Conceptualization of an Authorship Attribution Pipeline for Blog Articles

Final Report Graduation

Submitted by Jan Trienes

In fulfilment of the requirements for the degree
Bachelor of Science
To be awarded by the
Fontys University of Applied Sciences

Solingen, June 12, 2017

Information Page

Fontys Hogeschool Techniek en Logistiek
Postbus 141, 5900 AC Venlo

Final Report of Bachelor Thesis Project

Name of student:	Jan Trienes
Student number:	2219180
Course:	Informatics - Software Engineering
Period:	February 2017 – June 2017
Company name:	codecentric AG
Address:	Hochstraße 11
Postcode, City:	42697, Solingen
Country:	Germany
Company coach:	N. Blättermann
Email:	`niko.blaettermann@codecentric.de`
University coach:	J. Jacobs
Email:	`jan.jacobs@fontys.nl`
Examinator:	F. van Odenhoven
External domain expert:	I. Bischofs
Non-disclosure agreement:	no

Abstract

The question which author has written a certain text has been around since people have started to put their ideas and thoughts into writing. Recent advancements in statistics and the capabilities of powerful machine learning algorithms have created a research field which is known as authorship attribution. This field is of great significance for many applications in humanities, journalism and law. For instance, authorship attribution can be used to detect fraudulent product reviews on popular online platforms.

The company codecentric AG, as an innovator in agile software development, is constantly interested in state-of-the-art technologies and best practices. This bachelor thesis project is concerned with the conceptualization of an automated authorship attribution pipeline that contributes to the product portfolio of codecentric AG.

This project systematically analyzes the main components of machine learning based authorship attribution by conducting a literature review. Furthermore, comparison criteria are defined which are used to assess the ability of machine learning models to detect the author of a text. Two attribution approaches and a set of different stylistic markers are empirically compared in experiments using a real-world blog article dataset.

Finally, the insights of the literature research and the experiments are integrated into a reusable authorship attribution library. This project specifies the requirements of the library from a functional and non-functional perspective. Several design issues to make this library reusable and extensible are discussed and solved by using popular software engineering design patterns. A prototype of this library is implemented, that unifies modern natural language processing technologies in the Python ecosystem.

Zusammenfassung

Seitdem Autoren ihre Ideen und Gedanken verschriftlichen können, kommt immer wieder die Frage auf, wer der Urheber eines umstrittenen Textes ist. Sowohl die Fortschritte in der Statistik, als auch die Möglichkeiten, die moderne Algorithmen des maschinellen Lernens mit sich bringen, haben ein neues Forschungsgebiet hervorgebracht, welches als Autorenerkennung bekannt ist. Die Forschungen in diesem Gebiet haben eine hohe Wichtigkeit für interdisziplinäre Anwendungen in den Bereichen Geisteswissenschaften, Journalismus und Rechtswissenschaften. Die Autorenerkennung kann zum Beispiel von Online Plattformen genutzt werden, um gefälschte Produktbewertungen zu erkennen.

Die codecentric AG gilt als Vorreiter in der agilen Softwareentwicklung. Daher ist die Firma kontinuierlich an neuen Trends und modernsten Technologien interessiert. Diese Bachelorarbeit beschäftigt sich mit der Konzeptualisierung einer automatischen Pipeline zur Autorenerkennung. Die Erkenntnisse, welche innerhalb dieses Projektes gewonnen werden, tragen zu dem Portfolio der codecentric AG bei.

Dieses Projekt analysiert systematisch die Hauptbestandteile der Autorenerkennung, welche auf den Prinzipien des maschinellen Lernens basiert ist. Damit verschiedene Lösungsmöglichkeiten zur Autorenerkennung miteinander verglichen werden können, identifiziert dieses Projekt ein gemeinsames Kriterium. Zwei Lösungsansätze zur Autorenerkennung und verschiedene stilistische Merkmale werden nach diesem Kriterium empirisch anhand eines Blogartikel Datensatzes verglichen.

Die Erkenntnisse der Literaturrecherche und den Experimenten, werden abschließend in einer Wiederverwendbaren Bibliothek zur Autorenerkennung gebündelt. Dieses Projekt spezifiziert die Anforderungen einer solchen Bibliothek von einer funktionalen, als auch nicht-funktionalen Perspektive. Populäre Architekturmuster werden genutzt, damit die Bibliothek in der Zukunft wiederverwendbar und erweiterbar ist. Ein Prototyp der Bibliothek, welcher moderne Sprachverarbeitungstechnologien des Python Ökosystems vereint, wird implementiert.

Statement of Authenticity

I, the undersigned, hereby certify that I have compiled and written the attached document / piece of work and the underlying work without assistance from anyone except the specifically assigned academic supervisors and examiners. This work is solely my own, and I am solely responsible for the content, organization, and making of this document / piece of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents / pieces of work provided by my course / my academic institution, and I understand that this document / piece of work will not be accepted for evaluation or for the award of academic credits if it is determined that it has not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g. ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document / piece of work and the underlying work will not be accepted for evaluation or for the award of academic credits if it is determined that it embodies plagiarism.

Name:	Jan Trienes
Student Number:	2219180
Place/Date:	Solingen, June 12, 2017

Signature:

Jan Trienes

Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	Definition
API	Application Programming Interface
CSV	Comma Separated Value
CV	Cross Validation
EDA	Exploratory Data Analysis
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JGAAP	Java Graphical Authorship Attribution Platform
kNN	k-Nearest Neighbours
LDA	Latent Dirichlet Allocation
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
POS	Part-of-Speech
SQL	Structured Query Language
SRS	Software Requirements Specification
SVM	Support Vector Machine
UML	Unified Modeling Language
WYSIWYG	What You See Is What You Get

Glossary

Term	**Definition**
Bag-of-words	The bag-of-words model is a representation of a written text in form of a vector recording how often each token occurs. Grammar and word order is disregarded.
Corpus	A collection of texts written by a number of different authors.
Feature	A feature is a measured property of an instance. For example, the number of times an author has used the word "of" is a feature.
Instance	A real-world object with a set of properties. Multiple instances are used by a machine learning algorithm to learn the mapping of the properties to a class label. For example, a text written by an author is an instance.
kNN	The machine learning algorithm k-Nearest Neighbors (kNN) computes the distance of an unlabeled instance to the nearest k labeled instances. Afterwards, the most frequent class of these neighbors is assigned to the unlabeled instance.
MLP	A Multilayer Perceptron (MLP) is a machine learning algorithm. It transforms an instance to a class label by applying a series of functions. It is a form of neural network that adapts itself upon a misclassification.
n-grams	An n-gram is a sequence of n adjacent elements occurring in a text. Common forms are bigrams (two elements) or trigrams (three elements).
POS-tags	Part-of-Speech (POS) tags denote the grammatical function of a word (i.e. whether it is a noun, verb, adjective etc.).
Random forest	A random forest is a machine learning algorithm that can be used for classification. It randomly constructs multiple decision trees with different samples of the dataset and aggregates their outcomes.
SVM	A Support Vector Machine (SVM) is a machine learning algorithm. It tries to find a linear hyperplane that separates the instances of two classes from each other. New observations are assigned to either of these classes depending on the decision boundary.
Token	A token is a single unit within a text (i.e. a *word*). The process of extracting tokens is known as tokenization.

Chapter 1

Introduction

This report is written as part of the bachelor thesis project in the 8th semester of the Software Engineering course at Fontys University of Applied Sciences, Venlo, Netherlands. It depicts the final report of the project which was carried out in the period of February 2017 – June 2017. The purpose of this document is to prove that the needed competencies to realize a complex software project have been acquired during the studies. This chapter outlines the context of the project in further detail by providing information about the company in which this thesis is written and the problem that the data science department of this company has. Furthermore, background information about the problem domain is given.

1.1 The Company codecentric AG

The company codecentric AG is an expert in tailored software solutions and an innovator in agile software development. It has provided a wide range of services in the area of software development, infrastructure services, digitalization and consulting since 2005. The company has 15 subsidiaries in four European countries and its headquarters are located in Solingen, Germany. More than 350 employees work for codecentric AG.

This project was carried out in the document solutions department in Solingen. The team supports various clients with enterprise content management solutions. Although this thesis was written in the document solutions department, it is the data science department situated in Hamburg which is the customer of the project.

1.2 Problem Description

The newly founded data science department of codecentric is constantly interested in increasing their knowledge about state-of-the-art data science practices and to extend their product portfolio. The acquired knowledge is utilized in existing projects of codecentric while an extended product portfolio provides new business opportunities for the team. However, because of their limited resources, it is not always possible to investigate complex issues and new trends in data science in an extensive way.

Because of the increasing amounts of textual data available that stem from modern forms of communication such as blogs, emails, instant messaging and social media platforms, there is also a growing need to understand the content of this data and to find patterns in it. This topic area involving Natural Language Processing (NLP) is of special interest for the data science department. They want to learn about techniques and processes to utilize this data in an automated fashion, such that new business applications can be explored.

Many applications in the field of text mining and NLP share similar methodologies for tasks like data acquisition, preparation and utilization. The sub-field of authorship attribution is of high interest for the department, because it involves a large variety of different techniques and provides good business opportunities.

1.3 Problem Domain: Authorship Attribution

The fields of text mining and NLP are concerned with the extraction of patterns and knowledge in large amounts of unstructured textual data. The sub-field of authorship attribution can be defined as the process of inferring characteristics of an author by inspecting text the author has written previously. This inferred knowledge is in turn used to attribute authorship to a piece of text with unknown authorship. A famous example where this science has been successfully applied is the novel "The Cuckoo's Calling" by J.K. Rowling. She published it under the pseudonym Robert Galbraith in 2013. Because Rowling mentioned in the past that she was interested in publishing a crime novel, rumours arose that "The Cuckoo's Calling" was written by her and not by Galbraith. Scientists used former writings of Rowling to learn her stylistic fingerprint. By comparing it with the characteristics of the new novel, they showed that Rowling is indeed the author of this novel, even before it was confirmed by her (Juola, 2013).

Authorship attribution has a long history starting in the 19th century, but recent developments in statistics, machine learning and NLP, alongside large amounts of available data, have brought new approaches and applications into the field (Stamatatos, 2009). One prominent example is fraud detection. In the past, the science has been used to successfully inspect the authorship of PhD theses or the authenticity of online reviews on popular websites such as Amazon and Yelp (Ramnial, Panchoo, and Pudaruth, 2015; Shrestha, Mukherjee, and Solorio, 2016). While authorship attribution can be useful to spot fake reviews on online platforms or plagiarism in a published work, it also raises a number of ethical questions. For instance, it could be used to uncover the identity of a whistleblower. Interestingly, a research group is concerned with writing style anonymization which is a closely related field (McDonald et al., 2012). To summarize, authorship attribution has many applications in interdisciplinary fields including humanities, forensics, journalism and law which make it an interesting topic for the data science department.

1.4 Document Structure

The remainder of this document is divided into seven chapters. Chapter two provides a definition of the assignment and describes the project planning. The theoretical part of this assignment was executed as a research project. The main research goals as well as the design of it is given in chapter three. Chapter four outlines the general components of authorship attribution approaches and provides a list of methods that were evaluated during the experiments of this project. Chapter five explains how these methods were evaluated under common criteria, how the methods performed, and which practical issues need to be considered when performing authorship attribution. Chapter six explains the requirements elicitation process and states the functional and non-functional requirements of the pipeline. The design and implementation of the pipeline is discussed in chapter seven. The last chapter concludes this report and gives a reflection on the results of this project.

Chapter 2

Project Description

The following chapter illustrates the assignment in greater detail. High level project phases, their deliverables and the scope of the project will be defined. Furthermore, the project management and quality assurance approach is explained. This chapter is concluded by a time planning and a description of the development tools used in this project.

2.1 Assignment Definition

The data science department has asked for further investigation into the field of authorship attribution, since it entails a wide range of different technologies and processes that the department can utilize in projects involving NLP. In consultation with the customer, the use case of attributing authorship to blog articles has been formulated. A pipeline that combines various techniques of NLP to detect the author of blog posts is developed accordingly by this project. A scenario that the pipeline should be able to solve is the following: *"given the articles published in the years 2008 to 2016, detect the authors of articles written in 2017"*.

The data from the codecentric blog[1] is used during the development of this project. This dataset serves as a starting point for analysis and development. However, the resulting system and approaches are not limited to it. Instead, they are applicable to any kind of authorship attribution problem/dataset, and in a broader picture also applicable to any other text mining problem.

2.2 Project Scope

This project aims to acquire general knowledge about NLP techniques based on the use case of authorship attribution. The project performs a literature review and several experiments. These activities are used to analyze the problem domain and to assess the practicability of authorship attribution. Data science related issues, that come up during these activities, are covered from a practical perspective to the limited extent that is needed to develop the pipeline. It is beyond the scope of the project to investigate theoretical problems specific to authorship attribution, because they do not contribute towards the main goal of the customer. This scope constraint is also reflected in the type of literature that this project considers to get acquainted with the problem domain. Research surveys are consulted that discuss the most popular approaches in authorship attribution. The selection of literature is discussed in Section 3.3.4.

Furthermore, the implementation of the authorship attribution pipeline is not meant to be a fully functional tool which achieves perfect results on the given dataset. Instead, it is a prototype that demonstrates how NLP techniques can be automated and combined within the Python ecosystem. This technology requirement is due to the fact that the customer uses Python in a majority of their projects. Besides that, Python is the preferred language of the machine learning community. Although this project uses the codecentric blog article dataset during the experiments and development of the pipeline, the resulting software artifact should be applicable to any kind of authorship attribution problem.

[1]see `https://blog.codecentric.de/en/`

2.3 Project Phases and Deliverables

This section describes the high level project phases which were identified in consultation with the company supervisor and the customer. Each phase consists of a set of activities and their deliverables that were delivered to the customer at the end of the project.

Research Project (Phase 1)

The first project phase analyzed the problem domain of authorship attribution. In order to get acquainted with the state-of-the-art of the topic itself, an initial literature review was conducted that collected relevant literature. Following the literature review, a research project was designed that *identified* and empirically *compared* authorship attribution approaches. The execution of this research project provided the needed theoretical and practical knowledge to conceptualize an automated processing pipeline and thus formed the basis for phase two. The following were the main deliverables of the first phase:

- Collection of relevant literature
- Research design (text document)
- Assessment criteria for authorship attribution approaches (text document)
- Identified authorship attribution approaches (text document)
- Approach comparison (experiment source code, approach prototypes, graphics)
- Comparison conclusions (text document)

Pipeline Specification and Prototype Development (Phase 2)

The theoretical and practical knowledge obtained during the first phase of the project was used to specify the requirements of an authorship attribution pipeline. These requirements were documented by means of user stories and non-functional requirements. This phase concluded with the development of a prototype that implements the specified requirements. The deliverables of this phase included:

- Requirements specification of pipeline (text document)
- Pipeline design (text document, Unified Modeling Language (UML) diagrams)
- Pipeline prototype (source code, documentation, executable)

Recommendation (Phase 3)

After the use case of authorship attribution was examined in detail, overall consequences, ideas and approaches were drawn for future work on NLP that the data science department carries out. A collaborative discussion was carried out to identify and document them. Furthermore, this phase finalized the project by assembling the final report as well as the presentation. The deliverables of this phase included:

- Recommendations for future work on NLP (text document)
- Final report
- Final presentation

2.4 Project Management

Since this project was carried out individually, formal project management approaches such as the classic waterfall model, but also agile approaches like SCRUM and Kanban were neither directly applicable nor worthwhile to employ. This relates to the fact that they generally make assumptions about certain roles and processes. These project management approaches and their guidelines would have added increased complexity and unnecessary overhead to the project.

Therefore, it was necessary to find a low-overhead solution that still had quality promoting mechanisms and reduced the risk that the project would diverge from the customers' requirements. The approach used in this project embraced agile principles such as frequent reviews of (sub-)deliverables, user stories, test driven development and employed a formal framework for the research activities. The following paragraphs explain these issues in further detail.

One of the main phases of this project was the analysis phase. It provided the required knowledge about both the theoretical and practical issues in machine learning based authorship attribution. Since this knowledge translated into the requirements of the pipeline software, it was important to enhance the quality and reproducibility of the analysis activities undertaken. This was achieved using the formal framework for the research part defined by Verschuren et al. (2010).

Following the research phase was the specification of the pipeline prototype requirements. Instead of specifying them in the traditional format of functional requirements, user stories were used which are often employed in an agile project setting. The motivation for this decision and the format of the user stories can be found in Section 6.6. The user stories and their implementation progress was maintained in a product backlog that was shared with the customer.

The development of the pipeline followed a semi-iterative process. Because the resulting software is a prototype with a limited scope and the requirements were not expected to change throughout the process, it was possible to specify the requirements and an initial design before the actual implementation. During the realisation, each user story has been incrementally implemented. Intermediate reviews of the current implementation status enabled the customer to provide additional ideas or remarks.

There were very few uncertainties in the project, so it was not necessary to integrate any risk management procedures into the project management. Nevertheless, one risk has been identified with respect to the quality of communication with the customer. Since they were located at another branch of codecentric, no frequent inter-personal communication was possible. However, good communication and travel facilities at codecentric helped to keep this risk at a minimum.

2.5 Quality Assurance

The quality assurance measures that were used in this project ranged from reviews of (sub-)deliverables to automated static and dynamic testing practices. These issues will be further explained in this section.

Reviews of the (sub-)deliverables defined in Section 2.3 were scheduled two-week intervals together with the involved stakeholders from the data science department. Besides the possibility of finding quality issues in the intermediate results, the short interval catered for any adjustments that had to be made to the product or the scope. Although these reviews were in principle applicable to any kind of reviewable artifact, they were more suitable to artifacts which concern the main expertise of the department. Therefore, additional software development quality assurance mechanisms had to be employed.

During the development of the pipeline prototype in the second phase, the standard quality assurance mechanisms for software development were used. That means that an automated testing procedure was in place and written source code was checked against compliance with the language specific coding conventions. Besides that, the implementation status of a user story was verified with user acceptance tests. The quality assurance plan in Appendix A.2 defines these procedures in greater detail. Furthermore, a domain specific quality promoting measure that relates to projects involving machine learning, was the use of proper algorithm validation and evaluation methods to verify the results and performance of an algorithm. These methods are described in Section 5.3.

2.6 Time Planning

Table 2.1 shows the time planning of the project phases and the individual activities. A Gantt chart of this planning can be found in Appendix Figure A.1.

Phase	Week	Activity	CW
1	1 – 4	Planning & initial literature research	5 – 8
	5	Research project design	9
Milestone I	**02.03.17**	**Project plan**	**9**
1	6	Research project design	10
	7	Develop criteria to compare approaches	11
	8, 9	Identify authorship attribution approaches	12, 13
Milestone II	**28.03.17**	**Mid-term report**	**13**
1	10	Experiments	14
Milestone III	**10.04.17**	**Mid-term presentation**	**15**
1	11, 12	Experiments	15, 16
2	13	Define requirements of authorship attribution pipeline	17
	14 – 16	Develop pipeline prototype	18 – 20
3	17 – 19	Writing	21 – 23
3	20	Printing & presentation preparation	24
Milestone IV	**13.06.17**	**Final report**	**24**
Milestone V	**21**	**Final presentation**	**25**

Table 2.1: Time planning of project phases and activities.

2.7 Development Tools

A number of tools have been used throughout the process of the project. Table 2.2 briefly explains each of them.

Name	**Purpose**
Git	Git is a distributed version control system. It has been used to maintain a history of the project artifacts (e.g. text documents, source code, UML diagrams) and to share them with the involved stakeholders.
GitHub Enterprise	This is a private version of the GitHub platform hosted by codecentric. It has been used to maintain a product backlog and to share the intermediate status of a deliverable with involved stakeholders. The integrated issue tracking system and the project boards provide a very lightweight alternative to a feature-rich issue and bug tracking software such as Atlassian JIRA. Since this project was executed individually, this lightweight issue management was sufficient to maintain and communicate the implementation status of user stories with the stakeholders.
Python	Python is a general purpose object oriented programming language which has been used throughout all phases of the project. Python is widely used in the machine learning and NLP research communities. The third party library *pandas* has been used for data manipulation. Furthermore, *scikit-learn* has been used as the machine learning framework. Additionally, a number of NLP libraries have been investigated which are described in Section 7.3. The testing framework *pytest* has been used to write automated unit, integration, and acceptance tests. Python as well as the third party libraries are open sourced.
pip	The Python package manager *pip* has been used for dependency management and packaging of the pipeline prototype.
Jupyter	Jupyter is an open source interactive interpreter to execute Python code which is widely used in the scientific community. The Jupyter ecosystem features so called notebooks which unify executable code, their results and documentation into a single artefact. This environment has been used to execute the experiments in this project and to communicate their results.
Docker	Docker is a virtualization software that has been used for a variety of purposes. For example, it has been used to isolate the experiment environment consisting of a MySQL database and a Jupyter server, such that it can be easily installed on another machine to reproduce the results. Also, it has been used to execute the automated tests of the pipeline prototype in a clean and isolated environment.
Umlet	Umlet is a simple open source software program, used to create any kind of UML diagram. It has been used throughout all phases of the project.

Table 2.2: Development tools used throughout this project.

Chapter 3

Research Process Design

The first phase of this project entailed the review of related literature, the identification of possible approaches for the attribution problem and the empirical comparison of these approaches. The activities were executed in the bounds of a research framework. This chapter defines the main objective of this research, poses three central questions and provides a scope definition.

3.1 Research Framework

The following sections describe the conceptual design of the research project which has been created in accordance to the framework proposed by Verschuren et al. (2010). The technical design which describes the execution and the research methods can be found in Appendix B. The main objective of this research project can be defined as follows:

> *The research objective is to provide a criteria based assessment of authorship attribution techniques to the data science department, by identifying and comparing commonly used approaches that assign authorship to a piece of text.*

Figure 3.1 provides a schematic presentation of the research framework that was used to achieve the research goal. It shows the individual parts of the research and their relation to each other.

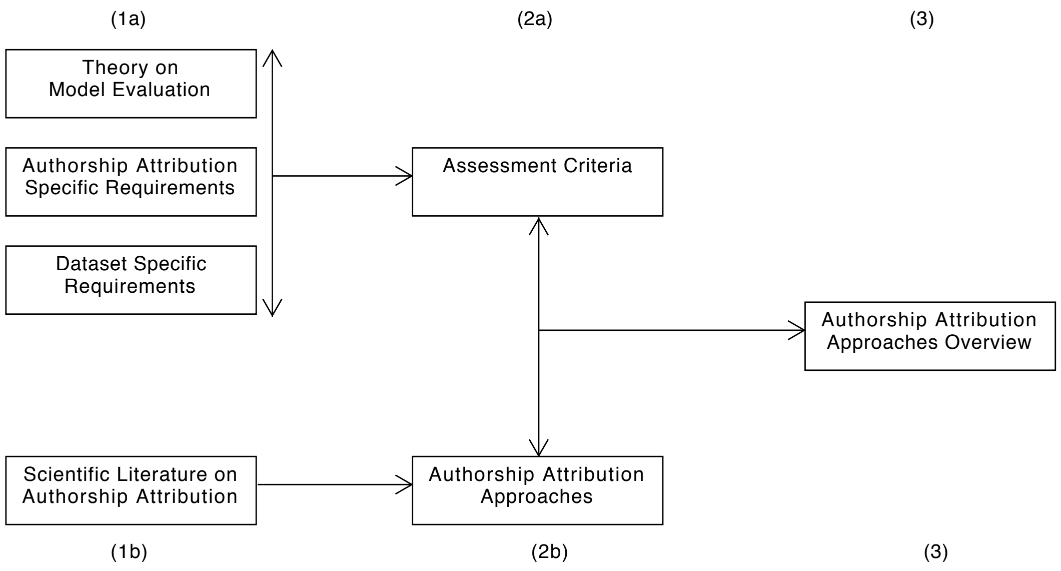
(1a)
(2a)
(3)
Theory on Model Evaluation
Authorship Attribution Specific Requirements
Dataset Specific Requirements
Assessment Criteria
Authorship Attribution Approaches Overview
Scientific Literature on Authorship Attribution
Authorship Attribution Approaches
(1b)
(2b)
(3)

Figure 3.1: Research framework to identify and compare authorship attribution approaches.

The research framework reads as follows: *"Studying related literature on model evaluation, in combination with the specific requirements that authorship attribution problems and the blog article dataset have (1a), provides the assessment criteria (2a). A review of existing scientific literature on authorship attribution (1b) provides a number of possible approaches to investigate (2b). A confrontation of the criteria and the approaches results in an overview of authorship attribution approaches and their capabilities (3)."*

3.2 Research Questions

The following questions are to be answered within this research project. The sequence of the questions also represents the order in which they are answered (i.e. RQ3 requires the results of RQ1 and RQ2 to be answered).

RQ1 What criteria are relevant when assessing and comparing the performance of multiple authorship attribution approaches?

1. What are commonly used performance indicators for machine learning algorithms?
2. What are commonly used performance indicators in the field of authorship attribution?
3. Which characteristics of a text corpus have an influence on the performance of an approach?

RQ2 Which authorship attribution approaches can be identified in literature?

RQ3 Which of the identified approaches perform well under the defined criteria?

3.3 Definitions of Key Concepts

This section defines the key concepts which have been used during this project. These definitions further refine and demarcate the scope by stating explicitly how they are used throughout this project.

3.3.1 Authorship Attribution Approach

For the purpose of this research project, an authorship attribution approach is defined as the combination of (a) the style markers of an author (i.e. features) and (b) an algorithm that assigns authorship to a piece of text. One could make an exhaustive comparison of different feature spaces and algorithms to evaluate their respective performance and to find the most adequate approach to solving blog article attribution problems. It is neither feasible to perform this comparison in limited time, nor would it contribute to the main goal of the project. Therefore, it is considered out of scope. Instead, this project limits itself to the most popular approaches that can be found in research surveys. The definition of a popular approach and the selection of literature can be found in Section 3.3.4.

3.3.2 Blog Article Dataset

A test dataset is used in order to compare particular approaches. This dataset consists of 1300 articles published in the codecentric blog in the period 2008 – 2016. The dataset is available as a Structured Query Language (SQL) dump. The conclusions that will be drawn from experiments are limited in their validity to this particular dataset. More reliable statements could be made by incorporating additional datasets. Since the compilation and preparation of these datasets is very time consuming and this research does not aim to provide a complete evaluation of the approaches, it is considered as optional only if time permits.

3.3.3 Performance Indicators

Research question RQ1 asks for criteria which assess the performance of an authorship attribution approach. Because machine learning based approaches use the same learning algorithms that are also used in other domains, the same performance indicators can be used. The performance of an approach is to be understood in a way that asks how well it has learned the characteristics of an author (i.e. whether it is able to distinguish between multiple authors). It is out of scope of this project to gather benchmarks concerning other properties of an attribution algorithm such as computational efficiency. This is due to the fact that the computational efficiency depends on a number of factors that are very use-case specific (e.g. the amount of data, available hardware) and are difficult to generalize.

3.3.4 Commonly Used Approaches

The problem of authorship attribution has a long history starting in the 19th century. Accordingly, a large number of possible approaches have been proposed by research (Stamatatos, 2009). Since comparing and testing all of these approaches is neither feasible nor within the scope of this research, a reasonable selection has to be made. There are 3 popular research surveys (judging by the citation count on google scholar)[1] that examine the previous work done in the field:

- "Authorship attribution" by Juola (2008).
- "A survey of modern authorship attribution methods" by Stamatatos (2009).
- "Computational methods in authorship attribution" by Koppel, Schler, and Argamon (2009).

This research focuses on the authorship attribution approaches described in these surveys. Using more than one research survey will increase the reliability of the individual approaches since they have been reviewed multiple times. There is another popular survey by Holmes (1994) which is not considered by this project. Because the other papers are more recent, it is expected that they reflect the current state of the field, while also respecting new trends in machine learning. It has to be noted, that the above mentioned literature does not cover deep learning, which is a popular trend in artificial intelligence. Because it has been shown that simpler techniques achieve good performance in authorship attribution, this project limits itself to these techniques.

The selection of relevant literature has the potential disadvantage that approaches, which are very suitable and would have a good performance under the defined criteria, might be missed. However, since this research does not aim for a complete overview of all possible approaches, this disadvantage is of lesser concern.

[1]Note: the surveys have been found by using the search term `authorship attribution`

Chapter 4

Identification of Approaches

This chapter describes the general components of an authorship attribution problem. Machine learning based attribution will be emphasized, which is based on the extraction of stylistic markers and a certain attribution method. Furthermore, this chapter highlights the approaches which have been identified as part of the literature review (RQ2), and which are examined in practice as part of the experiments (RQ3).

4.1 Overview of Approaches

The majority of methods developed in the field of authorship attribution can be separated into three high-level categories: *unitary invariant*, *multivariate analysis* and *machine learning*, where the latter one is the most recent approach (Koppel, Schler, and Argamon, 2009). With unitary invariant, one seeks a decision boundary that is able to discriminate the writing of one author from the writing of another, such as word lengths in relation to their frequency of occurrence (Mendenhall, 1887). Multivariate analysis considers a more sophisticated range of style markers such as the distribution of function words (Mosteller and Wallace, 1964). These words (e.g. articles and pronouns) do not have much lexical meaning but provide important information about the grammatical structure of a text.

These first attempts in authorship attribution form the basis for many modern approaches that utilize the capabilities of powerful machine learning algorithms. The basic set of style markers (i.e. features) remains similar to ones historically used, but modern learning mechanisms such as Support Vector Machines (SVMs) are able to deal with large amounts of sparse data. This allows one to collect a wide range of different features.

On an abstract level, machine learning approaches utilize a set of style markers and a specific attribution method (i.e. the learning algorithm) to assign authorship to an unknown text. An overview of the main components in machine learning approaches can be found in the following Figure 4.1 (Stamatatos, 2009). Also, Appendix C.3 shows the overall process of machine learning based approaches. The next sections will further explain the different types of style markers available, as well as the general categories of attribution methods.

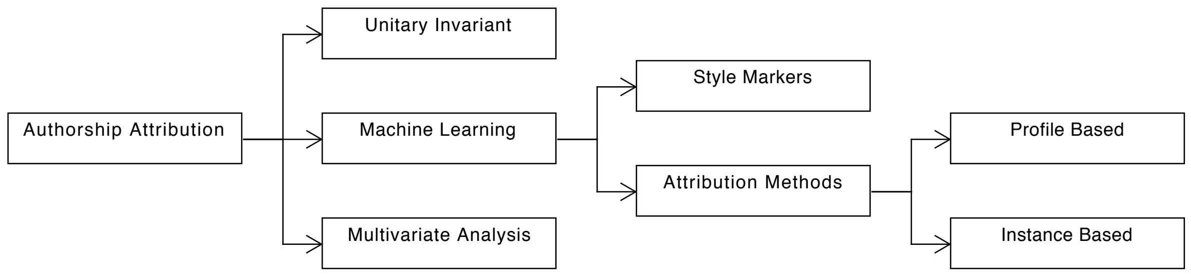

Figure 4.1: Overview of the different authorship attribution approaches.

4.2 Types of Style Markers

In general, every machine learning algorithm learns the mapping of a set of observations to a variable that is dependent on these observations. In the scenario of authorship attribution, these observations are referred to as style markers while the dependent variable is the label of an author. A number of different style markers has been proposed that help discriminate one author from another. These style markers can be roughly grouped into 5 categories (Stamatatos, 2009) and are described Table 4.1.

Category	**Description**
Lexical	These features consider each word, punctuation symbol and number as a separate unit. A number of different features can be derived, such as complexity measures (e.g. word/sentence-length, vocabulary diversity). One common representation of lexical features is the bag-of-words model which considers the text as a collection of words and their frequencies. In authorship attribution, function words are typically employed, since they are unconsciously used by the author and are expected to remain stable across different topics but not across different authors and genres (Argamon and Levitan, 2005).
Character	Unlike lexical features, this category considers each character as an individual unit (characters, digits, punctuation symbols). Therefore, measures such as the alphabetic character count or uppercase/lowercase ratio can be derived. Also character n-grams are frequently used. However, they might capture topic specific characteristics (because they are closely coupled to a specific word) rather than stylistic properties (Koppel, Schler, and Argamon, 2009).
Syntactic	Syntactic features consider the grammatical structure of the sentences. These features can be extracted by using Part-of-Speech (POS) taggers. Mostly, these POS tags are used as n-grams to capture complex grammatical structures.
Semantic	Semantic features also consider the meaning and dependencies of a specific word. These features are more complex and more difficult to extract.
Application specific	While all other features are independent of the particular text and genre, application specific features can provide additional information about the style of an author. These features could be the Hypertext Markup Language (HTML) formatting of a blog article, the salutations and greetings in a letter/email or the number and length of individual paragraphs in a book.

Table 4.1: Style markers in machine learning based authorship attribution.

4.3 Types of Attribution Methods

The style markers described in the previous section are means to capture the stylistic preferences of an author. However, they have to be extracted and utilized in a certain way. Literature has identified two general approaches to accomplish this: *instance based attribution* and *profile based attribution* (Stamatatos, 2009, pp. 13–18). The following describes their main mechanics and highlights key differences.

Instance Based This method considers each text of an author as a separate unit that can be represented as a vector of features. These vectors depict the input for the machine learning algorithm. Conversely, the algorithm sees multiple examples of an authors' writing to learn from it.

Profile Based This attribution method concatenates each text of an author into a single document (i.e. the profile). Features are then extracted based on this profile. The more similar the profile of a disputed document is to the profile of a certain author, the more likely it is that this author has written the document.

Both methods have certain practical issues that one needs to consider. For example, in an instance based scenario it might be difficult to handle an author for which only a single long text exists. In this case, it should be split into equal sized segments such that the features are comparable to those of other texts. Furthermore, the number of texts should not differ between authors so that a machine learning algorithm is not biased towards a specific author due to an imbalanced class distribution. Both of these situations are likely to occur in a real-world dataset.

While these two issues are of lesser concern in a profile based scenario, other difficulties arise. For instance, once all texts get combined into a single document, the stylistic deviations across individual texts are no longer perceivable. Furthermore, some features derived from a combination of all texts might be very different than the features obtained from a single document. It is beyond the scope of this project to examine these issues in detail, because they are a general theoretical issue in NLP related research.

4.4 Identified Approaches

The previous two sections have described the two main components in machine learning based authorship attribution: style markers and attribution method. A wide range of different techniques has been proposed for each of these components. Some of them are more popular than others and have also provided more stable results. With respect to style markers, Koppel, Schler, and Argamon (2009) have identified the most commonly used feature sets and compared them with each other: *function words*, *POS unigrams/bigrams* and *most common character trigrams*.

This feature set is further examined in this project, since it is expected to provide insights about different types of features (lexical, syntactic, semantic etc.) and how they can be extracted. Due to the special nature of the blog article dataset, the experiments also investigate how structural features such as formatting (e.g. color, bold font, font size etc.) can be used to distinguish between authors. Structural features have been applied to the attribution of web forum messages (Abbasi and Chen, 2005), and are in general able to provide reasonable hints about authorship (Juola, 2008, p. 266). In the scenario of blog articles, structure is typically controlled by the author when using a What You See Is What You Get (WYSIWYG) editor. A summary of style markers that are used in the experiments can be found in Appendix C.1.

With respect to the attribution method, several studies have achieved very good results by using SVMs that outperform other methods such as Neural Networks and Decision Trees (Abbasi and Chen, 2005; Zheng et al., 2006). Therefore, this report will mainly make use of the SVM algorithm. In order to gain some insights about the general implications that the choice of a classification method has, this project additionally experiments with the k-Nearest Neighbours (kNN) algorithm, the Multilayer Perceptron (MLP) algorithm and random forests. A brief explanation of these algorithms can be found in Appendix C.2.

As text pre-processing, feature extraction and model learning work differently for instance based and profile based methods, the above mentioned approaches will be used in both fashions. Generally, it is assumed that they will provide a good range of insights that can be utilized for further work on an authorship attribution pipeline.

Chapter 5

Assessment of Attribution Approaches

In order to make statements about performance and applicability of the solutions defined in the previous chapter, one has to assess them under a well-defined set of criteria and conditions. These criteria as well as how they can be measured were determined as part of research question RQ1 and are given in this chapter. Because the properties of a test corpus have a high influence on the performance of a solution, the main characteristics of the blog article dataset are described as well. This chapter is concluded by the practical insights which have been gained during the experiments.

5.1 Definition of Approach Assessment Criteria

Testing and evaluating the performance of a model is a standard task in machine learning. Accordingly, a large number of quantitative measures has been developed which can be used to assess the usefulness of a classifier. Because the selection of the appropriate measure is very dependent on the situation at hand, a stable list of criteria is difficult to establish upfront. However, there are certain standard measures which are often used as a starting point in the evaluation, which are *accuracy*, a *loss function* for probabilistic classifiers and *kappa score* (Witten, Frank, and Hall, 2011, pp. 147–177). Furthermore, this project can draw from criteria which are commonly used in authorship attribution scenarios. For example, the PAN community uses the *F1-score* in their annual authorship attribution competitions (Juola et al., 2011). Table 5.1 summarizes these criteria.

#	Criterion	Description
PC1	Accuracy	Fraction of right classifications over all classifications.
PC2	Quadratic Loss Function	Is used in a probabilistic prediction situation (i.e. 70% sure that author X has written article). Measures the cost associated to a missclassification.
PC3	Kappa	Difference in observed accuracy compared to expected accuracy (by chance). Standardized on a scale of $-1 \leq x \leq 1$ where 1 is perfect agreement, 0 is expected accuracy and < 0 is less than expected accuracy. More meaningful than accuracy alone.
PC4	F1 Score	Harmonic mean of precision and recall (value range $0 \leq x \leq 1$)
PC5	Accuracy Baseline	Accuracy achieved by always predicting the most common class in the dataset (i.e. dummy classifier).

Table 5.1: Quantitative assessment criteria for an authorship attribution approach.

The outcomes of these criteria are dependent on the dataset that is used to measure the performance of a solution. Therefore, it is important to determine how changes in the characteristics of this dataset influence the outcomes of the criteria. For instance, a model that only has to distinguish between two candidate authors is more likely to have a high accuracy than a model which has to distinguish between more than 100 candidate authors. Thus, the experiments examine how an approach performs when changing the number of candidate authors.

5.2 Characteristics of the codecentric Blog Dataset

As already discussed in the previous section, the performance of a solution is highly dependent on the dataset that is being used. For this reason, the following will summarize the characteristics of the codecentric blog article dataset which has been used as evaluation corpus in the experiments. The characteristics have been determined as part of an Exploratory Data Analysis (EDA) performed at the start of the project.

The corpus consists of 1300 articles, which have been written by 154 different authors between the year 2008 and 2016. A majority of these articles has been published in two languages (English and German). An author translates an article himself, but there is no information available whether the English or the German article is the original one. Also, there are slightly more English articles, because it is not required that a text has a translation.

As previously mentioned, the individual text length is an important property of the corpus, because it is more difficult to extract variations in style from a text which is 200 words long than from a text which is 2000 words long. The text length distribution of this corpus is highly positively skewed as can be seen in Figure 5.1. The same can be observed for the number of articles written by an author (see Appendix Figures C.2 and C.3). The top contributor has written over 150 articles and is thus overrepresented which may cause a model to be biased.

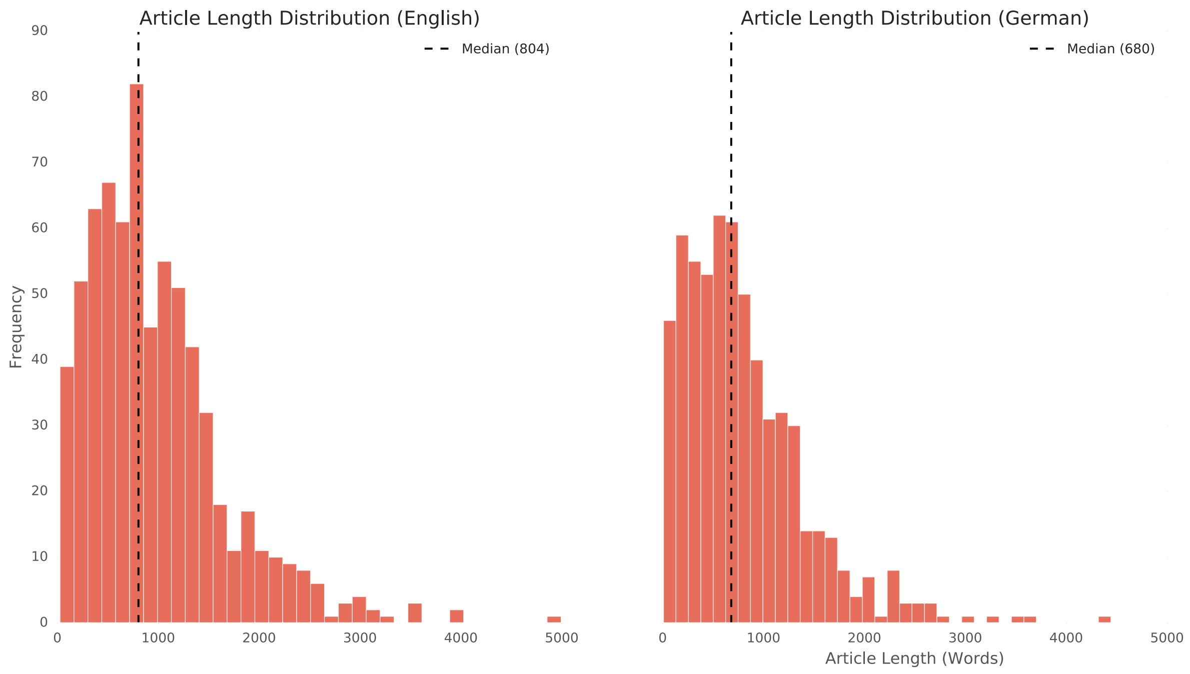

Figure 5.1: English and German article length distribution (in words).

With respect to the genre of the articles, it can be said that it remains consistent across all texts in the corpus, while the topics are highly diverse. When grouping the articles into 10 distinct categories, approximately 80% of the articles are covered (see Figure 5.2). In order to cover the remaining 20% of articles, one would need more than 30 additional topics to categorize the articles. A previous topic analysis by the data science department has also shown that the popularity of a certain topic changes over time (Radtke, 2017). In addition to that, the text length and the number of published articles per month has increased over the years (see Appendix Figures C.4 and C.5).

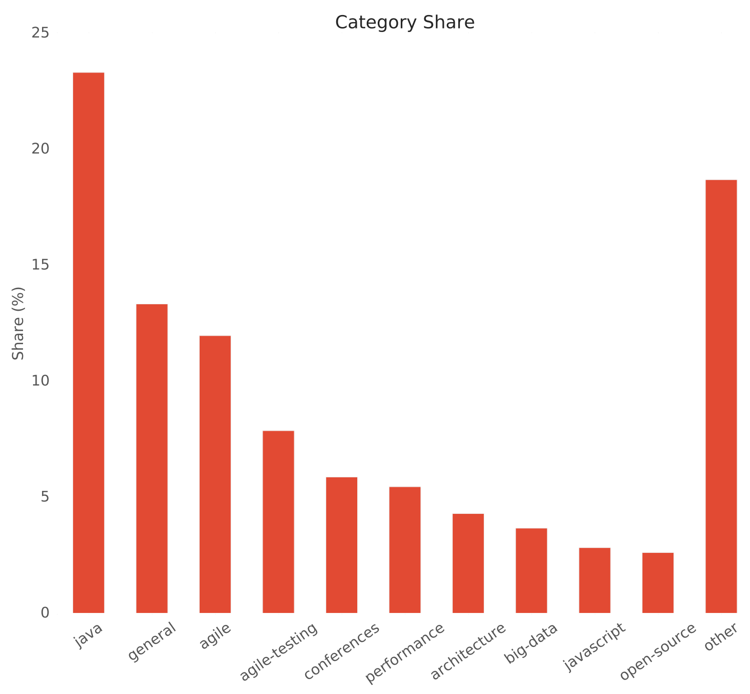

Figure 5.2: Share (in percent) of the top 10 article categories.

So far, the majority of authorship attribution studies have used very synthetic corpora in their evaluations. These corpora typically have a balanced class distribution, a large amount of available training texts and a small amount of candidate authors. These properties result in very optimistic results, since they are rarely present in practice (Luyckx and Daelemans, 2008). Because this project deals with a real-world corpus, it is of particular interest how the methods defined earlier perform on it.

5.3 Criteria Operationalization

In order to compare the authorship attribution approaches and to operationalize the criteria given in this chapter, an evaluation method has to be selected. This method describes the way, in which the available data is used to train and test the machine learning model to achieve valid and reliable performance measures. The following evaluation techniques are widely used in practice (Witten, Frank, and Hall, 2011):

1. Derive error rate from the same data that has been used to train the model (resubstitution error)
2. Hold-out (use a random subsample of data for testing purposes only)
3. Cross Validation (CV), k-fold CV, stratified k-fold CV
4. Leave-one-out CV
5. Bootstrap

Because the 10-fold-stratified CV is the recommended method in most model evaluation scenarios (Kohavi, 1995), this project will mainly make use of it. However, since there is no evaluation method which is superior in all scenarios, the specific nature of the dataset and the machine learning model has to be considered when making the experiments in RQ3. The functionality to perform a CV as well measuring the criteria defined in Section 5.1, is readily available in the Python library scikit-learn (scikit-learn developers, 2016).

5.4 Assessment Process

The experiments evaluate multiple authorship attribution methods under the criteria which have been described in Section 5.1. In addition to that, the individual style markers are considered in isolation to determine which of them are more discriminative than others and therefore depict a good default configuration for the authorship attribution pipeline. Each experiment consists of the following four steps:

1. Sample the data
2. Extract feature sets
3. Train the attribution method
4. Test the quality of the attribution method

The sampling step draws 8 different samples from the codecentric blog article dataset in order to test an attribution method under varying dataset characteristics. Each sample resembles a subset of the original dataset. There are two instance-based samples and two profile-based samples for the English language. Per text representation, one sample excludes authors which have written more than 20 articles (i.e. outliers), whereas the other sample includes them. The same sampling is applied to articles written in the German language. A detailed overview of the sample characteristics can be found in Appendix Table C.2.

After sampling the data, 4 feature sets are extracted and assessed individually per sample. These feature sets are combined to determine which combinations of style markers provide good classification results. Because it is unfeasible to assess all possible combinations of the feature types, a base-feature set consisting of POS-tags and function words is used. Research has found that this combination is a reasonable choice (Koppel, Schler, and Argamon, 2009, pp. 15).

5.5 Assessment Results

The experiments confirm the general trend, that function words and POS-tags depict a sensible default feature set. Appendix Table C.3 shows the classification accuracy per sample and feature set. These results have been achieved using an SVM classifier with a linear kernel. It has been shown that a linear kernel is preferable over a non-linear kernel when the number of features is much larger than the number of instances (Hsu, Chang, Lin, et al., 2016). As this situation is given in an NLP problem, the linear kernel has been favored. The following summarizes some of the major insights of the experiments.

Feature Sets The results show that a combination of feature sets almost always results in a minor accuracy boost. The combination of all feature types is among the best performing ones. However, this is also the expected behavior since more stylistic variations are captured with an increasing feature space.

Candidate Set It can be observed that a larger set of candidate authors results in a lower classification accuracy. For instance, while a set of 15 candidate authors achieves an accuracy of up to 87% (sample 5 in Appendix Table C.2), a much larger dataset with 85 candidate authors has a maximum accuracy of 45% (sample 7 in Appendix Table C.2).

Feature Limit Feature types that generate a large number of dimensions such as POS-trigrams have been limited to the 50 most frequent features. This limitation offered a good trade-off between computational efficiency and accuracy with the used dataset. It is worthwhile to explore different settings for this limit when another dataset is used during hyperparameter tuning which will be explained in Section 5.7.3.

Languages It can be observed that the German samples achieve slightly lower performance scores than English samples (see sample 1/2 and sample 3/4 in Appendix Table C.2). This may relate to the fact that the German set of function words is considerably smaller than the English set of function words. Nevertheless, it is still the best performing feature set when used in isolation.

Class Imbalance Since the blog article corpus is a real-world dataset, a number of unfavorable properties exist. One of them is that certain authors are overrepresented which is known as class imbalance. However, it has been found that accuracy does not significantly drop when overrepresented authors are added to a sample (see sample 1/2 in Appendix Table C.2). It is assumed, that this behavior relates to the way how an SVM optimizes the decision boundary. However, it has been beyond the scope of this project to test this assumption.

5.6 Issues with Assessment Process

Two critical issues have been encountered during the assessment of multiple authorship attribution approaches. These difficulties are shortly explained in this section. First of all, the criteria determined as part of RQ1 have only been partly useful during the experiments. The *accuracy*, and *baseline accuracy* are the two major metrics that give enough information when comparing a large number of feature sets and attribution approaches. Classification statistics such as the *f1 score*, *precision* and *recall* do not help when comparing multiple classifiers because they are very application specific. However, they may be used to further optimize a single attribution method after selecting it based on the accuracy. For instance, an authorship attribution classifier that is employed in a legal investigation may be optimized for the recall behavior to minimize the false negative rate (i.e. avoiding to miss a piece of evidence).

The second issue relates to the evaluation method that has been used to test the attribution method. A majority of the authors in the codecentric blog dataset have written less than 5 articles. This complicates evaluation, because methods such as 10-fold cross validation are not applicable. For this reason, the instance based samples have been limited to authors which have written more than 10 articles. Alternative evaluation methods such as the leave-one-out cross validation or the bootstrap may be more appropriate because they maximize the amount of available training data (Witten, Frank, and Hall, 2011, pp. 154–156). Due to the time constraints of this project, it has not been possible to further investigate these issues.

5.7 Practical Considerations for Pipeline Development

One of the main goals of the experiments has been to determine how authorship attribution can be performed in practice. It has been shown that it is possible to achieve satisfying results with rather simple methods using the codecentric dataset. However, a number of machine learning techniques has been discovered that are necessary to further improve these results. Because these techniques depict functionality that is important for a pipeline prototype, they are shortly explained in this section.

5.7.1 Feature Normalization

A common issue with bag-of-words feature types is that term frequencies are higher for longer documents. This negatively affects the performance of a classifier because it may be biased towards longer documents. Therefore, it is important to normalize the feature vectors to eliminate the information about the original document length. Normalizing the feature vector with the euclidean unit length has been found to be an effective normalization for authorship attribution with the SVMs classifier (Diederich et al., 2003).

5.7.2 Feature Selection

The goal of feature selection is to remove features that are redundant and uninformative in order to improve the classification performance. This practice is of particular interest for NLP problems, because they typically deal with large feature spaces. Figure 5.3 shows the impact of a chi^2 feature selection with respect to the performance of various classifiers. Over 17000 features have been extracted from sample 1 (see Appendix Table C.2). The feature selection algorithm selects the K-best features (x-axis) which are then in turn used to train and test the classifier for its accuracy (y-axis). It can be observed that the classification accuracy remains rather stable for the SVM classifier, because it is designed to handle large dimensions. Thus, it does not require feature selection. However, other classifiers such as kNN are negatively affected by a large feature space which relates to the curse of dimensionality phenomenon. A closely related and very popular practice in NLP are term weighting schemes such as *tf-idf* which assign lower weights to less important features (Witten, Frank, and Hall, 2011).

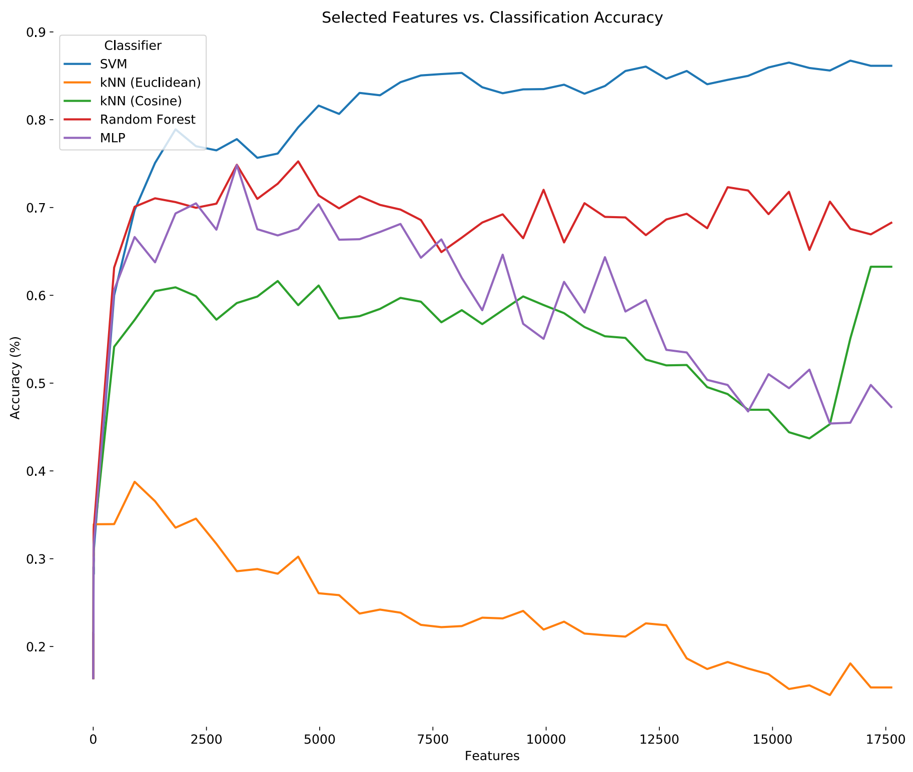

Figure 5.3: Impact of chi^2 feature selection on the classification performance of various classifiers.

5.7.3 Hyperparameter Tuning

A number of parameters can be adjusted during the feature extraction and classification process. For instance, an upper bound for most common POS-tags can be defined. Other parameters may be the size of the feature subset during feature selection, or the regularization parameter of an SVM classifier. These settings are usually referred to as hyperparameters. A crucial issue involves the search for optimal settings of these parameters in order to maximize the classification accuracy. It is a common practice to perform an exhaustive search for these parameters. An authorship attribution pipeline can automatically execute this action to achieve optimal results.

Chapter 6

Pipeline Requirements Specification

This chapter describes the general characteristics of an authorship attribution pipeline, the user groups it is targeted to and sets it into context with existing authorship attribution applications. Also, development constraints are stated under which the pipeline prototype has been implemented. Furthermore, the requirements elicitation process is described, which has been used to determine the functional and non-functional requirements of the pipeline.

6.1 Product Context

As previously discussed, there are many use cases for authorship attribution. Some examples are fraud detection or the provision of evidence and tips in a legal investigation. Similar techniques and processes are used regardless of the actual use case. Therefore, there is a potential to provide a reusable piece of software that can be applied to many authorship attribution problems. It has been agreed in collaboration with the customer to develop a pipeline library. This library shall bundle the techniques and approaches examined earlier in this report. The library can be reused to achieve a larger business goal using authorship attribution techniques. Besides resulting in a reusable tool for authorship attribution, the implementation will provide knowledge about the encapsulation of a data science pipeline into a Python library. It will be emphasized how this library can be designed to be reusable and extensible. This knowledge can be utilized in future projects of the data science department.

6.2 User Groups

The research phase of this project has identified a large number of different strategies to perform authorship attribution which have been described in Chapter 4. Some of these strategies are independent of the actual application and therefore provide a good starting point in an authorship attribution problem. For example, the experiments have shown that instance based attribution using function words as the feature set and an SVM as the classification model, depicts a reasonably good baseline (see Section 5.5). This circumstance makes an authorship attribution library approachable for two main user groups: *machine learning experts* and *non-machine learning experts*. In the context of the codecentric AG, both user groups are present and are potentially interested in the pipeline. An example for the first user group would be the data science department. The latter group might be the document solutions department, whose daily business involves processing large number of textual artifacts. The characteristics of the user groups will be briefly introduced in the following paragraphs.

The first user group, machine learning experts, are interested in finding an optimal model for the problem at hand. They are capable of engineering application specific features such as usage patterns in an authors' HTML formatting of a blog article. They also want to experiment with different classification models, tune their hyperparameters and compare their attribution quality. In contrast, the main goal of non-machine learning experts is to integrate authorship attribution in a broader application context. They are unfamiliar with the theories of NLP and machine learning. Therefore, they can benefit from a pipeline that uses a default configuration which is expected to give reasonably good results across many use cases.

Both user groups can benefit from a reusable piece of software that assists along all phases involved in the authorship attribution process. This yields the conclusion that the pipeline library should offer two main modes of operation to support the needs of both user groups:

Default Configuration	A sensible default configuration shall be provided, that relieves the user from studying more sophisticated issues in authorship attribution and enables him to use the pipeline right-away.
Custom Configuration	It should still be possible for an experienced user to adjust the pipeline such that it fits the needs of the use case at hand. The aim of the user might be to improve the performance by using a custom set of features or an ensemble of several learning algorithms.

The two modes of operation have been considered equally during the design and implementation of the library.

6.3 Overall Constraints

There were two main constraints with respect to the software development that had to be considered. The first constraint relates to the type of software. As previously indicated, the software should not be a standalone application. Instead, it shall be a library that is designed to be reusable. Any functionality that is needed to perform authorship attribution is considered to be in scope of the library as long as it is not specific to a certain use case (i.e. the broader context). Thus, a transforming routine which extracts grammatical features is considered in scope of the library since it is expected to be reused in multiple contexts. On the contrary, a cleaning routine which parses articles for specific formatting elements that a content management system of a blog inserts, is regarded to be out of scope of the library.

The second constraint has been given by the customer and defines the programming language to be Python. Python is a general-purpose object oriented programming language which is used in many domains including web development and science. The language constraint relates to the fact that the data science department uses Python in a majority of their projects. As one of the major goals of this work is to provide knowledge in NLP and machine learning that can be utilized in future projects, it is reasonable to implement the library within the environment the department is using in their daily business. Although the programming language has been constrained, there were no limitations with respect to the third party libraries that can be used. Therefore, the development of the software also consisted of an assessment of the available libraries to find suitable candidates for tasks relating to NLP. This assessment can be found in Section 7.3.

6.4 Product Perspective

A few applications have already been developed that perform authorship attribution. This section briefly introduces the most popular examples and explains how the library of this project relates to them.

The first application to mention is the Java Graphical Authorship Attribution Platform (JGAAP). It is an open source project developed by a research team at Duquesne university. Its primary goal is to allow non-machine learning experts to perform authorship attribution with modern algorithms and techniques using a Graphical User Interface (GUI) (Evaluating Variations in Language Lab, 2017). Furthermore, it serves as a framework in research to execute large scale experiments that assess the effectiveness of new authorship attribution practices (Juola, 2009). JGAAP has been used to determine that J.K. Rowling is the author of "The Cuckoo's Calling" (Juola, 2013). A closely related software in the field of authorship attribution is the open source Java application JStylo (McDonald et al., 2012). It is part of a larger framework called JSAN that aims to anonymize the writing style of an author. Its implementation is based on JGAAP and augments the functionality with transformations needed for writing style anonymisation. Both JStylo and JGAAP leverage the capabilities of the Java machine learning framework WEKA.

There are two further tools which are related but have a slightly different scope. Stylene is a web-application that analyzes stylometric features and the readability of Dutch texts only (Daelemans and Hoste, 2013). Signature is another authorship attribution software for textual analysis, which is closed source and only available on Windows (Millican, 2003).

In conclusion, it can be said that there is currently no software package which aims to integrate authorship attribution into a broader application context. Therefore, the main goal of the authorship attribution library of this project is to provide this software package. It differs from existing applications such as JGAAP and JStylo when it comes to the diversity of stylometric functionality. They already provide a good platform when it comes to running large scale experiments. Instead, the major aim of the library is to provide an authorship attribution Application Programming Interface (API) that is easy to incorporate in an existing application.

6.5 Requirements Elicitation Process

This section shortly explains how the functional and non-functional requirements have been elicited. As a result of the nature of this project, it was not possible to gather the requirements from the customer in a traditional way. This is because the customer has been mainly interested in learning about NLP and machine learning. He has been less concerned with the details involved in the use case of authorship attribution since it was only the main driver to work towards his goal of acquiring knowledge. Therefore, an alternative way of determining the authorship attribution specific requirements on a functional and non-functional level had to be found.

The activities that were part of the research phase of this project provided the main means to elicit the requirements. The literature research contributed to the theoretical understanding of authorship attribution by identifying a number of possible approaches. These approaches have been assessed as part of the experiments and thus formed the practical context. In order to do this assessment, it was necessary to implement small prototypes of each of the individual approaches which provided enough functionality to asses their attribution quality. These prototypes revealed the technical details that are involved when processing text, extracting features and training an authorship attribution model. The technical details translate into the functional and non-functional requirements described in Section 6.7 and 6.8 respectively.

6.6 Functional Requirements Format

The functional requirements were documented in terms of user stories which are short and *non-technical* descriptions of the functionality that a user expects in a system. There were multiple reasons, why user stories have been employed instead of the traditional functional requirements found in a Software Requirements Specification (SRS) which are further described in this section. A summary of the user stories can be found in Section 6.7.

First of all, this project did not require a formal contract depicted by a requirements specification. It has been agreed in consultation with the customer, that a high level description of the functionality is sufficient. Secondly, defining functional requirements on a technical detail has been shown to be error-prone and also time consuming and thus would have hindered the development of the software (Cohn, 2004, p. 133). Lastly, user stories are a convenient way to specify what a user of the pipeline library expects. Otherwise, a large number of functionalities could have been specified, which are not directly tied to an actual user need.

Each user story is composed of an *identifiable code*, a *title*, a *description* in the form specified by Cohn (2008), an *acceptance test*, and an *importance*. The title and the identifier are the main means for traceability of each story. The acceptance test defines the high level steps needed to verify that the user story has been successfully implemented. The importance is measured numerically, where a higher number indicates a higher priority. This way of quantifying the importance follows the scheme proposed by Kniberg (2015, pp. 9–10).

The user stories, consisting of all the mentioned fields, were maintained in a product backlog. It was necessary to find a low-overhead issue tracking software that is suited for a small-scale, single developer project. The built-in issue tracker of the GitHub repository that has been used to version the source code of the library, was very well suited for this task. The tracker has been used to manage and share the product backlog with the involved stakeholders. It offered capabilities to communicate and document the current implementation status of an issue. Since it did not require any additional software installation, configuration or licensing, it has been favored over feature-rich alternatives such as Atlassian JIRA.

Unlike in a typical project setting, the user stories lack an effort estimation. This is based on the fact that this project has been executed individually and thus did not require coordination between multiple developers or the customer. Furthermore, it is important to mention that this project only developed a prototype of an authorship attribution library. This prototype serves the purpose of gaining knowledge in the domain of NLP and machine learning. If there was a request to develop a piece of software needed in a production context in a timely manner, effort estimation would have been an essential tool to negotiate the amount of functionality and to manage the expectations of stakeholders. Furthermore, in this single developer project, an effort estimation would have been highly biased since it is based on the opinion and experience of a single person. In a typical project setting, one would attempt to find a more realistic effort estimation by applying techniques such as planning poker or the Delphi method in collaboration with multiple project members.

6.7 Functional Requirements

This section summarizes the user stories which represent the functional requirements specification of the authorship attribution library. A full specification of the user stories can be found in Appendix D.1. The user stories can be grouped into the five high level steps that are present in any typical authorship attribution process: data loading, text preprocessing, feature extraction, model learning and finally authorship attribution. Each of these steps will be briefly explained in the following.

During the first phase, the data loading phase, a user wants to supply his own text corpora (see US-1). The format in which these corpora are available might differ from user to user since they are specific to the broader application context. For instance, the codecentric blog article corpus has been available as a MySQL database dump. Another corpus might be available as a Comma Separated Value (CSV) file, or as a directory hierarchy, mapping an author to his written texts. In order for a pipeline to work, this data needs to be loaded and transformed into a common format that the pipeline understands. The second phase prepares the text such that it is suitable for feature extraction. Depending on the type of text, different transformations may be necessary that a user of the library wants to specify (see US-2). For instance, one text corpus may only require the pipeline to transform the text into a uniform capitalization, while another requires the removal of HTML tags or the like.

Once the data has been loaded and any unwanted content has been removed, textual features can be extracted. Given that a user of the library knows the theory behind authorship attribution, he wants to specify which features should be extracted from the text (see US-3). In case the user is unfamiliar with the details of authorship attribution, the pipeline should use a default set of features which is expected to provide reasonably good results (see US-4). The features originating from the preceding phase are then utilized in the final two phases to train a machine learning model. This model is being used to predict the author of an unknown text. A experienced user typically wants to specify the machine learning algorithm that is being used. Finally, he likes to inspect the quality of the trained attribution model (see US-5 throughout US-8).

Lastly, it is important to mention that the priority of these user stories varies. For instance, having an indication about the training progress of the machine learning model is from an end-user perspective less important than attributing authorship to an unknown text. Therefore, user stories have been ranked in terms of importance, such that the results of the experiments can be reproduced as soon as possible. The remaining user stories have been implemented incrementally in sequence of their priority.

6.8 Non-functional Requirements

An inherent problem with user stories is their limited capability to constrain non-functional characteristics of a software system that are unobservable during execution time. Consider a property such as maintainability. An end-user of software is neither interested nor able to observe the maintainability of the underlying software architecture which makes it difficult to convey this information in a user story. Therefore, this section describes several non-functional dimensions under which the authorship attribution library can be assessed. These dimensions have become apparent during the execution of the experiments and can be elaborated on at development time and during the time the library is integrated in an application context.

Extensibility Since NLP research constantly develops new techniques that help performing authorship attribution, it should be possible to extend the pipeline with these techniques without affecting the existing ones. This requires considerations at design and development time that result in a modular and extensible architecture.

Performance Depending on the application context, certain performance criteria might exist. Some examples are the performance during model learning time, or the performance during prediction time. This performance criteria can be quantified in throughput of articles per time unit (e.g. 100 articles/second).

Quality The required attribution quality heavily depends on the application context in which the pipeline is used. For example, in a forensic context it may be of high importance to minimize the false negative rate of the attribution. Other measurement instruments of this non-functional requirement are the standard machine learning metrics (see Section 5.1). As with the performance criteria, this non-functional requirement needs to be defined per-application.

Chapter 7

Pipeline Library Design

This chapter describes the major design decisions which have been made during the implementation of the pipeline library. It illustrates the component based architecture of the library and explains how the steps of the pipeline have been modeled in an object oriented way that promotes interchangeability. Furthermore, design alternatives are discussed and the selection of a suitable NLP library is described.

7.1 Components of Library

When developing an object oriented software system, a crucial decision involves the logical organization of related functionality by bundling it into multiple independent units. The component diagram in Figure 7.1 illustrates the main organizational units of the library. Each of these components, except for the `Pipeline`, encapsulate the functionality required in the corresponding phase of an authorship attribution problem. Every component exposes its functional capabilities through an API which can be consumed by other components as denoted in the diagram. The Python language provides built-in support for a component based architecture through the concept of *modules*.

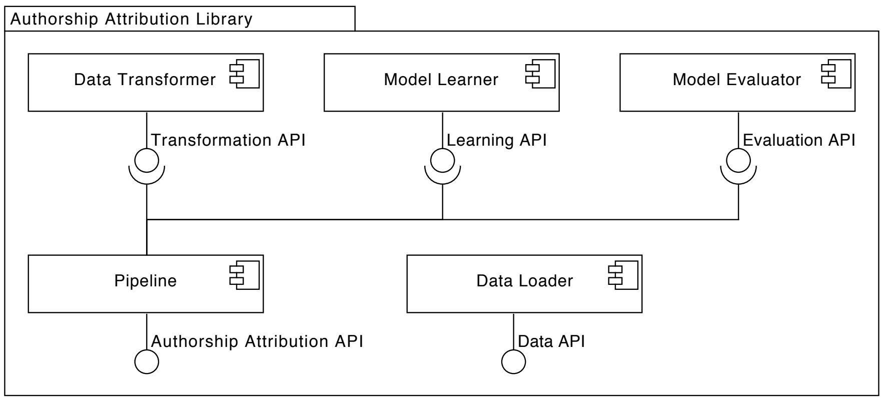

Figure 7.1: Component diagram of authorship attribution library.

The reason why the functionality has been divided into this set of components is to promote high cohesion. Each component provides only the functionality that is needed during the specific phase of an authorship attribution problem as described in Section 6.7. This modularity allows for the independent maintenance and modification of components. For instance, the `Data Transformer` component could be replaced by an alternative implementation without affecting the other components as long as the same interface is provided.

As previously mentioned, the `Pipeline` component does not directly correspond to a phase in an authorship attribution problem. This is due to the reason that it depicts an entry-point into the library. It acts as a facade that simplifies the overall usage of the other sub-systems present in the library (Gamma, Helm, et al., 1995, pp. 185–194). Thus, a client has minimal effort when executing the whole life-cycle of an authorship attribution problem. It has also been decided that none of the other components should depend on the `Data Loader` component. This relates to the fact that data loading is a very application specific task. In some contexts (e.g. a web service), loading the data from an external source may not be required because it is already in the state of the application. Therefore, a client of the library uses at a minimum only the facade. Optionally, it may also use any of the other components if more customizability is required.

Dividing the library into a number of components naturally defines the boundaries for individual testing strategies. Because the facade bundles all functionality that a user of the library expects, automated user acceptance tests have been written against its API during the implementation of the library. Integration and unit tests have been written on a component and class level respectively.

7.2 Processing Steps Architecture

Each step of the pipeline has been encapsulated into a class that has a very narrow responsibility scope. The main motivation behind this encapsulation is that the steps shall be as lightweight as possible in order to promote their interchangeability. The individual steps can be combined in a sequential manner in which each step consumes the results of the preceding step. Therefore, a class architecture needs to define a common interface for each step to allow them to be uniformly addressed. Furthermore, it needs to define a container (i.e. `Pipeline`) that sequentially interlinks them. The class diagram in Figure 7.2 visualizes this architecture.

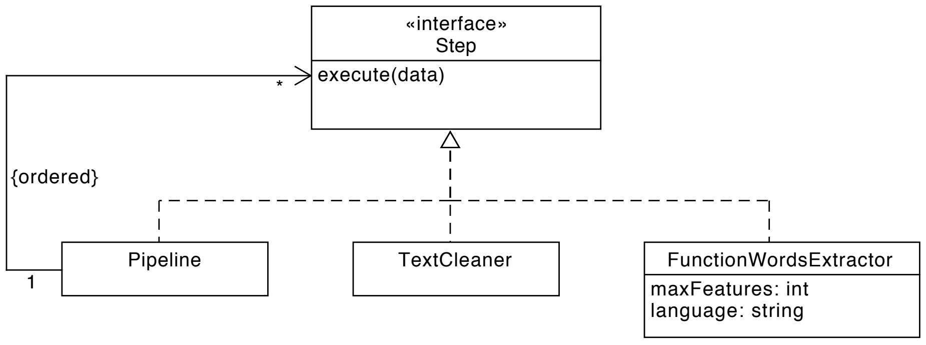

Figure 7.2: Pipeline step composition using the composite pattern.

The sequence diagram in Figure 7.3 demonstrates a very simplistic pipeline that cleans the text from formatting characters, extracts function word frequencies as features and trains a classifier. Although this pipeline already depicts a complete workflow that can be applied to an authorship attribution problem, it is unlikely to provide satisfying results due to the simplicity of the extracted feature set. However, in order to extract multiple feature sets, sequential chaining is no longer suitable. Each feature extraction step requires the same raw text input which is no longer available after the first transformation. This architectural problem is solved by the composite design pattern which allows for recursive class composition (Gamma, Helm, et al., 1995, pp. 163–174). A specialization of the `Pipeline` class is needed (referred to as `JoinStep`), that passes the same input data to all nested steps. After all of these steps terminate, the results can be merged. The composite pattern hides away the details of a specific step. It is unknown to the caller whether the step is a pipeline of sequential steps, or a single step because they share the same interface. The sequence diagram in Appendix D.1 shows how this architecture can be used to extract multiple feature sets.

The composite pattern has been applied for four main reasons. First of all, it is a proven pattern which has been used to solve similar problems in many popular frameworks such as JUnit (Gamma and Beck, 1999). This pattern allows the library to be extendable, which has been a major non-functional requirement of the

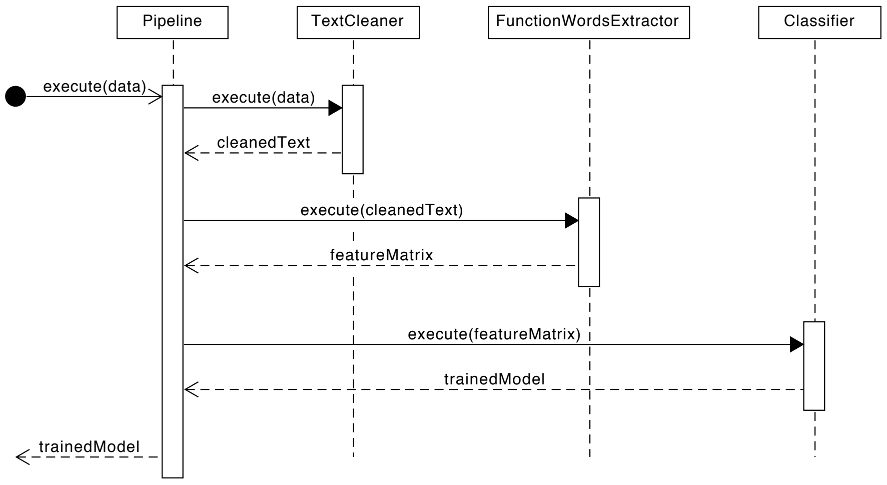

Figure 7.3: Control flow in a flat sequential pipeline.

library. Secondly, it ensures that the individual steps are as lightweight as possible, and helps to isolate them for automated tests. Also, other implementation alternatives including the “chain of responsibility” and the “pipes and filters” pattern have been found to be less flexible when nesting individual steps. This is because each step maintains a reference to the consecutive step which introduces additional dependencies and complicates the instantiation (Gamma, Helm, et al., 1995; Hohpe and Woolf, 2003). Lastly, the machine learning library scikit-learn uses the composite pattern internally to model transformations to feature vectors and classifiers (Buitinck et al., 2013). Therefore, this existing infrastructure can be reused which simplifies the implementation of the library. This decision has the potential disadvantage that a user of the library is coupled to scikit-learn. However, since the implementation is only a prototype, this disadvantage is of lower importance. If the library must be decoupled from any specific machine learning framework in future, the composite infrastructure would have to be recreated.

7.3 Assessment of Natural Language Processing Libraries

The Natural Language Toolkit (NLTK) has been used during the experiments to perform tokenization and POS-tagging. It has been found that a large fraction of the processing time is spent with these NLP tasks rather than with feature extraction and classification. Furthermore, NLTK has no built-in support for the German language. These factors have motivated a search and an assessment of alternative NLP frameworks. Appendix D.1 gives an overview of popular Python NLP libraries. NLTK and spaCy have been assessed along three dimensions: functional capabilities, multi language support and computational efficiency. The following shortly summarizes the main results of the assessment.

On the one hand, NLTK is superior to spaCy in terms of functional capabilities (see Appendix Table D.2). Since NLTK is commonly used in research, it supports various different algorithms for stemming and lemmatization and has fine-grained customization options, whereas spaCy focuses on the efficient implementation of a single algorithm per NLP task. On the other hand, spaCy has built-in support for the English, German and French languages (at the time of writing, 12 other language models are under development). Furthermore, it is superior when it comes to computational efficiency. During POS-tagging tasks, spaCy performs almost an order of magnitude better than NLTK (see Appendix Figure D.2).

Based on these assessment results, it has been decided to implement the initial version of the library using spaCy. The functional capabilities are sufficient to perform most authorship attribution techniques. Also, the lack of multi language support of NLTK has been a critical factor in this decision. When using NLTK, one would need to find a pre-tagged corpus per language that the library should support. In addition to that, classification models have to be trained, serialized and distributed in a binary form. These models are a heavy dependency for a library and require a large amount of time to maintain which has been out of scope of this project. Lastly, since spaCy is a rather new library, this decision also contributed to the overall goal of the customer to gain experiences with new technologies.

7.4 Interchangeable Natural Language Processing Back-End

Although the initial version of the pipeline library has been implemented using spaCy, the composite architecture allows for making the NLP framework interchangeable. This enables experienced users to parse documents with specific NLP algorithms that are not available in spaCy. In order to make the framework interchangeable, it is necessary to integrate the parser classes into the composite hierarchy and to define an abstract type that resembles a parsed document. These documents are created by each parser (e.g. spaCy, NLTK, etc.) and must be provided to the succeeding feature extraction steps of the pipeline. Figure 7.4 visualizes the architecture.

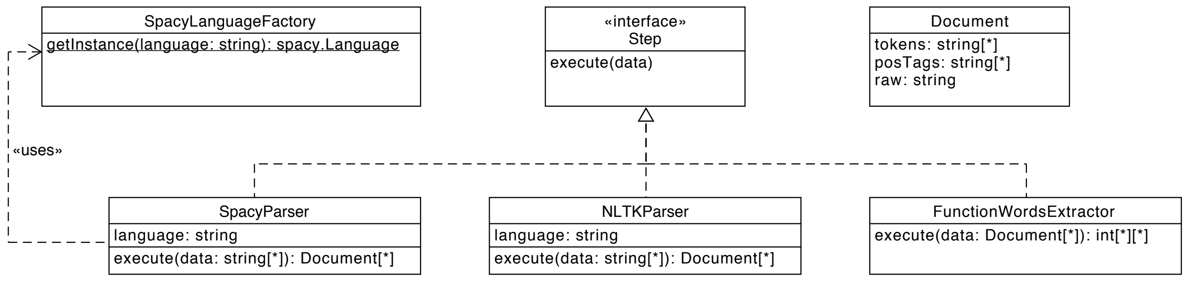

Figure 7.4: Introducing a document transfer object to allow interchangeability of the NLP framework.

A document in itself is a container that stores the individual results of each parser (e.g. `tokens` and `posTags`). Since the feature extraction steps are the successors of the parsing step, their execute method takes a collection of `Documents` rather than raw text. This architecture has a number of advantages from a functional point of view. First of all, the interchangeability of the NLP framework allows experienced users to utilize the capabilities of other libraries such as NLTK. Secondly, integrating document parsers into the composite hierarchy makes them usable side-by-side in a single pipeline. Furthermore, the document transfer object acts as a cache that stores the parsing results. Multiple feature extraction steps use these cached results. Lastly, it promotes maintainability of the library. In case an NLP library is no longer supported, it can be easily removed from the pipeline without major changes in the implementation.

A difficulty with respect to memory management has been discovered during the implementation of the `SpacyParser`. Every time the internal parser of spaCy is constructed, pre-trained machine learning models are loaded into the memory of the application (model size ranges from 500MB to 1GB). In order to avoid memory issues during execution time of the library, the construction of the spaCy internal parser has been encapsulated into the `SpacyLanguageFactory`. It maintains a collection of singletons where each instance depicts one natural language and only exists once during application runtime (Gamma, Helm, et al., 1995, p. 127). However, it has been found that this strategy is not suitable for multiprocessing, because the individual processes do not share the same memory space. Thus, depending on the degree of parallelization, memory issues can occur. Due to the time constrains of this project, it was not possible to further investigate this issue.

7.5 Implementation Details and Testing of Prototype

The following provides several technical details that relate to the implementation of the authorship attribution library. First of all, the user stories defined in Appendix D.1 have been successfully realized during the library implementation phase. The encapsulation of the pipeline steps allowed for implementing multiple classes that extract the same feature sets that were used in the experiments of this project. As new techniques in NLP and authorship attribution come available, additional feature types can be added to the library by providing an implementation of the `Step` interface.

User story US-8 specifies the need for a progress indication during lengthy tasks (e.g. text processing and model training). In order to provide a status indication that can be controlled for verbosity by the user of the library, it is necessary to make use of a logging framework instead of rudimentary `print` statements. The Python standard library includes a built-in logging system which has been used for this purpose. Because this mechanism is used by a majority of Python applications, the logging of the authorship attribution library integrates with the existing logging architecture of an application that aims to integrate the library.

In order to manage dependencies to third party modules and to bundle the authorship attribution library, the Python packaging software *setuptools* has been used. Applications which are defined according to the *setup-tools* standard can be installed with the the Python package manager *pip*. For build and test automation, the *make* utility has been used. Although unit tests, integration tests and acceptance tests have been successfully executed in an isolated Docker environment, further test isolation and automation may be needed as soon as the library undergoes further development. A continuous integration platform such as Jenkins can be used for this purpose. Because the development of the library has been done individually and the scope of this project defined it to be a prototype, establishing this infrastructure was not required.

Lastly, the library has been used to perform authorship attribution on an independent corpus to examine how capable it is to handle different datasets other than the codecentric blog article corpus. The corpus assembled by Brennan, Afroz, and Greenstadt (2012) has been used for this purpose. The dataset consists of 700 scholarly essays written by 45 authors. It has been possible to construct a pipeline with the help of the library that obtains comparable results to those achieved by the creators of the corpus.

Chapter 8

Conclusions

This report presented the final results of the bachelor thesis project which has been concerned with the conceptualization of an authorship attribution pipeline. The data science department of codecentric formulated this assignment in order to get an overview of state-of-the-art techniques used in NLP, to further extend their product portfolio and to broaden their knowledge.

These goals have been successfully achieved by providing both the required theoretical background and practical insights relevant to authorship attribution. The theoretical part consisted of the systematic identification of available approaches, that attribute authorship to a piece of text. Furthermore, guidelines that assess the quality of these approaches have been proposed. Practical issues relating to authorship attribution, and NLP in general, have been revealed during experiments that used the codecentric blog article dataset. A reusable pipeline library has been developed that demonstrates how a data science pipeline can be encapsulated and designed to be reusable and extensible. During this process, many insights about modern technologies in the Python ecosystem have been acquired.

It has to be mentioned that the theoretical results have been derived by using a limited amount of literature due to the restrictions in time and scope. One could achieve an even more comprehensive review of the topic by incorporating more literature. However, this task is very time consuming and would not have been in the interest of the involved stakeholders.

The execution of the project plan and the project management approach worked out as expected. Nevertheless, there were two issues that leave room for improvement and need to be considered in future projects. During the initial analysis of the problem domain of authorship attribution, the scope definition has been too broad. A fair amount of time has been spent studying relevant, but overly complex techniques and issues of authorship attribution. This problem was discovered in good time to narrow down the scope in consultation with the involved stakeholders. It has been an important lesson, that these scope limitations are crucial to ensure that the resulting conclusions are valid and at the same time fulfill the expectations of the customer.

Also, another important lesson relates to the communication with a customer at distance. Although the communication facilities at codecentric were excellent, they cannot replace interpersonal interactions. For example, practices such as a brain storming or pair programming are difficult to perform, when two people are not in the same room. The only ways to overcome this issue, is to increase the frequency of communication and arranging occasional meetings at the same location.

In the future, this project can be extended in a number of ways. First of all, the pipeline library can be made production-ready, by implementing additional feature extraction methods or an anomaly detection module. Also, the pipeline can be optimized for multiprocessing to achieve high computational efficiency. Furthermore, an integration of the library into the document processing software of the document solutions department is conceivable. Lastly, an investigation of deep learning is recommended, which has promising applications in the field of natural language processing.

References

Abbasi, A. and Chen, H. (2005). Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20(5), pp. 67–75.

Argamon, S. and Levitan, S. (2005). Measuring the usefulness of function words for authorship attribution. *Proceedings of the 2005 ACH/ALLC Conference*.

Brennan, M., Afroz, S., and Greenstadt, R. (2012). Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security (TISSEC)*, 15(3), p. 12.

Buitinck, L. et al. (2013). API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Cohn, M. (2004). *User stories applied: For agile software development*. Boston: Addison-Wesley Professional, pp. 4, 133.

Cohn, M. (2008). *Advantages Of The "As A User, I Want" User Story Template*. [online] Available at: `https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template` [Accessed May 23, 2017].

Daelemans, W. and Hoste, V. (2013). *STYLENE: an Environment for Stylometry and Readability Research for Dutch*. Tech. rep. Technical report, CLiPS Research Center, University of Antwerp, Antwerp, Belgium. 19.

Diederich, J., Kindermann, J., Leopold, E., and Paass, G. (2003). Authorship attribution with support vector machines. *Applied intelligence*, 19(1), pp. 109–123.

Evaluating Variations in Language Lab (2017). *Java Graphical Authorship Attribution Program*. [online] Available at: `https://github.com/evllabs/JGAAP` [Accessed May 18, 2017].

Gamma, E. and Beck, K. (1999). JUnit: A cook's tour. *Java Report*, 4(5), pp. 27–38.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., pp. 163–174, 185–194, 223.

Goodger, D. and Rossum, G. van (2001). *PEP 257 – Docstring Conventions*. [online] Available at: `https://www.python.org/dev/peps/pep-0257/` [Accessed Apr. 25, 2017].

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1), pp. 10–18.

Hohpe, G. and Woolf, B. (2003). *Enterprise Integration Patterns: Pipes and Filters*. [online] Available at: `http://www.enterpriseintegrationpatterns.com/patterns/messaging/PipesAndFilters.html` [Accessed May 26, 2017].

Holmes, D. (1994). Authorship attribution. *Computers and the Humanities*, 28(2), pp. 87–106.

Honnibal, M. and Montani, I. (2017). *spaCy API Documentation*. [online] Available at: `https://spacy.io/docs/api/` [Accessed May 26, 2017].

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2016). *A practical guide to support vector classification*. [online] Available at: `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf` [Accessed May 30, 2017].

Juola, P. (2008). Authorship attribution. *Foundations and Trends® in Information Retrieval*, 1(3), pp. 233–334.

Juola, P. (2009). JGAAP: A system for comparative evaluation of authorship attribution. *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*. Vol. 1. 1.

Juola, P. (2013). *How a computer program helped reveal JK Rowling as author of A Cuckoo's Calling*. [online] Available at: `https://www.scientificamerican.com/article/how-a-computer-program-helped-show-jk-rowling-write-a-cuckoos-calling/` [Accessed June 1, 2017].

Juola, P., Stamatatos, E., Argamon, S., and Koppel, M. (2011). *PAN 2011 Author Identification*. [online] Available at: `http://pan.webis.de/clef11/pan11-web/author-identification.html` [Accessed Mar. 23, 2017].

Kniberg, H. (2015). *Scrum and XP from the Trenches*. Lulu. com, pp. 9–10.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai*. Vol. 14. 2. Stanford, CA, pp. 1137–1145.

Koppel, M., Schler, J., and Argamon, S. (2009). Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1), pp. 9–26.

Luyckx, K. and Daelemans, W. (2008). Authorship attribution and verification with many authors and limited data. *Proceedings of the 22nd International Conference on Computational Linguistics*. Vol. 1. Association for Computational Linguistics, pp. 513–520.

McDonald, A., Afroz, S., Caliskan, A., Stolerman, A., and Greenstadt, R. (2012). Use fewer instances of the letter "i": Toward writing style anonymization. *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, pp. 299–318.

Mendenhall, T. C. (1887). The characteristic curves of composition. *Science*, pp. 237–249.

Millican, P. (2003). *The Signature Stylometric System*. [online] Available at: `http://www.philocomp.net/humanities/signature.htm` [Accessed May 19, 2017].

Mosteller, F. and Wallace, D. (1964). Inference and disputed authorship: The Federalist.

NLTK Project (2017). *NLTK documentation*. [online] Available at: `http://www.nltk.org/api/nltk.html` [Accessed May 26, 2017].

Radtke, M. (2017). *Topic Modeling of the codecentric Blog Articles*. [online] Available at: `https://blog.codecentric.de/en/2017/01/topic-modeling-codecentric-blog-articles/` [Accessed Mar. 24, 2017].

Ramnial, H., Panchoo, S., and Pudaruth, S. (2015). Authorship Attribution Using Stylometry and Machine Learning Techniques. *Intelligent Systems Technologies and Applications*, 1, p. 113.

Reich, Y. and Barai, S. (1999). Evaluating machine learning models for engineering problems. *Artificial Intelligence in Engineering*, 13(3), pp. 257–272.

Rossum, G. van, Warsaw, B., and Coghlan, N. (2013). *PEP 8 – Style Guide for Python Code*. [online] Available at: `https://www.python.org/dev/peps/pep-0008/` [Accessed Apr. 25, 2017].

scikit-learn developers (2016). *API Reference – scikit-learn*. [online] Available at: `http://scikit-learn.org/0.18/modules/classes.html` [Accessed Mar. 24, 2017].

Shrestha, P., Mukherjee, A., and Solorio, T. (2016). Large Scale Authorship Attribution of Online Reviews.

Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3), pp. 538–556.

Verschuren, P., Doorewaard, H., Poper, R., and Mellion, M. (2010). *Designing a research project*. Vol. 2. Eleven International Publishing The Hague.

Witten, I., Frank, E., and Hall, M. (2011). *Data Mining: Practical machine learning tools and techniques*. 3rd ed. Burlington: Morgan Kaufmann.

Zheng, R., Li, J., Chen, H., and Huang, Z. (2006). A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American society for information science and technology*, 57(3), pp. 378–393.

Appendix A
Project Planning

A.1 Gantt Chart

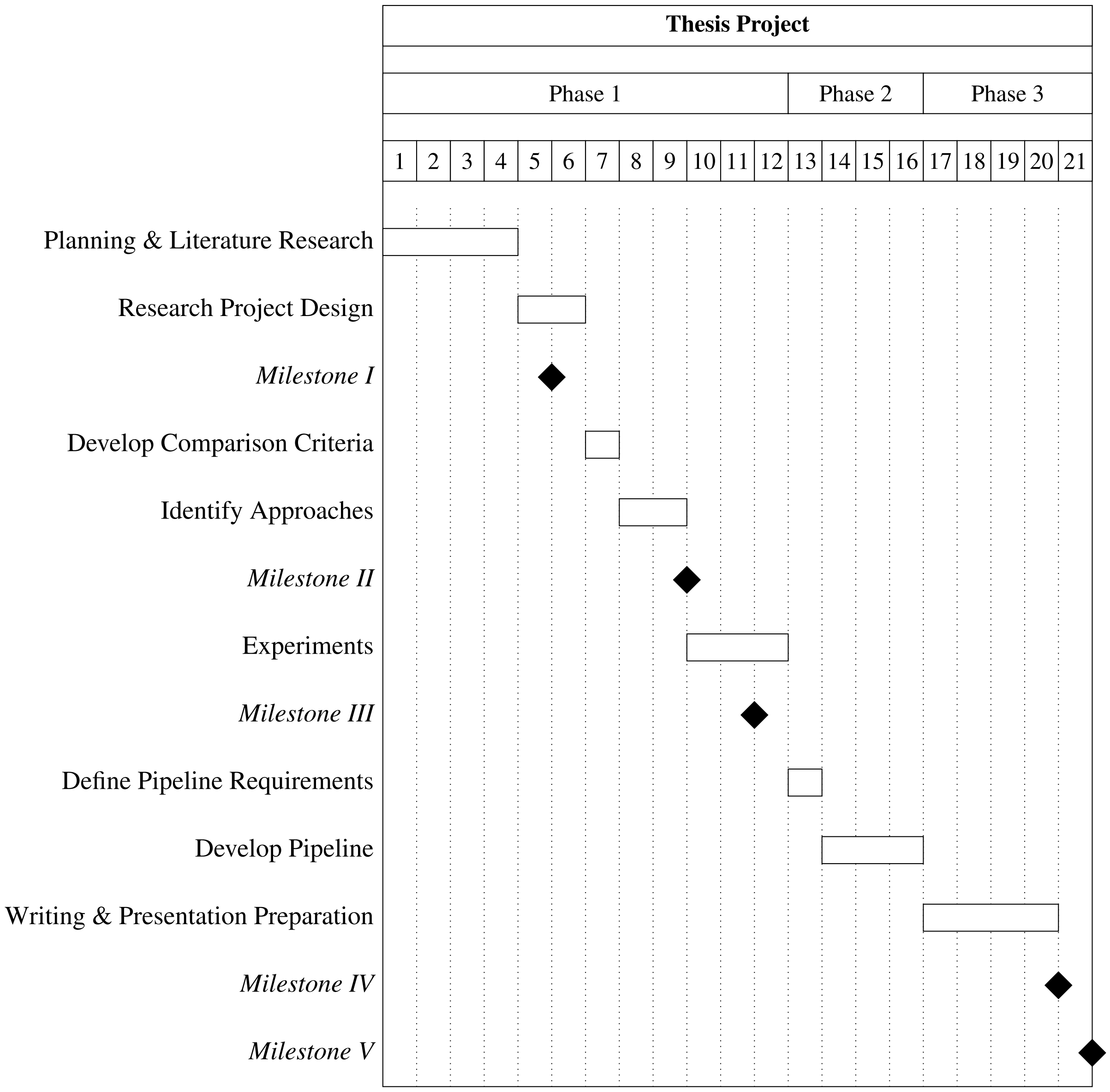

Figure A.1: Gantt chart of project phases and activities.

A.2 Quality Assurance Plan of Pipeline Implementation

This document serves as a quality assurance plan for the software development of the automated authorship attribution pipeline. It defines various quality assuring methods that cover the activities *implementation* and *test* of the software. This document does not follow a standardized format such as the IEEE 730 specification, because this project is undertaken in a very small-scale environment, and no formal quality contract between the customer and the developer is required. Instead, it is written in accordance with the general principles of an agile definition of done.

Coding Standards

The written code has to comply with the language specific coding conventions. For Python, these coding conventions are defined by the PEP8 (Rossum, Warsaw, and Coghlan, 2013). This document states guidelines for *naming*, *comments*, and general *code layout*. For documentation strings, the PEP257 standard has to be followed (Goodger and Rossum, 2001). In order to enforce these coding standards, a linter such as *Pylint* is to be used.

Testing Methods

Different testing strategies are used to enhance the quality of the software on various levels. The following tests must be written during the implementation of the software:

Unit Test	These tests apply to small fragments of the application (methods and classes).
Integration Tests	These tests consider multiple classes and their interactions with each other.
Code Coverage Tests	Measure the amount of code which is executed (i.e. covered) throughout the testing procedure.
Acceptance Tests	Testing on user story level (i.e. has the user story been fully implemented).

All of these tests must be automated. Where applicable, tests shall be written in a test-driven manner.

Appendix B

Technical Research Design

The following document describes the technical design of the research project. While the conceptual research design defines the boundaries and the overall goal of the research project, the technical design describes how to execute it. For each question, the research method (e.g. literature research, experiment, interview), the execution procedure and the analysis procedure is given. This document concludes with a time planning for the research project.

B.1 RQ1 - Evaluation Criteria

Question RQ1.1 asks for the criteria that can be used to assess the performance of an authorship attribution algorithm. A *literature review* has been selected as the method to answer this question. Because the systematic evaluation and comparison of machine learning models is an essential part of the data mining process (Witten, Frank, and Hall, 2011), it is expected that sufficient literature in this field exists.

In addition to the general evaluation criteria for machine learning models (RQ1.1), this research will identify which of these techniques are commonly used to assess authorship attribution models (RQ1.2 and RQ1.3). Thus, a *literature review* on authorship attribution specific material and a product review of an authorship attribution tool will be conducted. Since an evaluation criterion has to be operationalized later on in this research project, it is necessary to identify how to measure it. In order to achieve a higher validity of the individual results of RQ1.1 and RQ1.2, an interview with an expert in the field of machine learning and text mining will be conducted.

Search Terms and Environment

Due to time and scope restrictions, a preliminary list of related literature to establish the evaluation criteria has been identified[1]:

- “Data Mining: Practical machine learning tools and techniques” by Witten, Frank, and Hall (2011).
- “A study of cross-validation and bootstrap for accuracy estimation and model selection” by Kohavi (1995).
- “Evaluating machine learning models for engineering problems” by Reich and Barai (1999).
- “Authorship attribution” by Juola (2008).
- “A survey of modern authorship attribution methods” by Stamatatos (2009).

Besides the theory on model evaluation, the tool JStylo will be examined, which is a stylometry and authorship attribution research platform (McDonald et al., 2012). It provides various means for feature extraction

[1]search query `machine learning model evaluation|selection`

and authorship attribution using the classification algorithms available in WEKA (Hall et al., 2009). It also produces a wide range of different performance indicators. This research will draw from these indicators, since they are expected to be relevant to authorship attribution tasks.

Expert Interview

An expert interview provides the necessary practical context to augment the theoretical principles in model evaluation. The overall population of the interview are any experts in the field of machine learning and text mining. The sample which will be drawn from this population is an expert from the data science department of codecentric. On the one hand, a downside of this approach are any personal biases towards specific criteria and methodologies. On the other hand, the candidate will provide methodologies which are relevant and have proven to be useful in work relating to codecentric which increases the usefulness of the results.

The following questions shall be answered during the interview:

- What evaluation metrics do you typically use at codecentric to compare the performance of machine learning models?
- How do you ensure that the resulting figures are reliable?
- What are typical performance indicators for models in the field of NLP?

Execution Procedure

The execution procedure for this research question is as follows:

1. Gather information about model evaluation from literature.
2. Gather information about model evaluation from JStylo.
3. Contact expert in order to arrange a meeting for the interview.
4. Execute expert interview.

Analysis Procedure

The individual results will be aggregated as soon as all information is available. A possible form of this aggregation could be a table providing the name and the purpose of a criterion, alongside with information on how to measure it.

B.2 RQ2 - Approach Identification and Definition

The second research question aims to identify popular authorship attribution approaches. This will be accomplished by conducting another literature review. A literature review is an appropriate method to answer this question, since there are no experts directly available nor is the field in general very popular. Therefore, the main information sources are the scientific publications of researchers.

Due to limitations in time and scope, a selection of literature had to be made in advance. There are many publications in the field that would be relevant to this project, however it is infeasible to review all of them. A preliminary scan of this literature has resulted in the list of research surveys described in Section 3.3.4.

After the literature has been studied, an aggregated overview of the individual approaches alongside with their characteristics and requirements will be composed.

B.3 RQ3 - Approach Assessment

The last research question RQ3 aims to get insights about the performance and technical details of the authorship attribution approaches. This requires their evaluation and comparison in practice by using the criteria obtained from RQ1. Therefore, an experiment that empirically obtains the required information is a suitable approach.

Execution Procedure

Using the corpus of codecentric blog articles, the approaches will be assessed against the criteria identified in RQ1. Experiments using a specific dataset always expose the risk of providing unreliable results due to issues in sampling. In order to increase reliability, techniques such as 10-fold stratified cross-validation have proven to provide more stable results (Kohavi, 1995).

If applicable, existing authorship attribution tools such as JStylo and JGAAP can be used. If they do not support the individual approach, an experimental environment using Python will be used. One could also use the programming language R, which is a widely used environment for scientific work, but Python is the preferred language of the data science department at codecentric.

Analysis Procedure

The results of the individual experiments will be aggregated in a tabular form mapping the criteria from RQ1 to the respective measure of the experiments. This allows for a quantitative comparison of the approaches. Finally, conclusions for each approach will be drawn, which result in the overview required for the overall research objective.

B.4 Time Planning

The following Gantt chart B.1 provides an overview of the time planning. The total duration of this research is estimated to be 6 weeks (starting in week 7 of the bachelor thesis project, see Figure A.1). The experiments are expected to consume a large majority of this time.

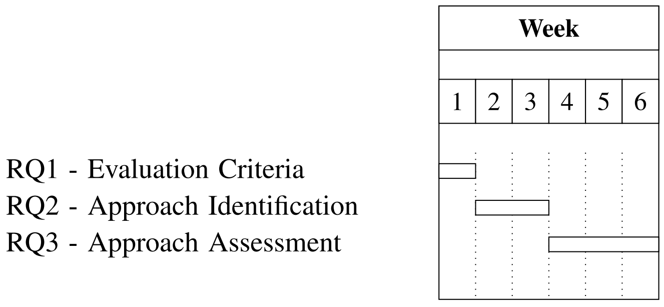

Figure B.1: Gantt chart of research project activities.

Appendix C

Research Project Results

C.1 Style Markers

The following Table C.1 describes the four main feature types that this project considers. The list of English and German function words has been extracted from the authorship attribution tool JStylo (McDonald et al., 2012).

Feature	**Description**	**Type**	**Extraction Tool**
FW	The N most frequent function words.	Lexical	Tokenizer
POS	The N most frequent part-of-speech tag unigrams, bigrams and trigrams.	Syntactic	Tokenizer, POS Tagger
CNG	The N most frequent character trigrams.	Character	not applicable
SF	Structural features (HTML formatting, URLs etc.)	Application Specific	HTML Parser

Table C.1: Selected set of style markers for the experiments.

C.2 Attribution Methods

Support Vector Machine (SVM)

Given a classification problem with a set of instances belonging to two different classes, an SVM tries to find a linear hyperplane that separates the classes from each other. New observations are assigned to either of these classes depending on the decision boundary. The SVM aims to maximize the margin of the hyperplane to the instances of the classes by incorporating a missclassification cost parameter. Non-linear problems are handled by mapping the data into a higher dimensional space using a *kernel*. In a multiclass classification problem, N classifiers are trained in a one-vs-all fashion.

k-Nearest Neighbors (kNN)

The kNN algorithm is a lazy learner, that computes the distance of an unlabeled instance to the nearest k labeled instances. The most frequent class (using a majority vote) under the k nearest neighbors is assigned to the unlabeled instance. An important hyperparameter is the choice of k and the distance metric. Two popular metrics are the *euclidean distance* and the *cosine similarity* whereas the latter one is preferable in a high dimensional feature space.

Multilayer Perceptron (MLP)

A MLP is a type of neural network that consists of multiple layers which each have a number of nodes. Each node in a layer is connected to the nodes of the succeeding layer. The network learns the mapping of input values to output values by associating weights to the connections and applying an activation function. The network learns by adapting the connection weights upon missclassification. A MLP can be used for non-linearly separable multiclass classification problems.

Random Forest

A random forest is a combination of multiple decision trees. A decision tree separates instances by splitting on the feature values. Eventually, the leaf of a tree assigns a class to the instance. The random forest constructs multiple decision trees and classifies an unlabeled instance with all of the trees. The predictions of the trees are aggregated by a majority vote.

C.3 Attribution Process

The following Figure C.1 illustrates the high level steps that are necessary when building and deploying authorship attribution models. The process is based on the framework for the attribution of online messages proposed by Zheng et al. (2006).

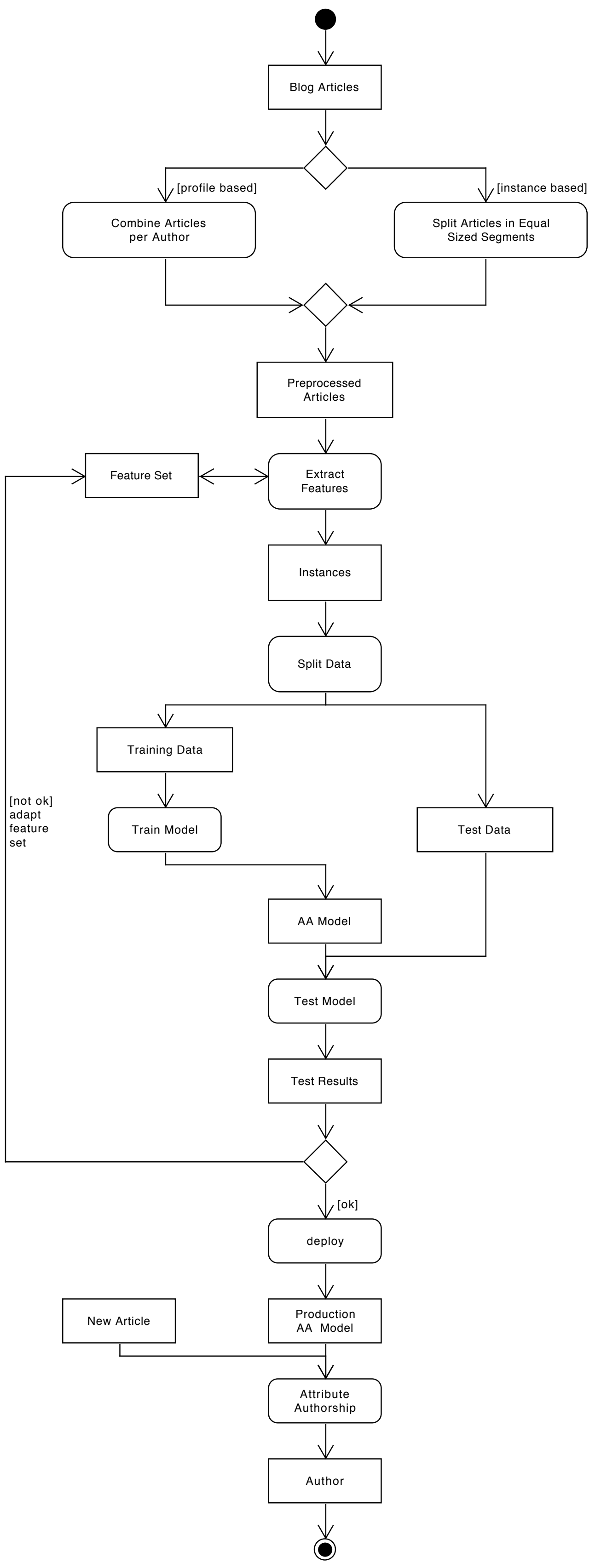

Figure C.1: Activity diagram of the authorship attribution process.

C.4 Exploratory Data Analysis

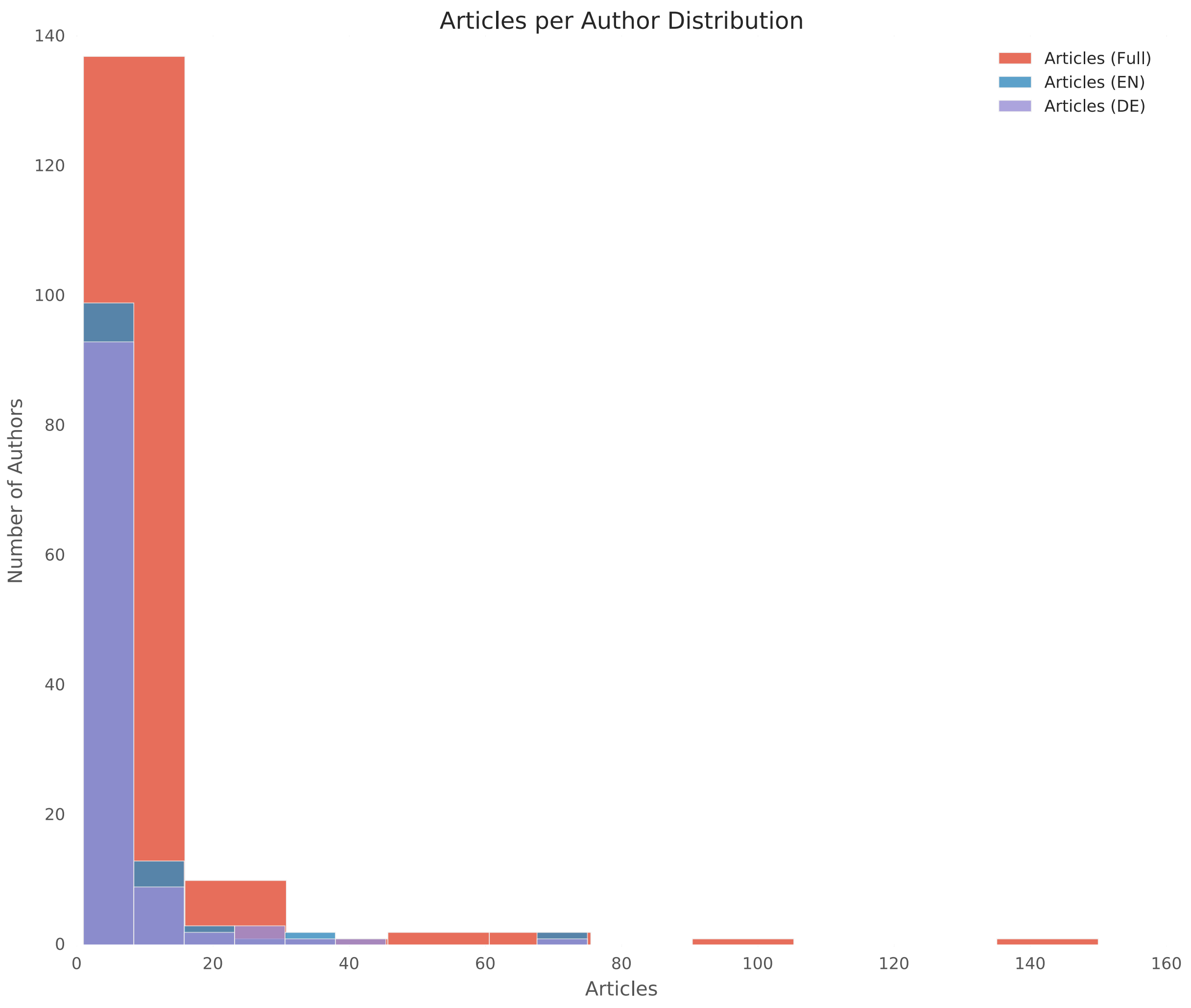

Figure C.2: Author distribution showing the English, German and full corpus.

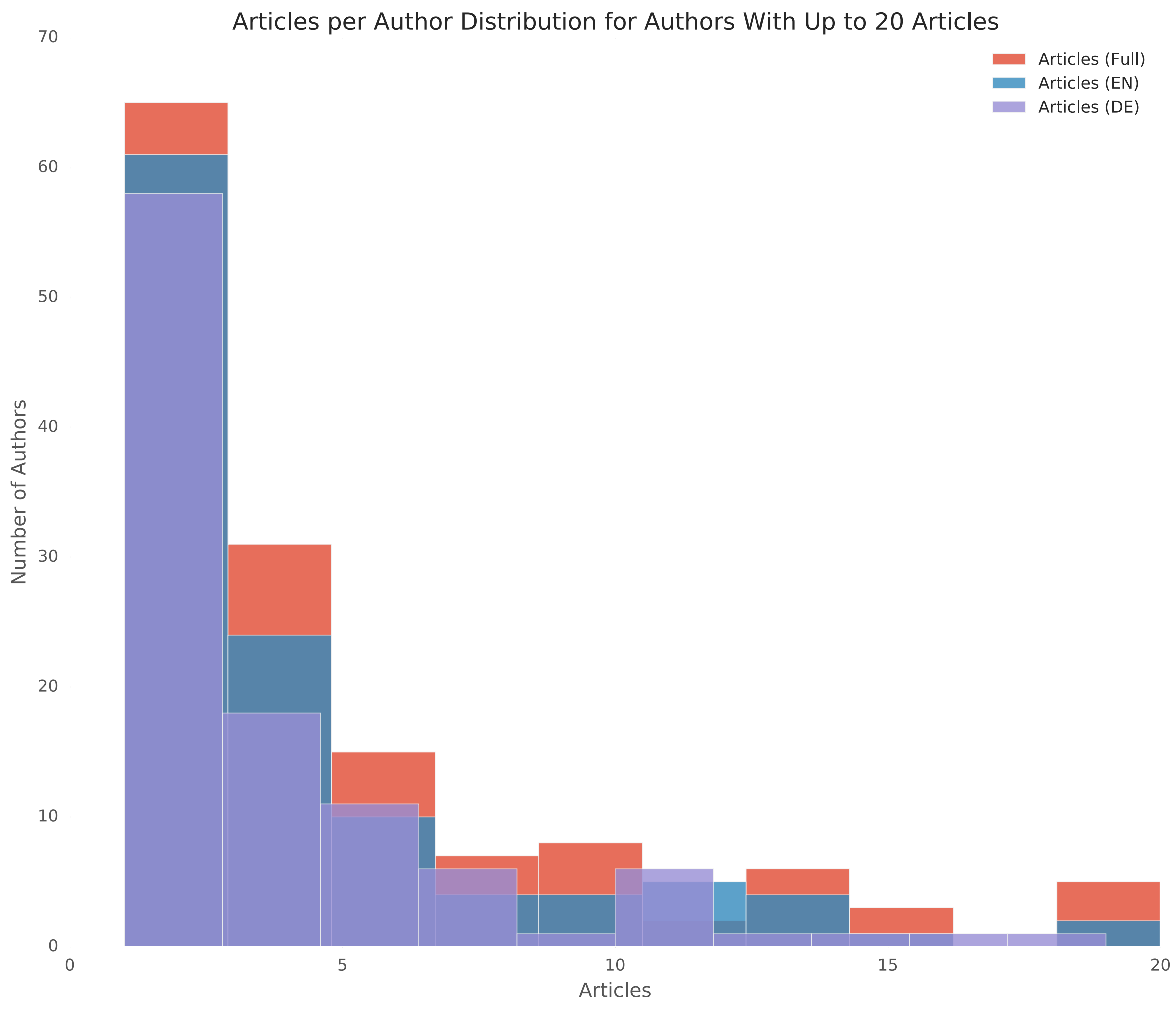

Figure C.3: Author distribution (limited to authors with a maximum of 20 articles).

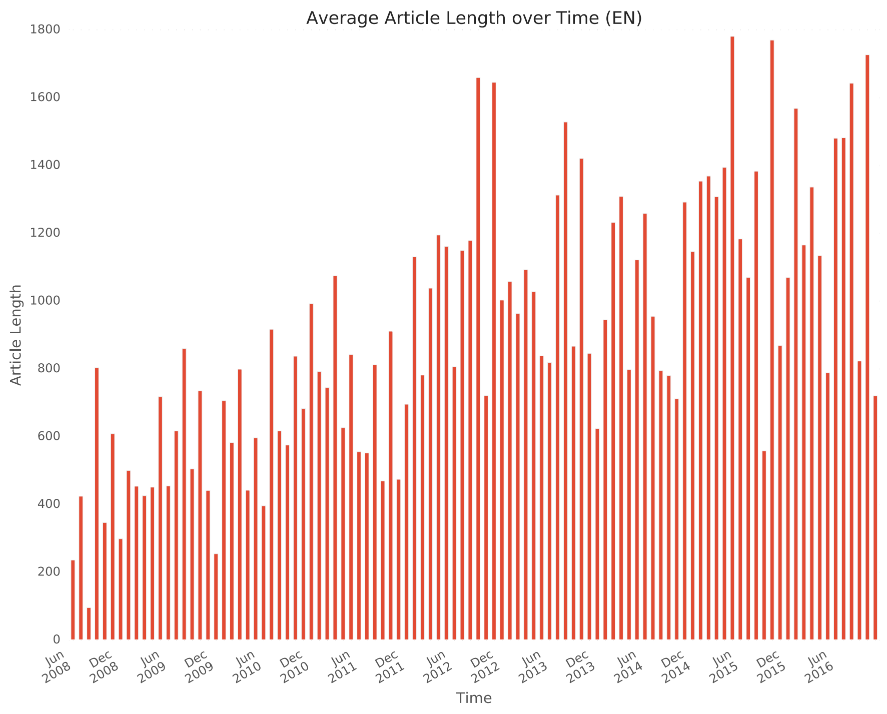

Figure C.4: Development of English article length (2008 – 2016, in words).

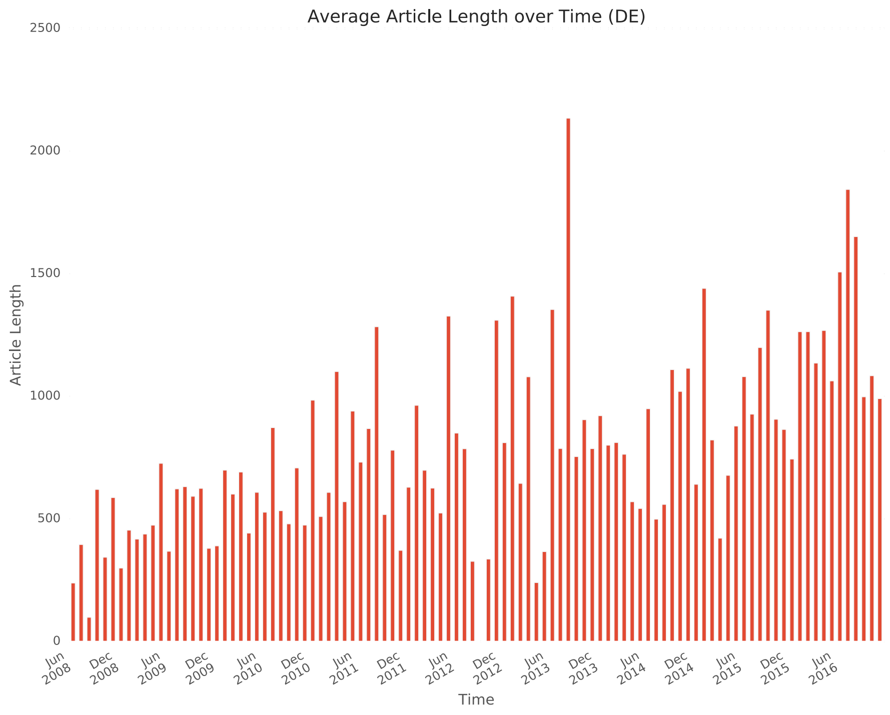

Figure C.5: Development of German article length (2008 – 2016, in words).

C.5 Experiment Results

	Language	Articles	Authors	Type	Benchmark	Articles/Author	Words/Author
sample1	en	198	15	instance	10 %	12	887
sample2	en	436	20	instance	17 %	13	880
sample3	de	122	10	instance	16 %	10	789
sample4	de	350	16	instance	21 %	15	692
sample5	en	198	15	profile	7 %	12	887
sample6	de	122	10	profile	10 %	10	789
sample7	en	661	85	profile	1 %	3	955
sample8	de	567	74	profile	1 %	4	655

Table C.2: Characteristics of the samples used in the experiments.

	sample1	sample2	sample3	sample4	sample5	sample6	sample7	sample8
fw	74	70	62	54	77	**85**	34	33
cng	34	41	51	33	73	70	37	27
pos	69	59	50	37	83	55	35	24
sf	40	37	49	39	53	40	25	12
fw+cng	76	70	70	56	**87**	80	38	35
fw+pos	**80**	75	68	58	**87**	**85**	43	36
fw+sf	72	70	67	55	87	80	39	28
fw+pos+cng	**80**	77	72	**63**	**87**	80	44	**40**
fw+pos+sf	77	77	**74**	**63**	**87**	80	42	34
fw+pos+sf+cng	79	**79**	**74**	**65**	**87**	80	**45**	38

Table C.3: Achieved accuracy score of feature sets per sample using an SVM classifier.

Appendix D

Pipeline Library Development

D.1 Functional Requirements

Identifier:	US-1	**Title:**	Data Loading	**Importance:**	20
Description:	As a user of the library, I want to load my text file based corpora so that I can use them with the authorship attribution pipeline.				
Acceptance Test:	Download the Brennan Greenstadt Extended corpus and load it with the authorship attribution library from the directory in which it is saved. All files in the directory should have been loaded, assigned to the respective author and in a format which allows further processing with the library.				

Identifier:	US-2	**Title:**	Custom Text Cleaning	**Importance:**	10
Description:	As a user of the library, I want to provide my own cleaning routines that specify which transformations are applied to the text so that I can handle my application specific text format.				
Acceptance Test:	Use the codecentric blog article corpus to create a cleaning routine that removes the content management specific formatting elements. These formatting elements should have been removed after the cleaning routine is applied.				

Identifier:	US-3	**Title:**	Feature Extraction	**Importance:**	40
Description:	As a machine learning expert using the library, I want to decide which style markers are used to quantify the style of an author so that I can fulfill my application specific use cases.				
Acceptance Test:	Construct two different pipelines using the codecentric blog article corpus. The first pipeline uses a set of style markers which are independent of the dataset such as function words and part-of-speech tags. The second pipeline makes use of structural features that quantify the usage of HTML formatting elements.				

Identifier:	US-4	**Title:**	Default Configuration	**Importance:**	30
Description:	As a non-machine learning expert using the library, I just want to provide my data to the library without having to worry about any configuration, such that I can focus on my business goals.				
Acceptance Test:	Construct a pipeline using the codecentric blog article corpus. Train the pipeline on a subset of the articles and test it with another subset. During this process, the user should not be required to configure features or attribution methods.				

Identifier:	US-5	**Title:**	Training Data	**Importance:**	40
Description:	As a user of the library, I want to provide the data which is used to learn the style of authors so that unknown authors can be detected based on this knowledge.				
Acceptance Test:	Split the blog article corpus into a training and testing set. Train the authorship attribution library with the training set. Afterwards, the library should be able to predict the authors of the articles in the testing set.				

Identifier:	US-6	**Title:**	Detect Author	**Importance:**	40
Description:	As a user of the library, I want to find the author of an unknown text so that I can use this knowledge in my use case.				
Acceptance Test:	Train the pipeline with any text corpus. Use a document which has not been part of the training set to predict the authorship. The outcome of this prediction should be an author label which has been part of the training process.				

Identifier:	US-7	**Title:**	Inspect Attribution Quality	**Importance:**	30
Description:	As a user of the library, I want be able to see the author detection quality so that I can make any adjustments to the type of style markers if needed.				
Acceptance Test:	Construct a pipeline and split a text corpus into a training and testing set. Afterwards, train the pipeline with the training set and predict with the testing set. The pipeline should output a number of machine learning performance metrics.				

Identifier:	US-8	**Title:**	Progress Indication	**Importance:**	10
Description:	As a user of the library, I want to be informed about the current status of lengthy operations so that I know when to expect the results.				
Acceptance Test:	Construct a pipeline and use a large testing corpora to train the pipeline. Verify that a status indication reports about the current progress.				

D.2 Control Flow of Complex Pipeline

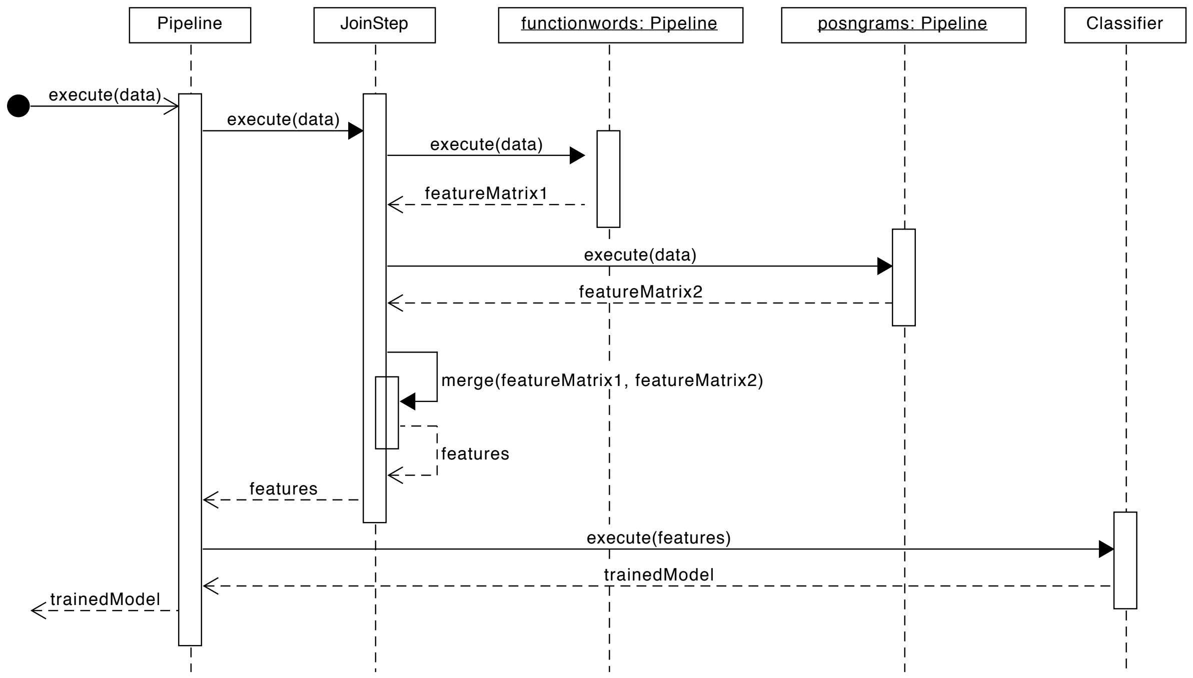

Figure D.1: Control flow in a nested pipeline to extract multiple feature sets.

D.3 Assessment of Natural Language Processing Frameworks

Overview of Frameworks

Library	Description
NLTK	The NLTK (v3.2.2) considers itself as a platform to perform NLP. It offers a wide range of different text processing functionality and implements several wrappers around other popular NLP libraries such as the Stanford CoreNLP. The NLTK aims to have large customizability and is widely used in research.
spaCy	spaCy (v1.8.2) aims to provide a simple API to perform NLP in a real-world application context. It is not as flexible as the NLTK, since it limits itself to a few high-performance algorithms that are suitable for large scale NLP applications.
TextBlob	TextBlob (v0.12.0) is another popular library, which is a wrapper around the NLTK. It provides sophisticated functional extensions including text translation and spelling correction. Since this functionality is not needed for this project, and the computational efficiency will be comparable to the NLTK, this library does not undergo further investigation.
genism	genism (v2.1.0) is related to the other NLP libraries, but focuses on model learning (e.g. Latent Dirichlet Allocation (LDA) and document similarity).

Table D.1: Overview of Python NLP frameworks.

Assessment of Functional Capabilities

The functional capabilities have been determined by inspecting the API documentation of both spaCy and the NLTK (NLTK Project, 2017; Honnibal and Montani, 2017). They have been summarized in the following Table D.2.

Feature	NLTK	spaCy
Word Tokenization	Yes	Yes
Stemming	Yes	No
Lemmatization	Yes	Yes
Spell Checker	No	No
POS Tagging	Yes	Yes
Language Support	EN, RUS	EN, DE, FR

Table D.2: Assessment of functional capabilities and multi-language support of the NTLK and spaCy.

Assessment of Computational Efficiency

Computational efficiency has been assessed using the English articles of the codecentric blog corpus. It consists of 696 documents (approximately 780.000 words). The benchmark has been executed on Linux *4.9.27* using a *Intel Core i7-4850HQ CPU @ 2.30GHz* processor with 4 available processor cores and Python *3.5.2*, NLTK *3.2.2* and spaCy *1.8.2*. Each computation has been repeated 3 times to create an average over all runs. The duration per NLP task can be found in Figure D.2.

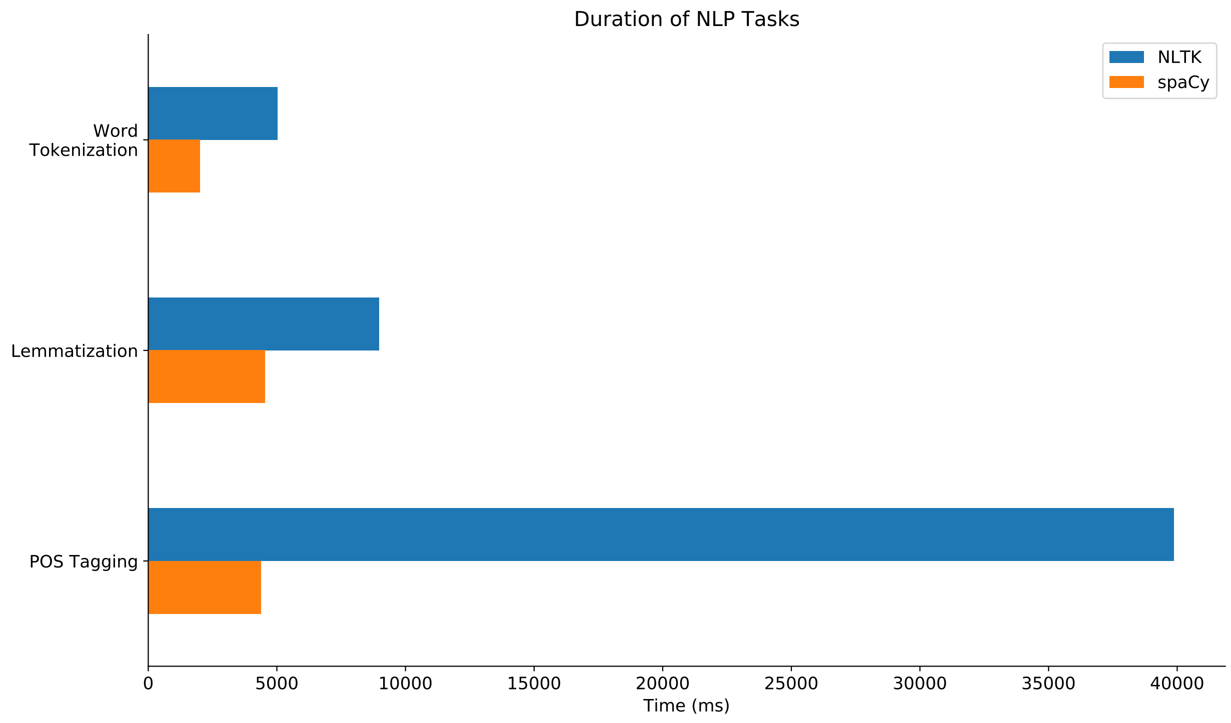

Figure D.2: Duration of common NLP tasks using NLTK and spaCy.