon. In the indefinite loop, articles are consumed from the stream. Then, the sentiment is calculated and pushed into the output stream. After this, the lexicon is updated using the designated sub-procedure.
Algorithm 7 AdaUSA algorithm for adaptive sentiment analysis.

Input: Stream<String> articleStream, Stream<{-1, 1}> articleSentiments
String learningBehavior, String forgettingBehavior,
Integer learningWindow, Integer forgettingWindow,
Map<String, Double> lexicon,
Procedure Tokenize, Procedure CalculateSentiment

1: procedure ADAUSA
2:  originalLexicon ← lexicon
3:  memory ← List()
4:  articleCount ← 0
5:  changeMemory ← List<Map<String, Double>>
6: while true do
7:    article ← articleStream.consume()
8:    articleWords ← Tokenize(article)
9:    sentiment ← CalculateSentiment(articleWords)
10:   articleSentiments.push(sentiment)
11:
12:  if forgettingBehavior == “T” then
13:    (articleCount, lexicon) ← TumbledForgetting(articleCount, forgettingWindow,
14:                      lexicon, originalLexicon)
15:  if forgettingBehavior == “S” then
16:    (lexicon, changeMemory) ← SlidForgetting(forgettingWindow, lexicon,
17:                      changeMemory)
18:  if learningBehavior == “T” then
19:    (memory, lexicon, changes) ← TumbledLearning(article, memory,
20:                      learningWindow, lexicon, ρ, α)
21:    changeMemory.append(changes)
22:  if learningBehavior == “S” then
23:    (memory, lexicon, changes) ← SlidLearning(article, memory,
24:                      learningWindow, lexicon, ρ, α)
25:    changeMemory.append(changes)
Chapter 7

Named Entity Post-Processing for Financial Information

Financial news articles might contain a lot of information overviewing the current state of the financial market. Such articles provide information on many companies and do not necessarily target a single entity. This chapter will introduce a post-processing method to reduce the number of such articles in the considered news stream and only consider the ones in which a definite target company can be identified. It will define the concept of absolute and relative dominance of mentioned companies which will quantify how well a company is distinguished as main target. The novel $ADom$-selection and $RDom$-selection algorithms will be introduced to deploy company dominance and improve the quality of existing methodology for entity recognition.

7.1 Company tagging

Identifying a target company using the entity recognition methodology described in Algorithm 1 is done in two steps. The first step (line 3 up and until 11) tags the entities from a sentence and filters the ones tagged as organization. The second step (line 12 and 13) selects the most occurring tagged company and returns it as the output. This method is based on the assumption that a company mentioned more often than others is more relevant to the news article.

A reoccurring problem in this method is the pattern in which companies are listed equally often. Peaking ahead to the results of current methodology in Chapter 9.1.3, a problem is found in news articles which provide an overview of the current state of the market, thus targeting many different companies. Below, two example articles are given which respectively show overviewing- and specific target behavior. In the first article, the true target company is not clear and might even be any of the companies mentioned. The second article shows Apple as a clear target.

No one in Silicon Valley wants to be called a “chip company” these days. Qualcomm, which makes the processors found in many Android phones and iPhone modems, describes itself as a “platform company”. Intel, once proudly “inside” most of the world’s PCs, now positions itself as a “data company”. Nvidia, whose soaring share price has made it one of the best-performing stocks of any sector in the past two years, describes its graphical processors as “amplifying human intelligence”, thanks to their growing role in deep learning research. Yet just as quickly as semiconductor companies are trying to rebrand themselves as something else, dealmakers are rushing into the sector. - Tim Bradshaw [39]
Some owners of Apple’s new iPhone X are finding an unusual bug in their $1,000 device: answering calls. The three-month-old smartphone boasts a new design, can be unlocked at a glance with facial recognition software and has been hailed by Tim Cook, Apple’s chief executive, as preparing the iPhone for its second decade. But hundreds of owners have complained on Apple forums that their pocket supercomputer cannot accomplish the most basic task that a $10 phone can: take an incoming call. When the phone rings, the iPhone’s touchscreen appears to them to be delayed from turning on for up to 10 seconds, preventing the user from tapping the virtual button required to answer it. - Tim Bradshaw [39]

Note, these articles are written by the same author. Similar behavior (overviews vs. specific targets) is found among many different authors and hence should be taken into account on article level rather than per author. To do this, a novel data post-processing method will be introduced next.

7.2 Dominant company selection

The proposed post-processing solution to the mentioned problem is that of collecting only news articles which will most likely follow the assumption made to identify target companies made in Section 7.1. In order to do this, dominance of companies will be defined to provide a metric describing the likelihood of adherence to the assumption.

Dominance of a company can be measured by the overall appearance of a company in an article (which will be called absolute dominance) or by how more often a company is mentioned than others (relative dominance). These post-processing steps will be defined and described in detail.

7.2.1 Absolute dominance (ADom)

Absolute dominance is defined by how often the most occurring company is mentioned in a news article. This dominance is measured by a positive integer $i$. If the most occurring company in an article occurs at least $i$ times, the article is member of a dominance set $Dom_i$.

Formally, let $f : C \times N \rightarrow \mathbb{N}$, with $C$ the set of all companies and $N$ the collection of news articles in the data stream, the frequency function defined by:

$$f(c, n) = m \quad \text{if } c \text{ occurs } m \text{ times in } n$$

with $c \in C$, $n \in N$ and $m \in \mathbb{N}$. Now, let $ADom_i \subseteq N$, with $i \in \mathbb{N}$, denote the absolute dominance set defined the set of articles where:

$$n \in ADom_i \iff \exists c \in C \quad f(c, n) \geq i.$$ 

Note, $ADom_0 = N$ and, for $j \geq i$, $ADom_j \subseteq ADom_i$.

Using this definition, one can alter the algorithm given in Chapter 5 to only use articles containing dominant companies. This algorithm is given in the next subsection.
7.3. NER POST-PROCESSING ALGORITHM

7.2.2 Relative dominance (RDom)

Similar to absolute dominance, one can define the notion of relative dominance. A company is relatively dominant if its frequency in an article is at least $i$ more than all other companies. This means that the most frequent company is listed significantly more often than others, presumably because it is more important.

Formally, again, let $f : C \times N \to \mathbb{N}$ be the frequency function as defined in the previous subsection. Additionally, let $c_n^*$ denote the most occurring company in article $n \in N$, defined by:

$$c_n^* = \arg\max_{c \in C} f(c, n)$$

Now, let the set $RDom_i \subseteq N$, with $i \in \mathbb{N}$, denote the relative dominance set defined the set of article where:

$$n \in RDom_i \iff \forall c \in C \setminus \{c_n^*\} f(c_n^*, n) - f(c, n) \geq i$$

Again, note that $RDom_0 = N$ and, for $j \geq i$, $RDom_j \subseteq RDom_i$. Furthermore, notice that, for all $i \in \mathbb{N}$, $RDom_i \subseteq ADom_i$. This last property indicates that $RDom$ holds a stronger condition on its members than $ADom$.

7.3 NER post-processing algorithm

To use both article set properties defined in the previous subsection, a modification to Algorithm 1 is necessary to incorporate the two post-processing steps. For $ADom$-selection, the modification is found in Algorithm 8 on lines 12 up and until 17. These lines select the most occurring company and check whether the news article belongs to $ADom_i$. If this is the case, the target company is returned. Otherwise, a null value is returned.

**Algorithm 8** $ADom$-selection in named entity recognition.

**Input:** String article, Integer $i$, Procedure Tokenize, Procedure NER

**Output:** String targetCompany

1: procedure SECTTARGET
2: companyCounts $\leftarrow$ dict()
3: articleWords $\leftarrow$ Tokenize(article)
4: articleTags $\leftarrow$ NER(articleWords)
5: for $i$ in range(length(articleWords)) do
6:   if articleTags[i] == "ORG" then
7:     company $\leftarrow$ articleWords[i]
8:     if company $\in$ companyCounts.keys() then
9:       companyCounts[company] $\leftarrow$ companyCounts[company] + 1
10:   else
11:     companyCounts[company] $\leftarrow$ 1
12: targetCompany $\leftarrow$ $\arg\max_{c \in \text{companyCounts.keys()}}$ companyCounts[c]
13: targetCount $\leftarrow$ companyCounts[targetCompany]
14: if targetCount $\geq$ $i$ then
15:   return targetCompany
16: else
17:   return null

73
As for RDom-selection, the modifications are found in lines 12 up and until 19 of Algorithm 9. These lines selects the most occurring company and its count. After this, the company is removed from the count dictionary and the highest count (of the second best company) is selected. If this second best count deviates enough from the target counts, the article belongs to RDom; and hence the target company is returned. If this is not the case, a null value is returned.

**Algorithm 9** RDom-selection in named entity recognition.

**Input:** String article, Integer i, Procedure Tokenize, Procedure NER

**Output:** String targetCompany

1: procedure SELECTTARGET
2:   companyCounts ← dict()
3:   articleWords ← Tokenize(article)
4:   articleTags ← NER(articleWords)
5:   for i in range(length(articleWords)) do
6:     if articleTags[i] == "ORG" then
7:       company ← articleWords[i]
8:       if company ← companyCounts.keys() then
9:         companyCounts[company] ← 1
10:     else
11:       companyCounts[company] ← companyCounts[company] + 1
12:   targetCompany ← argmax companyCounts[c] c∈companyCounts.keys()
13:   targetCount ← companyCounts[targetCompany]
14:   secondBestCount ← max companyCounts[c] c∈companyCounts.keys()
15:   if targetCount − secondBestCount ≥ i then
16:     return targetCompany
17:   else
18:     return null
Chapter 8

Experimental Setup

This section will be devoted to elaborating the experimental setup for the analysis of the research questions. The setup will be provided in a modular fashion, hence being one of the main contributions this thesis brings. The test cases described in this section will be used in later sections to form answers and conclusions to the thesis statement given in Chapter 3. Detailed descriptions of the test cases will be given respective to their corresponding analysis task (i.e. pre-processing, sentiment analysis, named entity recognition and named entity linking).

First, the overall experimental setup will be given, already making decisions in the used stream environment and different building blocks involved. Using this setup, the test system will be specified. After this, the building blocks in data pre-processing, sentiment analysis, named entity recognition and named entity linking are defined to execute the prediction performance test cases. The full setup will also be defined in a different execution environment to execute the computational test cases. Next, the validation data (‘golden standard’) is defined and described. Finally, the test cases for different elements of the system are given.

8.1 Experimental setup

The experimental setup will describe the different environments, resources and models used during the experimental phase of the thesis. Testing will be done by assessing both predictive and computational aspects of the considered frameworks and models. First, the predictive performance of different models for SA, NER and NEL will be assessed. A general overview of architecture used during these tests is found in Figure 8.1. Note, this system only makes use of the Apache Storm environment for streaming. This decision is made primarily because it provides multi-language functionality, hence giving the possibility to assess performance of both Python and Java-based methodology described in Chapter 5.

8.1.1 Test system description

The used resources for the execution of test cases will be fixed and kept as stable as possible on the machine the tests are executed on. Note, the stability of resources will only be of relevance during the computational test cases concerning the execution environments. The used operating system is a Ubuntu Linux environment natively run on a laptop. The used versions and available resources are shown in Figure 8.2.

Note, both the SA and NER blocks are run in an unsupervised manner. This means that they are not allowed to query the true label of a corpus. The true label for both tasks is added in the data stream for validation purposes, not for model training.
8.1. EXPERIMENTAL SETUP

Figure 8.1: Overview of the overall test architecture.

Figure 8.2: Overview of the test system requirements.
8.1. EXPERIMENTAL SETUP

8.1.2 Pre-processing design
The pre-processing block (or bolt as used in the terminology of Apache Storm) will follow the structure of the proposed block found in Section 5.2. The block consists of four different pre-processing steps, each making use of their own methodology. All of these methods are rule-based and deterministic. The proposed block in the methodology section already described the different considerations for the used pre-processing techniques and why they are relevant to the tasks at hand. The block will be constructed using unicode encoding for noisy character removal, regular expression matching for punctuation removal, the NLTK library for stopword removal and the inflect library for number transformations.

8.1.3 Sentiment analysis design
The sentiment analysis block consists of the methods needed to classify the sentiment of a corpus. The block will receive the textual bodies of the news article one by one along with their true sentiment. The textual body is used by different models to classify the sentiment. The classified and true sentiment are forwarded to the sentiment report bolt which will describe the results of the analysis of the considered method. The considered methodology for the test cases in this block are found below.

- **SentiWordNet 3** (Java): General purpose lexicon differentiating between different POS word types [7].
- **SentiStrength** (Java): General purpose lexicon-based on highly subjective short texts [104].
- **SenticNet** (Java): General purpose lexicon built using neural networks and many short input texts.
- **Difussion** (Java): General purpose lexicon.
- **Sentiment140** (Python): Lexicon build using an ensemble of naive bayes, maximum entropy and support vector machines [38].
- **NRC Hashtag** (Python): Lexicon proven to have high accuracy on news articles. The method is used in combination with Vader.
- **Textblob** (Python): Pre-trained sentiment model consisting of an ensemble of Naive Bayes, Random Forest and a general purpose lexicon lexicon.
- **Naïve Bayes in NLTK** (Python): Heuristic supervised classifier with high performance on sentiment analysis. Note, this model is used as a baseline. As the task handled by the thesis considers a stream of unlabeled data, this method will not be suitable for the final system.
- **AdaUSA**: The adaptive unsupervised sentiment analysis algorithm introduced in this work. More information is found in Chapter 6.

8.1.4 Named entity linking design
Having a similar functionality to that of the sentiment block, the NER block will test the different considered named entity recognition methodology. All methods use the input textual corpus to classify the company targeted in the article. The considered methods are listed below.
8.1. EXPERIMENTAL SETUP

- **Apache OpenNLP** (Java): Apache OpenNLP uses a pre-trained model to detect organizations in a piece of text. The most frequently extracted organization is deemed the target one.

- **Stanford CoreNLP** (Java): Stanford CoreNLP provides a pre-trained NER model to tag all relevant entities in a corpus. Filtering this tagging on organization and selecting the most frequent one will yield the target company.

- **CogComp NLP** (Java): Recently (2018) introduced NER model which utilizes an ensemble of OpenNLP and CoreNLP with custom novel improvements [61].

- **Spacy NER** (Python): Spacy NER uses a pre-trained NER model to tag a set of entities from sentences. Again, selecting the most frequently mentioned organization will yield the target company.

- **ADom-selection and RDom-selection**: NER post-processing algorithms introduced in this thesis. More information is found in Chapter 7.

8.1.5 Full setup architecture

Once the models for the sentiment analysis and NER blocks are in place, the results will be reported by reporting blocks. These blocks will report different performance metrics over time. The following performance metrics will be reported by the reporting blocks:

- **Accuracy**: Describing which proportion of the articles are classified correctly.

- **Precision**: Describing the proportion of positive articles are classified as positive.

- **Recall**: Describing the proportion of documents classified as positive which are actually positive documents

- **F1-score**: A metric to show the relation between precision and recall.

The metrics reported for sentiment analysis will use the performance metrics for binary classification and NER will use the metrics from multi-class classification. More information on these metrics is found in Section 2.1.2 and Section 2.2 for the sentiment analysis and NER block respectively.

In addition to the traditional metrics, windowed metrics will be used to monitor the performance of the models over time. These windowed version calculate the metrics by only looking at a history of the latest 100 reported results.

8.1.6 Execution environment

An important decision to make in the final system architecture is that of the execution environment. For this, two alternatives will be considered, Apache Storm and Apache Flink. As the environments do not contribute to the predictive performance of the system, they are used to measure the computational behavior (i.e. running time).

Note, the environments of Storm and Flink are considerably different in the sense that Flink, in essence, only supports Java as programming language. Therefore, only the constructed Java models from the Sentiment and NER blocks can be considered in the Flink environment. As for the pre-processing block, this means that the methodology for stopword removal and number transformation is changed to manually implemented rule-based methods found in Section 5.2.
8.1. EXPERIMENTAL SETUP

Figure 8.3: Overview of test architecture in Apache Storm and Apache Flink.
8.2 Validation data

To retrieve reliable performance metrics from the considered methodology, a suitable validation dataset is needed. This ‘golden standard’ will serve as the base input of the different test cases. Figure 8.4 shows the included information in this dataset. Note, the article input information will be available during production deployment of the system as well. The latter pieces of information will serve as true labels for the different classification and retrieval tasks in the analysis.

The validation dataset used in the test cases consists of 4494 financial articles written by a total of ten authors, publishing for the Financial Times ranging in publishing date between September 2nd 1994 and February 15th 2019. More detailed information on the used authors, the number of articles they published and their respective industry they operate in is found in Appendix B.

The windowed proportion of positive articles over time is shown in Figure 8.5. The figure shows a clear difference in positive and negative articles over time. This change over time is the result of a changing financial environment. In the analysis, it is interesting to see to what extent this difference in proportion over time correlates with the results of the different models. The overall ratios of positive and negative articles over the whole dataset are 0.542 and 0.458 respectively.

8.3 Test cases

Knowing the specifications of the experimental setup and validation data, suitable test cases can be formulated. These test cases will be divided in the two tested system aspects, predictive and computational performance tests. Each will define cases for different types of input models. The replication of test cases will depend on whether a model is deterministic (i.e. a pre-trained model is used) or probabilistic (i.e. training is needed). If a test case is replicated multiple times, the average performance metrics over all cases are used.
8.3. TEST CASES

8.3.1 Prediction performance measurement

Predictive performance test cases aim at testing the correctness of the behavior of a model. Shortly said, these cases look at the number of elements in the golden standard which are correctly classified by a target model. The test metrics describing the performance of each model are described in Section 8.1.5. For both sentiment analysis and named entity recognition, the corresponding true labels from the Golden Standard dataset will be used to compute the metrics.

Note, a number of models are pre-trained and will therefore show deterministic behavior in the test case. Because of this, there is no need for replication, as more repetitions will yield the same results. On the contrary, the probabilistic (trained) models do need test replication as each test will result in different performance. This method (naive bayes) will be replicated ten times and the final performance will be computed as the unweighted average of all the results.

The introduced AdaUSA algorithm for sentiment analysis uses a baseline lexicon and two behavioral parameters for the learning and forgetting behavior respectively. The test cases to test the performance of this method will make use of the best performing existing lexicon as baseline. For the test, the procedure in Algorithm 7 will be used. The different behavioral parameters tested are "No", "Tumbling" and "Sliding" which denote no adaptation, tumbled adaptation and slid adaptation respectively for both learning and forgetting procedures. Note, once the learning procedure is set to "No", the baseline lexicon will remain unchanged over the whole experiment and hence the forgetting mechanism will behave the same (i.e. nothing will be learned nor forgotten). Therefore, there will be seven test cases for the AdaUSA method. These test cases are given in Table 8.2.

The test cases for named entity recognition will make use of the procedure described in Algorithm 1. The ADom-selection and RDom-selection methods will only be used on the best performing existing NER method and will make use of the procedures in Algorithm 8 and Algorithm 9 respectively. Both algorithms will be tested for multiple values of $i$, ranging...
8.3. TEST CASES

<table>
<thead>
<tr>
<th>Method</th>
<th>Task</th>
<th>Model type</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet 3</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>SentiStrength</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>SenticNet</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Diffusion</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Sentiment140</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>NRC Hashtag</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Textblob Sentiment</td>
<td>Sentiment analysis</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Sentiment analysis</td>
<td>Probabilistic</td>
<td>10</td>
</tr>
<tr>
<td>AdaUSA</td>
<td>Adaptive sentiment analysis</td>
<td>Deterministic</td>
<td>1</td>
</tr>
<tr>
<td>Apache OpenNLP</td>
<td>Named entity recognition</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Stanford CoreNLP</td>
<td>Named entity recognition</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Illinois CogComp NLP</td>
<td>Named entity recognition</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>Spacy NER</td>
<td>Named entity recognition</td>
<td>Pre-trained</td>
<td>1</td>
</tr>
<tr>
<td>ADom-selection</td>
<td>NER Post-processing</td>
<td>Deterministic</td>
<td>1</td>
</tr>
<tr>
<td>RDom-selection</td>
<td>NER Post-processing</td>
<td>Deterministic</td>
<td>1</td>
</tr>
<tr>
<td>DBPedia Spotlight</td>
<td>Named entity linking</td>
<td>Deterministic</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.1: Test case replication per considered method.

<table>
<thead>
<tr>
<th>Learning</th>
<th>Forgetting</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Any</td>
</tr>
<tr>
<td>Tumbling</td>
<td>No</td>
</tr>
<tr>
<td>Tumbling</td>
<td>Tumbling</td>
</tr>
<tr>
<td>Tumbling</td>
<td>Sliding</td>
</tr>
<tr>
<td>Sliding</td>
<td>No</td>
</tr>
<tr>
<td>Sliding</td>
<td>Tumbling</td>
</tr>
<tr>
<td>Sliding</td>
<td>Sliding</td>
</tr>
</tbody>
</table>

Table 8.2: All behavioral parameter settings for the AdaUSA test cases.

from 1 to 8. Lastly, the test cases for DBPedia Spotlight for named entity linking will consider the outputs of each of the named entity recognition models as input. For each named entity recognition model, the linking test case will report a linking accuracy. A summary of the test cases is found in Table 8.1.

8.3.2 Computational performance measurement

Apart from testing the predictive performance of the considered models, the choice of the execution environment significantly influences the system overall effectiveness. Especially when considering system scalability (i.e. considering more input sources and articles), the running time of different methods in a certain execution environment is a crucial factor.

The two environments used for this test are Apache Storm and Apache Flink. The experimental setups for both environments are found in Figure 8.3. The running times considered are the ones measured over processing a stream of 1000 consecutive articles.

An important aspect to take into account during testing of running time is test system stability. The test system used should have a consistent allocation of resources to the architecture for fair comparison. In practice, this consistent allocation is not always achieved as the operating system might run background processes influencing the available resources and
thus the resulting computational performance. Stability is ensured by only making use of the vital resources on the test system running the tests. Nevertheless, it should be noted that there may be inconsistencies in resource allocation and hence test case replication is vital. Each of the tests will be run ten times and the performance metrics will be averaged.

8.4 Financial author assessment

After the test cases are executed, the task tackled in this thesis can be investigated. This task wants to automatically assess the quality of financial article authors. To do this, the full tested architecture will be used. To get understanding on how well the system performs per author, the following results will be filtered on articles written by the specific authors.

- **Sentiment analysis**: Which portion of polarities per author is classified correctly? Are there significant differences between different authors?

- **Named entity recognition**: Which portion of companies is correctly retrieved? Are there significant difference between different authors?

Using these insights, the linked information from the named entity linking procedure can be used to link the stock ticker and price to the sentiment of an article.

8.4.1 Stock ticker retrieval on DBPedia

Besides the performance of sentiment analysis and entity recognition in the validation dataset, it is interesting to see what insights the linked entities of the classified companies can give. For this, a suitable SPARQL query is needed to enrich the target company. As the linking engine DBPedia Spotlight links to entities on DBPedia, a logical first step would be to query that platform for company insights. To do this, the following SPARQL query will be used:

```sparql
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?stockSymbol where {
    OPTIONAL { <COMPANYURI> dbp:symbol ?stockSymbol. }
}
```

Listing 8.1: DBPedia ticker retrieval query.

In this query, the tag COMPANYURI will be replaced with the linked company. If the stock ticker symbol is given in DBPedia, this query will retrieve it. If it is not the case, the results will be empty. This query is executed on the DBPedia SPARQL API found on http://dbpedia.org/sparql. For example, when the linked entity of Apple would be used as input, the output is generated from this query is shown below. In this case, there is direct access to the company stock ticker.

```json
[{
    'stockSymbol': {
        'type': 'typed-literal',
        'datatype': 'http://www.w3.org/1999/02/22-rdf-syntax-ns#langString',
        'value': 'AAPL'
    }
}]
```
8.4.2 Stock ticker retrieval on Wikidata

Peaking ahead to the results from the above mentioned query, one will find that close to 10% of the target companies can be linked to their ticker using this approach. This shows that a lot of companies do not have their stock ticker available on DBPedia. To link more stock tickers, one can link additional information from other platforms to the entity. For this, the commonly used property \texttt{owl:sameAs} will be used. With the tag, similar entities form other platforms can be retrieved. For this, the query is changed to:

\begin{Verbatim}
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?stockSymbol ?similarEntities where {
  OPTIONAL { <COMPANYURI> dbp:symbol ?stockSymbol. }
  OPTIONAL { <COMPANYURI> owl:sameAs ?similarEntities. }
}
\end{Verbatim}

Listing 8.2: DBPedia ticker retrieval query with similar entities.

An example of a result for such query can be seen in car company BMW, for which the (truncated) JSON output given below is generated. Note the presence of Wikidata among the results. This Wikipedia backbone usually holds more information on entities than DBPedia and hence makes a useful resource for further information linking.

\begin{verbatim}
[{ 'similarEntities': { 'type': 'uri', 'value': 'http://cs.dbpedia.org/resource/BMW' }},
 { 'similarEntities': { 'type': 'uri', 'value': 'http://de.dbpedia.org/resource/BMW' }},
 { 'similarEntities': { 'type': 'uri', 'value': 'http://openei.org/resources/BMW' }},
 { 'similarEntities': { 'type': 'uri', 'value': 'http://www.wikidata.org/entity/Q26678' }},
 { 'similarEntities': { 'type': 'uri', 'value': 'http://data.nytimes.com/40106264331835457592' }}
]
\end{verbatim}

To work with Wikidata, a separate query is needed to access the linked information from the platform. This query found below. Again, it uses a COMPANYURI to access the linked entity. Note however, this is not the same URI as was used for DBPedia. In the example of BMW described above, the URI is given by \texttt{http://www.wikidata.org/entity/Q26678}. To retrieve the results from a SPARQL query on Wikidata, the query will be submitted to \texttt{http://query.wikidata.org/sparql}.

\begin{Verbatim}
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
SELECT DISTINCT ?ticker WHERE {
}
\end{Verbatim}

Listing 8.3: Wikidata ticker retrieval query.
8.4. FINANCIAL AUTHOR ASSESSMENT

For stock tickers, Wikidata holds information on both current and previous stock tickers of a company. For example, Apple used the stock ticker 6689 for the Tokyo Stock Exchange until 2004 (after which it became AAPL). Both of these tickers are retrieved using the above query and hence there is a need for a check whether the ticker is still active. For this, the following query is used.

```
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
SELECT DISTINCT ?ticker
WHERE {
  FILTER NOT EXISTS {
    ?exchange pq:P249 ?ticker ;
    pq:P582 ?enddate .
  }
}
```

Listing 8.4: Wikidata ticker retrieval query with filter for old tickers.

8.4.3 Stock comparison to sentiment

Now, using the queries in Listings 8.2 and 8.4, the stock ticker of linked companies given a specified publication date can be found. Using these stock tickers, the stock price of the company on the article date can be retrieved. To do this, the methodology for stock price retrieval given in Section 5.6 will be used. The stock compare block described in that section will finally assess the quality of a news article author.
Chapter 9

Results and Evaluation

This section will present the results of the experiments described in Section 8. These results will serve as the basis for answering the research questions around which this thesis is constructed. Later on, the results will also be used to formulate the conclusions, closing off the thesis in Section 10.

The remainder of this chapter is structured as follows: First, the experimental results per building block will be described. After this, the best performing methods are selected and the full architecture performance is assessed. These best methods are also used in the Flink environment to investigate the computational performance of the execution environment. Having all these insights, the answers to the research questions are formulated.

9.1 Experiment results

The experimental results of the tested architecture are given following the building blocks of the experimental setup given in Section 8.1.1. These blocks are described apart from each other next. The performance of each of the methods will be described and the best performing ones will be selected.

9.1.1 Sentiment analysis

The test cases run for sentiment analysis are described in Section 8.3.1. The results of these tests are found in Table 9.1. The results on the validation data show the best predictive performance for SentiWordNet 3 with an accuracy of 0.616 and F1-score of 0.730. Interestingly, none of the methods achieves an overall accuracy higher than 62%. The explanation for this is that all considered methods are trained on relatively short, subjective texts (mostly tweets or movie reviews) and will hence have less power on longer, objective news articles. Note, the results only show the overall performance metrics measured over the results.

The windowed metrics provide insights on the behavior of the models over time. Figure 9.1 shows these metrics for the SentiWordNet 3 lexicon. The remainder of the windowed results can be found in Appendix C.1. Some of the results show a clear pattern in the windowed metrics. This pattern corresponds to the one found in Figure 8.5 (the proportion of positive articles over time).

Investigating the statistical relation between the models and the proportion of positive articles, one can look at two aspects: the correlation of the curves over time and the equality of sample distributions. For the former, the auto-correlation function can be found, monitoring change
### 9.1. EXPERIMENT RESULTS

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentiwordnet 3</td>
<td>0.616</td>
<td>0.590</td>
<td>0.960</td>
<td>0.730</td>
</tr>
<tr>
<td>SenticNet</td>
<td>0.611</td>
<td>0.588</td>
<td>0.945</td>
<td>0.725</td>
</tr>
<tr>
<td>SentiStrength</td>
<td>0.562</td>
<td>0.620</td>
<td>0.496</td>
<td>0.551</td>
</tr>
<tr>
<td>Densify</td>
<td>0.610</td>
<td>0.688</td>
<td>0.514</td>
<td>0.588</td>
</tr>
<tr>
<td>Sentiment140</td>
<td>0.574</td>
<td>0.604</td>
<td>0.618</td>
<td>0.611</td>
</tr>
<tr>
<td>NRC Hashtag</td>
<td>0.495</td>
<td>0.642</td>
<td>0.151</td>
<td>0.245</td>
</tr>
<tr>
<td>TextBlob SA</td>
<td>0.570</td>
<td>0.562</td>
<td>0.933</td>
<td>0.702</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.591</td>
<td><strong>0.848</strong></td>
<td>0.332</td>
<td>0.477</td>
</tr>
</tbody>
</table>

Table 9.1: Performance metrics of different sentiment models.

Figure 9.1: Windowed metrics for SentiWordNet 3.
### 9.1. EXPERIMENT RESULTS

<table>
<thead>
<tr>
<th>Method</th>
<th>Correlation</th>
<th>KS test statistic</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet 3</td>
<td>0.952</td>
<td>0.227</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>SenticNet</td>
<td>0.951</td>
<td>0.215</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>SentiStrength</td>
<td>0.063</td>
<td>0.323</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>Densify</td>
<td>0.054</td>
<td>0.385</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>Sentiment140</td>
<td>0.525</td>
<td>0.286</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>NRC Hashtag</td>
<td>-0.952</td>
<td>0.226</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>Textblob</td>
<td>0.974</td>
<td>0.140</td>
<td>$&lt; 1 \cdot 10^{-15}$</td>
</tr>
</tbody>
</table>

Table 9.2: Correlation with respect to the proportion of positive articles.

![QQ-plot](image)

Figure 9.2: QQ-plot of distribution of proportion of positive articles and windowed accuracy of SWN3 along with the line $y = x$.

in curves over time. The latter can be achieved by a conducting a two-sample Kolmogorov-Smirnoff test. This statistical test tests whether the distribution of the two random variables from which the samples are drawn follow the same distribution. The null hypothesis states the two distributions are the same. The test statistic for the test is the maximal distance between the empirical distributions.

Conducting both tests, one will find the results in Table 9.2. The correlation is particularly high with SentiWordNet 3, SenticNet and Textblob. The similarity of the former two methods is as expected, as SenticNet used SentiWordNet as basis for its development. The high correlation of these methods corresponds with the high recall of the methods, thus meaning that they prefer to classify articles as positive. Interestingly however, the distributions of none of the lexicon results are following the same distribution as the portion of positive articles. To gain more insights in the differences of the distribution, the QQ-plot shown in Figure 9.2. This figure shows that the distribution of the classifications is similar, yet the windowed accuracy, on average, is higher. These results together indicate that the classifier is somewhat biased towards positive articles, yet this is not at the cost of misclassification of negative articles.
9.1.2 Adaptive sentiment model

Introduced in Chapter 6, the AdaUSA algorithm is tested using the SentiWordNet 3 lexicon, which had the highest performance in the previous test cases. Executing the test cases described in Chapter 8 on the validation dataset, one will find the results given in Table 9.3. Note, the first row of this table denotes the static baseline lexicon (i.e., no adaptations are applied). The rolling metrics of the different test cases are found in Appendix C.2.

Compared to this baseline, one can see that tumbled learning does not yield any significant increases in performance. Also, the forgetting mechanism tied to tumbled learning does not make a large difference in the performance of the method over time. On the other hand, slid learning does have an impact on the performance of the method. One can observe that the forgetting behavior in this case makes the difference. The best results are achieved by slid learning and tumbled forgetting, achieving an accuracy of 0.666 and an F1-score of 0.757. With a relative increase of 8.1%, this method significantly increase the performance of the lexicon in the financial domain over time.

Again, one can investigate the correlation and distributional statistic of the two-sample Kolmogorov-Smirnoff test. The autocorrelation for AdaUSA with slid learning and tumbled forgetting is 0.928 and the $p$-value for the statistical test is again smaller than $1.0 \cdot 10^{-15}$, thus showing that the two samples are not drawn from the same distribution. Similar to the baseline SWN3 lexicon, the QQ-plot of AdaUSA with slid learning and tumbled forgetting is found in Figure 9.3. From this figure, one can see that the distribution of accuracy is now much higher than that of the proportion of positive articles. This, once again, shows that the procedure is likely to classify positive as positive and is able to classify negative articles as negative (but might fail as well).

![Figure 9.3: QQ-plot of distribution of proportion of positive articles and windowed accuracy of AdaUSA with slid learning and tumbled forgetting along with the line $y = x$.](image)
9.1. EXPERIMENT RESULTS

<table>
<thead>
<tr>
<th>Learning</th>
<th>Forgetting</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Any</td>
<td>0.616</td>
<td>0.590</td>
<td>0.960</td>
<td>0.730</td>
</tr>
<tr>
<td>Tumbling</td>
<td>No</td>
<td>0.617</td>
<td>0.590</td>
<td>0.964</td>
<td>0.732</td>
</tr>
<tr>
<td>Tumbling</td>
<td>Tumbling</td>
<td>0.616</td>
<td>0.589</td>
<td>0.963</td>
<td>0.731</td>
</tr>
<tr>
<td>Tumbling</td>
<td>Sliding</td>
<td>0.616</td>
<td>0.589</td>
<td>0.963</td>
<td>0.731</td>
</tr>
<tr>
<td>Sliding</td>
<td>No</td>
<td>0.603</td>
<td>0.580</td>
<td>0.966</td>
<td>0.725</td>
</tr>
<tr>
<td>Sliding</td>
<td>Tumbling</td>
<td>0.666</td>
<td>0.622</td>
<td>0.967</td>
<td>0.757</td>
</tr>
<tr>
<td>Sliding</td>
<td>Sliding</td>
<td>0.627</td>
<td>0.597</td>
<td>0.961</td>
<td>0.736</td>
</tr>
</tbody>
</table>

Table 9.3: Results for different configurations of the AdaUSA method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tagged</th>
<th>Classified</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache OpenNLP</td>
<td>0.580</td>
<td>0.354</td>
<td>0.301</td>
<td>0.314</td>
<td>0.181</td>
</tr>
<tr>
<td>Stanford CoreNLP</td>
<td>0.851</td>
<td>0.487</td>
<td>0.299</td>
<td>0.397</td>
<td>0.250</td>
</tr>
<tr>
<td>Illinois CogComp NLP</td>
<td>0.835</td>
<td>0.401</td>
<td>0.289</td>
<td>0.354</td>
<td>0.192</td>
</tr>
<tr>
<td>SpaCy NER</td>
<td>0.839</td>
<td>0.350</td>
<td>0.187</td>
<td>0.293</td>
<td>0.126</td>
</tr>
</tbody>
</table>

Table 9.4: Performance metrics of different named entity recognition models.

9.1.3 Named entity recognition

The results of the experiments on the named entity recognition models are shown in Table 9.4. The table shows whether the true company is tagged as an organization by the model (column ‘Tagged’) and whether it classified it as the targeted company (column ‘Classified’). Additionally, the precision, recall and F1-score of the company classifications are given. From these results, it can be concluded that Stanford CoreNLP achieves the highest accuracy in classifying the target company of the text. Note, all of the models achieve mediocre to bad classification accuracy.

News articles oftentimes include references to or comments from other companies which interferes with finding the correct company. For example, an article targeting airplane manufacturer Boeing might also involve an interview asking the experiences of flight organization Ryanair using Boeing aircrafts. In roughly 35% of the cases, this was the reason of failure in the CoreNLP model. Another explanation for this behavior is that many news articles provide overviews of many different companies as an industry-wide update. Especially articles written by Hannah Kuchler and Pilita Clark (14.3% of articles) write these overviews. These two factors explain most of the mistakes in experiments.

Looking at the windowed accuracy over time, one can observe the results for the CoreNLP model found in Figure 9.4. In this figure, one can see a clear increase in accuracy over time. Especially from 2012 onward, the average accuracy rises to over 90% tagging and 60% classification. This increase is suspected, as the models for entity recognition are trained recently (CoreNLP is introduced in 2014) and hence are likely to be trained on more recent data sources. The windowed accuracies of the remaining tested entity recognition models are found in Appendix C.3.

Knowing the results from the tagging algorithm, the Stanford CoreNLP model can now be used as recognizer in Algorithms 8 and Algorithm 9 to test the post-processing steps of ADom-selection and RDom-selection. The tests are executed for a value of $i$ varying between 1 and 8. The results of this experiment are found in Table 9.5. The ‘Portion’ column of this table indicates which portion of articles was selected in the dominance set. Furthermore, the results
9.1. EXPERIMENT RESULTS

Figure 9.4: Windowed accuracy of Stanford CoreNLP over time.

<table>
<thead>
<tr>
<th>$i$</th>
<th>Portion</th>
<th>Tagged</th>
<th>Classified</th>
<th>Portion</th>
<th>Tagged</th>
<th>Classified</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0.851</td>
<td>0.487</td>
<td>1</td>
<td>0.851</td>
<td>0.487</td>
</tr>
<tr>
<td>1</td>
<td>0.995</td>
<td>0.856</td>
<td>0.490</td>
<td>0.995</td>
<td>0.856</td>
<td>0.493</td>
</tr>
<tr>
<td>2</td>
<td>0.864</td>
<td>0.877</td>
<td>0.533</td>
<td>0.479</td>
<td>0.911</td>
<td>0.681</td>
</tr>
<tr>
<td>3</td>
<td>0.630</td>
<td>0.907</td>
<td>0.612</td>
<td>0.330</td>
<td>0.925</td>
<td>0.740</td>
</tr>
<tr>
<td>4</td>
<td>0.463</td>
<td>0.921</td>
<td>0.670</td>
<td>0.238</td>
<td>0.941</td>
<td>0.775</td>
</tr>
<tr>
<td>5</td>
<td>0.341</td>
<td>0.930</td>
<td>0.704</td>
<td>0.171</td>
<td>0.941</td>
<td>0.787</td>
</tr>
<tr>
<td>6</td>
<td>0.254</td>
<td>0.932</td>
<td>0.723</td>
<td>0.124</td>
<td>0.941</td>
<td>0.796</td>
</tr>
<tr>
<td>7</td>
<td>0.190</td>
<td>0.935</td>
<td>0.728</td>
<td>0.086</td>
<td>0.940</td>
<td>0.805</td>
</tr>
<tr>
<td>8</td>
<td>0.142</td>
<td>0.929</td>
<td>0.741</td>
<td>0.062</td>
<td>0.937</td>
<td>0.812</td>
</tr>
</tbody>
</table>

Table 9.5: Results for $ADom$-selection and $RDom$-selection in combination with Stanford CoreNLP.

show the tagging and classification accuracies which were also reported for the pre-trained models before. As one can see in the results table, there is a trade-off between achieving a high classification accuracy and the number of articles which are not considered because they are not in the dominance sets.

9.1.4 Named entity linking

Knowing the results from Section 9.1.3, it is now possible to assess the number of companies that can be linked to their corresponding online entity. This will be done by deploying a locally hosted version of DBPedia Spotlight. The performance of the linking module will be measured over both the full NER result set and the correctly classified entries. The latter result will show the truly correct performance of the model.
### 9.1. EXPERIMENT RESULTS

<table>
<thead>
<tr>
<th>Method</th>
<th>All classifications</th>
<th>Correct classifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache OpenNLP</td>
<td>0.466</td>
<td>0.575</td>
</tr>
<tr>
<td>Stanford CoreNLP</td>
<td>0.561</td>
<td>0.716</td>
</tr>
<tr>
<td>Illinois CogComp NLP</td>
<td>0.772</td>
<td>0.850</td>
</tr>
<tr>
<td>Spacy NER</td>
<td>0.741</td>
<td>0.818</td>
</tr>
</tbody>
</table>

Table 9.6: Proportion of correctly linked entities with respect to the all classified companies and all correctly classified companies.

The results of this experiment are found in Table 9.6. The best performing method for linking the companies is Illinois CogComp NLP. However, due to its low performance on finding the target company, the overall performance of the method is a correct classification of 1532 articles, is still worse than Stanford CoreNLP which achieves a correct classification of 1569 articles.

#### 9.1.5 Full architecture

Having the results of both sentiment and NER blocks, the performance of the whole system can be assessed. As explained in Section 3.2.3, the results for the full architecture will consist of a logical AND operator on the results of both sentiment and NER tasks. This final evaluation will be done using the results of the AdaUSA algorithm with slid learning, tumbled forgetting, using the SentiWordNet 3 sentiment model and Stanford CoreNLP NER model with RDom-selection. These models for the two tasks are chosen because these showed the highest performance in their respective experiments.

Using only the baseline methodology (without the novel solutions), one can see the best performing sentiment lexicon to be SentiWordNet 3 with an accuracy of 0.616 (as is seen in Table 9.1). For named entity recognition, one can see a maximal classification accuracy for Stanford CoreNLP achieving an accuracy of 0.487 (seen in Table 9.4). Using a logical AND over the results per articles, the overall accuracy is given by 0.412. Using the methods introduced in Chapter 6 and Chapter 7, the results of the subtasks are improved. In Table 9.3 an accuracy of 0.666 is achieved by AdaUSA and in Section 9.1.3 an accuracy of 0.812 are found or RDom3-selection. Inserting these results in the logical AND, an overall accuracy of 0.611 is achieved. Compared to the baseline methods, both algorithms introduced in this thesis contributed to a significant increase in model performance. More detailed insights on this will be given in Section 10.2.

#### 9.1.6 Computational performance results

Apart from the predictive performance, an important aspect of the full system is the environment which will utilize the sentiment and entity recognition models. A comparison is made between Apache Storm and Apache Flink. The only predictive performance related trade-off between the two is the support of only Java in Flink and support for Java with Python sub-processes in Storm. Both AdaUSA and RDom-selec are platform independent method. Both SentiWordNet 3 and Stanford CoreNLP are Java-based models, hence the platform limitation will not influence the predictive performance of one framework over the other.
Therefore, the relevance in choice between the frameworks is that of computational performance. As mentioned in Section 8.1.6, the computational complexity is measured in the time it takes to process a 1000 articles. The results of this experiment are found in Table 9.7. Apache Flink achieves the best results with an average processing time of 436.22 seconds per 1000 articles.

<table>
<thead>
<tr>
<th>Environment \ Task</th>
<th>Sentiment task</th>
<th>NER task</th>
<th>Full system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Flink</td>
<td>1.00 sec</td>
<td>431.66 sec</td>
<td>436.22 sec</td>
</tr>
<tr>
<td>Apache Storm</td>
<td>122.13 sec</td>
<td>1121.45 sec</td>
<td>1293.07 sec</td>
</tr>
</tbody>
</table>

Table 9.7: Average computational performance over 1000 articles in seconds.

9.2 Financial author assessment

The final part of task tackled in this thesis is the assessment of financial article authors using the constructed architecture. This architecture will consist of the best performing methods from the SA and NER tasks on the validation data used in the computationally best performing execution environment. From the previous results, one can conclude that this is a Flink streaming environment, with the AdaUSA algorithm with slid learning, tumbled forgetting and SentiWordNet 3 as baseline lexicon for sentiment analysis and Stanford CoreNLP with RDom-selection for named entity recognition. Reviewing the portions of articles selected for each index of RDom-selection, the choice is made to use RDom\textsubscript{3}-selection, since this set contains a significant amount of the input articles while also achieving high predictive performance on both tagging and classification of companies.

In addition to the tested architecture, additional functionality for stock price retrieval is needed. This functionality is described in Section 8.4. The SPARQL queries from that section will, together with the stock compare methodology and block described in Section 5.6, be used to assess the true quality of the authors in the validation dataset. All results per author will be given next.

9.2.1 Author analysis results

To gather insights on how well the proposed architecture can analyze the articles written by a certain author, the results from Sections 9.1.1 and 9.1.3 are grouped per author. The results of this test case are found in Table 9.8. From this table, one can see deviating results of the classification for both sentiment analysis and named entity recognition per author. For example, the achieved accuracy of articles written by Steve Johnson is significantly higher than that of Andy Sharman, especially in sentiment analysis.

In a general sense, the results of the sentiment analysis split show a clear distinction between well performing authors (Marriage, Skypala, Hollinger, Clark, Johnson and Bradshaw) and poorly performing authors (Sharman, Kuchler, Chaffin and Wright). In terms of NER, the results are less divided, yet there is still a distinction between the different authors.

A possible explanation for the difference in sentiment accuracies is the level of emotion put in the articles written by an author. If less emotion is put in the text, the sentiment model will have difficulties in classifying the sentiment, which will result in ‘guessing’ behavior (i.e. tossing a coin), meaning the accuracy ends up around 50%. This behavior is seen in the less-performing group of authors.
9.2. FINANCIAL AUTHOR ASSESSMENT

<table>
<thead>
<tr>
<th>Author name</th>
<th>SA accuracy</th>
<th>NER accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy Sharman</td>
<td>0.426</td>
<td>0.778</td>
</tr>
<tr>
<td>Hannah Kuchler</td>
<td>0.530</td>
<td>0.731</td>
</tr>
<tr>
<td>Joshua Chaffin</td>
<td>0.574</td>
<td>0.697</td>
</tr>
<tr>
<td>Madison Marriage</td>
<td>0.728</td>
<td>0.782</td>
</tr>
<tr>
<td>Pauline Skypala</td>
<td>0.822</td>
<td>0.671</td>
</tr>
<tr>
<td>Peggy Hollinger</td>
<td>0.731</td>
<td>0.633</td>
</tr>
<tr>
<td>Pilita Clark</td>
<td>0.704</td>
<td>0.725</td>
</tr>
<tr>
<td>Robert Wright</td>
<td>0.562</td>
<td>0.654</td>
</tr>
<tr>
<td>Steve Johnson</td>
<td>0.829</td>
<td>0.782</td>
</tr>
<tr>
<td>Tim Bradshaw</td>
<td>0.782</td>
<td>0.802</td>
</tr>
</tbody>
</table>

Table 9.8: Results of sentiment analysis and named entity recognition grouped by author.

9.2.2 Author quality assessment

Besides the performance of the tested architecture, the overall performance of the articles compared to stock prices can be investigated. It is interesting to link the predicted sentiments and classified companies of the final architecture to the stock tickers and stock prices of article publishing dates. For this, three factors of interest need to be measured:

- Which proportion can be linked to a semantic entity?
- Which proportion of the linked entities is linked to a stock ticker on either DBPedia or Wikidata?
- Which portion of the articles has the same polarity as the movement in stock price on that day?

The results of these questions per author are shown in Table 9.9. From this table, one can see that particular authors achieve significantly better results than others. For example, the linked portion of articles is higher for Tim Bradshaw than for Pauline Skypala. Also, when looking at the portion of tickers found for the different authors, Peggy Hollinger stands out for having a lower score. Linking this to her expertise (editor of aerospace and defence industries), it reasonable to think that most of the companies she writes about do not concern exchange listed entities.

<table>
<thead>
<tr>
<th>Author name</th>
<th>Linked portion</th>
<th>Tickers linked</th>
<th>Correctly written</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy Sharman</td>
<td>0.587</td>
<td>0.553</td>
<td>0.487</td>
</tr>
<tr>
<td>Hannah Kuchler</td>
<td>0.623</td>
<td>0.633</td>
<td>0.556</td>
</tr>
<tr>
<td>Joshua Chaffin</td>
<td>0.561</td>
<td>0.378</td>
<td>0.364</td>
</tr>
<tr>
<td>Madison Marriage</td>
<td>0.590</td>
<td>0.576</td>
<td>0.223</td>
</tr>
<tr>
<td>Pauline Skypala</td>
<td>0.414</td>
<td>0.448</td>
<td>0.601</td>
</tr>
<tr>
<td>Peggy Hollinger</td>
<td>0.653</td>
<td>0.344</td>
<td>0.304</td>
</tr>
<tr>
<td>Pilita Clark</td>
<td>0.538</td>
<td>0.551</td>
<td>0.443</td>
</tr>
<tr>
<td>Robert Wright</td>
<td>0.516</td>
<td>0.392</td>
<td>0.581</td>
</tr>
<tr>
<td>Steve Johnson</td>
<td>0.559</td>
<td>0.511</td>
<td>0.333</td>
</tr>
<tr>
<td>Tim Bradshaw</td>
<td>0.672</td>
<td>0.553</td>
<td>0.466</td>
</tr>
</tbody>
</table>

Table 9.9: Results of linking and assessment grouped by author.
Lastly, the assessment scores show large differences in expertise of the different considered authors. For example, Madison Marriage achieves a correctness score of 0.223, which indicates that her writing behavior does not mirror the flow of the stock prices of the companies she writes about. On the other hand, Pauline Skypala has a much higher assessment score, indicating that she has a higher impact on stocks with the articles she writes.

9.3 Research question answers

Knowing the results of different prediction and computationally oriented test cases, it is now possible to formulate the answers to the different research questions proposed in Section 3.1. The questions will be answered according to the domain they arose from.

9.3.1 Sentiment analysis

For sentiment analysis, Section 2.4.4 and 3.1 mentioned that most of the current lexicon-based approaches to sentiment analysis are powerful on short, subjective texts. As this thesis involves longer, more objective news articles, it is interesting to identify the performance of the same methods on the article dataset.

Q1.1: With what predictive performance do different existing pre-trained lexicons perform on financial news articles over time?

The predictive performance of the different methods is found in Section 9.1.1. These results show a clear difference in performance of the considered methods over time. Due to the methods having different origins than the one used in the thesis, one can see a clear drop in quality of all methods. Three of the considered methods (SentiWordNet 3, SenticNet and Textblob) show a clear correlation with the proportion of positive articles in the validation dataset, thus revealing a bias towards positive articles.

Q1.2: Which of these existing methods on sentiment analysis performs best?

Reviewing the results in Tabel 9.1, the best performing method is SentiWordNet 3, having an accuracy of 0.616 and F1-score of 0.748. Note however, as already indicated in the previous answer, this method is biased towards positive classifications. Therefore, it will most likely decrease in performance if the number of negative articles would increase.

Q1.3: Does a stream lexicon adaptation method achieve higher accuracy than batch-based ones for sentiment analysis over time?

Chapter 6 introduced a novel method for adaptive unsupervised sentiment analysis (AdaUSA). This method has different behavioral parameters and uses a baseline lexicon as input. The best performing configuration of the method was using the SentiWordNet 3 lexicon, slid learning and tumbled forgetting. This configuration achieved an accuracy of 0.666 and an F1-score of 0.757 on the validation dataset. This score is an increase of 8.1% accuracy compared to the fixed SentiWordNet 3 lexicon.
Q1: With what performance can the sentiment of financial news articles be classified using sentiment lexicon methodology?

The best performance found on the validation dataset is originating from the AdaUSA model using SentiWordNet 3 as base lexicon, slid learning and tumbled learning. This method showed an accuracy of 0.666 and F1-score of 0.757. Note that this method outperformed the supervised naive bayes classifier and other (non-adaptive) lexicons. However, the base lexicon SWN3 is shown to be biased towards positive classification and should therefore be used with caution when deployed in AdaUSA over a long period of time.

9.3.2 Named entity recognition and linking

Next to sentiment analysis, a vital part of the work considered named entity recognition and linking. For this, four different methods for NER and a named entity linking engine were used. The results of the experiments can now be used to formulate the answers to the two proposed research questions.

Q2.1: With what accuracy can a company targeted by a news article be recognized from the text of a news article?

The predictive performance of four different pre-trained named entity recognition models was tested against the validation dataset. The best performing model is from Stanford CoreNLP achieving a tagging accuracy of 0.851 and classification accuracy of 0.487 with an F1-score of 0.250. The thesis introduced a novel post-processing method for target selection in Chapter 7. Using the RDom-selection technique together with Stanford CoreNLP, a maximum tagging accuracy of 0.941 and classification accuracy of 0.812 is achieved.

Q2.2: What portion of the identified companies can be correctly linked to their corresponding semantic web entity using named entity linking?

Using DBPedia Spotlight for named entity linking, the performance of Stanford CoreNLP is a linking portion of 0.716 over the correctly classified companies. Looking at the other named entity recognition models, the Illinois CogComp model achieved the highest portion of linked companies with an accuracy of 0.850 over the correctly classified companies. However, due to the lower performance of the CogComp model on entity recognition, the absolute number of articles classified and linked correctly is still higher for Stanford CoreNLP with RDom-selection.

9.3.3 Full architecture

Combining the results of the last two subsections, the full architecture performance can now be assessed. The predictive performance of the full system is given by selecting the best performing methods from sentiment analysis and entity recognition. Now, the computational complexity of both methods can be assessed. For this, the answers to the two proposed research questions for the full system will be answered.
Q3.1: Which streaming environment will process financial news articles with the lowest computational complexity?

A comparison is made between the streaming frameworks Apache Flink and Apache Storm in which the average processing time of 1000 documents is measured. The fastest processing time is found with Apache Flink taking an average of 436.22 seconds to process 1000 articles. A potential limitation of the frameworks was the limitation of Apache Flink to only work with Java. However, as the best performing sentiment and entity recognition models are Java-based, this limitation is not influencing the overall system performance. Therefore, the best performing execution environment is Apache Flink.

Q3.2: With what accuracy can the field of expertise of financial news article authors be described given the sentiment of their article and companies they write about?

This question is answered by combining the results of the sentiment analysis, named entity recognition and streaming research questions. Additionally, this questions involves assessing the quality of news article authors used in the validation dataset. To assess this metric, the AdaUSA method with SentiWordNet 3 as baseline is combined with the Stanford CoreNLP NER model with RDom-selection in the Flink execution environment. The sentiment model achieved an accuracy of 0.666 and the named entity recognition model achieved a maximum of 0.812. The combined results achieved are an accuracy of 0.611 and a computational performance 436.22 seconds per 1000 articles.

Using this final architecture, the performance of the different tasks can be used to assess the quality of the authors in the validation dataset. Grouping the performances of the tasks per author shows different performances for each of them. Especially in sentiment analysis, there is a significant distinction between well performing authors and poorly performing ones. This difference is most likely due to the level of emotion put in articles by their authors. If there is a high level of emotion, the sentiment is classified more easily. If the level of emotion is lower (i.e. the article is written in a neutral way), the sentiment model is less likely to be able to classify the sentiment, thus will purely guess the sentiment. This guessing is similar to flipping a coin and hence will result in a score of around 50%, as is seen among the poorly scoring authors.

Looking at the performance of linking and quality assessment of authors, again, the results per author deviate significantly from one to another. In particular, the full assessment score for each author shows large differences. As can be seen in Table 9.9, Madison Marriage is, according to the used architecture, the author with the lowest adherence to the stock prices of the companies she writes about with a score of 0.223. The highest impact among authors in the validation dataset on the corresponding stock prices is found for Pauline Skypala with a score of 0.601.
Chapter 10

Conclusion

This chapter will aim at reviewing the work done in the thesis. It will recap the steps taken to come up with the final architecture and method configuration. The results of Chapter 9 are discussed and potential risks or problems with the analysis are identified. Using these, the limitations of the work are identified after which open opportunities for future work will be presented.

10.1 Thesis conclusion

This thesis combined the current knowledge on three different fields into a descriptive architecture for financial news article authors. Sentiment analysis, named entity recognition and linking and stream data mining were combined to create a real-time assessment method for unstructured financial information. The work aimed at investigating the best possible combination of methods for the task.

The work started with formulating research questions in the three different targeted fields of expertise. For sentiment analysis (Q1, Q1.1 and Q1.2), the performance of eight current pre-trained and probabilistic (supervised) methods was identified. Additionally, the AdaUSA method was introduced which adapts a baseline lexicon by contextual sentiments over time (Q1.3). Named entity linking (Q2.1 and Q2.2) was split to named entity recognition (Q2.1) and linking (Q2.2). Four NER engines were considered and the best performing one was picked for the final model. The post-processing methods ADom-selection and RDom-selection are introduced to strengthen the results of the NER methods. Lastly, stream mining (Q3.1 and Q3.2) was used to combine the former two models into a stream environment. The environment was built in Apache Flink and Apache Storm to assess the computational complexity between both.

To assess all considered methods, an experimental setup is created in Chapter 8. This setup is different from the architecture used in practice in that it specifically aims at reviewing the classified results against their true labels. During construction of the setup, a modular design was kept in mind. The setup makes it more straightforward to test additional methods in the future without having to make major changes to the existing structure. Using this setup, all models used in the work were validated and their performance over time was reported.

The final proposed architecture is an Apache Flink environment for data stream mining. Data pre-processing was performed by removing noisy words (stopwords, bad encoding, etc.) using the pre-existent methodology provided by the NLTK NLP toolbox. For sentiment analysis, the AdaUSA algorithm, using the SentiWordNet 3 general purpose lexicon, slid learning and tumbled forgetting achieved the highest predictive performance with an accuracy of 0.666 and F1-score of 0.757. Named entity recognition was performed most accurately by the
10.2 Discussion

The results of the work shown in Chapter 9 will be discussed according to their sub-task. After this, the full-system will be discussed and potential limitations will be addressed. Altogether, this section aims at providing more insights into the results found for the different models.

Firstly, taking a closer look at the results of the sentiment models, one can see a clear shift in performance over time in many of them. For example, the best performing method, SentiWordNet 3, has a clear performance peak between 2007 and 2013, as was seen in Figure 9.1. This performance shift has three clear reasons. The first is the bias towards classifying articles as positive, as was shown in Table 9.2. The table shows a very high correlation between the model and proportion of positive articles over time. The second reason is the fact that the SWN3 lexicon is a general purpose model, thus aiming at achieving fair performance across multiple domains. Lastly, the baseline lexicons did not take into account changes of sentiments over time and hence showed suboptimal results.

Using AdaUSA, the latter two issues are resolved by adapting the lexicon to both a domain- and time-specific one. Although showing a significant increase in performance, the AdaUSA still showed a bias towards positive articles as the recall was very high. The approach of AdaUSA in this thesis was based on a beam-search technique, meaning that the best performing baseline lexicon (SWN3) was supposed to also generate the best results when used with AdaUSA. The results of the adaptive algorithm could hence potentially be improved by optimizing the hyper-parameters and considering different baseline lexicons as input.
Secondly, the named entity recognition showed a large difference between the ability to tag an organization and classify the correct single target company. The hypothesis for this failure given in Section 9.1.3 is that certain authors in the validation set do not clearly target a single company in their article. Reviewing the results of the CoreNLP model, one observes the predictive performance per author shown in Table 9.8. Indeed, there is a large difference between authors.

To improve the results of the NER company tagging, the post-processing methods \textit{ADom}-selection and \textit{RDom}-selection were introduced. These methods increased the performance of company classification significantly. However, one should take into account that both methods provide filters on input articles. When these filters are deployed to tight, one can observe a lot of information being discarded. An example of this is found in \textit{RDom}s-selection. This method achieves the highest accuracy on the validation data, yet it only considers 6.2% of the input articles. During deployment, this trade-off between accuracy and information loss should be accounted for.

Another component of the study was named entity recognition and semantic retrieval. This part took the classified company from the NER task and linked it to its specific semantic entity to retrieve more insights on the entity. A limitation in this task was the availability of one suitable linking engine: DBPedia Spotlight. This engine only allowed linking of entities to the DBPedia platform, whereas other platforms for linked data might hold more insights. This was especially the case with retrieving the stock ticker of a particular company, which was mostly retrieved from Wikidata instead of DBPedia. If more platforms could be linked to the entity (e.g. Open Corporates, etc.) there is more functionality for linking entities to their stock tracker.

Next, the streaming engine was assessed in the study. The results for the processing time of 1000 articles showed a significant difference in the usage of Apache Storm and Apache Flink. A study into benchmarks of both systems already showed this same result between the two platforms [59]. Especially when arrival rates are fluctuating (as is the case with news article publication), Apache Flink shows significantly better results than Storm, thus acknowledging the findings in this thesis.

Lastly, the full system architecture was deployed to assess news article authors automatically. For this, stock ticker and price were retrieved from semantic web knowledge bases and a stock API. The results found in Section 9.2.2 show a clear difference in article impact. Note, the architecture created is a decision support system. This means that it not inclined to make decisions on its own, rather it is created to assist authors in the choices they make. Having this much difference in impact hence means that a stock trader most likely wants to follow the advice given by a better performing author.

The results of the author assessment should however be used with caution. As can be seen in Table 9.9, the number of entities and tickers linked significantly shrink the total number of articles considered. Reviewing the results of the full architecture, one can see that the entity recognition post-processing step keeps only 33% of input information. Next, 71.6% can be linked and, of the linked entities, roughly 50% can be linked to a stock ticker. This means that, from all input articles, only approximately 11.8% are used to assess the quality of authors. Discarding this much information could be resulting in biased results towards certain authors. However, due to the assumptions and scope of this thesis as defined in Section 3.2, providing insights on this bias is left as future work.
10.3 Main contributions

As mentioned in Section 1.3, the main contribution of this thesis is the analysis of unstructured financial information in a stream mining environment. However, as this full system consists of different building blocks, the contributions can be divided in multiple smaller pieces. The following contributions are distinguishable from the work done in this thesis:

1. **Introduce a novel unsupervised lexicon adaptation method for financial news articles streams:** After assessing the performance of the state-of-the-art in lexicon-based sentiment analysis on financial news articles, the thesis introduced a novel method which adapts sentiment scores over time. AdaUSA uses windowed learning and forgetting behavior on contextual sentiment scores to adapt sentiments in a baseline lexicon. On the used validation dataset, the algorithm showed an relative increase of 8.1% compared to the baseline SentiWordNet 3 lexicon without adaptation.

2. **Introduce a novel post-processing technique for target company extraction from financial news articles:** Named entity recognition methods were used to classify companies targeted by financial news articles. To enhance the results of this classification, two novel post-processing methods to filter relevant articles were introduced. ADom-selection and RDom-selection define a metric to assess whether a certain company stands out in an article. Using these metrics, a maximal increase in classification accuracy of 32.5% can be obtained. However, there is a trade-off between loosing information and achieving better results, as is seen in the results received from the validation data tests.

3. **Create a cross-programming language stream mining test architecture for modular machine learning model testing:** The Apache Storm-based test system for stream data mining models can be utilized for validation in a wide scale of contexts. The test framework is not limited to sentiment analysis and NER but can be scaled to work with other machine learning tasks (e.g. topic modelling, time series analysis, etc.) as well. Models from different programming languages can be used in the test framework. Although tested with models in Java and Python, the architecture is capable of handling other languages, such as JavaScript, Ruby, Go, C(++)

4. **Full financial news article stream classification architecture in both Apache Storm and Apache Flink:** The final best performing models for sentiment analysis and entity recognition are deployed in both an Apache Flink and an Apache Storm environment and, with minor adjustments, can be deployed for real-time article analysis. This environment combined with the novel AdaUSA and RDom-selection algorithms improved the performance of the architecture significantly and made it achieve stable performance over time. Using the classification architecture, the impact of financial news article authors was investigated, showing deviating performances over the different considered authors.

10.4 Thesis limitations

As time for creation of this work was limited, several assumptions and limitations were defined to narrow down the scope and hence time spent on execution. This section will review the main limitations of the work and how to overcome them in future work.
10.5. FUTURE RESEARCH

- **Data availability**: The data used for the work was retrieved from the academic database LexisNexis. Although this platform includes many different properties per articles (e.g. author, publication date, target company, etc.), there was information missing (sentiment score) which was added manually throughout the project. This manual labelling is both time-consuming and prone to human error. The former aspect resulted in a quantity limitation to the available validation data. Previous literature suggests validating models against pre-constructed, renowned validation sets which have labels already in place. However, these validation sets did not fit the need of the work as they mainly consisted of short, highly subjective information.

- **Stream environment**: The data stream environment used in this thesis is, besides a feature, a limitation. Many state-of-the-art models use complex deep neural structures for sentiment analysis and NER. Despite these methods achieving very high accuracy (90+%) on both tasks, they did not fit the need for this work and hence could not be used. A solution to this problem, widely used in practice, is the hybrid streaming environment described in Section 2.7.6. This solution, however, looses the main benefit of stream data mining and hence is not feasible to fit this thesis.

- **Efficient Market Hypothesis**: The main assumption of the work in this thesis is the EMH described in Section 1.2. This assumptions states that stock prices of a company are directly and instantaneously influenced by the news articles written thereon. This hypothesis has only been empirically validated in previous research, yet no hard evidence is provided confirming it. To get a better view on the influence of news articles on stock prices, a preliminary study should be conducted into this influence on the available validation data. Because of time limitations, this thesis did not consider this study and just accepted the hypothesis as a main assumption.

10.5 Future research

The limitations described in the previous section open up opportunities of future research after the results of this work. These open problems will mainly concern improvements of the models in a data stream mining environment as that was the feature with the highest influence on model choice.

10.5.1 Sentiment analysis

The work in this thesis introduced a novel solution for adaptive unsupervised sentiment analysis. This method uses a number of parameters as input. The experiments showed a significant increase in performance using the SentiWordNet 3 lexicon, slid learning and tumbled forgetting. Due to limitation in time, there are no similar experiments conducted using the other tested lexicons found in this thesis. Future research might therefore take a deeper look into the behavior of other lexicon-based methods in combination with AdaUSA.

Additionally, more experiments might be conducted into different other hyperparameters of the AdaUSA model. For example, the window sizes of learning and forgetting are fixed in the conducted experiments. Better results might be achieved when different sets of hyperparameters are used. One can play around with the window sizes, values for ρ and α in the adaptation algorithm and so on.
10.5. FUTURE RESEARCH

The adaptive algorithm is introduced using three different behaviors for both learning and forgetting. No adaptation, slid adaptation and tumbled adaptation on a count basis do not pose a complete set of adaptation techniques. Especially when more input source, thus more input articles are used, the considered count-based windows might not suffice as windows are closed too quickly. Alternatively, a time-based window might be used to have more control on the timeline at which articles are used for adaptations. One might consider time-based windows of a month, year, or even more to adapt sentiment scores over time. According to [92], the sentiment of words changes significantly over a yearly period, hence this might be a good point to start improving the method.

Lastly, the learning and forgetting behavior of AdaUSA are not necessarily limited to lexicon-based methods. When considering supervised methods over time (i.e. online or incremental learning), the adaptation windows can be used to adapt a model over time. For example, using tumbled learning, one can learn an ensemble of classifiers on the data observed over time. The forgetting behavior can then guide the algorithm to forget models from the ensemble, thus providing means to deploy the algorithm indefinitely.

10.5.2 Named entity linking

As is seen in the results in Table 9.4, there is significant difference in whether a company is recognized by the model and it is actually classified as target for that article. A main challenge is to make this translation more accurate. In this thesis, the target company is chosen from the list of tagged companies by selecting the most frequently tagged one. To get better classifications, other rules or methods could be used to strengthen the translation from tagging to result.

Additionally, the thesis only considered pre-trained models as labeled NER data was very costly to construct. Future applications of the methods might include creating a custom NER tagger which is trained using labeled news articles. For example, many custom trained NER models using deep neural networks exist in literature. Once trained, these methods can be deployed in stream environment, potentially achieving better results [119][29].

Lastly, tagging the target company could be done using methods different from named entity recognition. For example, direct text classification might be used to solve the task. The main reason for not choosing such a method in this work is the limitation of the number of target classes in most of these methods. As they are trained on a fixed label space, they will never be able to recognize a company not included in the training data. Models without this limitation should be used to solve the task. Examples of available methods to do this in a batch mode are Lee et al [62] and Yan et al [117].

10.5.3 Financial news article author assessment

The main task handled in this thesis was the assessment of the impact made by financial news article authors. This has been done using the proposed architecture presented in Chapter 5. Using this full architecture, information from open linked data sources was linked with sentiment polarity scores to assess the impact made. The main limitation is this method is the dependence on the efficient market hypothesis (EMH). This hypothesis states that there is immediate response from stock prices when important articles are written on concerned companies. The study hence used sentiment scores and stock prices from the same day to assess the impact made. More analysis can be done into the relevance of the EMH. Next studies can show generate insights on the actual speed a market responses to certain news. A time-gap could be identified, thus quantifying how long to wait for certain news to be digested by the market.
Furthermore, the study used different steps in which input articles were discarded. The named entity recognition post processing algorithm discarded an approximate 33% of article from the input stream. Next, 72% of the remaining articles could be linked to a semantic web target entity. Finally, approximately 50% of the linked entities were linked to a stock ticker, leaving 12% of the input articles used in the analysis. Future research can strengthen these results by trying to increase the final number of articles considered. For this, the best way forward is focusing on the processing step discarding the most articles, thus the post-processing algorithm.
Bibliography


BIBLIOGRAPHY


Appendix A

Detailed Methodology Description

This appendix provides detailed information on the different methods used throughout the thesis. The two tables below list the specifications of the used methods, as well as a short description on how they work and the url to access more information.

<table>
<thead>
<tr>
<th>Method</th>
<th>Task</th>
<th>Used version</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet 3</td>
<td>SA</td>
<td>3.0</td>
<td>Java/Python</td>
</tr>
<tr>
<td>SentiStrength</td>
<td>SA</td>
<td>2.2</td>
<td>Java</td>
</tr>
<tr>
<td>SenticNet</td>
<td>SA</td>
<td>5.0</td>
<td>Java/Python</td>
</tr>
<tr>
<td>Densify</td>
<td>SA</td>
<td>1.0</td>
<td>Java/Python</td>
</tr>
<tr>
<td>Vader</td>
<td>SA</td>
<td>1.0</td>
<td>Python</td>
</tr>
<tr>
<td>Sentiment140</td>
<td>SA</td>
<td>1.0</td>
<td>Java/Python</td>
</tr>
<tr>
<td>NRC Hashtag</td>
<td>SA</td>
<td>0.2</td>
<td>Java/Python</td>
</tr>
<tr>
<td>TextBlob SA</td>
<td>SA</td>
<td>0.15.2</td>
<td>Python</td>
</tr>
<tr>
<td>NLTK</td>
<td>SA</td>
<td>3.4</td>
<td>Python</td>
</tr>
<tr>
<td>Apache OpenNLP</td>
<td>NER</td>
<td>1.9.1</td>
<td>Java</td>
</tr>
<tr>
<td>Stanford CoreNLP</td>
<td>NER</td>
<td>3.9.2</td>
<td>Java</td>
</tr>
<tr>
<td>CogComp NLP</td>
<td>NER</td>
<td>3.0</td>
<td>Java</td>
</tr>
<tr>
<td>Spacy NER</td>
<td>NER</td>
<td>2.0</td>
<td>Python</td>
</tr>
<tr>
<td>DBPedia Spotlight</td>
<td>NEL</td>
<td>1.0.0</td>
<td>Java/Python</td>
</tr>
<tr>
<td>Apache Storm</td>
<td>Streaming</td>
<td>1.3.2</td>
<td>Java</td>
</tr>
<tr>
<td>Apache Flink</td>
<td>Streaming</td>
<td>0.9.2</td>
<td>Java/Python</td>
</tr>
</tbody>
</table>

Table A.1: Version details of the used methods of this thesis.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet 3</td>
<td>General purpose lexicon distinguishing different word types (verbs, adverbs, pronouns, etc.). Classifies sentiment on words basis and combines the result to retrieve document-based sentiment.</td>
<td><a href="https://github.com/aesuli/sentiwordnet">https://github.com/aesuli/sentiwordnet</a></td>
</tr>
<tr>
<td>SentiStrength</td>
<td>General purpose lexicon. Makes use of Ngrams size 1 to 4. The lexicon is wrapped in a compiled Java package which makes it more easy to use.</td>
<td><a href="http://sentistrength.wlv.ac.uk/">http://sentistrength.wlv.ac.uk/</a></td>
</tr>
<tr>
<td>SenticNet</td>
<td>General purpose lexicon constructed using recurrent neural networks on a widespread number of contexts and datasets. The method is utilized using Ngrams of size 2 to 4 as these yield the highest predictive performance.</td>
<td><a href="http://sentic.net/">http://sentic.net/</a></td>
</tr>
<tr>
<td>Densify</td>
<td>General purpose lexicon constructed by an ensemble of different heuristic classifiers. In the thesis, this method is utilized making use of Ngrams size 1 to 4.</td>
<td><a href="https://www.aclweb.org/anthology/W18-3107">https://www.aclweb.org/anthology/W18-3107</a></td>
</tr>
<tr>
<td>Vader</td>
<td>Rule-based sentiment assistance. Vader uses five pre-specified rules for strengthening or bending the sentiment classified by a lexicon. For example, it will flip sentiment when it sees 'not' or 'but'.</td>
<td><a href="https://github.com/cjhutto/vaderSentiment">https://github.com/cjhutto/vaderSentiment</a></td>
</tr>
<tr>
<td>Sentiment140</td>
<td>Twitter-based lexicon trained on a large set of tweets across multiple domains. The method is validated by SentiBench as well-performing on longer, more informal texts [87].</td>
<td><a href="http://www.sentiment140.com/">http://www.sentiment140.com/</a></td>
</tr>
<tr>
<td>NRC Hashtag</td>
<td>Lexicon constructed using a Distant Supervised Approach. The lexicon is learned by tweet hashtags and will associate sentiment to contextual words [87].</td>
<td><a href="http://saifmohammad.com/WebPages/lexicons.html">http://saifmohammad.com/WebPages/lexicons.html</a></td>
</tr>
<tr>
<td>TextBlob SA</td>
<td>Pre-trained sentiment analysis model. The model uses an ensemble of Naive Bayes, Random Forst and Logistic Regression to classify the sentiment of a document.</td>
<td><a href="https://github.com/sloria/TextBlob">https://github.com/sloria/TextBlob</a></td>
</tr>
<tr>
<td>NLTK</td>
<td>Most commonly used NLP toolbox in Python. Includes many different classifiers, transformers and models. NLTK is used for the supervised benchmark using a Naive Bayes classifier and the Bag of Words model.</td>
<td><a href="https://www.nltk.org/">https://www.nltk.org/</a></td>
</tr>
</tbody>
</table>

Table A.2: Detailed descriptions of the used sentiment models of this thesis.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache OpenNLP</td>
<td>Apache licenced NLP engine with pre-trained NER models. OpenNLP is used to tag entities in a sentence. The most frequent organization tag is the classified target company.</td>
<td><a href="https://opennlp.apache.org/">https://opennlp.apache.org/</a></td>
</tr>
<tr>
<td>Stanford CoreNLP</td>
<td>Developed on Stanford University, Stanford CoreNLP is an NLP toolbox for Java. The NLP engine is based on the Apache OpenNLP engine with major improvements based on numerous NLP models created over time.</td>
<td><a href="https://stanfordnlp.github.io/CoreNLP/">https://stanfordnlp.github.io/CoreNLP/</a></td>
</tr>
<tr>
<td>CogComp NLP</td>
<td>Created by Illinois University, CogComp NLP is another NLP toolbox. Its NER engine uses CoreNLP as a baseline and adds profound rule-based heuristics to enhance the results.</td>
<td><a href="http://cogcomp.org/">http://cogcomp.org/</a></td>
</tr>
<tr>
<td>Spacy NER</td>
<td>Spacy is an NLP toolbox for Python. The main difference between NLTK and Spacy is that Spacy only utilizes the best performing model, rather than swiss-army-knife NLTK. Spacy contains a pre-trained NER model trained using neural networks.</td>
<td><a href="https://spacy.io/">https://spacy.io/</a></td>
</tr>
<tr>
<td>DBPedia Spotlight</td>
<td>DBPedia Spotlight is a named entity linking framework developed by DBPedia. Spotlight takes text as input and will output all entities linked on DBPedia, Wikidata and other open data platforms.</td>
<td><a href="https://www.dbpedia-spotlight.org/">https://www.dbpedia-spotlight.org/</a></td>
</tr>
<tr>
<td>Apache Storm</td>
<td>Apache Storm is one of the top level projects from Apache. It is a streaming engine in which the flow of data can be created using a Directed Acyclic Graph (DAG). It makes use of data sources called &quot;Spout&quot; and data transformation &quot;Bolt&quot; blocks to analyze data. Storm is designed to be functioning with any available programming language able to read from a stdin and write to a stdout.</td>
<td><a href="https://storm.apache.org/">https://storm.apache.org/</a></td>
</tr>
<tr>
<td>Apache Flink</td>
<td>Apache Flink is one of the top level project. Created by DataArtisans, Flink is a streaming engine especially designed for multithreaded stream mining. It makes use of an extended MapReduce paradigm to efficiently analyze streams of information.</td>
<td><a href="https://flink.apache.org/">https://flink.apache.org/</a></td>
</tr>
</tbody>
</table>

Table A.3: Detailed descriptions of the used NER and NEL methods and streaming environments of this thesis.
Appendix B

Validation Data Details

Below, one can find a detailed overview of the available news articles per author in the validation dataset. All considered articles originate from The Financial Times magazine.

<table>
<thead>
<tr>
<th>Author name</th>
<th>Expertise</th>
<th># Articles</th>
<th>Date from</th>
<th>Date until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy Sharman</td>
<td>Motor vehicles and cars industry correspondent</td>
<td>448</td>
<td>02-09-1994</td>
<td>29-03-2017</td>
</tr>
<tr>
<td>Hannah Kuchler</td>
<td>Biotech, digital health correspondent</td>
<td>387</td>
<td>04-01-2001</td>
<td>16-10-2018</td>
</tr>
<tr>
<td>Joshua Chaffin</td>
<td>New York correspondent</td>
<td>604</td>
<td>18-12-2009</td>
<td>15-02-2019</td>
</tr>
<tr>
<td>Madison Marriage</td>
<td>Accounting and tax correspondent</td>
<td>334</td>
<td>09-10-2009</td>
<td>03-02-2019</td>
</tr>
<tr>
<td>Pauline Skypala</td>
<td>Freelance economics journalist</td>
<td>497</td>
<td>22-02-2002</td>
<td>12-02-2019</td>
</tr>
<tr>
<td>Peggy Hollinger</td>
<td>Editor aerospace and defence industries</td>
<td>415</td>
<td>08-04-2012</td>
<td>30-01-2019</td>
</tr>
<tr>
<td>Pilita Clark</td>
<td>Business columnist</td>
<td>256</td>
<td>28-09-1999</td>
<td>18-11-2018</td>
</tr>
<tr>
<td>Robert Wright</td>
<td>Social policy correspondent</td>
<td>735</td>
<td>11-04-2008</td>
<td>01-02-2019</td>
</tr>
<tr>
<td>Tim Bradshaw</td>
<td>Global tech correspondent</td>
<td>521</td>
<td>28-12-2007</td>
<td>23-03-2016</td>
</tr>
</tbody>
</table>

Table B.1: Detailed author information from the validation dataset.
Appendix C

Detailed Validation Results

This appendix will describe the results for the windowed metrics mentioned in Section 8.1.5. Per sentiment method, the windowed precision, recall, accuracy and F1-score will be shown. Per named entity recognition model, the windowed performance of both tagging and classification accuracy are shown.
C.1 Sentiment models

C.1.1 SentiWordNet 3
C.1.2 SentiStrength

![Graphs showing Rolling Recall, Rolling F1-Score, Rolling Precision, and Rolling Accuracy over time.](image-url)
C.1.3 SenticNet
C.1.4 Densify

![Graphs showing rolling recall, f1 score, precision, and accuracy over time.](image-url)
C.1.5 Sentiment140

Rolling Recall

Rolling F1-Score

Rolling Precision

Rolling Accuracy
C.1.6 NRC Hashtag

![Graphs showing performance metrics over time for NRC Hashtag](image)
C.1.7 Textblob
C.2 AdaUSA results

C.2.1 AdaUSA tumbled learning and no forgetting
C.2.2 AdaUSA tumbled learning and tumbled forgetting
C.2.3 AdaUSA tumbled learning and slid forgetting
C.2.4 AdaUSA slid learning and no forgetting
C.2.5 AdaUSA slid learning and tumbled forgetting

![Graphs showing rolling recall, rolling F1-score, rolling precision, and rolling accuracy for AdaUSA results.](image)
C.2.6 AdaUSA slid learning and slid forgetting

![Graphs showing rolling recall, rolling F1-score, rolling precision, and rolling accuracy over time.](image)
C.3 Named entity recognition models

C.3.1 Spacy NER
C.3.2 Apache OpenNLP
C.3.3 Stanford CoreNLP

[Graph showing windowed accuracies with blue and orange lines]

1. Tagged proportion
2. Classified proportion
C.3.4 Illinois CogComp NLP
Acronyms

**ADom-selection** absolute dominance selection. 1, 7, 71, 80, 84, 93, 94, 103, 105, 106

**RDom-selection** relative dominance selection. 1, 7, 71, 80, 83, 93–96, 99, 100, 103–106

**TDOC** term degree of correlation. 60, 63, 64

**AdaUSA** adaptive unsupervised sentiment analysis. 1, 5, 7, 83, 84, 92, 93, 95, 96, 98–100, 103, 104, 106–108

**D&RA** Data & Reporting Advisory. 5, 12

**EMH** efficient market hypothesis. 13, 33, 34, 107, 108

**IR** information retrieval. 19, 20

**ML** machine learning. 11, 25, 52

**NEL** named entity linking. 5, 7, 28, 39, 55, 77, 118, 120

**NER** named entity recognition. 1, 5, 7, 28, 52, 54, 55, 77, 79, 80, 83, 94–96, 99, 100, 103, 105–108, 118, 120

**NLP** natural language processing. 7, 11, 19–21, 34, 43, 44, 49–52, 103, 119, 120

**RDF** resource description framework. 27

**SA** Sentiment Analysis. 12, 21, 77, 96, 118

**SWN3** SentiWordNet 3. 5, 91, 92, 99, 104

**TF-IDF** term frequency-inverse document frequency. 20, 22, 34, 60