Johannes Daub
Matriculation Number: 3145320

Tool Support for the Automatic Analysis of Natural Language User Statements

Master Thesis

Winter Semester 20/21 - Summer Semester 21
Supervisors: Prof. Dr. Barbara Paech, Michael Anders
Faculty of Mathematics and Computer Science
Ruprecht-Karls-University Heidelberg

September 17, 2021

Abstract

[**Context & Motivation**] Developers need to learn about the requirements of software users, who give their feedback mostly in form of natural language statements. Processing these statements through manual coding, however, is an elaborate task and makes it unsuitable for big datasets. By extracting concepts from these statements, developers can get insights about the point of view of the software user. A software tool that provides automatic processing can help with this process.

[**Contributions**] This thesis explores the state-of-the-art topic modeling methods for user forums and applies suitable methods in the context of concept detection to a manually collected and annotated interview dataset. A software tool for automatic language processing, named "Feed.UVL" is created and the selected methods are integrated into this tool. The created software tool provides dataset management, which means that datasets can be stored, reviewed and deleted with the software. The implemented methods can be used to analyze these datasets for concepts. With the result visualization, the analysis results can be reviewed and the performance can be evaluated via the F1-score on a ground truth. Feed.UVL uses a micro-service architecture, which means it can be extended easily with new methods or functions. The integrated methods are then evaluated for the task of concept detection. A set of quality assurance measures, including static code analysis, component and system tests, have also been performed on the created tool.

[**Conclusion**] The main part of the thesis was the creation of a novel tool for natural language processing. The tool has a clean and user-friendly design and supports researchers in their analysis. Automatic analysis tasks can be handled and the user interface provides a rich display of results, including the metrics *false positives*, *false negatives*, *precision*, *recall* and *F1-score*. The current design and micro-service architecture ensures that the tool can be extended easily for further analysis methods and future research goals. At the moment, two state-of-the-art topic modeling methods (LDA and SeaNMF) are integrated, which were adapted for the use in concept detection. The evaluation has shown that while their precision is relatively high (0.84 for LDA and 0.83 for SeaNMF), their recall is rather low compared to a manually annotated ground truth for use in concept detection, which leaves space for improvements and future works.

Zusammenfassung

Werkzeugunterstützung zur automatischen Analyse von natürlichsprachlichen Nutzeräußerungen

[**Kontext & Motivation**] Entwickler müssen die Anforderungen von Software-Nutzern, die ihr Feedback meist in Form von Aussagen in natürlicher Sprache geben, kennenlernen. Die Verarbeitung dieser Aussagen durch manuelle Kodierung ist jedoch eine aufwändige Aufgabe und für große Datenmengen ungeeignet. Durch die Extraktion von Konzepten aus diesen Aussagen können Entwickler Einblicke in die Sichtweise der Software-Nutzer gewinnen. Ein Software-Werkzeug, das eine automatische Verarbeitung ermöglicht, kann bei diesem Prozess helfen.

[**Beiträge**] Diese Arbeit untersucht den Stand der Technik von Methoden zur Themenmodellierung für Benutzerforen und wendet geeignete Methoden im Kontext der Konzepterkennung auf einen manuell erhobenen und annotierten Interviewdatensatz an. Es wird ein Software-Werkzeug zur automatischen Sprachverarbeitung mit dem Namen "Feed.UVL" erstellt und die ausgewählten Methoden in dieses Werkzeug integriert. Das erstellte Software-Werkzeug bietet eine Datensatzverwaltung, d. h. Datensätze können mit der Software gespeichert, überprüft und gelöscht werden. Mit den implementierten Methoden können diese Datensätze auf Konzepte hin analysiert werden. Mit der Ergebnisvisualisierung können die Analyseergebnisse überprüft und die Leistung über den F1-Score auf einer Ground Truth bewertet werden. Feed.UVL verwendet eine Microservice-Architektur, d. h. es kann leicht um neue Methoden oder Funktionen erweitert werden. Die integrierten Methoden werden dann für die Aufgabe der Konzepterkennung evaluiert. Eine Reihe von Qualitätssicherungsmaßnahmen, einschließlich statischer Code-Analyse, Komponenten- und Systemtests, wurden ebenfalls mit dem erstellten Werkzeug durchgeführt.

[**Schlussfolgerungen**] Der Hauptteil der Arbeit war die Entwicklung eines neuartigen Werkzeugs für die Verarbeitung natürlicher Sprache. Das Werkzeug hat ein klares und benutzerfreundliches Design und unterstützt Forscher bei ihrer Analyse. Es können automatische Analyseaufgaben durchgeführt werden, und die Benutzeroberfläche bietet eine umfangreiche Ergebnisanzeige, einschließlich der Metriken *false positives*, *false-negatives*, *Precision*, *Recall* und *F1-Score*. Das aktuelle Design und die Microservice-Architektur stellen sicher, dass das Werkzeug leicht für weitere Analysemethoden und zukünftige Forschungsziele erweitert werden kann. Derzeit sind zwei moderne Methoden zur Themenmodellierung (LDA und SeaNMF) integriert, die für die Verwendung in der Konzepterkennung angepasst wurden. Die Evaluierung hat gezeigt, dass ihre Präzision zwar relativ hoch ist (0,84 für LDA und 0,83 für SeaNMF), ihr Recall im Vergleich zu einer manuell annotierten Ground Truth für den Einsatz in der Konzepterkennung jedoch eher gering ist, was Raum für Verbesserungen und zukünftige Arbeiten lässt.

Contents

1 Introduction

In this chapter, an overview of this thesis is presented. Section 1.1 motivates the benefits of tool support for the automatic analysis of natural language user statements. Subsequently, Section 1.2 lists and explains the goals of the thesis. Finally, Section 1.3 presents the structure of this thesis.

1.1 Motivation

When developers are creating software applications, they try to fit the applications to the needs of the user. These needs are captured as requirements of the software by the developers. Having precise and correct requirements is a key element in building successful software applications that are accepted by the user. Requirements can be collected from user feedback, which can have various forms. The feedback can be in the form of interviews or surveys, where specific information is gathered, or it can be more unspecific like in app store reviews, where any software property can be addressed. Software users are also talking about software in social media, websites or user forums, which can also be valuable sources for feedback. Those feedback sources share one thing in common: User feedback usually occurs in form of natural language.
To extract requirements from natural language statements, the statements have to be processed. One way of processing is the extraction of concepts from these statements, which provide information about what element of the application a user is discussing. The extraction of concepts can be done with coding and is usually done manually, which is a very labor-intensive work. For example, the Corona-Warn-App has more than sixty thousand app reviews in the Apple App Store[1], of which analysis would result in high cost if done in a reasonable amount of time. In the work of Nelson et al. [28], the researchers are only coding about 15% of about 8,500 news articles, stating this as a well-known limitation of coding methods.
A software tool to automatically process natural language user statements with methods

[1]Apple App Store: https://apps.apple.com/de/app/corona-warn-app/id1512595757 Last accessed: 23.08.2021

from machine learning or natural language processing (NLP), would reduce the time needed to analyze user feedback. User feedback can be collected as a dataset and be managed inside an application. Within the application, the datasets can be analyzed with different machine learning methods and the results are visualized. This can help researchers with annotating the data. As a result, the feedback loop could be much smaller, and requirements can be more precise and up to date with the current needs of a user.

1.2 Goals

The main goal of this thesis is to create a software tool that does support the automatic processing of natural language user statements. The tool is going to be integrated into feed.ai (Section 2.4) which is an existing tool, providing functionalities for Twitter and app store analysis. As feed.ai is a web-based application with a micro-service architecture, this structure is kept for this tool, which is named "Feed.UVL". The user interface (UI) of feed.ai is extended by everything that is needed for Feed.UVL and both parts share the same look and feel, which means that Feed.UVL is adapted accordingly to the style of feed.ai.
The focus of this thesis lies on the tool creation and the integration of topic modeling methods that have been applied to user forums. For this, a literature review is performed and appropriate methods are selected for implementation. These methods are evaluated on the task of concept detection with a given dataset. Different parameters are tested for those methods and the results are compared to a ground truth.
A set of requirements is compiled for the implementation. The tool allows the selection of different methods in the graphical user interface (GUI), along with the method parameters. After an analysis run has been performed, the tool user is able to view and verify the results. The analysis results are visualized by the tool, according to the applied method, and presented to the tool user. To compare the analysis results with manually extracted concepts, the tool also supports the comparison with this ground truth. A measurement of precision, recall and F1-score is helpful for the tool user to evaluate the performance of the applied method. The tool also makes it suitable for the tool user to manage datasets, which can be uploaded, viewed and deleted. The dataset contents are pre-processed to the requirements of the implemented methods, which is also part of the automatic processing of Feed.UVL. To enable analysis of further methods, the tool is designed in a way, such that it can be easily extended and changed, according to future needs. The resulting tool is tested to ensure code quality and functionality.

1.3 Overview

Following this chapter, the fundamentals, which are necessary for understanding this thesis, are introduced in Chapter 2. A research part follows in Chapter 3, where a literature review has been performed based on predefined research questions. Afterwards, the requirements for Feed.UVL are provided in Chapter 4. Next, Chapter 5 describes the design and the implementation of Feed.UVL. Within Chapter 6, the tests that are performed to ensure the quality of the tool, are presented. Chapter 7 explains how the work is evaluated. Finally, Chapter 8 summarizes and discusses the thesis, and gives an outlook on future improvements.

2 Fundamentals

The fundamentals chapter describes the concepts, methods and technologies on which this thesis is built upon. Section 2.1 explains the coding of natural language. The implementation of this thesis uses a micro-service architecture, which is explained in Section 2.2. The containerization platform Docker that is used to implement the micro-service architecture is described in Section 2.3. The software platform feed.ai that this thesis is built upon is introduced in Section 2.4. The following Section 2.5 gives a short introduction to topic modeling and related areas. Latent Dirichlet Allocation and non-negative matrix factorization are both methods that are used in the implementation of this thesis. These are both explained in Sections 2.6 and 2.7 respectively.

2.1 Coding

Coding refers to assigning tags to segments of text. This can be used to analyze natural language texts as part of a Qualitative Content Analysis (QCA) [20], which is a research method for analyzing qualitative data. Codes can be different types of categories (also called coding frames), e.g. topics, concepts or sentiment. A segment of text is at least a single word, as it is the smallest semantic unit of natural language. Coding is usually performed manually, which means that the text segments have to be selected and codes have to be defined and assigned by a human. However, there are approaches that perform coding with machine learning [28]. Alternatively, software tools such as MAXQDA[1], Nvivo[2], Inception[3], Gate[4] can be used to support manual coding.

[1]MAXQDA: https://www.maxqda.com/ Last accessed: 13.08.2021
[2]Nvivo: https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home Last accessed: 17.08.2021
[3]Inception: https://inception-project.github.io/ Last accessed: 17.08.2021
[4]Gate: https://gate.ac.uk/ Last accessed: 17.08.2021

2.2 Micro-service Architecture

The micro-service architecture [37] offers an alternative to monolithic software applications. Instead of building one huge monolithic application with many components, the software functions are split into multiple services with their own code. Micro-services often contain only a single function. The communication between micro-services is often handled with a technology-agnostic protocol as HTTP, which means that they are deployed in a network- or cloud-environment. Figure 2.1 shows an example micro-service architecture.

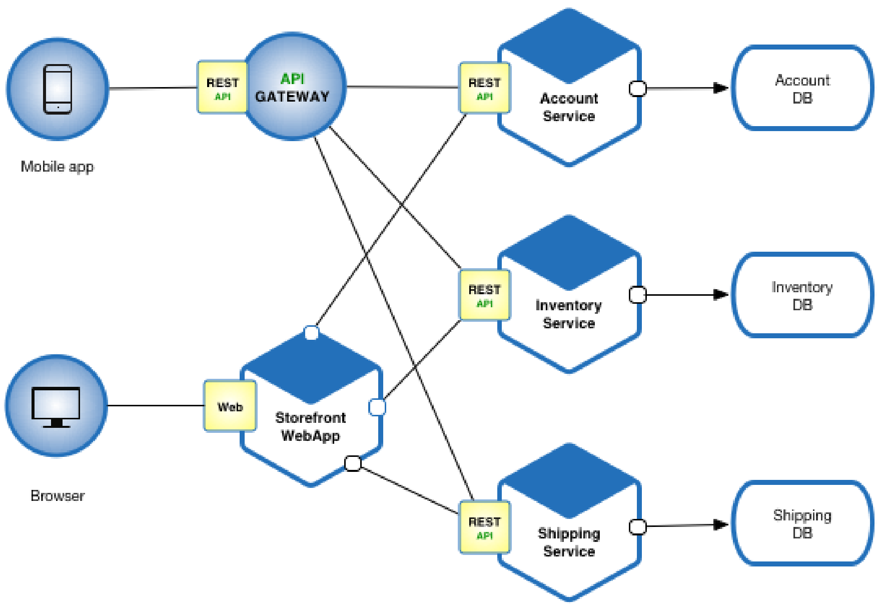


Fig. 2.1: Example of the Micro-service Architecture [5]

The usage of this architecture has many benefits. When changes to the code have to be made, the monolithic application has to be rebuilt at once to deploy the changes. With the micro-service architecture, only the service containing the changes needs to be rebuilt, the rest of the system remains stable.
Micro-services can be replaced and are independent of their programming language as long as function and the application programming interface (API) are preserved. This also means a micro-service can use different kinds of technologies or databases than the other services. Testing and deployment are also done on the micro-service level, which makes it easier to maintain them.
The system also becomes more resilient to software crashes. When a micro-service crashes, the rest of the system usually remains intact, only losing a part of its func-

[5] https://microservices.io/patterns/microservices.html Last accessed: 12.08.2021

tionality, whereas in a monolith, the whole system crashes, leading to a full outage. Afterwards, it is also easier to track down the error, as the crashed micro-service also indicates the source of failure.
When the system has to deal with high load, it may be necessary to scale it. A micro-service that encounters high load, can be replicated and spread across multiple servers as needed. For a monolith, the whole system has to be replicated, which creates unnecessary redundancy for functions that do not encounter higher load, and which is much slower due to the application size. Figure 2.2 shows an example of replication and scaling for both architectures.

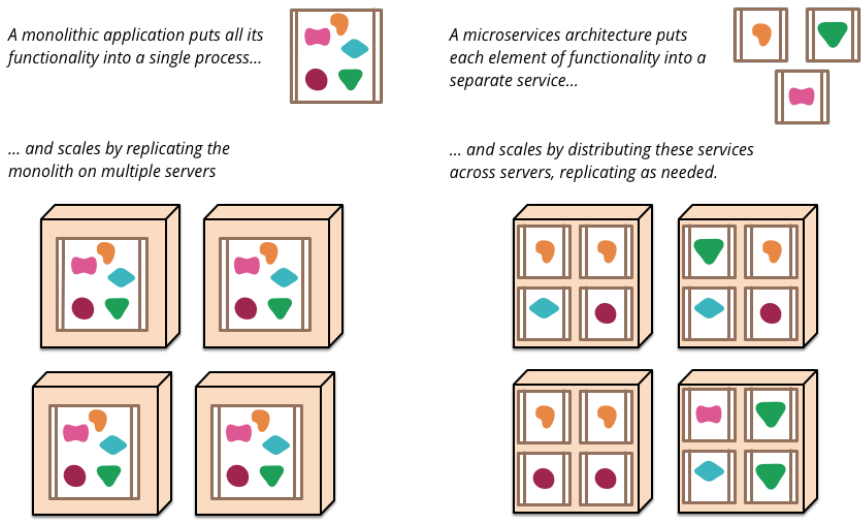

A monolithic application puts all its functionality into a single process...
... and scales by replicating the monolith on multiple servers
A microservices architecture puts each element of functionality into a separate service...
... and scales by distributing these services across servers, replicating as needed.

Fig. 2.2: Scaling for Monolithic Software and Micro-services[6]

2.3 Docker

Docker[7] is an open source platform for building, deploying, and managing containerized applications. With Docker, applications can be containerized, which means packaging the application together with operating system libraries and dependencies. A container can be run independently of the host operating system. Containerization can be seen as a lightweight alternative to deployment with virtual machines, which require more resources to run an operating system. See Figure 2.3 for the difference in overhead needed by virtual machines and docker containers. A Docker container can be used to deploy a micro-service as part of a micro-service architecture (Section 2.2). Those containers can then be connected via a network to enable their communication.

[6]https://martinfowler.com/articles/microservices.html Last accessed: 12.08.2021
[7]Docker: https://www.docker.com Last accessed: 12.08.2021

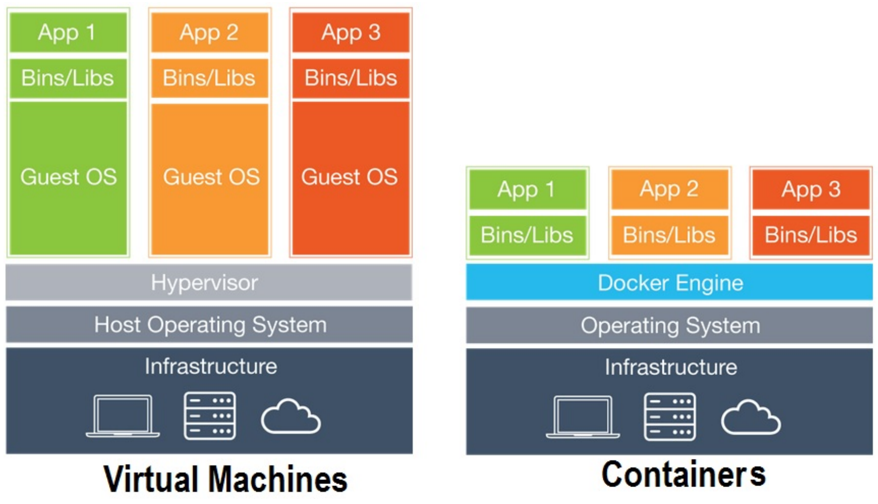


Fig. 2.3: Virtual Machine Setup compared to Docker Container Setup[8]

2.4 Feed.ai

Feed.ai [38] is an open-source software tool[9] developed by Christoph Stanik in his doctoral thesis as part of the OpenReq project[10]. Some basic functions of feed.ai are the collection, analysis and visualization of tweets that contain user feedback. Feed.ai is built with a micro-service architecture (Section 2.2), which means functions are split into different services. Those services are deployed with Docker (Section 2.3) and are connected in a network. The communication between services is done via HTTP-based web API requests. The full stack for the tweet data domain is depicted in Figure 2.4. Its components are described as follows:

- **Tweet Data Storage.** This micro-service is responsible for saving and loading data from tweets and classifications. It is connected to a database and offers an API to read and write data.

- **Tweet Collector.** A micro-service that pools configured Twitter accounts regularly for new tweets, which are then sent to the tweet data storage.

- **Tweet Orchestrator.** The tweet orchestrator periodically checks the storage for new unprocessed tweets and retrieves them. Those new tweets will then be sent to the data analytics layer for processing. Classified tweets are then sent back to the storage.

[8] https://techglimpse.com/docker-installation-tutorial-centos/ Last accessed: 12.08.2021
[9] Feed.ai GitHub: https://github.com/openreqeu Last accessed: 16.09.2021
[10] OpenReq: https://openreq.eu/ Last accessed: 31.08.2021

- **Tweet Sentiment Classification/Tweet Intention Classification.** Tweets that this micro-service receives are processed with machine learning techniques to classify their sentiment and intention. The results are sent back to the orchestrator.

- **Data Visualization.** The visualization micro-service provides an interface of the data for the user. The data can be accessed via a web application that has various visualizations for the tweets and their date and time of posting, for the classified sentiment and intention, and for the settings of the collector.

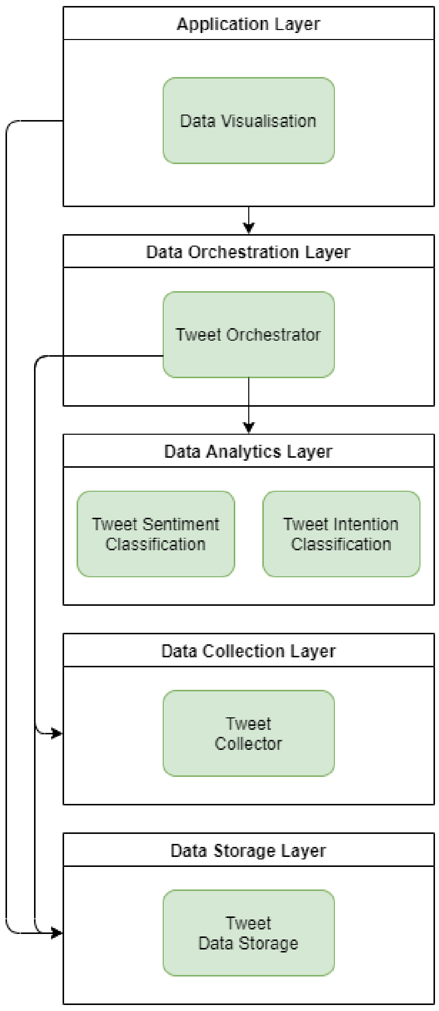


Fig. 2.4: Feed.ai Micro-service Structure for Tweet Data [38]

2.5 Topic Modeling

Topic modeling is the unsupervised classification of documents into topics. This can be compared to clustering on numeric data, where data is grouped based on certain criteria. In mathematical terms, a topic model is a probabilistic generative model, which models the distribution of topics in a set of documents. Topic models are used to organize, understand and summarize large collections of textual information. Furthermore, a topic model can reveal hidden topic patterns, e.g. the topics discussed in forum threads might not represent the forum category structure. Topic models are often used to analyze news and social media. Besides topic modeling, there are two areas that are closely connected:

Topic labeling [40] is the process of generating or finding appropriate labels to topics, that were found by topic modeling. It can be seen as the next step of processing, as topic labels have to be found manually otherwise.

Topic detection and tracking (TDT) [1] is the process of finding topics and tracking them over a time period. This can be relevant for tracking topics in news, social media, or web forums, as the topics may change over time. A topic model can be used for TDT if the topic model supports adaption to new data.

For topic modeling, the following terms are defined and will also be used during this thesis [6]:

- A *word* is the basic unit of discrete data, defined to be an item from a vocabulary, indexed by $1, ..., V$. Words are represented as unit-basis vectors that have a single component equal to one and all other components equal to zero.
- A *document* is a sequence of N words, denoted by $W = (w_1, w_2, ..., w_N)$, where w_n is the nth word in the sequence.
- A *corpus* is a collection of M documents denoted by $D = w_1, w_2, ..., w_M$.
- A *dataset* contains a corpus and may contain additional information (metadata). For example, a forum post dataset may include post structure, timestamps, and author information.
- A *topic* consists of words of the corpus. The number of words in a topic can be either fixed to some value or be based on a threshold of some score, depending on the method.

2.6 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) by Blei et al. [6] is a method used for topic modeling. It is explained here, as it has been encountered in Chapter 3 very often. Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. The occurrence of a topic in a document (topic prevalence) is described with a Dirichlet distribution as a prior probability. An approximation of this distribution can be calculated via Gibbs Sampling, an algorithm for statistical inference. Topics are the x most probable words for a distribution, which are then called *topic words*. The top words of a topic are then presented to the human user to determine the topic label (see Section 2.5, topic labeling). Table 2.1 shows an example output of topic words, along with the topic description by a human annotator. The number of topics k is a parameter that is chosen before using the method. To apply the method, the following steps are performed [5]:

- Assign each word in each document one of k topics, the distribution is drawn from a Dirichlet distribution.

- For each document d and each word w, compute:

  $p(topic\ t \mid document\ d)$: the proportion of words in d that are assigned to topic t, except current word w.

  $p(word\ w \mid topic\ t)$: the proportion of assignments to topic t that come from word w.

- Update the probability of word w belonging to topic t:

$$p(word\ w\ with\ topic\ t) = p(topic\ t \mid document\ d) * p(word\ w \mid topic\ t)$$

If a lot of words from document d belong to topic t, it is also more probable that word w belongs to t. Since a document can contain multiple topics, if w has a high probability of being in topic t, documents containing w become more strongly associated with t. The application of LDA is not limited to topic modeling [6], which makes it a very versatile and popular method. It has been extended for many different applications [16].

Table 2.1: Example LDA Topic Word Output and Annotated Topic Description [17].

Top LDA Words	**Description**
problem error bug fix version time chang line miss build wrong solv open messag expect	Bugs and errors
diagram model creat editor uml select view element menu open add click explor packag show	Graphical editor issues mostly related to Papyrus and UML2 (e.g., hyperlinks between two models)
uml model implement metamodel specif extend gener languag instanc provid custom understand read time tool	Extending UML tools
uml ecor gener model code emf genmodel convert creat eclips metamodel project import map transform	UML code generation and related transformations from Ecore to UML
papyru eclips model plugin version code project sysml camil custom extens provid palett instal mart	Papyrus related issues, including installation and compatibility with Eclipse versions and source code
model project eclips uml tool emf code tutori creat develop vlad plugin inform revers start	Requests for tutorials and documentation
eclips instal updat uml featur site download tool depend build sdk version requir emf plugin	Update site issues and plugin dependency problems
profil stereotyp appli uml model defin creat static editor load problem applic packag definit save	UML profile and stereotype issues

2.7 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a matrix factorization method that requires all matrix entries to be either zero or positive. A matrix A with m rows and n columns can be factorized into two matrices W and H of size $m \times k$ and $k \times n$, so that $A \approx WH$ The entries of W and H also have to be positive or zero. The value k has to be $k <= min(m, n)$ and can be selected by the user. The matrices W and H can be found by minimizing the Frobenius norm $||A - WH||^2$, which can be achieved by applying a block coordinate descent optimization algorithm. NMF can be applied to various problems, as long as the matrix A is non-negative. As a result of Chapter 3, NMF has also been applied to topic modeling.
When applied to topic modeling. The matrix A is the word-document matrix and W and H can be seen as word-topic and topic-document matrices, where k is the number of topics, selected by the user. With this, each word can be assigned a probability to belong to a topic and each document has a probability to belong to a topic.

3 Literature Review

To get an overview about the scientific state of the art with regard to the research questions, a systematic literature review was conducted. Chapter 3.1 presents the research questions that define the goals of the literature review. Chapter 3.2 describes the methodology for finding relevant research papers and criteria to narrow down the search results. In Chapter 3.3 the found works will be displayed in detail. In Chapter 3.4 the search results will be compared according to certain criteria. Finally, Chapter 3.5 summarizes the results.

3.1 Research Questions

The research questions are used as a starting point for the literature review. From those questions, search terms and criteria are derived to efficiently find relevant literature. The main question RQ1 is designed to get an overview of topic modeling approaches that are used in the context of online forums. The sub-questions are looking more detailed at these approaches in terms of methods, automation, domain-specificity, problems and their handling. With this overview, suitable approaches and requirements can be selected for this work. The questions are shown in Table 3.1.

Table 3.1: Research Questions for the Literature Review

Name	Research Question
RQ1	**Which approaches for fully or semi-automatic topic modeling of natural language user statements in online forums exist and what are their characteristics?**
RQ1.1	Which machine learning or natural language processing methods are used by the approaches for which steps of the classification process?
RQ1.2	What topics are identified by these methods?
RQ1.3	What problems arise when using data from online forums (e.g., domain dependency, short texts, syntactical errors, etc.)?
RQ1.4	How do the approaches deal with these problems?

3.2 Methodology

To ensure the reproducibility of the literature review each step is documented extensively. The first part of the literature review is search term based. The selection of relevant works is done by defining criteria of relevance based on the research questions and checking the search results against them. Afterwards, a snowballing based approach is used to find further relevant works.

3.2.1 Criteria of Relevance

To distinguish relevant from irrelevant works, the criteria of relevance are specified. At first, it is important that the approaches are properly researched and scientifically sound (CoR3). Also, they need to be accessible (CoR2) and comprehensible (CoR1). Title and abstract of a paper are checked to get an overview on whether it appears to be initially relevant (CoR4). Both title and abstract are taken into account, as the title alone did not offer enough information about relevancy for many works. After that, the remaining works are checked thoroughly (CoR5-6). The criteria of relevancy are shown in Table 3.2.

Table 3.2: Criteria of Relevance

Name	**Criterion**
CoR1	The paper is written in either English or German
CoR2	The paper needs to be available for free or with University's access
CoR3	The paper has been peer reviewed
CoR4	Title and abstract show relevance to the research topic
CoR5	The presented approach is using a method for topic modeling in online forums
CoR6	The paper offers sufficient new content to the research topic

3.2.2 Search Term Based Research

The search term based approach is using search queries on online publication databases. To ensure the quality of the publications, the search is performed on the digital libraries of the Institute of Electrical and Electronics Engineers (IEEE)[1] and the Association for Computing Machinery (ACM)[2]. The search terms were chosen based on the main

[1]IEEExplore: https://ieeexplore.ieee.org/ Last accessed: 17.02.2021
[2]ACM Digital Library: https://dl.acm.org/ Last accessed: 17.02.2021

research question, which focuses on *topic modeling*, but this term did not cover the research area alone. As a result, more search terms, like *topic detection* or *topic labeling* (as seen in Section 2.5) have been selected for broader coverage of the research field. The term *forum* was used to focus the search on forum related works only, thus limiting the amount of unrelated works. The term *discussion* has been found to be used synonymously to *forum*, which has been discovered during reading abstracts of search results. The term *BBS* stands for bulletin board system and is also a synonym for *forum*, although it only influenced the amount of search results for the IEEE search. The search terms are limited to the title, because it offers a narrow context for those terms. Without the limitation, the words can occur in any context in the text, leading to thousands of search results. The search terms are displayed in Table 3.3. Note that *title* in the search term refers to *Title* in the ACM search and to *Document Title* in the IEEE search.

Table 3.3: Search Terms for the Literature Search

Name	**Search Term**
ST1	[Title: *topic*] AND [Title: *modeling*] AND [[Title: *forum*] OR [Title: *discussion*] OR [Title: *bbs*]]
ST2	[Title: *topic*] AND [Title: *detection*] AND [[Title: *forum*] OR [Title: *discussion*] OR [Title: *bbs*]]
ST3	[Title: *topic*] AND [Title: *labeling*] AND [[Title: *forum*] OR [Title: *discussion*] OR [Title: *bbs*]]

There have been some problems with the search term based research. A keyword search was tested, as some works have keywords like *topic detection* defined, but yielded search results that did not contain the keywords. The abstract has also been used for the search, but this led to many irrelevant search results, as the terms were used in a different context. The results of the search term based research with applied criteria for initial relevancy are shown in Table 3.4. Four works have been found in both databases [9], [26], [43], [50].

3.2.3 Snowballing

Snowballing refers to checking citations of works for further relevant approaches. Backward snowballing analyzes the papers a work has cited, which ought to be older. Forward snowballing is looking at papers that have cited a work. Digital libraries like ACM, IEEE and others have built-in support for the snowballing process, which makes it easier to find relevant works. Table 3.5 shows a complete overview of the search and

Table 3.4: Search Term Based Research results

Library	Term	Results	Initially Relevant	References
ACM	ST1	17	9	[4], [12], [46], [9], [14], [41], [30], [17], [10]
ACM	ST2	4	3	[50], [26], [43]
ACM	ST3	3	3	[2], [3], [10]
IEEE	ST1	15	10	[25], [45], [34], [18], [23], [24], [9], [44], [39], [19]
IEEE	ST2	14	12	[29], [13], [48], [21], [47], [26], [43], [42], [49], [50], [8], [22]
IEEE	ST3	1	1	[40]

snowballing results. There were not many relevant approaches found by snowballing. This is due to multiple reasons. Many are citing topic modeling methods, but those have not been applied to forums (CoR5). Some of them are using the same method for forum analysis, which would lead to redundant information (CoR6). Most papers are not about topic modeling and refer to another aspect related with the relevant papers (CoR4).

Table 3.5: Results of Search Term Based Approach and Snowballing

Author	Ref.	Title	Process	Library
Atapattu et al.	[2]	A Framework for Topic Generation and Labeling from MOOC Discussions	Search Term	ACM
Athira et al.	[3]	Multi-label Topic Classification of Patient Generated Content in a Breast-cancer Community Forum	Search Term	ACM
Chen et al.	[8]	Topic Detection over Online Forum	Search Term	IEEE
Cheng et al.	[9]	Linked Topic and Interest Model for Web Forums	Search Term	ACM&IEEE
Hsiao et al.	[14]	Topic Facet Modeling Semantic Visual Analytics for Online Discussion Forums	Search Term	ACM
Kahani et al.	[17]	The Problems with Eclipse Modeling Tools: A Topic Analysis of Eclipse Forums	Search Term	ACM
Li et al.	[21]	Forum topic detection based on hierarchical clustering	Search Term	IEEE
Liu et al.	[23]	Analyzing Topics of JUUL Discussions on Social Media Using a Semantics-assisted NMF model	Search Term	IEEE
Reich et al.	[31]	Computer-Assisted Reading and Discovery for Student-Generated Text in Massive Open Online Courses	Snowballing [2]	SoLAR[3]
Tang et al.	[40]	A Topic Label Extraction Method for the University BBS	Search Term	IEEE
Vytasek et al.	[41]	Topic Models to Support Instructors in MOOC Forums	Search Term	ACM
Wu et al.	[43]	Topic Detection in Online Discussion using Non-Negative Matrix Factorization	Search Term	ACM&IEEE
Zhang et al.	[48]	Document similarity measure for topic detection in BBS	Search Term	IEEE

[3]Journal of Learning Analytics: https://learning-analytics.info/ Last accessed: 15.09.2021

3.3 Review Results

A very practical work is done by Kahani et al. [17], who focus on gaining insight through topic analysis of forums. Their goal was to find issues with the Eclipse modeling tools[4], and forums are usually a reference point for users who encounter issues. The method they are using to extract topics from various Eclipse forums is called Latent Dirichlet Allocation (LDA). This method is also used in most other works that are following. Other works are more theoretical and are extending LDA for better performance by including forum-related information or by complementing it with other methods. As the work by Kahani et al. is very practical, they are very detailed with its usage, explaining parameters of LDA and providing reference values. The provided values are referenced from papers which are also using LDA, though not in the context of web forums. Their approach uses the MALLET framework[5], which is a NLP framework written in Java. Table 2.1 shows an example output of topic words.
Another work that uses LDA is done by Cheng et al. [9]. In their work, the goal is to improve the performance of LDA by including author information of posts into the algorithm. This is done by adding another set of hidden variables for authors. So this model can not only determine the topics of forum posts, but also the topics related to authors, which they refer to as author interest. For their evaluation, they are using the perplexity metric and are reporting better performance than standard LDA.
The work of Hsiao et al. [14] presents a different approach, where they introduce a Topic Facet Model (TFM). They are using Sequential LDA (SLDA) as a base, which also takes the positions of words into account, and have modified it to also determine topic facets. Their assumption is that all words in a single sentence are generated from one topic facet, which means topic facets offer more specificity. Additionally, they have created a web interface for viewing the forum threads and their analytics.
The paper by Vytasek et al. [41] has a focus on forums of Massive Open Online Courses (MOOC), which can be overwhelming with their content because of the huge amount of participants. Topic models can help instructors getting an overview of ongoing discussion topics and can categorize individual posts into topics. The authors are using the MALLET framework for topic modeling. In total, they are using four different models. The first approach uses a 20-topic model, which makes it difficult to navigate through posts, because each topic still has a huge amount of posts. For the second approach, they classified the posts into content/non-content related posts beforehand and then used those with a 10-topic classifier. This helped to give context to organizational

[4]Eclipse modeling tools: https://www.eclipse.org/downloads/packages/release/2020-12/r/eclipse-modeling-tools Last accessed: 08.03.2021

[5]McCallum, Andrew Kachites. "Mallet: A machine learning for language toolkit": http://mallet.cs.umass.edu, 2002. Last accessed: 08.04.2021

questions in those forums. Model three and four explored ways to organize topics into subtopics to reduce the number of post per topic, where just increasing the number of total topics would be more difficult for the instructors as they need to organize those topics. As this work is only a short paper, there is not much detail given on the used models.
The publication of Atapattu et al. [2], which is marked as work in progress, is also working with MOOC forums. In their approach, a set of candidate topic labels (key terms) is extracted from the course materials. The forum posts are classified with a Naive Bayes classifier into course weeks using the key terms. Candidate topic labels are generated by measuring the similarity between the topic terms of discussions and the extracted key terms of the corresponding week.
The regular LDA approach only yields a set of words that belong to a topic, but the topic label itself needs to be determined manually. The approach of Tang et al. [40] adds a topic label extraction process to LDA. After regular LDA clustering, a keyword extraction is applied, which will pick a keyword for each topic set. The keywords are selected by their $tf - idf$ and their correlation to other topics, where a low correlation means that these keywords are representing a particular topic better. The topic label is then selected by taking the keyword that has the highest probability to be suitable. Additionally, an artificial feedback mechanism is applied, where the user can modify the results, giving a certain label a higher weight. This information will be saved for later uses where this feedback will be applied again. In their evaluation, the algorithm shows an improvement in performance regarding the needs of the user over time. This is evaluated by asking the users about their agreement about the generated topic labels, which improves in subsequent iterations. Figure 3.1 displays the complete process.

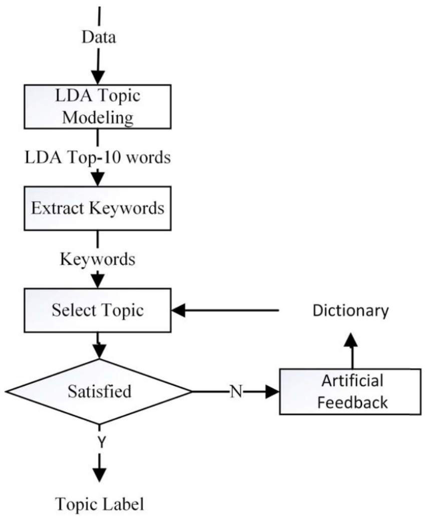


Fig. 3.1: Topic label extraction process by Tang et al. [40].

An approach that does not use LDA is presented by Wu et al. [43]. Their topic detection uses the user participation and non-negative matrix factorization (NMF). A $n \times m$ Matrix denotes that n discussions are participated by m users. Instead of term frequency-inverse document frequency ($tf - idf$) a procedure that uses participation frequency (pf) of users instead is used, which is then called participation frequency-inverse document frequency ($pf - idf$). After applying NMF, the resulting matrices indicate cluster membership. In addition, a word-discussion matrix is created, which can also be clustered with NMF. This will result in a list of words, which each represent a cluster. Both matrices are then combined and a joint-factorization is applied. In the end, each cluster from the first matrix corresponds to a word of the second matrix, indicating the topic of that cluster. Their experiments show useful results, and a purity measure shows that the joint factorization works better for clustering than a single matrix.
In [23], Liu et al. are presenting a semantics assisted NMF (SeaNMF) approach. In their paper, they are stating that LDA is not optimal for short, informal, and noisy texts that their dataset contains. Given their dataset, there are N posts and M unique words. Assuming K topics, a $M \times K$ matrix W and a $N \times K$ matrix H are created. A word-context matrix S is approximated by applying the Skip-Gram algorithm to W and W_c^T where W_c^T is the context matrix based on W. The resulting semantics are merged with the conventional NMF model and a Block Coordinate Descent algorithm is adopted for optimization. In their results, Lui et al. are reporting a higher topic coherence than LDA. Similar to LDA, topics are represented by topic keywords.
The work by Athira et al. [3] presents a multi-label semi-supervised topic classification using a support-vector-machine (SVM). For their approach, they collected a total of 11,000 posts from a breast cancer community forum, from which they manually labeled 1,000 posts. After a number of pre-processing steps, a vector representation of posts is created with the use of Doc2vec. The vectors give the posts a representation in space, which can then be used to create clusters. For that, the labeled posts are used as cluster centers and unlabeled posts are assigned to the nearest clusters using Euclidean distance. It is also stated that not every post may end up in a cluster, as they limited the proximity. These unlabeled posts can be used to find new labels that are not represented by the other labels. Afterwards, they select the most relevant labels for each post with fuzzy logic and a set of keywords that have been extracted of the manually labeled posts. Figure 3.2 shows the complete framework.

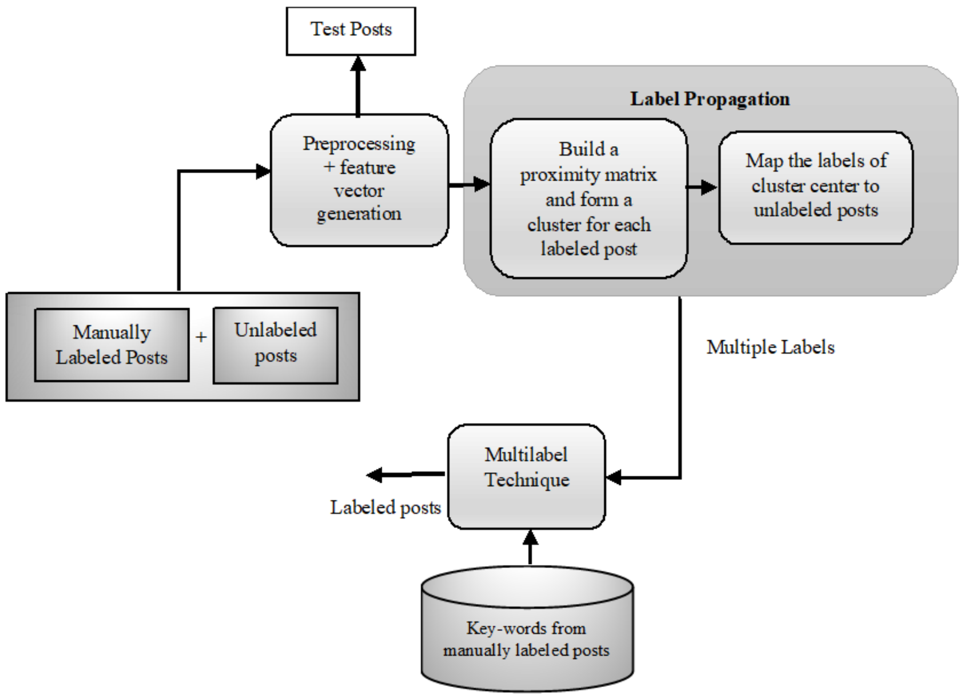

Test Posts
Label Propagation
Preprocessing + feature vector generation
Build a proximity matrix and form a cluster for each labeled post
Map the labels of cluster center to unlabeled posts
Manually Labeled Posts
+
Unlabeled posts
Multiple Labels
Multilabel Technique
Labeled posts
Key-words from manually labeled posts

Fig. 3.2: Multi-label Framework as proposed by Athira et al [3]

Zhang et al. [48] are presenting an improved approach based on $tf - idf$. In their context, $td - idf$ is used to measure the document similarity, which says that similar documents are about the same topic. It is not stated whether a document stands for a post or a thread in their context, although they are stating that those texts can be very short, which could mean that they are working with posts as documents. Their focus lies on the big variance of text size in forums, which can affect the traditional $tf - idf$ measure. To overcome this, they divide the words of the texts into five feature classes, from which they will generate feature weights. In contrast to previous feature selection strategies, they are selecting the top m weights for each class. As they state, m has to be selected carefully. If m is too big, there will be a difference for texts of different sizes and if m is too small, important information will be lost. They are reporting a performance improvement compared to the traditional $tf - idf$ measure, but are stating that this is still not enough for practical use.
Li et al. [21] are using a classical clustering approach using agglomerative hierarchical clustering (AHC). A document will be represented as a vector in the feature space. For calculating their feature weights, they are using term frequency of $tf - idf$ and information gain based on the principle of maximum entropy. Information gain $g(j, i)$ can be described as the difference of the empirical entropy $H(j)$ of a document j and the conditional entropy $H(j|i)$ of the document and a feature word i. The clustering algorithm then groups up posts into clusters based on a similarity measure of the

feature vectors. If the similarity between clusters is below the threshold, the algorithm will stop. The performance is measured with the precision and recall metrics and the authors are reporting an improvement on these metrics for their new weight calculation.
Another clustering approach is done by Chen et al. [8]. First, they are creating custom features. The features are differentiated by local and global features. Local features are term frequency and term integrity. Global features are inverse document frequency and burst, which tracks a topic appearance over a time span. The features are then used with single pass incremental clustering. Their evaluation is reporting a better performance than LDA and the topics can be described with the keyword that received the highest weight for a topic.
Reich et al. [31] propose the use of the structural topic model (STM) for forum analysis. The STM by Roberts et al. [32] is incorporating meta-data into the detection process. This way, meta-data is used for estimating topic prevalence and topic content. The authors state that this model explicitly models topic correlation, which LDA does not. The topic output is given with topic words that resemble a label and a number of topic words, like the LDA output. It is also stated that STM has better predictive and qualitative interpretability than LDA. Figure 3.3 shows the top ten topics of student responses about their motivation to register for an online learning course as an example and the top topic words.
Two different topic modeling metrics are mentioned in the literature. The perplexity metric is a statistical measure of how well a probabilistic model predicts a sample. It is used to determine the optimal number of topics. The lower the perplexity score, the better the number of topics is chosen. However, this metric does not guarantee that the topic words of a topic are interpretable by a human [7]. The topic coherence measure is better suitable to optimize for human interpretability. It measures the semantic similarity between the topic words of a single topic. A higher similarity results in a higher topic coherence score, which changes with the number of topics. The optimal number of topics for a dataset is found, when the topic coherence metric has the highest score.

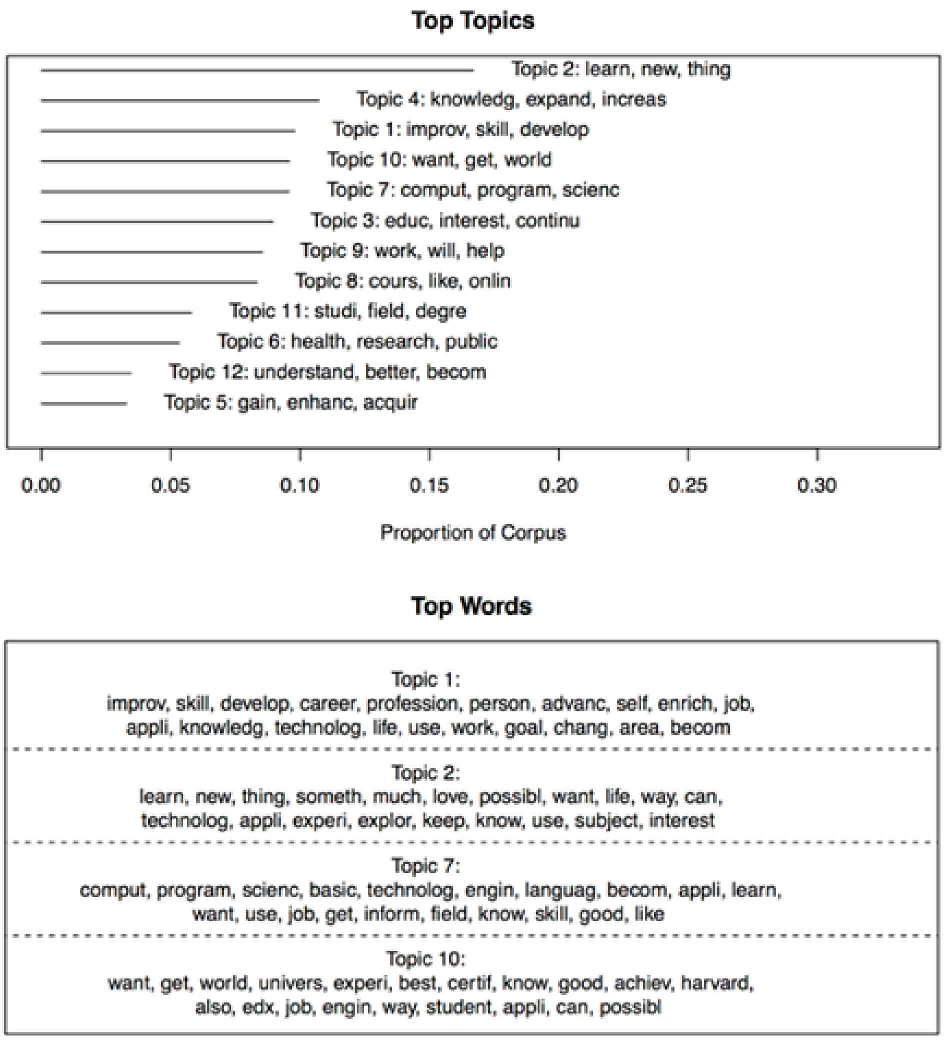


Fig. 3.3: Example Topics and Topic Words for STM [31].

3.4 Synthesis

In the synthesis, the found approaches are compared and the most suitable ones are selected for the implementation. For this, additional characteristics, which refine the research questions of Section 3.1, are taken into account. These additional characteristics are necessary, as the selected methods have to be implemented in this thesis and have to be applicable to concept detection. In the following, the characteristics are explained and extracted from the approaches. Afterwards, Table 3.11 at the end of the chapter is providing an overview of the synthesis.

Which method has been used in the approach?
The first characteristic is important for an overview of methods, especially since some approaches are using similar methods or are using the same base method. It is based on RQ1.1, but only takes the main processing part of topic modeling into account. Preprocessing methods are not considered. Table 3.6 gives an overview of the methods. LDA is the most used method and often mentioned by other approaches that use different methods.

Table 3.6: Approaches and their Methods

Ref	Method
[17]	LDA
[2]	LDA + naive Bayes
[40]	LDA + feedback mechanism
[9]	LDA modified with metadata
[14]	Modified SLDA
[43]	NMF
[23]	SeaNMF
[48]	Document similarity clustering
[8]	Single pass incremental clustering
[21]	AHC
[3]	SVM+doc2vec
[31]	STM
[41]	Topic model, classifier, not described in detail

Which additional data beyond documents is required for the approach?
This is an addition to RQ1 as it is not covered by the other sub-questions. For different datasets, not all information may be available for certain approaches. Approaches that are including more data generally report an improvement in performance. On the other hand, approaches that are using text only are more versatile for different applications. For the implementation of this thesis, it is not assumed that datasets only contain

documents from forums or have specific forum metadata available for analysis. This makes all approaches that use additional data unsuitable for implementation. Table 3.7 shows the approaches and the data that is needed.

Table 3.7: Approaches and the Additional Data needed

Ref	Data
[14], [17], [21], [23], [41]	Text only
[2]	MOOC course information
[40]	User feedback
[9], [43]	Post author
[48]	Post title (optional)
[8]	Date and time of post
[3]	Manually annotated posts
[31]	All metadata available (e.g. user age, gender, time)

What is the topic output format of this approach?

This characteristic is a refinement to RQ1.2 because not only the kind of topics are important, but also the way they are presented, as an automatic concept extraction is planned. Most methods return topic words, which are the words that are most likely to be part of that topic and thus are describing this topic. These topic words can be considered as concept words for the task of concept extraction in this thesis. Other approaches are just returning a cluster of posts, where the user then has to read posts of a cluster to determine the label, which is not useful as the output cannot be interpreted as concept words. Three approaches are assigning labels to the topics, but require user input. Table 3.8 is listing the approaches and their output.

Table 3.8: Approaches and their Output

Ref	Output
[8], [9], [17], [23], [31], [41], [48]	Topic words
[14]	Topic words and topic facet
[2], [40]	Topic words and proposed topic label
[3]	Posts with topic labels
[43]	Post cluster
[21]	Not described (should be a cluster)

How is the method performance compared to others?

This characteristic is a refinement of RQ1, as it is not covered by other sub-questions. As the approach is going to be implemented, it should have a state-of-the-art performance. When methods are compared to other methods, there could be a ranking that covers all methods. Table 3.9 is listing the approaches and their performance comparison. Some approaches are only comparing themselves to a baseline method, e.g. when they made some improvements to a method. Other approaches are only providing a qualitative evaluation that has been performed by humans, e.g. they looked at the topic word output and checked whether it is understandable. This makes it harder to compare them to other methods. A comparison by a metric (e.g. topic coherence) with a common dataset would make the different approaches more comparable. LDA is often used as a baseline, also for approaches that use other methods. There is no global ranking recognizable.

Table 3.9: Approaches and Performance Comparison

Ref	**Comparison**
[17]	LDA is very popular and a comparison for other approaches
[2]	Not compared to others
[40]	Better performance than LDA (qualitative)
[9]	Better performance than LDA (perplexity metric)
[14]	Better performance than LDA (qualitative)
[43]	Better performance than baseline, not compared to other methods
[23]	Better performance than LDA (topic coherence)
[48]	Better performance than baseline, not compared to other methods
[8]	Better performance than LDA (NMI metric)
[21]	Better performance than baseline
[3]	Only compared to other variants in this paper
[31]	Better interpretability than LDA (qualitative)
[41]	No evaluation given

Is there an implementation of the approach available (e.g. with open source libraries)?

This is a very practical characteristic. It is necessary as the approaches will be implemented as part of this thesis. A paper that is more detailed on their implementation could be preferable, since the reproduction will be easier. Table 3.10 shows what can be derived from the information given by the papers. For some approaches, there are only implementations of the baseline methods available, but not of their improvements. Some approaches are very detailed, offering information about the parameterization of the method, while others do not have any implementation details. This makes it hard to recreate these approaches. An implementation that is already tested is preferable, as it reduces the chance of being incorrect.

Table 3.10: Approaches and Implementation Details

Ref	Implementation
[17]	Many frameworks available, detailed parameters for usage
[2]	Base methods available, no implementation details
[40]	Base method available, label proposition and feedback mechanism not available
[9]	Base method available, modification not detailed enough for implementation
[14]	SLDA implementation available, no details on their modifications
[43]	NMF implementation available, no details on their modifications
[23]	SeaNMF implementation available
[48]	No implementation details
[8]	No implementation details
[21]	Base method available, no implementation details
[3]	Base method available, no implementation details
[31]	Framework available
[41]	No implementation details

Which approach should be selected for the implementation in this thesis?
For this question, the following criteria are specified, based on the aforementioned characteristics of the approaches:

- The approach only needs documents as data
- The output of the approach is in form of topic words
- There is an implementation of the method of the approach available

The performance characteristic is not used as criterion, as there is no global performance ranking recognizable.
Based on these criteria, at least one suitable method is selected. The approaches [2], [40], [9], [43], [48], [8], [3], [31] are not suitable, as they require additional data besides documents.
Of the remaining approaches ([14], [17], [21], [23], [41]) , [21] has not specified their output, while the others have at least topic words as an output. Only [17] (LDA) and [23] (SeaNMF) have an implementation of their methods in use available. [41] is not offering any implementation details and for [14], there is only the base method without their modifications available, which makes those two methods obsolete for use in this thesis.
Finally, there is [17] (LDA) and [23] (SeaNMF) remaining. LDA is widely used as a topic modeling method and is not only used in the context of user forums, but also for other social media platforms [27]. It only requires documents as data, which makes

it flexible and independent of special metadata (e.g. forum metadata). The output is in form of topic words, which can be used to label a topic. This output is sufficient, as the topic words can be used as concept words, when applied to concept detection. Although some approaches are achieving some form of better performance, mostly by incorporating additional data, which makes them less flexible, LDA is still a state-of-the-art topic modeling method. This explains its wide distribution and its many implementations that are available.

SeaNMF is an improvement over NMF, which is performing better than LDA on short and noisy texts. It uses the same data and output format as LDA, making it also very flexible. There is an implementation available that is also very detailed on the parameter use, which makes it easy to use in this thesis.

Both LDA and SeaNMF are being implemented as part of this thesis. To compare the methods' performance in topic modeling, the topic coherence metric is computed as it is done by Shi et al. [35]. It is not known whether the topic coherence metric can be applied for concept detection. This will be part of the evaluation in Chapter 7.

Table 3.11: Synthesis Overview

Ref	Method	Data	Output	Comparison	Implementation	Selected
[17]	LDA	Text only	Topic words	Popular method	Framework, very detailed	Yes
[2]	LDA + naive Bayes	MOOC course data	Topic words, label	Not compared to other methods	Base methods, no details	No
[40]	LDA + feedback mechanism	User feedback	Topic words, label	Better than LDA (qualitative)	Base method, no details	No
[9]	LDA modified with metadata	Post author	Topic words	Better than LDA (perplexity)	Base method, few details	No
[14]	Modified SLDA	Text only	Topic words, facet	Better than LDA (qualitative)	Base method, no details	No
[43]	NMF	Post Author	Post clusters	Not compared to other methods	Base method, no details	No
[23]	SeaNMF	Text only	Topic words	Better than LDA (topic coherence)	Available	Yes
[48]	Document similarity clustering	Post title	Topic words	Not compared to other methods	No details	No
[8]	Single pass incremental clustering	Date & time of post	Topic words	Better than LDA (NMI)	No details	No
[21]	AHC	Text only	not described	Not compared to other methods	Base method, no details	No
[3]	SVM+doc2vec	Annotated posts	Posts with labels	Not compared to other methods	Base method, no details	No
[31]	STM	All metadata	Topic words	Better than LDA (qualitative)	Framework	No
[41]	No details	Text only	Topic words	No evaluation given	No details	No

3.5 Review Summary

This chapter summarizes the reviewed works based on the research questions.

RQ1: Which approaches for fully or semi-automatic topic modeling of natural language user statements in online forums exist and what are their characteristics?

RQ1.1: Which machine learning or natural language processing methods are used by the approaches for which steps of the classification process?
The approaches are either unsupervised clustering methods [8], [22], [23], [43], [48] or topic models [2], [9], [14], [17], [40], [41], which means, that a topic cluster needs to be labeled manually. Some approaches try to generate topic labels, but they require additional input in form of feedback [40]. [3] is using a SVM classifier, which requires some annotated posts. Vitasek et al. [41] are not explicitly mentioning the methods they used for topic modeling, although they are using the Mallet framework, which is used by [17] for LDA. LDA is a very popular topic modeling method used by many approaches. These are extending the method to improve its performance [2], [9], [14], [40], or using it in a practical approach for forum analysis [17]. An overview of the methods can be seen in Table 3.6. Additionally, [3] is using doc2vec for their document representation and [2] is using naive Bayes to classify their posts and [23] uses Skip-Gram for word-level representation, all three methods are applied before the main topic detection method.

RQ1.2: What topics are identified by these methods?
The LDA-based approaches are discovering latent topics in the available data. The topics are in most cases presented as topic words, that require manual labeling, which means that the topic is interpreted by the user. Since the topic words are part of the forum texts, the topics can be arbitrary, based on the contents discussed in the forum. Although it is possible to classify forum posts beforehand into content/non-content posts, which will then result in content-related topics and non-content (e.g. organizational) topics [2]. The artificial feedback mechanism of [40] will adjust proposed topic labels to the preferences of the user. The topics seem to be in the forum context, e.g. a digital camera forum will yield camera-related topics [9]. Except for [8], where they mention to only detect hot topics, there is no restriction in topics mentioned.

RQ1.3: What problems arise when using data from online forums (e.g., domain dependency, short texts, syntactical errors, etc.)?
The reviewed papers are not talking in much detail about problems with forum data. Forum data is described as having an imprecise, terse and conversational style of writing [43]; being of varying length from very short to long texts [48]. [21] reports sparse and short forum texts, [23] short, informal and noisy forum posts and [2] the brevity of posts. [9] states that their dataset contains a lot of abbreviated words, misspelled words and unorganized contents. In most cases, this is described as the nature of forum data, which is not further addressed.

RQ1.4: How do the approaches deal with these problems?
Not all papers are mentioning that pre-processing steps are done because of the issues mentioned in RQ1.3, they are describing them as part of their procedure. [31] is doing stop word removal and stemming as pre-processing. [17] is doing the same, but since they are working with a software forum, they are also removing code snippets from forum texts. [14] parsed out program codes, but did not mention additional steps. [40] is removing emoticons and other nonsense syllables. Three papers are dealing with short texts: [2] is merging posts into threads. [48] is specifically developed to deal with forum texts of varying size. [23] is using word-level information additionally to post-level information, because of short texts.

4 Requirements

This chapter contains all requirements for the created software tool. Section 4.1 describes the initial coarse requirements from which features were derived. Section 4.2 depicts a typical tool user persona. In Section 4.3 the domain data for this thesis is introduced. Next, Section 4.4 describes the functional requirements, which include user tasks, subtasks and system functions. Afterwards, the relevant non-functional requirements are described in Section 4.5. Section 4.6 introduces the workspaces that are part of the tool, followed by the mock-ups in Section 4.7.

4.1 Coarse Requirements

The implementation of Feed.UVL is based on the following coarse requirements, with the ultimate goal of providing support to the researcher during the analysis of natural language. The functionalities derived from the coarse requirements are divided logically into different workspaces, which focus on certain tasks of the researcher.

R1: Feed.UVL allows different methods to be selected for the analysis of natural language. These methods are suitably parameterized. A given data set is used to perform analysis with the help of several methods.

R2: Feed.UVL accepts at least two different file formats for the input of data (csv, xlsx, etc.) and supports the pre-processing of this data according to the requirements of the chosen methods. Data consists of user statements and can be manually fed in via Feed.UVL's user interface. Input data contains the ground truth and the data to be analyzed itself.

R3: The results generated by Feed.UVL are visualized for easy analysis. On the one hand, basic statistics such as precision and recall of the method are displayed, on the other hand, the results can be compared in detail with the previously defined ground truth. In addition, Feed.UVL supports the analysis of false positives or false negatives. The visualization of the data takes place via the individual widgets used by Feed.UVL.

R4: The interface of Feed.UVL is easy to use. Within Feed.UVL, the existing data formats and infrastructures of feed.ai are retained.

R5: The integration of new methods or visualizations should be easy to implement within Feed.UVL. In particular, Feed.UVL should not be bound to a single analysis target, but remain universally usable for different targets.

R6: Feed.UVL offers methods for fully or semi-automatic topic modeling on individual statements. Appropriate methods are found through the literature review and implemented in Feed.UVL.

4.2 Personae

The main user of the software tool is a researcher as depicted in Table 4.1 who analyzes natural language user statements for language features like concepts to get information about the user's view on software. The tools functionalities focus on the needs of the researcher, which should increase the efficiency of their tasks. At the same time, the tool should not be confusing or complex, which may decrease the tool usability for the researcher.

Table 4.1: Persona: Researcher

Field	**Value**
Job	Researcher
Biography	Age 34, doctoral degree in Computer Science, currently researching the user view on software applications. Has previously finished other research projects regarding requirements from software users.
Knowledge	Experience with using evaluation software for natural language texts and using coding tools
Needs	A tool for concept analysis of natural language user statements that is easy to use, configurable and error-free.
Frustrations	Software crashes or has unexpected errors. The data displayed by a software application does not match the actual data. Confusing or unclear software.
Ideal Features	Result visualization, which gives an overview of detected concepts. Evaluation metrics to check performance of applied methods with a ground truth. Compare detected concepts to a ground truth. Clarity about runs and method micro-services. There should be a status indicator. There should be an overview over results. Datasets should be manageable inside the application.

4.3 Domain Data

The domain data for Feed.UVL is depicted in Figure 4.1. A result is based on one method applied on one dataset. A dataset contains at least one document and can have an optional ground truth, which consists of at least one truth element.

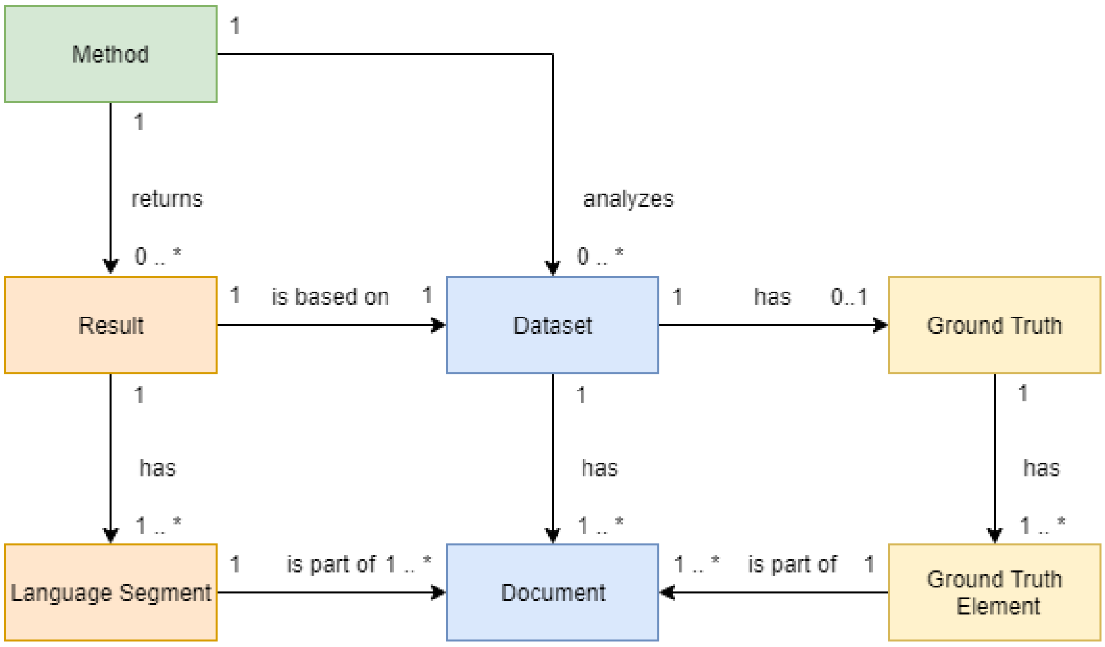


Fig. 4.1: Domain Data Model for Feed.UVL

4.4 Functional Requirements

The functionalities of Feed.UVL are described in this section. The researcher, as defined in Section 4.2 has one central user task, which contains several subtasks, as described in Section 4.4.1. From those subtasks, the system functions are derived and listed in Section 4.4.2.

4.4.1 User Tasks and Subtasks

UT1: Analyze Natural Language User Statements
The researcher analyzes user statements with different kinds of methods. Metrics and visualization techniques are used to evaluate those methods and draw conclusions from the results. The researcher applies different methods for different tasks and needs to be able to select the correct one for each task. The datasets that are analyzed as well as any analysis result need to be stored and managed.

ST1.1: Manage Datasets
The researcher needs to have an overview over existing datasets. A dataset should be storable within the application. The contents of a dataset have to be reviewed to decide whether the dataset is still needed. If the dataset is no longer needed, the researcher can remove it or in case there is a fault within the data, the dataset can be updated. For some datasets, there may be a ground truth that is used for evaluation. This ground truth needs to be stored with the dataset it belongs to. Ground truth data should also be reviewable when reviewing the dataset.

ST1.2: Analyze Datasets with different Methods
The researcher selects a method to analyze a specific dataset. Methods that have parameters are configured accordingly. Afterwards, the analysis run is started.

ST1.3: Analyze Results
The researcher takes an analysis result and uses appropriate visualization methods. The results can be made visible in the dataset directly. Also, metrics suitable to the method are applied to check the performance. The applied parameters are also consulted again. Afterwards, the researcher draws conclusions from the results.

ST1.4: Manage Results
After analyzing a result, the researcher may delete it because it is obsolete, or assign a name to it to make it easier to recognize again. The researcher may filter stored results by method to look for specific ones.

4.4.2 System Functions

The system functions (SF) are derived from the user subtasks and describe the tool features from a system point of view. Each system function is fully described with their conditions, inputs, outputs, exceptions and rules along with the subtasks it supports. A user story is added to every system function to underline the role and motivation of the user.

SF1: Navigate between workspaces

As a researcher, I want to be able to navigate between the views of the software, so I can accomplish my tasks. It has to be clear, in which part of the software I am currently.

Table 4.2: SF: Navigate between workspaces

Field	**Value**
Preconditions	None
Input	W1: Navigation View, one of W2-W6
Postconditions	None
Output	W1, one of W2-W6
Exceptions	None
Rules	(R1) Theme matches workspace domain
Supports	Subtask 1.1, 1.3

SF2: Upload dataset

As a researcher, I want to add my datasets into the application. Usually, my datasets are xlsx or csv files.

Table 4.3: SF: Upload dataset

Field	Value
Preconditions	File f exists
Input	W4: Upload Dataset View, f
Postconditions	Data in f is added to storage and is ready to use
Output	W4, tooltip displayed
Exceptions	(E1) f is corrupt, is too large, IO Exception occurs, or other error occurs
	(E2) f is of the wrong type or format
	(E3) User cancels operation
Rules	(R1) Allowed file types are: xlsx, csv, txt
	(R2) The dataset will be saved by its filename without ending.
Supports	Subtask 1.1

SF3: Show dataset

As a researcher, I want to view my dataset in the application, so I do not have to check the files manually.

Table 4.4: SF: Show dataset

Field	Value
Preconditions	Dataset d exists in storage
Input	W4: Upload Dataset View or W6: Dataset View, d
Postconditions	None
Output	W6, documents of d are shown
Exceptions	None
Rules	None
Supports	Subtask 1.1

SF4: Filter dataset contents

As a researcher, I want to be able to search my dataset for a specific word, this word should be highlighted in the containing documents.

Table 4.5: SF: Filter dataset contents

Field	**Value**
Preconditions	Result r with dataset d exists
Input	W5: Document View, r selected, filter string (optional)
Postconditions	None
Output	W5, list of documents which contain the filter string
Exceptions	None
Rules	(R1) Order of documents is kept
Supports	Subtask 1.1

SF5: Delete dataset

As a researcher, I want to remove my old dataset from the application, so it does not get messy.

Table 4.6: SF: Delete dataset

Field	**Value**
Preconditions	Dataset d exists in storage
Input	W4: Upload Dataset View or W6: Dataset View, d
Postconditions	Dataset d removed from storage
Output	W4 or W6, tooltip showing message about deletion or failure status
Exceptions	(E1) User cancels operation
Rules	None
Supports	Subtask 1.1

SF6: Upload ground truth

As a researcher, I want to add a ground truth to a dataset in the application. This helps with managing the data.

Table 4.7: SF: Upload ground truth

Field	Value
Preconditions	File f exists, dataset d exists
Input	W4: Upload Dataset View, f
Postconditions	Data in f is added to d in storage
Output	W4, tooltip message displayed
Exceptions	(E1) f is corrupt, is too large, IO Exception occurs, or other error occurs (E2) f is of the wrong type or format (E3) User cancels operation
Rules	(R1) Allowed file types are: xlsx, csv, txt
Supports	Subtask 1.1

SF7: Highlight ground truth

As a researcher, I want to be able to know, which segments of the documents of my dataset are part of the ground truth. With this, I can review and verify the data, or use it for further analysis.

Table 4.8: SF: Highlight ground truth

Field	Value
Preconditions	Dataset d with ground truth g exists in storage
Input	W6: Dataset View, d, g
Postconditions	None
Output	W6, g is highlighted in documents of d
Exceptions	None
Rules	None
Supports	Subtask 1.1

SF8: Start analysis run

As a researcher, I want to be able to perform analysis runs with some method on my datasets in the application.

Table 4.9: SF: Start analysis run

Field	Value
Preconditions	Dataset d exists, method m exists and is applicable to the dataset
Input	W2: Start Concept Detection View, d, a, parameters
Postconditions	Result r is stored in database
Output	W2, tooltip message displayed
Exceptions	(E1) Error with contacting backend
	(E2) Error while retrieving dataset from storage
	(E3) Error while parsing file or executing method
Rules	(R1) Output is dependend on method
	(R2) Some methods may have parameters
Supports	Subtask 1.2

SF9: Filter run results

As a researcher, I want to filter my results by method, so I can find and compare data from one method more easily.

Table 4.10: SF: Filter run results

Field	Value
Preconditions	Method m exists
Input	W3: Detection Results View or W5: Document View
Postconditions	None
Output	W3 or W5, m selected, only results of m are displayed
Exceptions	None
Rules	(R1) Only finished results are displayed
Supports	Subtask 1.4

SF10: Display run result

As a researcher, I want to see the results from an analysis run inside the application with appropriate visualization methods.

Table 4.11: SF: Display run result

Field	Value
Preconditions	A result r of method m is stored in storage
Input	W2: Start Concept Detection View or W3: Detection Results View, r
Postconditions	None
Output	W3, a visualization of r is shown in the UI
Exceptions	None
Rules	(R1) Visualization of r is based on m
Supports	Subtask 1.3

SF11: Match concepts with documents

As a researcher, I want to see all concepts of a result that belong to a certain document. This shows me the context of the concept.

Table 4.12: SF: Match concepts with documents

Field	Value
Preconditions	Result r with dataset d exists
Input	W5: Document View, r
Postconditions	None
Output	W5, concepts of r are listed beside documents of d
Exceptions	None
Rules	(R1) Only concepts that are in current document are shown besides
Supports	Subtask 1.3

SF12: Rename run result

As a researcher, I want to be able to give a name to a run result, so I can mark certain runs and I am able to find them again easier.

Table 4.13: SF: Rename run result

Field	Value
Preconditions	Result r exists in storage
Input	W2: Start Concept Detection View or W3: Detection Results View, r
Postconditions	Name of r is changed in storage
Output	Tooltip alterting user about success or failure, UI displays new name
Exceptions	(E1) User cancels operation
Rules	None
Supports	Subtask 1.4

SF13: Download run result

As a researcher, I want to extract the raw result data from the application, so that I can perform other analyses on my own outside of the software, or that I can share the data with other researchers.

Table 4.14: SF: Download run result

Field	Value
Preconditions	Result r exists in storage
Input	W3: Detection Results View, r
Postconditions	None
Output	r is offered as download
Exceptions	None
Rules	(R1) The file to download is a json file
	(R2) Only finished run results can be downloaded
Supports	Subtask 1.4

SF14: Delete run result

As a researcher, I want to remove old and irrelevant run results.

Table 4.15: SF: Delete run result

Field	**Value**
Preconditions	Result r exists in storage
Input	W2: Start Concept Detection View or W3: Detection Results View, r
Postconditions	Result r removed from storage
Output	Tooltip alerting user to deletion or failure status
Exceptions	(E1) User cancels operation
Rules	None
Supports	Subtask 1.4

4.5 Non-Functional Requirements

Non-functional requirements (NFRs) are system attributes that are not described in system functions. Those attributes can be system-related, like maintainability or user-related, like functionality. The ISO / IEC 25010 [15] quality model defines a standard for NFRs. In the model, eight different NFRs are defined, of which three were selected as most important for this thesis. Those NFRs are subdivided into more specific requirements. An important factor of NFRs is, that they have to be measurable, so that it is clearly possible to identify, whether they have been fulfilled. The three NFRs that have been selected are functionality, performance efficiency and maintainability, and are described in Sections 4.5.1, 4.5.2 and 4.5.3 respectively. The descriptions and the definition are taken from the standard[1], the metrics were defined to measure the fulfillment.

[1]ISO/IEC 25010 Standard: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 Last accessed: 27.08.2021

4.5.1 Functionality

This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.

Functional completeness

Definition: Degree to which the set of functions covers all the specified tasks and user objectives.

Metrics:

- **System function coverage.** The system functions are covering all the user tasks and subtasks. User tasks and subtasks are based on the coarse requirements. The System functions are derived from the subtasks, thus they can be traced back to them. Each subtask has at least one system function that supports it.

Functional correctness

Definition: Degree to which a product or system provides the correct results with the needed degree of precision.

Metrics:

- **Detection method code review.** Detection methods are implemented with reviewed code bases. The software packages used for the implementation can be tracked back to a code repository, which has indicators that the packages have passed some form of review.

- **System function correctness.** Every system functions' correctness is verified with system tests. If all system tests pass for a system function, it indicates its correctness.

4.5.2 Performance Efficiency

This characteristic represents the performance relative to the amount of resources used under stated conditions.

Time behavior

Definition: Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

Metrics:

- **View load time.** Views are loaded in less than 2 seconds. A stopwatch tool is used to track the time needed to load a view.
- **Result visualization time.** Result visualization takes less than 10 seconds for results where $\textit{documents} \times \textit{concepts} < 100,000$. This is tested with a big enough dataset, whose analysis result yields enough concepts. This result is then visualized and the time is measured with a stopwatch tool.
- **Computational complexity of methods.** Implemented detection methods have a maximal computational complexity of $O(n^2)$. The complexity of most methods is found in the literature. Otherwise, this can be verified mathematically.

4.5.3 Maintainability

This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. This characteristic is composed of the following sub-characteristics:

Modularity

Definition: Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Metrics:

- **Micro-service architecture.** New backend components are implemented using a micro-service architecture realized as docker containers. At least database access, analysis methods, and user interface service should be in different containers.

Modifiability

Definition: Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Metrics:

- **Micro-service resilience.** Stopping a method micro-service does not cause other micro-services to crash. This is tested by shutting down a method micro-service and performing all the system tests. Only system tests related to analysis runs are allowed to differ from expected behavior. An analysis micro-service can also be shut down during analysis to check if any other micro-service crashes.

- **Maintainability issues.** Code issues that affect maintainability are detected and resolved. An appropriate static analysis tool that can check code for maintainability issues is applied and the found issues are resolved afterwards. In the end, the tool detects no more maintainability issues.

Testability

Definition: Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Metrics:

- **System function behavior.** All system functions have an expected output. Section 4.4.2 shows the expected output for each system function, this can be used for testing.

- **System function coverage.** All criteria are tested with system tests. For each system function, there is one test that checks for the expected behavior.

- **Component test coverage.** A code coverage of at least 85% is achieved with component tests for all relevant components. A software tool is used to measure the code coverage that is achieved during the test execution.

4.6 Workspaces

Workspaces (W) define logical work areas of the software, which contain system functions. In this thesis, Feed.UVL introduces several workspaces. The navigation view (W1) is the central unit to navigate between the other workspaces. W2 depicts the view where a new concept detection run can be parameterized and started. In the future, it can be used not only for concept detection but also for other analysis targets. An analysis run result can be displayed in W3, which will contain proper visualization for concept detection results. W4 is used to upload new datasets to the software as well as to upload a ground truth for a dataset. Next, W5 is also related to run results and provides the function to match detected concepts with documents. Lastly, the dataset view (W6) can display the documents of a dataset and highlight the ground truth in the documents, if available. The workspaces and their connections are shown in Figure 4.2.

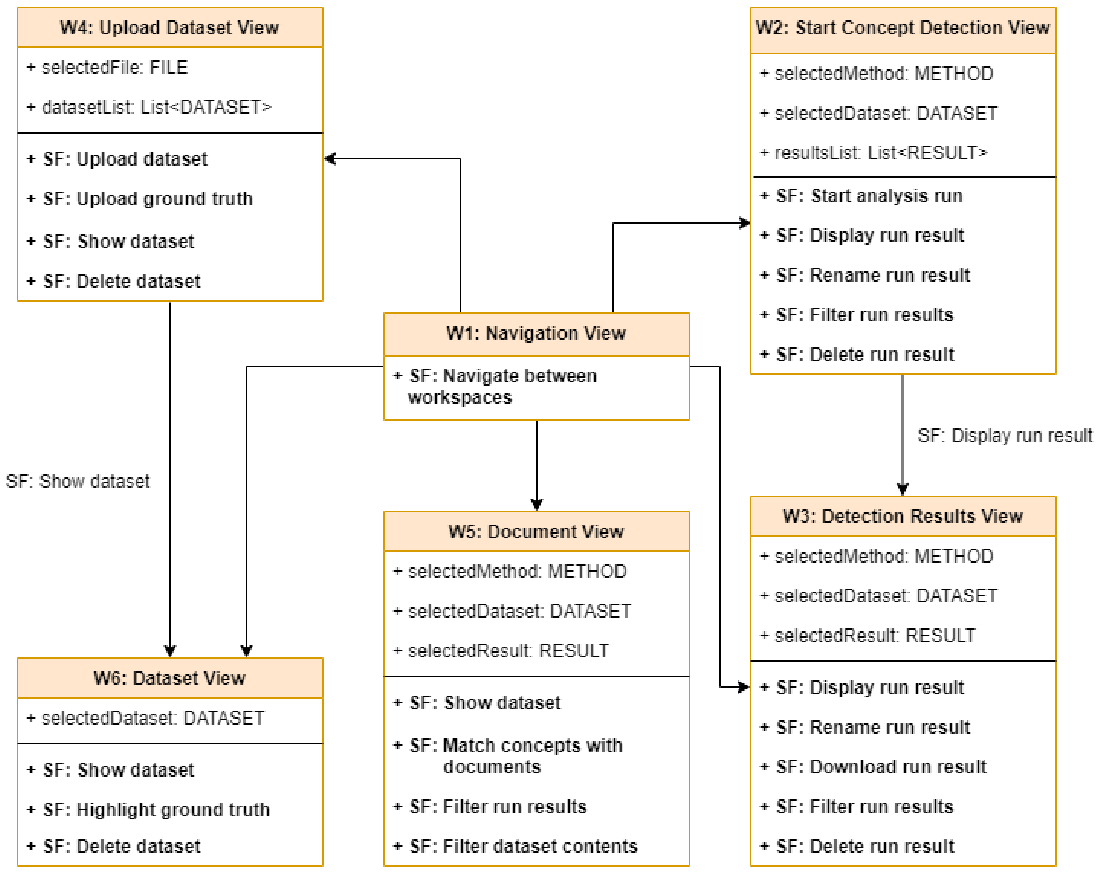


Fig. 4.2: UI-Structure Diagram of all Feed.UVL Workspaces

4.7 Mock-ups

The mock-ups introduced in this section are based on the previously described workspaces (Section 4.6). Note that the mock-ups are only a first realization idea, which changed during the implementation process.

Figure 4.3 shows the mock-up for the navigation view. The top part shows the navigation elements of feed.ai. A horizontal line is introduced as separator for the navigation elements of Feed.UVL, which are at the bottom part.

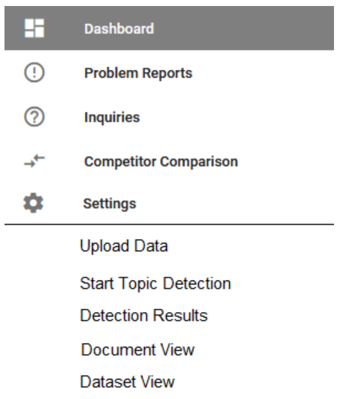
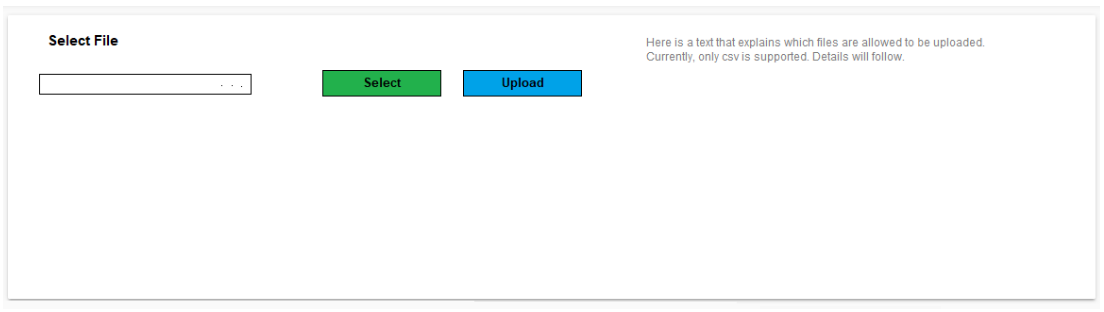


Fig. 4.3: Navigation Mock-up

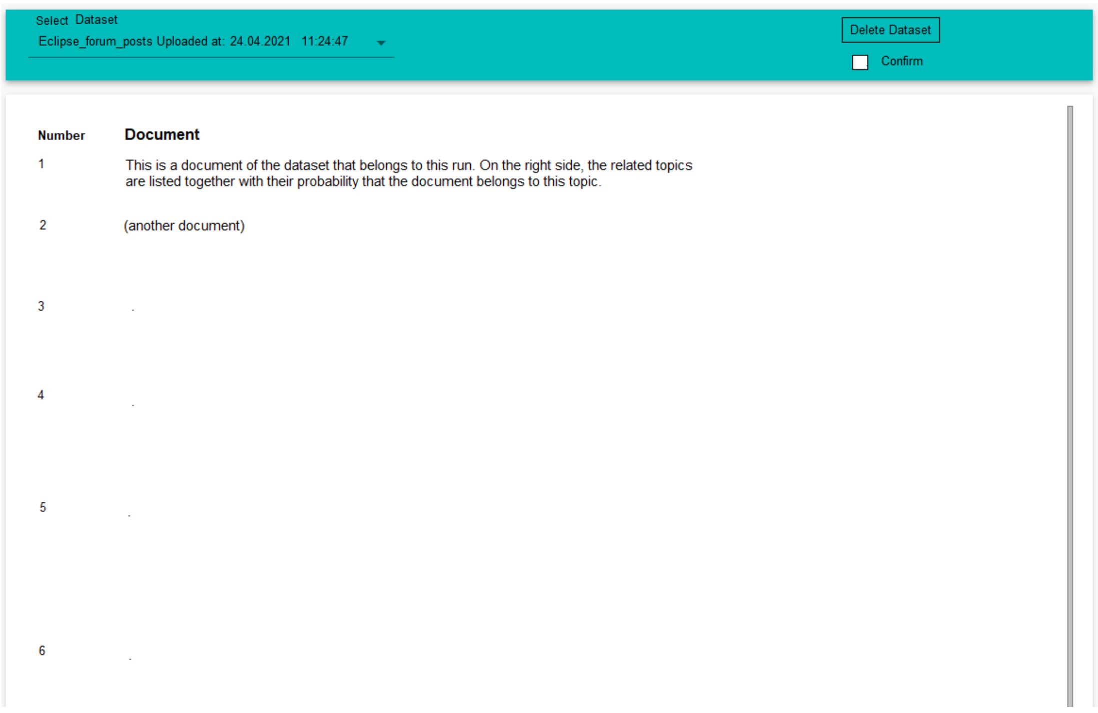
In Figure 4.4, the upload dataset view (W4) is shown. It mainly contains form elements for a file upload, which will be the dataset file and hints about the file format.



Here is a text that explains which files are allowed to be uploaded. Currently, only csv is supported. Details will follow.

Fig. 4.4: File Upload Mock-up

Next, Figure 4.5 displays the dataset view (W6). A dataset can be selected via a drop-down and the documents of the dataset will be listed as a table. There is also a button for the deletion of the dataset and a confirmation checkbox for security.



Number	Document
1	This is a document of the dataset that belongs to this run. On the right side, the related topics are listed together with their probability that the document belongs to this topic.
2	(another document)
3	.
4	.
5	.
6	.

Fig. 4.5: Dataset View Mock-up

Figure 4.6 shows the start concept detection view (W2). The method and dataset can be selected via a dropdown. Parameter input fields are shown according to the selected method. The green button can be pressed to start the detection run. Below, the last runs are listed in form of a table. This helps to get an overview of past runs without switching the view.

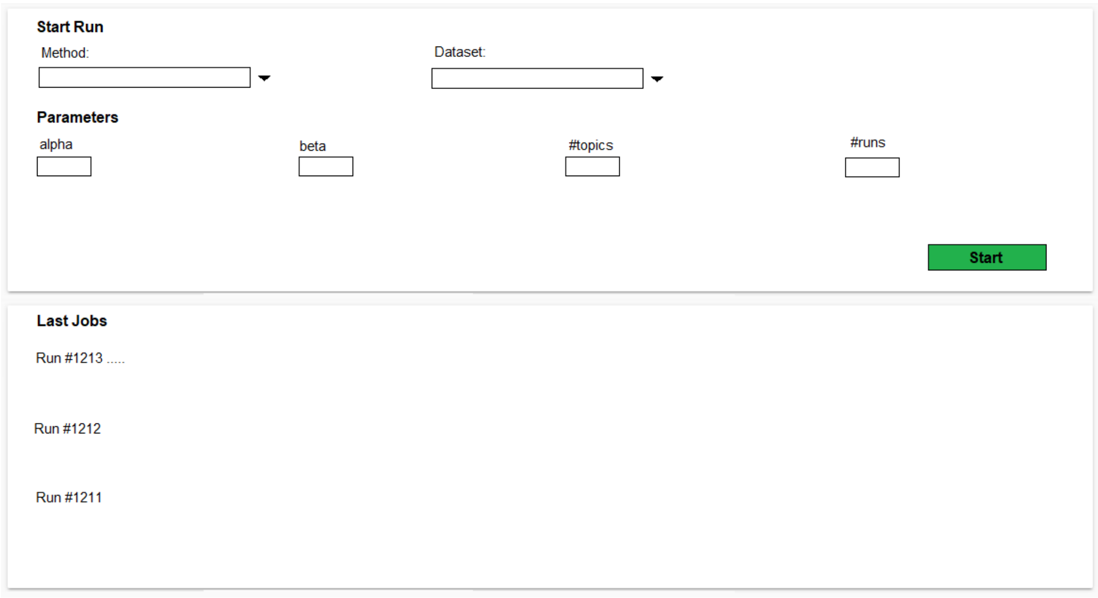
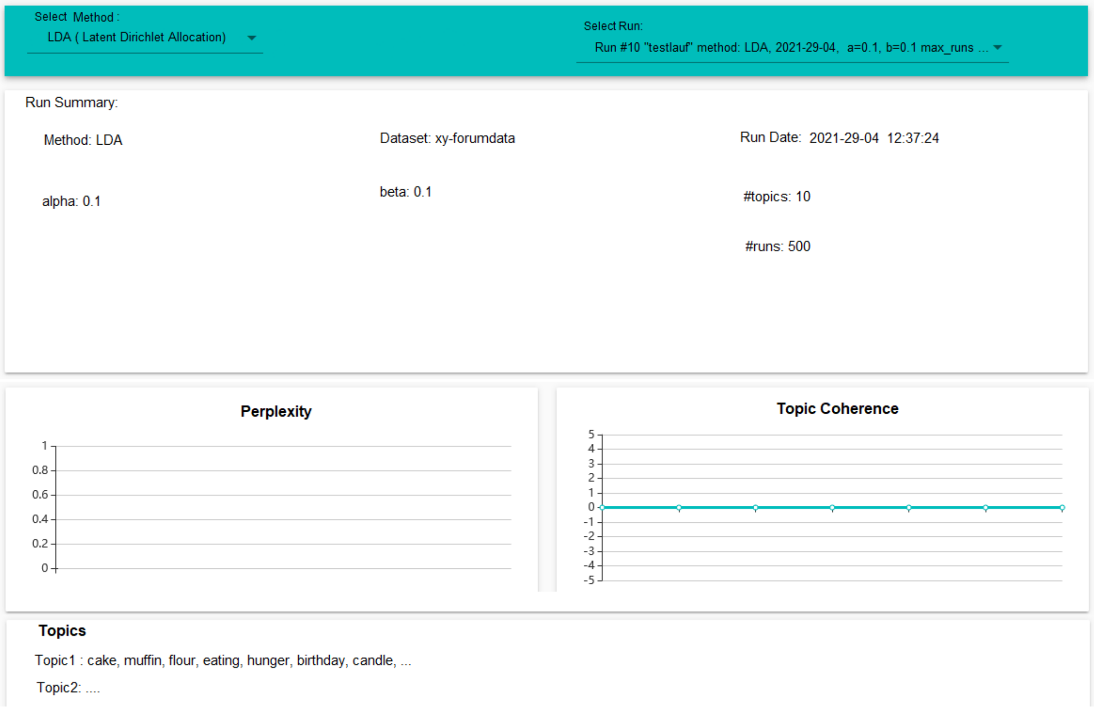


Fig. 4.6: Start Detection Mock-up

In Figure 4.7, the mock-up for the detection results view is shown. In the top left corner, a method can be selected, which will filter the finished runs for the selected method ind the dropdown in the top right corner. Below, the parameters of the selected run are shown, followed by example visualizations of the methods metrics and detection output.

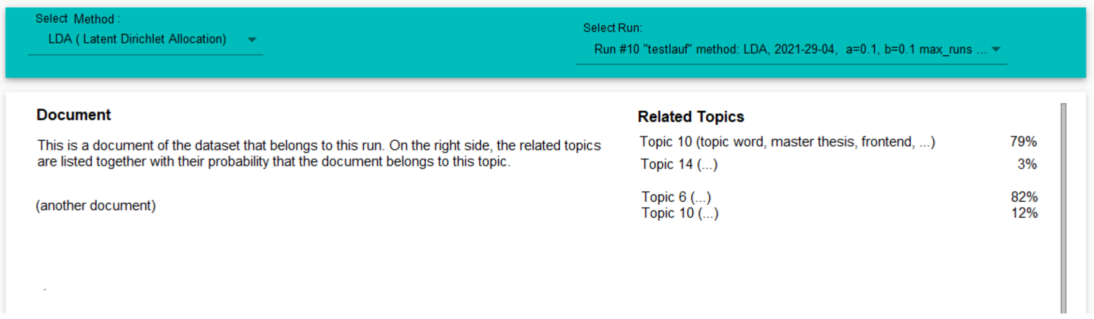

Select Method :
LDA (Latent Dirichlet Allocation)
Select Run:
Run #10 "testlauf" method: LDA, 2021-29-04, a=0.1, b=0.1 max_runs ...

Run Summary:
Method: LDA
Dataset: xy-forumdata
Run Date: 2021-29-04 12:37:24
alpha: 0.1
beta: 0.1
#topics: 10
#runs: 500

Topics
Topic1 : cake, muffin, flour, eating, hunger, birthday, candle, ...
Topic2: ....

Fig. 4.7: Detection Result Mock-up

Lastly, Figure 4.8 shows the mock-up for the document view (W5). The top dropdowns work the same as for Figure 4.7. Below, the documents of the dataset of the selected run are displayed along with the detection result of this document.


Select Method :
LDA (Latent Dirichlet Allocation)
Select Run:
Run #10 "testlauf" method: LDA, 2021-29-04, a=0.1, b=0.1 max_runs ...

Document
This is a document of the dataset that belongs to this run. On the right side, the related topics are listed together with their probability that the document belongs to this topic.
(another document)

Related Topics
Topic 10 (topic word, master thesis, frontend, ...) 79%
Topic 14 (...) 3%
Topic 6 (...) 82%
Topic 10 (...) 12%

Fig. 4.8: Document-centered Result Mock-up

5 Design and Implementation

In this chapter, the design and implementation are described in detail. Firstly, Section 5.1 explains the micro-service architecture that is deployed. Next, Section 5.2 describes the data classes that are based on the domain data. Finally, Sections 5.3 and 5.4 specify the backend and frontend micro-services respectively. Service and micro-service are used synonymously. The class diagrams of specific services show their repository name in brackets. The implementation is completely open-source and can be found on GitHub[1].

5.1 Micro-Service Architecture

As it is defined in the NFRs in Section 4.5.3, a micro-service architecture is used. The data visualization micro-service of feed.ai is kept and extended to new views for Feed.UVL. An orchestration micro-service handles the data upload and manages the analysis runs. The analyses are done by method micro-services in the data analytics layer. There is one service for LDA and one for SeaNMF. The data is stored in a MongoDB[2] which is only accessed by the storage micro-service. The whole structure can be seen in Figure 5.1. The data collection layer that is present in feed.ai (Figure 2.4) is not adopted, as there is no data collection service needed. Datasets are added by the user via the data visualization service, processed by the orchestrator and stored via the storage service.
Because each micro-service is separated from the others, their implementations can be completely different. Storage and orchestrator are implemented in the *Go* programming language, which is adopted from the corresponding services of feed.ai, while the method micro-services are using *Python*, because there are implementations for the methods in *Python*. The visualization service is the same as for feed.ai and is implemented in *JavaScript* using the *VueJS Framework*[3].

[1]Feed.UVL GitHub: https://github.com/feeduvl Last accessed: 16.09.2021
[2]MongoDB: https://www.mongodb.com/ Last accessed: 14.09.2021
[3]VueJS Framework: https://vuejs.org/ Last accessed: 14.09.2021

Each micro-service is managed in its own code repository. When a new version is commited to a repository, Jenkins[4], a tool for continuous integration and deployment, builds the service and sends it to the Docker daemon (Section 2.3) for deployment. Docker then manages the services in containers. Those containers are connected in a network, so the services can communicate with each other through their APIs.

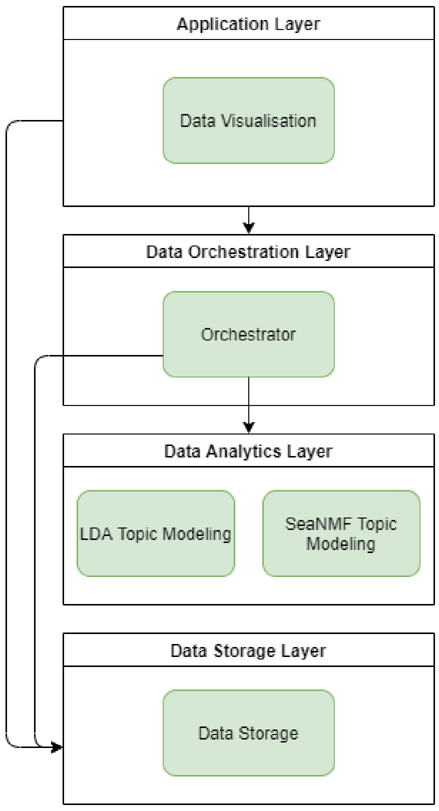


Fig. 5.1: Feed.UVL Micro-service Structure

[4]Jenkins: https://www.jenkins.io/ Last accessed: 14.09.2021

5.2 Data Classes

All micro-services use the same data classes, which were derived from the domain data in Section 4.3. Figure 5.2 shows these data classes. The classes *Dataset* and *Result* are used by all micro-services, except for their validate function, which is only needed in the storage service. The *Run* class is used in the communication between the orchestrator and the method services. It is an intermediate object that contains the method name, dataset and parameters and is used to start the analysis run. After the analysis has finished, the result is saved in the *Result* class. The implementation of these classes depend on the programming language of the micro-service.

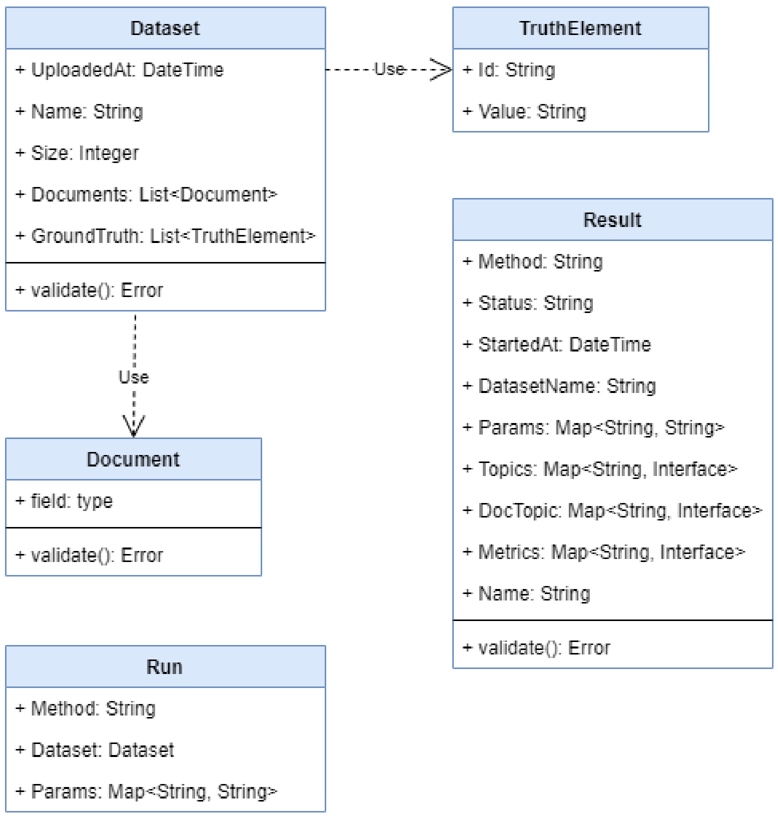


Fig. 5.2: Data Classes of the Micro-services (all repositories)

5.3 Backend Services

This section describes all the micro-services that are part of the backend. Sections 5.3.1 and Section 5.3.2 are describing the storage service and orchestration service respectively, followed by the LDA in Section 5.3.3 and SeaNMF in Section 5.3.4. Each service is depicted with a class diagram. Note that the *Go* programming language does not feature classes, which means that the class diagrams of the storage and orchestration service are abstracted from their actual structure.

5.3.1 Storage Service

This service is the connection between the MongoDB and other micro-services. The data can be managed via an API. Those API functions are handled by the router class as seen in Figure 5.3, which makes use of the *MongoHandler* class that defines the functions to access the MongoDB.

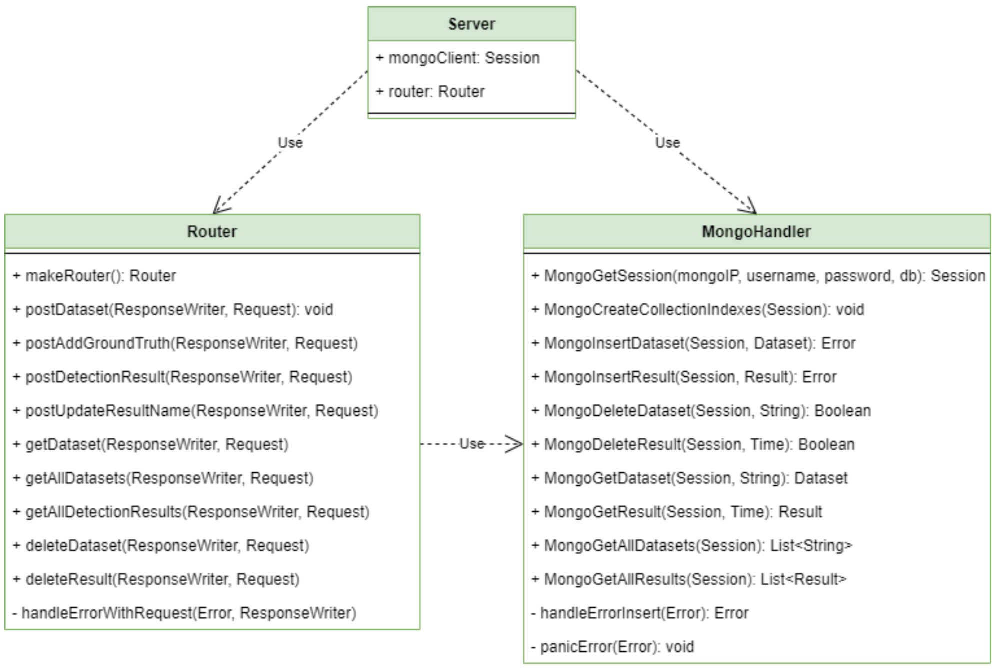

Server
+ mongoClient: Session
+ router: Router
Use
Use
Router
+ makeRouter(): Router
+ postDataset(ResponseWriter, Request): void
+ postAddGroundTruth(ResponseWriter, Request)
+ postDetectionResult(ResponseWriter, Request)
+ postUpdateResultName(ResponseWriter, Request)
+ getDataset(ResponseWriter, Request)
+ getAllDatasets(ResponseWriter, Request)
+ getAllDetectionResults(ResponseWriter, Request)
+ deleteDataset(ResponseWriter, Request)
+ deleteResult(ResponseWriter, Request)
- handleErrorWithRequest(Error, ResponseWriter)
Use
MongoHandler
+ MongoGetSession(mongoIP, username, password, db): Session
+ MongoCreateCollectionIndexes(Session): void
+ MongoInsertDataset(Session, Dataset): Error
+ MongoInsertResult(Session, Result): Error
+ MongoDeleteDataset(Session, String): Boolean
+ MongoDeleteResult(Session, Time): Boolean
+ MongoGetDataset(Session, String): Dataset
+ MongoGetResult(Session, Time): Result
+ MongoGetAllDatasets(Session): List<String>
+ MongoGetAllResults(Session): List<Result>
- handleErrorInsert(Error): Error
- panicError(Error): void

Fig. 5.3: Class Diagram for the Storage Service (uvl-storage-concepts)

5.3.2 Orchestration Service

The orchestration service has a similar structure as the storage. A router handles the API functions that are available. Instead of a database handler, there is a *RestHandler*, which manages outgoing API calls to other services. Figure 5.4 shows the class diagram for the orchestrator.

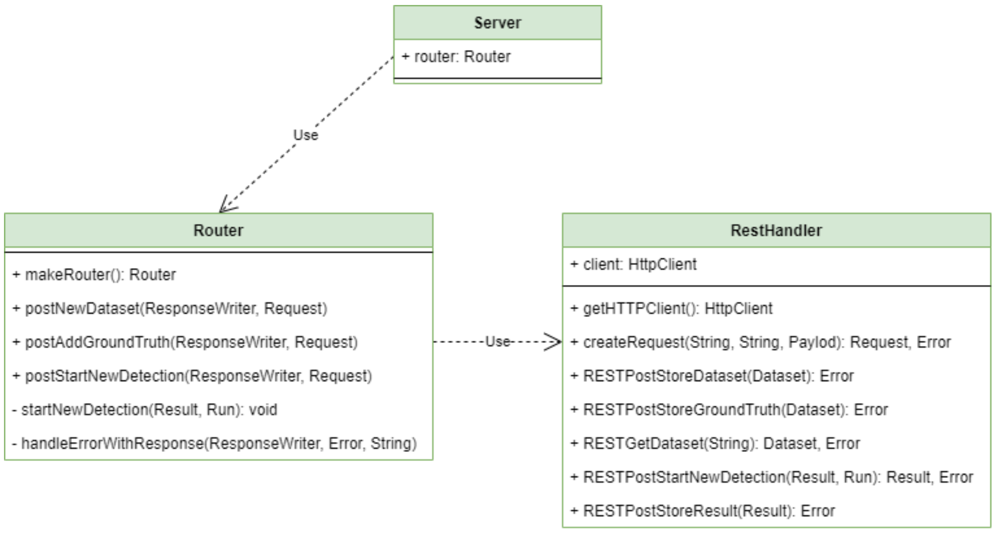

Server
+ router: Router
Use
Router
+ makeRouter(): Router
+ postNewDataset(ResponseWriter, Request)
+ postAddGroundTruth(ResponseWriter, Request)
+ postStartNewDetection(ResponseWriter, Request)
- startNewDetection(Result, Run): void
- handleErrorWithResponse(ResponseWriter, Error, String)
Use
RestHandler
+ client: HttpClient
+ getHTTPClient(): HttpClient
+ createRequest(String, String, Paylod): Request, Error
+ RESTPostStoreDataset(Dataset): Error
+ RESTPostStoreGroundTruth(Dataset): Error
+ RESTGetDataset(String): Dataset, Error
+ RESTPostStartNewDetection(Result, Run): Result, Error
+ RESTPostStoreResult(Result): Error

Fig. 5.4: Class Diagram for the Orchestration Service (uvl-orchestration-concepts)

5.3.3 LDA Method Service

This micro-service handles the analysis of datasets using LDA. The method is implemented using Gensim[5], a library for topic modeling. There is one API function for analysis and one that returns the status of the service. This is used by the visualization service to show the user whether the method service is online or not. Figure 5.5 shows the class diagram for this micro-service. The method parameters are listed and explained next:

- **chunksize**: Number of documents to be used in each training chunk. Default: 2,000
- **passes**: Number of passes through the corpus during training. Default: 1
- **iterations**: Maximum number of iterations through the corpus when inferring the topic distribution of a corpus. Default: 500

[5]Gensim Topic Modeling: https://radimrehurek.com/gensim/ Last accessed: 14.09.2021

- **n_topics**: The number of topics that shall be detected. Higher topic coherence indicates a better n_topics. Can be any number > 0. Default: 10

- **stemming**: Include stemming in preprocessing. Default: False

- **fix_random**: Set to *true* to fix random seed to 0. This will make the results reproducible. Default: *false*

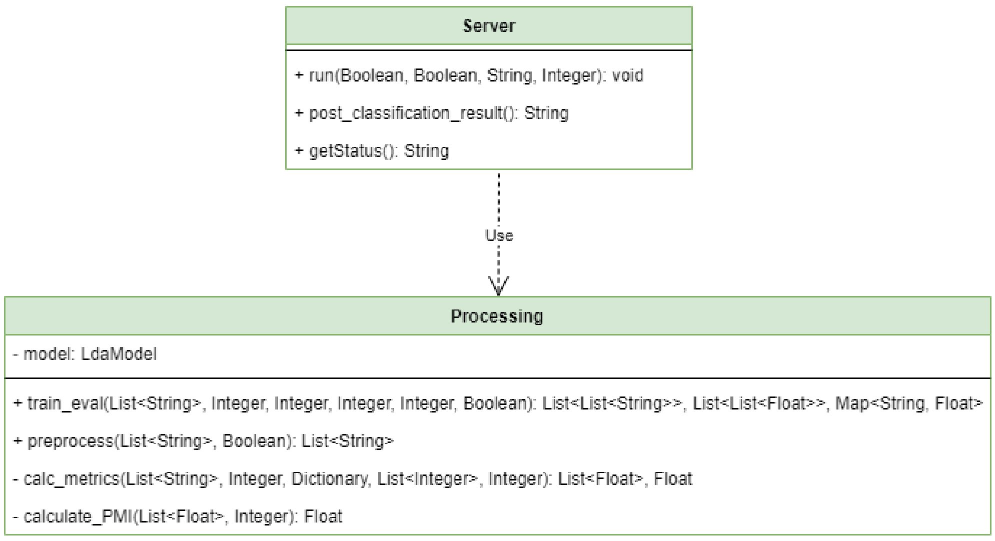

Server
+ run(Boolean, Boolean, String, Integer): void
+ post_classification_result(): String
+ getStatus(): String
Use
Processing
- model: LdaModel
+ train_eval(List<String>, Integer, Integer, Integer, Integer, Boolean): List<List<String>>, List<List<Float>>, Map<String, Float>
+ preprocess(List<String>, Boolean): List<String>
- calc_metrics(List<String>, Integer, Dictionary, List<Integer>, Integer): List<Float>, Float
- calculate_PMI(List<Float>, Integer): Float

Fig. 5.5: Class Diagram for the LDA Method Service (uvl-analytics-concepts-lda)

5.3.4 SeaNMF Method Service

The class structure of this micro-service is similar to the LDA method service. The method implementation is from the original method authors [35]. Figure 5.6 shows the class diagram for this micro-service. The parameters of the method are listed and described next:

- **alpha**: Factorization weight for word-semantic correlations, higher alpha may increase coherence, but reduce interpretability of topics. Can be in (0,1] range. Default: 0.1

- **beta**: Sparsity factor, increase beta (e.g. 0.1) for SparseSeaNMF (SSeaNMF), for normal SeaNMF this parameter is not needed (=0). Can be in [0,1] range. Default: 0

- **n_topics**: The number of topics that shall be detected. Higher topic coherence indicates a better *n_topics*. Can be any number > 0. Default: 10

- **max_iter**: Maximum number of iterations that will be performed. Default: 500
- **max_err**: Error threshold. The processing will stop when the error is smaller than *max_err* or *max_iter* is reached. Default: 0.1
- **fix_random**: Set to *true* to fix random seed to ‘0‘. This will make the results reproducible. Default: *false*
- **vocab_min_count**: Only words that occur at least *vocab_min_count* times will be added to vocabulary, if the dataset is small and the *vocab_min_count* to high, the processing will fail. Default: 3

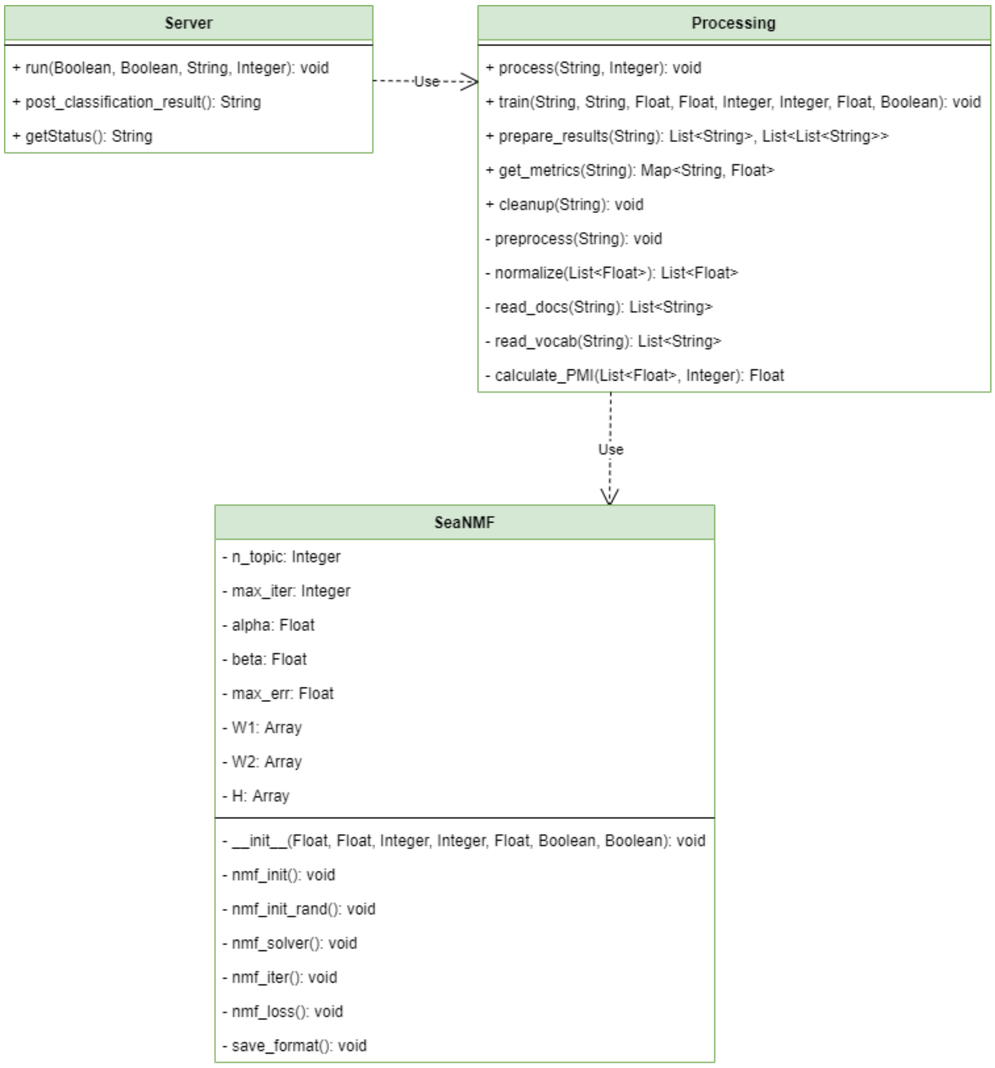

Server
+ run(Boolean, Boolean, String, Integer): void
+ post_classification_result(): String
+ getStatus(): String
Use
Processing
+ process(String, Integer): void
+ train(String, String, Float, Float, Integer, Integer, Float, Boolean): void
+ prepare_results(String): List<String>, List<List<String>>
+ get_metrics(String): Map<String, Float>
+ cleanup(String): void
- preprocess(String): void
- normalize(List<Float>): List<Float>
- read_docs(String): List<String>
- read_vocab(String): List<String>
- calculate_PMI(List<Float>, Integer): Float
Use
SeaNMF
- n_topic: Integer
- max_iter: Integer
- alpha: Float
- beta: Float
- max_err: Float
- W1: Array
- W2: Array
- H: Array
- __init__(Float, Float, Integer, Integer, Float, Boolean, Boolean): void
- nmf_init(): void
- nmf_init_rand(): void
- nmf_solver(): void
- nmf_iter(): void
- nmf_loss(): void
- save_format(): void

Fig. 5.6: Class Diagram for the SeaNMF Service (uvl-analytics-concepts-seanmf)

5.4 Frontend

The visualization micro-service is the frontend of the software tool, which means it is the part of the application visible to the tool user. The following sections are presenting the different views of the tool, with screenshots of the final result and class diagrams describing the class structure. Since a complete class diagram is too big to display, each class diagram only shows a part of the whole application. As the frontend is extending the implementation of feed.ai, only new classes are described. The implemented views are based on the mock-ups that were created (Section 4.7). However, there are differences in the final result, which will be mentioned.
As already stated, the frontend is implemented in JavaScript using the *VueJS Framework*. GUI elements are implemented with *Vue* components, which can be nested. This provides some form of modularity. One central part of the frontend is the *Vuex Storage*. It is a central storage class that can be accessed by *Vue* components. When the contained data changes, all components using this data are notified and can change their data display accordingly. The class diagram in Figure 5.7 shows the *Vuex Storage* and all the classes that are accessing it. A blue colored class is a data related class. Yellow classes are the main views, while orange classes are subcomponents. Grey classes are external components.
The tooltips mentioned in the following view descriptions are new components that were not present in feed.ai. It did not feature any kind of notification for the user. As it was needed for the implementation, it has been decided to use tooltips for displaying messages to the tool user. As an alternative, dialog messages have been considered, but it was decided against them because they can be very intrusive and disrupt a smooth workflow.

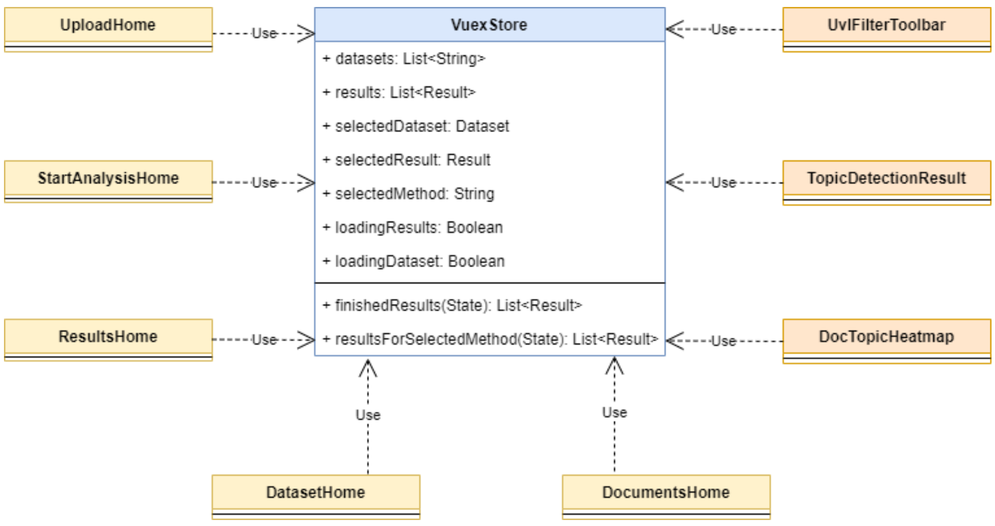


Fig. 5.7: Class Diagram for the Vuex Store (ri-visualization)

5.4.1 Navigation View

The navigation view implements workspace W1 (Figure 4.2) and is used to navigate between different views. It is displayed in Figure 5.8. There is a line to separate the navigation elements of Feed.UVL (bottom side) from the ones of feed.ai (top side). A headline was added to clearly indicate the domain of the navigation elements, which is a change compared to the mock-up in Figure 4.3. The class diagram related to the navigation view can be seen in Figure 5.9. The *TopNavigationDrawer* class is the navigation view. It holds the *ViewInfo* for each view it links to, which contains the display icon, name and theme. The displayed view is changed by manipulating the *Routes* class, which contains a *Route* class that loads the corresponding *Vue* component for each view.

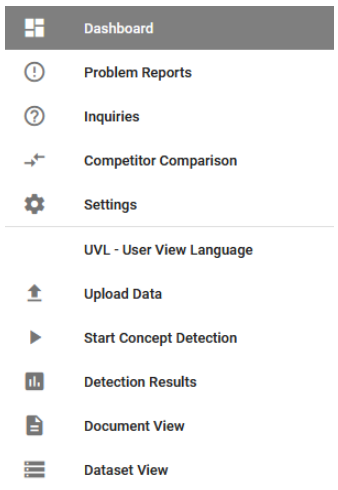


Fig. 5.8: Navigation View

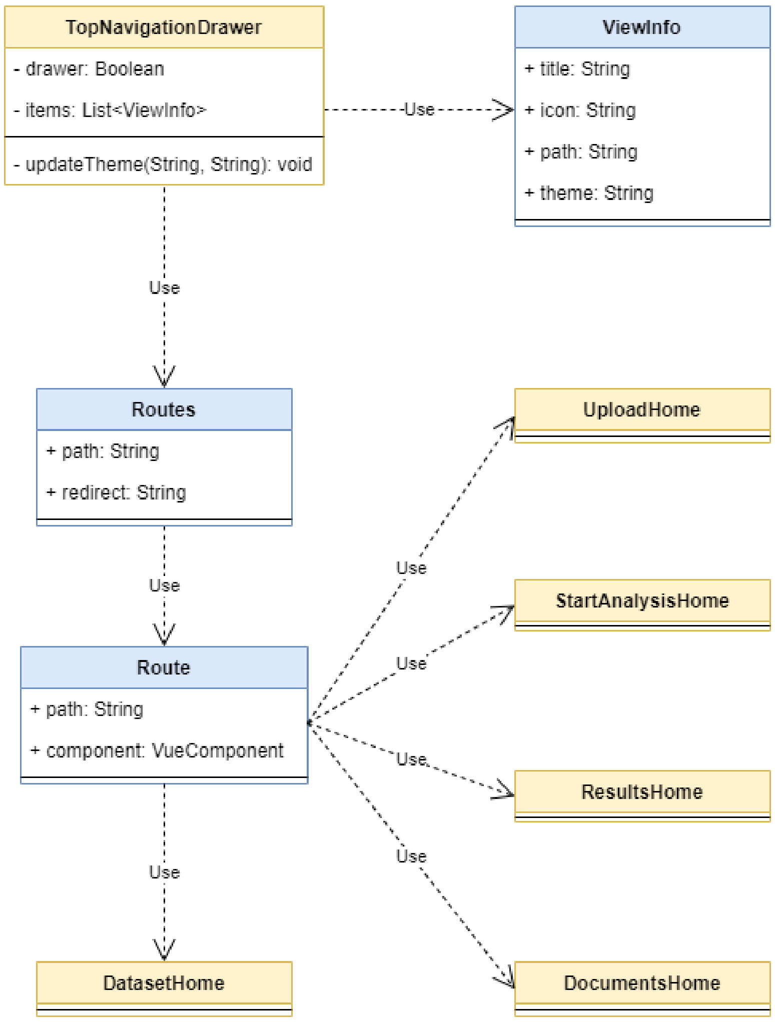
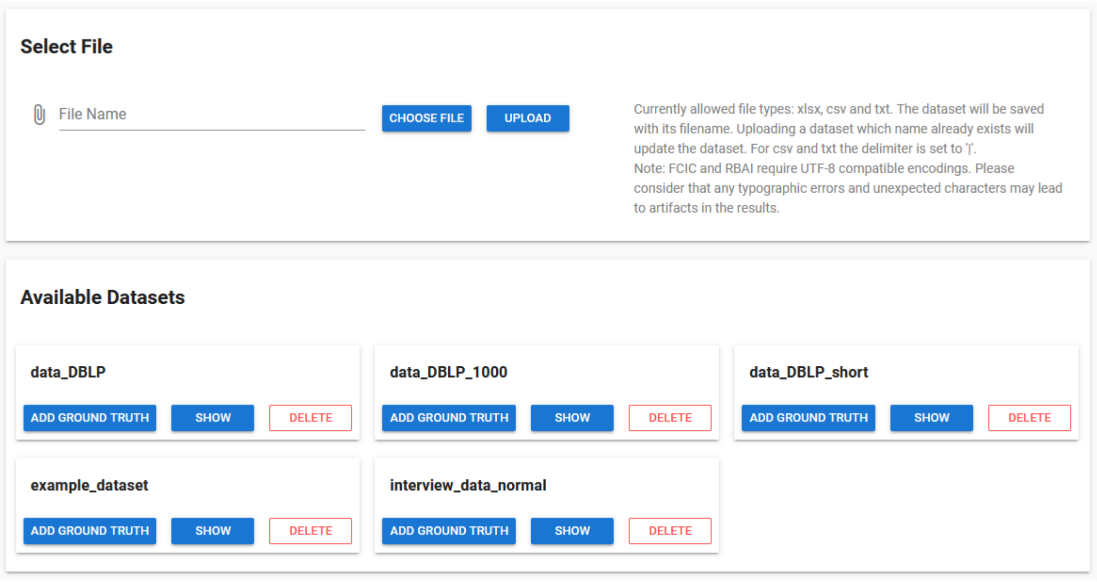


Fig. 5.9: Class Diagram for the Navigation View (ri-visualization)

5.4.2 Upload Dataset View

In this view, the user can upload datasets and manage them as planned in W4 (Figure 4.2). Figure 5.10 shows the upload dataset view. The top part is used to upload a dataset. The dataset file can be selected and uploaded via two buttons. There is a text that describes the allowed file types and formats. It is possible to upload xlsx, csv and txt files as specified in SF2: Upload dataset (Section 4.4.2). After an upload, a tooltip will be displayed to inform the user about success or failure. On the bottom part, the uploaded datasets can be seen. There are buttons for each dataset that allow further actions (add a ground truth, show dataset, delete dataset) to be performed. This bottom section was not yet part of the mock-up in Figure 4.4.

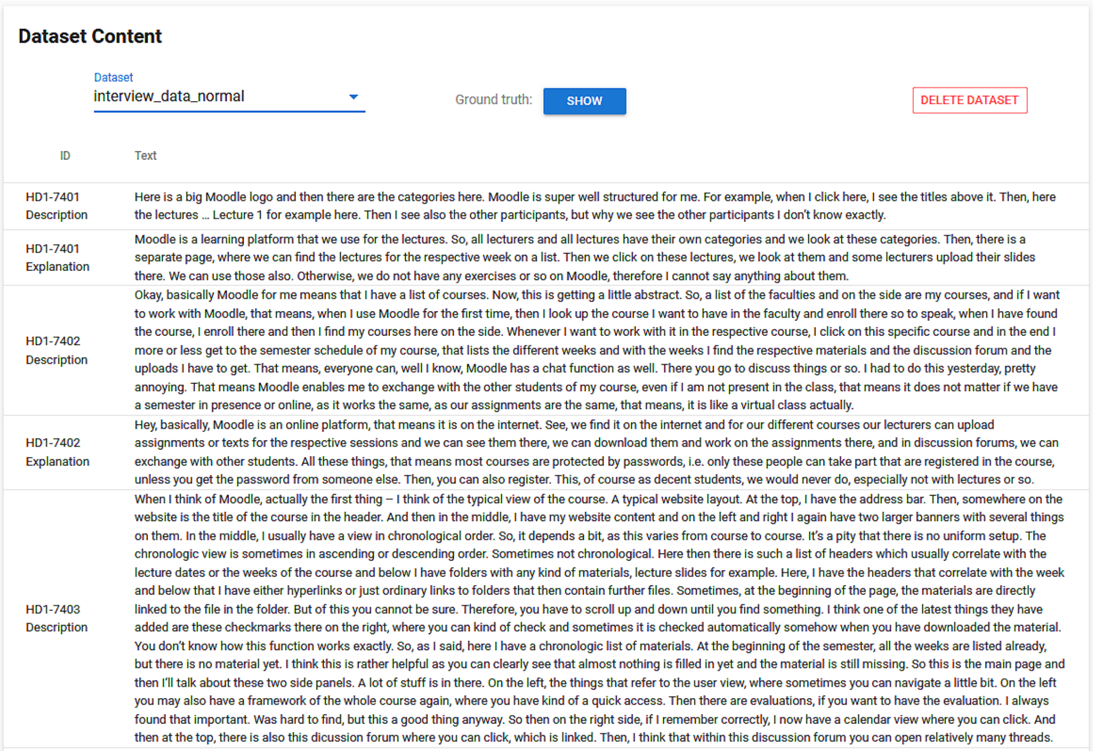


Currently allowed file types: xlsx, csv and txt. The dataset will be saved with its filename. Uploading a dataset which name already exists will update the dataset. For csv and txt the delimiter is set to '|'.
Note: FCIC and RBAI require UTF-8 compatible encodings. Please consider that any typographic errors and unexpected characters may lead to artifacts in the results.

Fig. 5.10: Upload Dataset View

5.4.3 Dataset View

In the dataset view, the documents of a dataset can be displayed. This view implements the workspace W6 (Figure 4.2). First, a dataset is selected via the dropdown. The dataset is loaded from the storage service and its documents are displayed in the table below. Figure 5.11 shows the dataset view with a dataset selected. The left column shows the IDs of the documents, while the right column contains the documents. If the dataset has a ground truth, there is a button, which when pressed will cause the ground truth to be highlighted in the documents. There is also a button for dataset deletion. This can be used by the tool user to review a dataset and delete it, in case it is no longer needed. The implementation only differs slightly optically from the mock-up in Figure 4.5. There is no more confirmation checkbox, as the confirmation of the deletion is handled with a tooltip. Therefore, there is a button for displaying the ground truth, which is not in the mock-up.

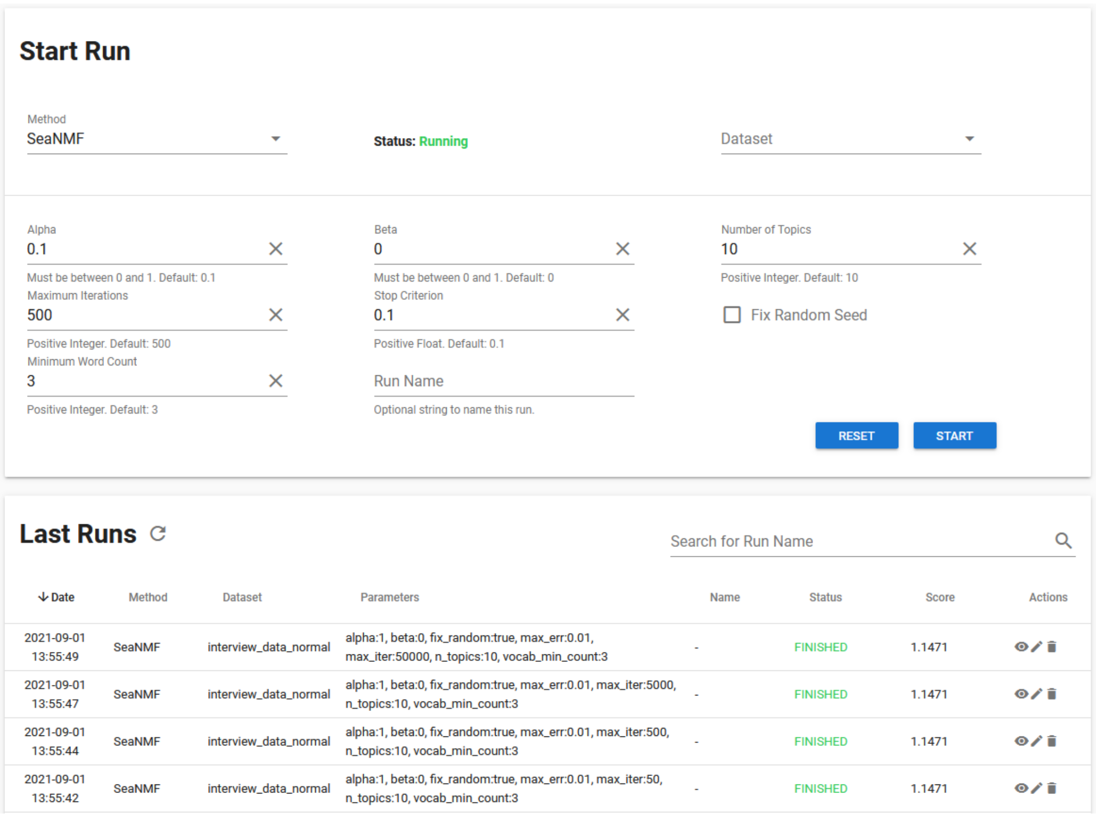


ID	Text
HD1-7401 Description	Here is a big Moodle logo and then there are the categories here. Moodle is super well structured for me. For example, when I click here, I see the titles above it. Then, here the lectures … Lecture 1 for example here. Then I see also the other participants, but why we see the other participants I don't know exactly.
HD1-7401 Explanation	Moodle is a learning platform that we use for the lectures. So, all lecturers and all lectures have their own categories and we look at these categories. Then, there is a separate page, where we can find the lectures for the respective week on a list. Then we click on these lectures, we look at them and some lecturers upload their slides there. We can use those also. Otherwise, we do not have any exercises or so on Moodle, therefore I cannot say anything about them.
HD1-7402 Description	Okay, basically Moodle for me means that I have a list of courses. Now, this is getting a little abstract. So, a list of the faculties and on the side are my courses, and if I want to work with Moodle, that means, when I use Moodle for the first time, then I look up the course I want to have in the faculty and enroll there so to speak, when I have found the course, I enroll there and then I find my courses here on the side. Whenever I want to work with it in the respective course, I click on this specific course and in the end I more or less get to the semester schedule of my course, that lists the different weeks and with the weeks I find the respective materials and the discussion forum and the uploads I have to get. That means, everyone can, well I know, Moodle has a chat function as well. There you go to discuss things or so. I had to do this yesterday, pretty annoying. That means Moodle enables me to exchange with the other students of my course, even if I am not present in the class, that means it does not matter if we have a semester in presence or online, as it works the same, as our assignments are the same, that means, it is like a virtual class actually.
HD1-7402 Explanation	Hey, basically, Moodle is an online platform, that means it is on the internet. See, we find it on the internet and for our different courses our lecturers can upload assignments or texts for the respective sessions and we can see them there, we can download them and work on the assignments there, and in discussion forums, we can exchange with other students. All these things, that means most courses are protected by passwords, i.e. only these people can take part that are registered in the course, unless you get the password from someone else. Then, you can also register. This, of course as decent students, we would never do, especially not with lectures or so.
HD1-7403 Description	When I think of Moodle, actually the first thing – I think of the typical view of the course. A typical website layout. At the top, I have the address bar. Then, somewhere on the website is the title of the course in the header. And then in the middle, I have my website content and on the left and right I again have two larger banners with several things on them. In the middle, I usually have a view in chronological order. So, it depends a bit, as this varies from course to course. It's a pity that there is no uniform setup. The chronologic view is sometimes in ascending or descending order. Sometimes not chronological. Here then there is such a list of headers which usually correlate with the lecture dates or the weeks of the course and below I have folders with any kind of materials, lecture slides for example. Here, I have the headers that correlate with the week and below that I have either hyperlinks or just ordinary links to folders that then contain further files. Sometimes, at the beginning of the page, the materials are directly linked to the file in the folder. But of this you cannot be sure. Therefore, you have to scroll up and down until you find something. I think one of the latest things they have added are these checkmarks there on the right, where you can kind of check and sometimes it is checked automatically somehow when you have downloaded the material. You don't know how this function works exactly. So, as I said, here I have a chronologic list of materials. At the beginning of the semester, all the weeks are listed already, but there is no material yet. I think this is rather helpful as you can clearly see that almost nothing is filled in yet and the material is still missing. So this is the main page and then I'll talk about these two side panels. A lot of stuff is in there. On the left, the things that refer to the user view, where sometimes you can navigate a little bit. On the left you may also have a framework of the whole course again, where you have kind of a quick access. Then there are evaluations, if you want to have the evaluation. I always found that important. Was hard to find, but this a good thing anyway. So then on the right side, if I remember correctly, I now have a calendar view where you can click. And then at the top, there is also this dicussion forum where you can click, which is linked. Then, I think that within this discussion forum you can open relatively many threads.

Fig. 5.11: Dataset View

5.4.4 Start Analysis View

Based on W2 (Figure 4.2) and a mock-up (Figure 4.6), this view as shown in Figure 5.12 is used to start a new run. On the top side, a method can be selected. The middle part below will load the parameter component of this method. Allowing to customize the parameters of this method. There are hints about the default parameter values and special hints will be displayed, when the parameter input is invalid. There is also a button to reset the parameters to default. A status field besides the method drop down indicates the service status of the current method. It can either be running or be offline. On the top left side, the dataset that is going to be analyzed is selected. On the bottom side, there is a table listing the last runs that were started. It can give the tool user an overview about what has already been done. When a method is selected on the top side, the last runs are filtered by this method. The runs are shown with their parameters, name and status. Optional there is a score field that can be used. In this case, it shows the topic coherence score, as SeaNMF is a topic modeling method. The status of a run can be started, when the analysis is still ongoing, finished, or failed, in case there was some kind of error and the run could not finish properly. Small icons indicate actions that can be done with a run (view result, rename, delete). Lastly, there is a search bar, which can be used to filter for a specific run name, that was previously given to a run. The class diagram of this view can be seen in Figure 5.13. The *StartAnalysisHome* class has access to the *Methods* class, which contains the individual *Method* objects. The classes *LdaParameter* and *SeanmfParameter* are loaded when the corresponding method is selected.



Start Run

Method: SeaNMF

Status: Running

Dataset

Alpha: 0.1
Must be between 0 and 1. Default: 0.1

Maximum Iterations: 500
Positive Integer. Default: 500

Minimum Word Count: 3
Positive Integer. Default: 3

Beta: 0
Must be between 0 and 1. Default: 0

Stop Criterion: 0.1
Positive Float. Default: 0.1

Run Name
Optional string to name this run.

Number of Topics: 10
Positive Integer. Default: 10

Fix Random Seed

RESET START

Last Runs

Search for Run Name

↓ Date	Method	Dataset	Parameters	Name	Status	Score	Actions
2021-09-01 13:55:49	SeaNMF	interview_data_normal	alpha:1, beta:0, fix_random:true, max_err:0.01, max_iter:50000, n_topics:10, vocab_min_count:3	-	FINISHED	1.1471	
2021-09-01 13:55:47	SeaNMF	interview_data_normal	alpha:1, beta:0, fix_random:true, max_err:0.01, max_iter:5000, n_topics:10, vocab_min_count:3	-	FINISHED	1.1471	
2021-09-01 13:55:44	SeaNMF	interview_data_normal	alpha:1, beta:0, fix_random:true, max_err:0.01, max_iter:500, n_topics:10, vocab_min_count:3	-	FINISHED	1.1471	
2021-09-01 13:55:42	SeaNMF	interview_data_normal	alpha:1, beta:0, fix_random:true, max_err:0.01, max_iter:50, n_topics:10, vocab_min_count:3	-	FINISHED	1.1471	

Fig. 5.12: Start Analysis View

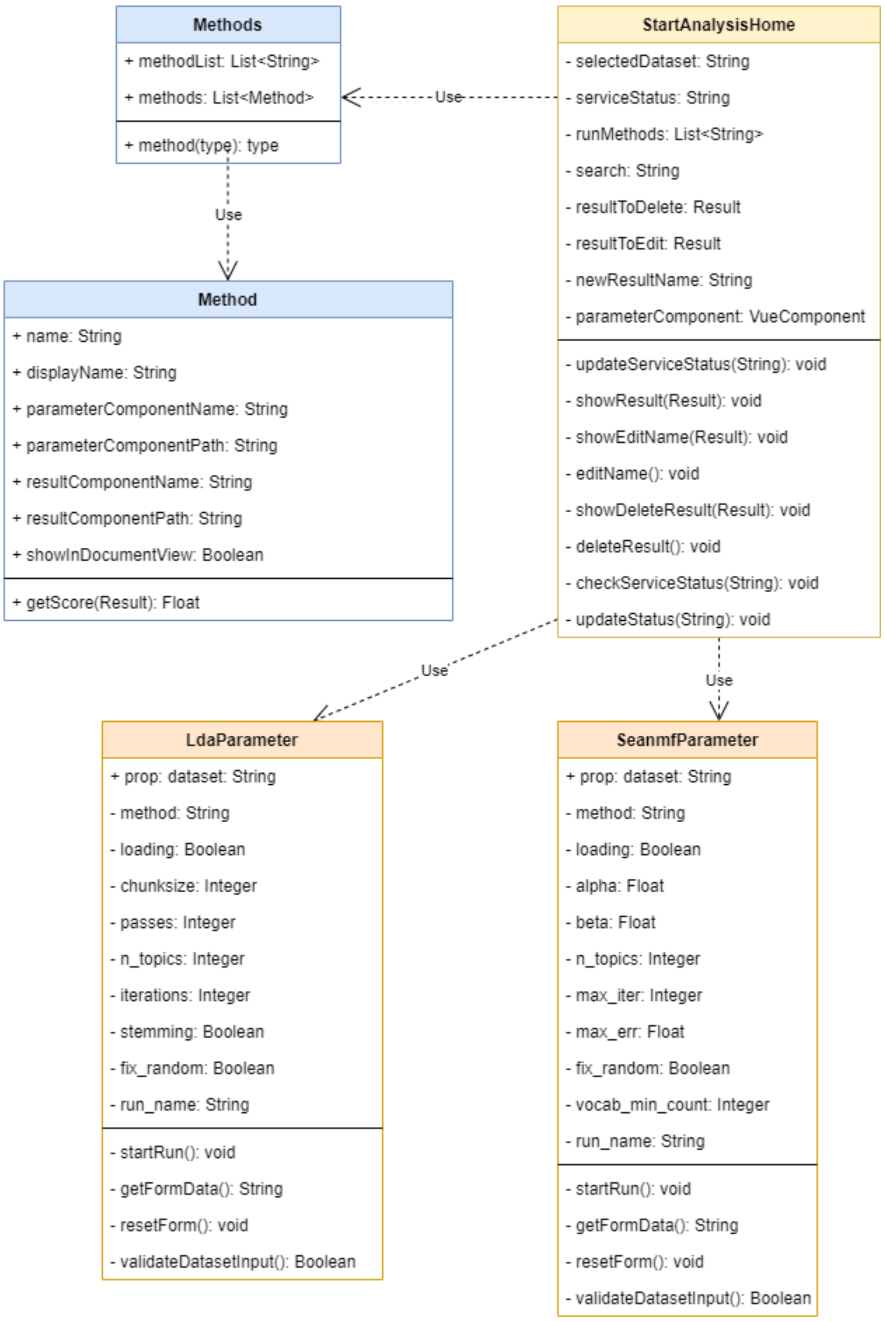
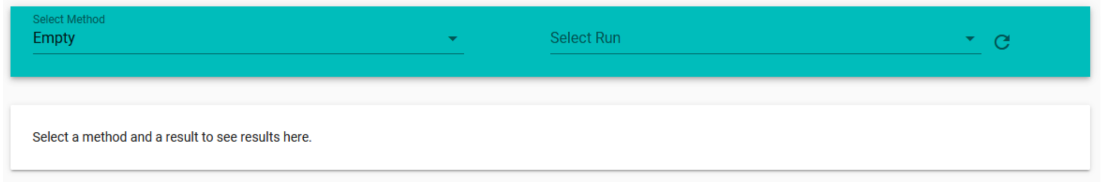


Fig. 5.13: Class Diagram for the Start Analysis View (ri-visualization)

5.4.5 Detection Result View

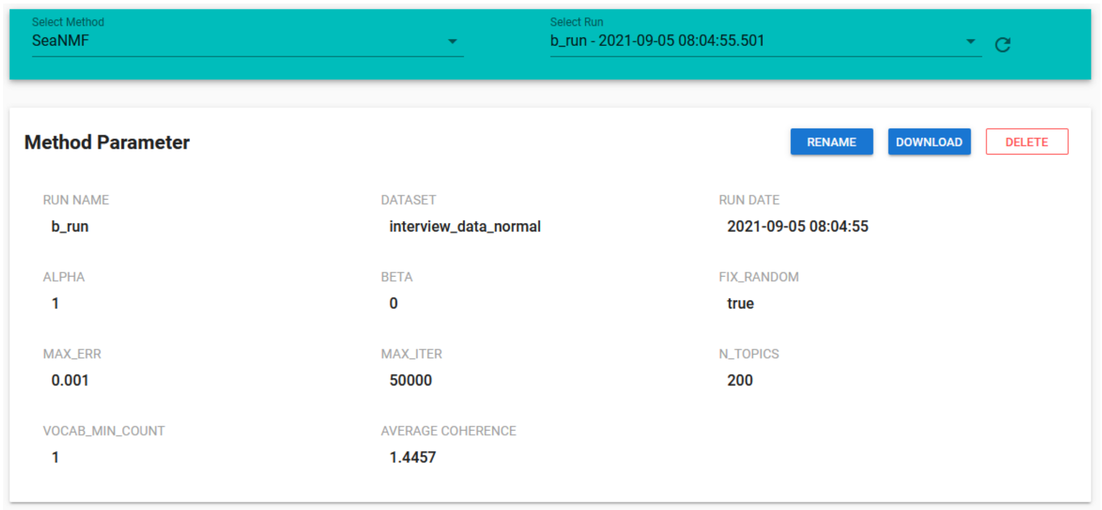
In this view, which implements workspace W3 (4.2), the results are visualized. It is designed in a way that it can be used for visualization of any methods' result. Initially, when no method is selected, it looks as shown in Figure 5.14. The *Vue* component that is used to visualize a specific method is loaded as soon as the method is selected. That way, each method can have their own visualization. However, for LDA and SeaNMF visualization, the same component is used, as their results share the same format. The exact visualization display is different from the mock-up (Figure 4.7), as the visualization options were not yet known. When a result is selected, the view updates with the result data as seen in Figure 5.15. There is a section, where the method parameters are displayed for review. It adapts the display on the parameters as they are stored in the result. That way, it can be used for LDA and SeaNMF, although they have different parameters. There are also buttons to perform further actions with the result (rename, download, delete). Below the parameter display, there is another section with a word-cloud as seen in Figure 5.16. It displays concept words in different sizes, where a bigger size indicates that the concept word appeared more often in the dataset. This gives the user a first overview over the results. Afterwards, all concept words are listed in a column, as seen in Figure 5.17. If there is a ground truth, its elements will be displayed in a column beside the concept words. A color coding is applied to the concepts and ground truth elements to indicate a match. The legend can be clicked to show only matching or non-matching concept words or ground truth elements. The next section as shown in Figure 5.18 is related to this one, as it shows metrics about the concept words and ground truth matching. The amount of unique concepts detected is listed, together with the amount of unique ground truth elements, true positives, false positives and false negatives. Below, there is precision, recall and F1-score displayed. This is also shown visually with gauges. This allows a quick evaluation of the performance of the method on concept detection. Lastly, there is a heatmap, showing the distribution of the concept words and documents as displayed in Figure 5.19. Documents are listed with their IDs. This heatmap can give the user hints, as to which concept should be evaluated further. Figure 5.20 shows the class diagram for the result view. The *ResultsHome* class is mainly the container of the individual result visualizations. It uses the *UvlFilterToolbar* component, which contains the dropdown for method and run selection, as seen in Figure 5.14.

The main visualization component is the *TopicDetectionResult* class for both LDA and SeaNMF. It contains the *GroundTruthComparison* class, which is the implementation of the ground truth comparison widget as shown in Figure 5.18. Furthermore it also contains a *WordToDocumentHeatmap* class for the heatmap component (Figure 5.19), which is based on the *EChart* class, and a *Cloud* class for the wordcloud component (Figure 5.16).



Select a method and a result to see results here.

Fig. 5.14: Empty Result View



SeaNMF
b_run - 2021-09-05 08:04:55.501

Method Parameter

RUN NAME b_run	DATASET interview_data_normal	RUN DATE 2021-09-05 08:04:55
ALPHA 1	BETA 0	FIX_RANDOM true
MAX_ERR 0.001	MAX_ITER 50000	N_TOPICS 200
VOCAB_MIN_COUNT 1	AVERAGE COHERENCE 1.4457	

Fig. 5.15: Parameter Display of the Result View

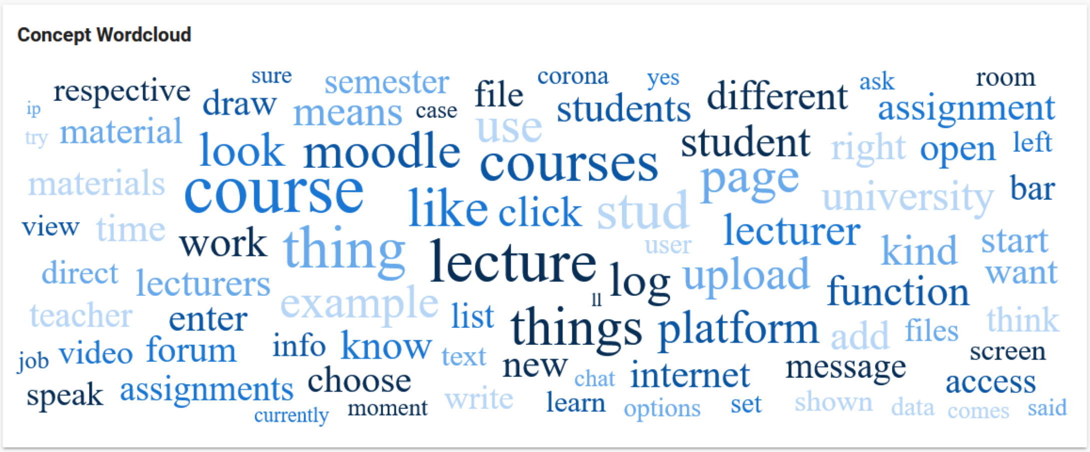
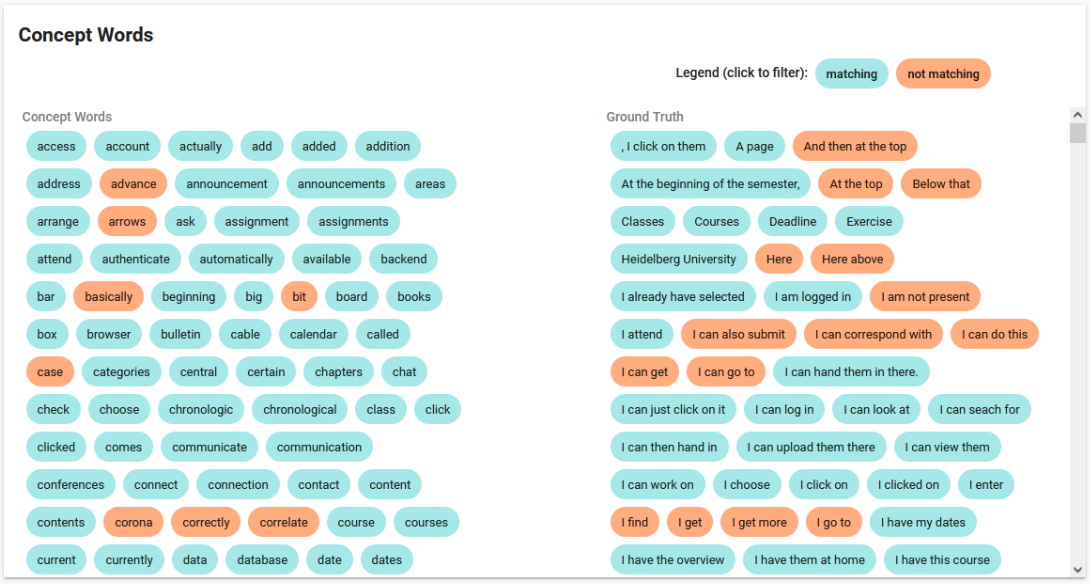


Fig. 5.16: Wordcloud of the Result View



Concept Words

Legend (click to filter): matching | not matching

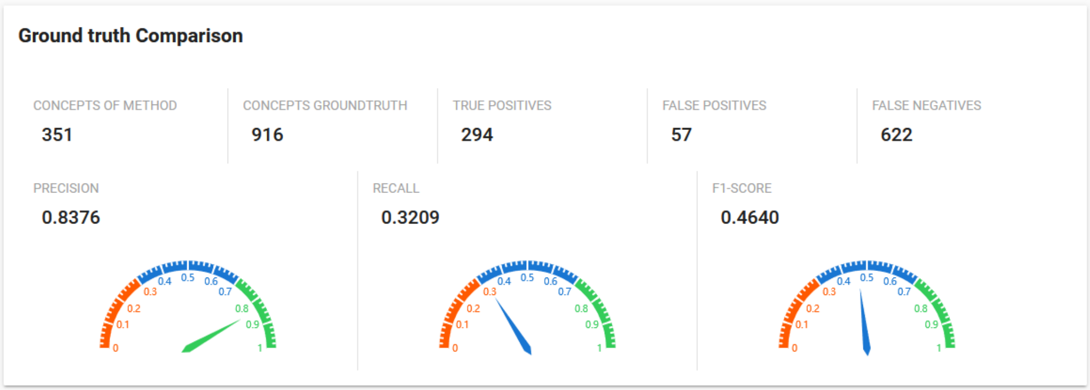
Concept Words

access, account, actually, add, added, addition, address, advance, announcement, announcements, areas, arrange, arrows, ask, assignment, assignments, attend, authenticate, automatically, available, backend, bar, basically, beginning, big, bit, board, books, box, browser, bulletin, cable, calendar, called, case, categories, central, certain, chapters, chat, check, choose, chronologic, chronological, class, click, clicked, comes, communicate, communication, conferences, connect, connection, contact, content, contents, corona, correctly, correlate, course, courses, current, currently, data, database, date, dates

Ground Truth

, I click on them, A page, And then at the top, At the beginning of the semester, At the top, Below that, Classes, Courses, Deadline, Exercise, Heidelberg University, Here, Here above, I already have selected, I am logged in, I am not present, I attend, I can also submit, I can correspond with, I can do this, I can get, I can go to, I can hand them in there., I can just click on it, I can log in, I can look at, I can seach for, I can then hand in, I can upload them there, I can view them, I can work on, I choose, I click on, I clicked on, I enter, I find, I get, I get more, I go to, I have my dates, I have the overview, I have them at home, I have this course

Fig. 5.17: Concept Word and Ground Truth Display of the Result View



Ground truth Comparison

CONCEPTS OF METHOD	CONCEPTS GROUNDTRUTH	TRUE POSITIVES	FALSE POSITIVES	FALSE NEGATIVES
351	916	294	57	622

PRECISION	RECALL	F1-SCORE
0.8376	0.3209	0.4640

Fig. 5.18: Ground Truth Comparison Widget

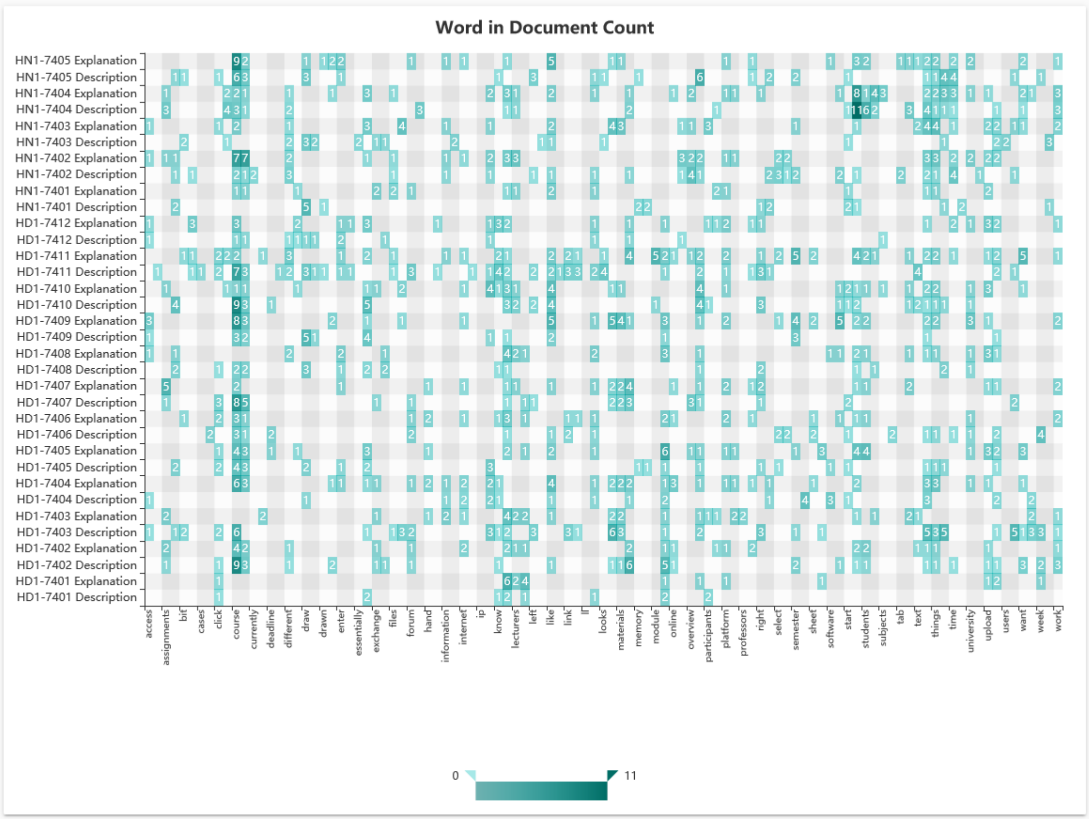


Fig. 5.19: Word-to-Document Heatmap

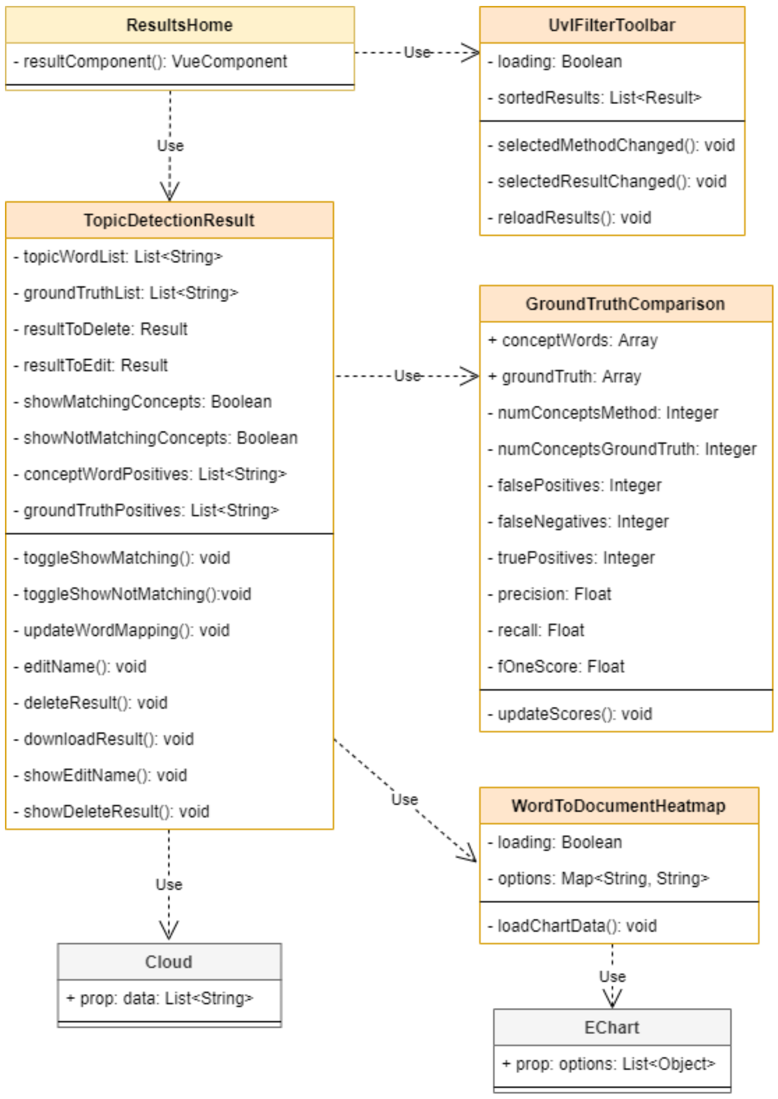
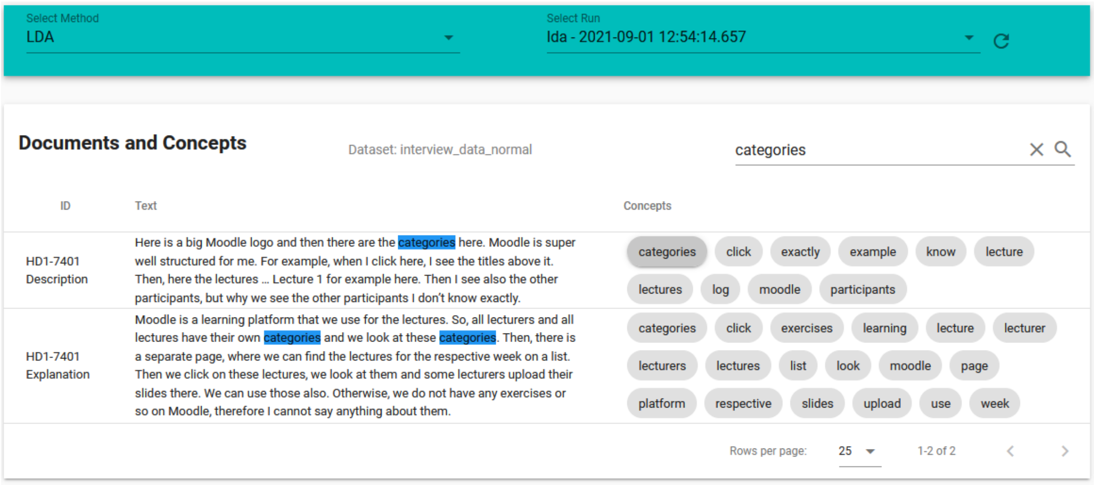

ResultsHome
- resultComponent(): VueComponent
Use
UvlFilterToolbar
- loading: Boolean
- sortedResults: List<Result>
- selectedMethodChanged(): void
- selectedResultChanged(): void
- reloadResults(): void
Use
TopicDetectionResult
- topicWordList: List<String>
- groundTruthList: List<String>
- resultToDelete: Result
- resultToEdit: Result
- showMatchingConcepts: Boolean
- showNotMatchingConcepts: Boolean
- conceptWordPositives: List<String>
- groundTruthPositives: List<String>
- toggleShowMatching(): void
- toggleShowNotMatching():void
- updateWordMapping(): void
- editName(): void
- deleteResult(): void
- downloadResult(): void
- showEditName(): void
- showDeleteResult(): void
Use
GroundTruthComparison
+ conceptWords: Array
+ groundTruth: Array
- numConceptsMethod: Integer
- numConceptsGroundTruth: Integer
- falsePositives: Integer
- falseNegatives: Integer
- truePositives: Integer
- precision: Float
- recall: Float
- fOneScore: Float
- updateScores(): void
Use
WordToDocumentHeatmap
- loading: Boolean
- options: Map<String, String>
- loadChartData(): void
Use
Cloud
+ prop: data: List<String>
Use
EChart
+ prop: options: List<Object>

Fig. 5.20: Class Diagram for the Result View (ri-visualization)

5.4.6 Document View

This view is also used for result display, but centered around the documents of the dataset. It resembles the workspace W5 (Figure 4.2). As the result view can be very large, depending on method visualization, this view has been separated from it. The top bar as shown in Figure 5.21 is the same as in the result view. When a method or result has been selected in one view, it will also be updated in the other. Which means that when the user was viewing a result in the result view, the same result will already be selected in this view, when the user switches the view. The documents of the dataset that was analyzed are displayed similar to the dataset view but there is another column, where the concepts of the result that are part of the document, are shown. By clicking on a concept, it will be entered into the search bar, which will trigger the filter function. Only documents that contain the concept are shown, and the concept will be highlighted blue in the documents. This makes reviewing the context of the concept word easier.
Initially, the idea was to show the topics of a document with their probability (see the mock-up in Figure 4.8), as returned by the implemented topic modeling methods, but this was not useful for concept detection. Topics are not displayed, only the topic words, which are considered as concept words in this thesis. This was necessary to transfer the topic modeling methods for concept detection.



Select Method: LDA

Select Run: lda - 2021-09-01 12:54:14.657

Documents and Concepts

Dataset: interview_data_normal

categories

ID	Text	Concepts
HD1-7401 Description	Here is a big Moodle logo and then there are the categories here. Moodle is super well structured for me. For example, when I click here, I see the titles above it. Then, here the lectures … Lecture 1 for example here. Then I see also the other participants, but why we see the other participants I don't know exactly.	categories, click, exactly, example, know, lecture, lectures, log, moodle, participants
HD1-7401 Explanation	Moodle is a learning platform that we use for the lectures. So, all lecturers and all lectures have their own categories and we look at these categories. Then, there is a separate page, where we can find the lectures for the respective week on a list. Then we click on these lectures, we look at them and some lecturers upload their slides there. We can use those also. Otherwise, we do not have any exercises or so on Moodle, therefore I cannot say anything about them.	categories, click, exercises, learning, lecture, lecturer, lecturers, lectures, list, look, moodle, page, platform, respective, slides, upload, use, week

Rows per page: 25 1-2 of 2

Fig. 5.21: Document View

6 Quality Assurance

In this chapter, the quality assurance process for the created software tool is described. Firstly, Section 6.1 introduces the test structure and measures that have been performed. Next, Section 6.2 presents the static code testing of the development process. Component testing is then explained in Section 6.3. Afterwards, the system tests are described in Section 6.4.

6.1 Test Concept

Due to the micro-service architecture, the different parts of the tool are divided into their own code repositories. This allows the testing of each repository on its own. The code repositories themselves are managed via Git and hosted on GitHub, which makes the integration of different testing tools possible.

The static code analysis has been employed with Codefactor[1], which is set up to analyze the code base after each commit. One benefit of Codefactor is, that it supports all programming languages that are used in this project, so all repositories can be checked. The detected issues are categorized into style issues and maintainability issues. Style issues refer to the way the code is written, like formatting issues, but do not affect functionality. On the other hand, maintainability issues include unused variables or import statements and code smells, which would make later code editing more difficult. Those issues are very relevant for fulfilling NFR: Modifiability (see Section 4.5.3). Fixing those issues improves overall code quality and readability and resolves problems before component or system tests are executed.

The component tests are used to test the micro-services that are part of the backend of Feed.UVL, namely storage and orchestrator. The tests are implemented in the native *Go* environment, which features code coverage reporting, and are part of the respective code repository. Component testing is declared as successful when the code coverage is above the specified 85% for all relevant methods, which is required for NFR: Testability (see Section 4.5.3), and all the tests are passing.

[1]Codefactor: https://www.codefactor.io/ Last accessed: 26.08.2021

System tests use a specified scheme with an expected outcome and a step-by-step procedure for each test. When the outcome deviates from what is expected, a bug report is issued, which will be investigated further. The system tests are declared successful when each test results in the expected outcome. The testing has been performed during the development process and especially when an implementation task has been finished.

6.2 Static Code Tests

The static code testing with Codefactor has been performed on all code repositories at the end of the development phase. Codefactor reported two types of issues: Style issues, which refer to code formatting and styling rules, that can affect readability, and maintainability issues, which can impair later code editing. Every issue, that has been found, has been fixed afterwards. Table 6.1 lists the repositories along with the detected style and maintainability issues.

Table 6.1: Result of Static Code Testing

Repository	**Style Issues**	**Maintainability Issues**	**Total Issues**
ri-visualization	3	11	14
uvl-analytics-concepts-lda	0	3	3
uvl-analytics-concepts-seanmf	0	3	3
uvl-orchestration-concepts	0	0	0
uvl-storage-concepts	0	0	0

6.3 Component Tests

Due to dependency issues that occurred when using Jenkins for testing, the tests were run on a local machine before a commit is made and the code coverage report was included in the repository. This way, it was made sure that bugs are detected before a new version got deployed. The coverage is measured with the native *Go* tool cover and is defined as the percentage of code lines that have been reached at least once during test execution. Testing constraints are satisfied when a minimum coverage of 85% is achieved. Table 6.2 lists all relevant files for the backend micro-services and their coverage. Since Go does not feature classes, the coverage is reported per file.

Table 6.2: Component Test Coverage

Repository	**File**	**Coverage**
uvl-orchestration-concepts	starter.go	91.8%
	rest_handler.go	89.0%
uvl-storage-concepts	starter.go	90.6%
	model.go	93.3%
	mongodb.go	89.8%
Average		**90.07%**

6.4 System Tests

This section describes the system tests (TCS) that were performed in order to check the functionality of all system functions as listed in Section 4.4.2. All the steps and conditions have been fully documented. If there is no workspace in the expected GUI result listed, then it remains unchanged.

SF1: Navigate between workspaces

The tests for SF1 are listed in Table 6.3. Their focus lies on the correct display of the UI elements and the display of the correct workspace.

Table 6.3: System Tests SF1: Navigate between workspaces

Field	**Value**
ID	TCS1.1
Preconditions	None
Preconditions GUI	W1: Navigation View
Test Steps	1. Go to non-Feed.UVL workspace 2. Go back to W1
Expected Result GUI	W1 correctly loaded. Correct display of Feed.UVL headline and logo
Expected Exception	None
Postcondition System	None
ID	TCS1.2
Preconditions	None
Preconditions GUI	W1: Navigation View
Test Steps	1. Go to W2: Start Concept Detection View
Expected Result GUI	W2 has been loaded, Feed.UVL theme and headline of W2 is shown
Expected Exception	None
Postcondition System	None

SF2: Upload dataset

Table 6.4 lists the tests for SF2. These tests are checking the correct behavior for different input files and storage states.

Table 6.4: System Tests SF2: Upload dataset

Field	**Value**
ID	TCS2.1
Preconditions	File f with dataset exists, file type is csv, xlsx or txt
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Select file f for upload
	2. Attempt upload
Expected Result GUI	Tooltip indicating upload success is shown
Expected Exception	None
Postcondition System	Dataset from f is stored
ID	TCS2.2
Preconditions	File f with dataset exists, file type is neither csv, xlsx nor txt
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Select file f for upload
	2. Attempt upload
Expected Result GUI	Tooltip indicating that this file type is not allowed
Expected Exception	File is not uploaded
Postcondition System	None
ID	TCS2.3
Preconditions	Invalid file f exists, file type is csv, xlsx or txt
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Select file f for upload
	2. Attempt upload
Expected Result GUI	Tooltip error that dataset could not be uploaded
Expected Exception	File is not uploaded
Postcondition System	None
ID	TCS2.4
Preconditions	Dataset d1 with name n is in database, file f with dataset d and name n exists
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Select file f for upload
	2. Attempt upload
Expected Result GUI	Tooltip indicating upload success is shown
Expected Exception	None
Postcondition System	Dataset d1 is replaced with dataset d in storage

SF3: Show dataset

The test in Table 6.5 checks the correct display of a dataset. The test is performed on both workspaces from the precondition GUI. The GUI result is always W6.

Table 6.5: System Tests SF3: Show dataset

Field	**Value**
ID	TCS3.1
Preconditions	At least one dataset d exists
Preconditions GUI	W4: Upload Dataset View or W6: Dataset View
Test Steps	1. If in W4, click "show dataset" button from d, else if in W6 select dataset d from dropdown
Expected Result GUI	W6, dataset d displayed
Expected Exception	None
Postcondition System	None

SF4: Filter dataset contents

The tests for SF4 are listed in Table 6.6. The tests focus on different input values and the correct outputs for the filtering.

Table 6.6: System Tests SF4: Filter dataset contents

Field	Value
ID	TCS4.1
Preconditions	Dataset d with result r exists
Preconditions GUI	W5: Document View, r selected
Test Steps	1. Enter an empty string into the search field
Expected Result GUI	All documents are displayed
Expected Exception	None
Postcondition System	None
ID	TCS4.2
Preconditions	Dataset d with result r exists
Preconditions GUI	W5: Document View, r selected
Test Steps	1. Enter a string into the search field, that does exist in at least one document, but not in all documents
Expected Result GUI	Only documents that contain the string are displayed. The string is marked blue in the texts.
Expected Exception	None
Postcondition System	None
ID	TCS4.3
Preconditions	Dataset d with result r exists
Preconditions GUI	W5: Document View, r selected
Test Steps	1. Enter a string into the search field, that does not exist in any document
Expected Result GUI	No document is displayed
Expected Exception	None
Postcondition System	None

SF5: Delete dataset

Table 6.7 lists the test for SF5. There is only one test as there are no variations of data or inputs.

Table 6.7: System Tests SF5: Delete dataset

Field	**Value**
ID	TCS5.1
Preconditions	At least one dataset d exists
Preconditions GUI	W6: Dataset View or W4: Upload Dataset View
Test Steps	1. Select dataset d (W6 only)
	2. Press "Delete Dataset" (W6) or "Delete" for d (W4)
	3. Confirm deletion
Expected Result GUI	Tooltip shows message that dataset d has been deleted
Expected Exception	None
Postcondition System	Dataset d is deleted

SF6: Upload ground truth

The tests for SF6, which are listed in 6.8, are analogous to the tests of SF2. Details on file type and file content were omitted when it is expected to be valid.

Table 6.8: System Tests SF6: Upload ground truth

Field	**Value**
ID	TCS6.1
Preconditions	Dataset d exists, file f with ground truth g exists
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Click "Add ground truth" button from d 2. Select file f for upload 3. Attempt upload
Expected Result GUI	Tooltip indicating upload success
Expected Exception	None
Postcondition System	Ground truth g is stored with dataset d
ID	TCS6.2
Preconditions	Dataset d exists, file f with ground truth but invalid file type exists
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Click "Add ground truth" button from d 2. Select file f for upload 3. Attempt upload
Expected Result GUI	Tooltip indicating that this file type is not allowed
Expected Exception	File is not uploaded
Postcondition System	None
ID	TCS6.3
Preconditions	Dataset d exists, file f with invalid file contents exists
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Click "Add ground truth" button from d 2. Select file f for upload 3. Attempt upload
Expected Result GUI	Tooltip error that ground truth could not be uploaded
Expected Exception	File is not uploaded
Postcondition System	None
ID	TCS6.4
Preconditions	Dataset d exists, file f with ground truth g exists
Preconditions GUI	W4: Upload Dataset View
Test Steps	1. Click "Add ground truth" button from d 2. Select file f for upload 3. Attempt upload
Expected Result GUI	Tooltip indicating upload success
Expected Exception	None
Postcondition System	Ground truth from dataset d is replaced with g

SF7: Highlight ground truth

The tests for SF7 check whether the GUI shows correct behavior based on the existence of a ground truth, and are listed in Table 6.9.

Table 6.9: System Tests SF7: Highlight ground truth

Field	Value
ID	TCS7.1
Preconditions	Dataset d with ground truth g exists
Preconditions GUI	W6: Dataset View
Test Steps	1. Select d
	2. Click button "Show"
Expected Result GUI	Ground truth from dataset is marked blue in displayed documents.
Expected Exception	None
Postcondition System	None
ID	TCS7.2
Preconditions	Dataset d without ground truth exists
Preconditions GUI	W6: Dataset View
Test Steps	1. Select d
Expected Result GUI	Hint is displayed that no ground truth is available
Expected Exception	None
Postcondition System	None

SF8: Start analysis run

Table 6.10 lists the system tests for SF8 and focuses on different UI inputs and a special case where the method micro-service is offline.

Table 6.10: System Tests SF8: Start analysis run

Field	Value
ID	TCS8.1
Preconditions	None
Preconditions GUI	W2: Start Concept Detection View
Test Steps	1. Choose a method 2. Select a dataset 3. Attempt run
Expected Result GUI	Tooltip message indicating that run has been started
Expected Exception	None
Postcondition System	Run result saved in storage
ID	TCS8.2
Preconditions	None
Preconditions GUI	W2: Start Concept Detection View
Test Steps	1. Choose a method 2. Attempt run
Expected Result GUI	Tooltip error that dataset should be selected
Expected Exception	Run is not started
Postcondition System	None
ID	TCS8.3
Preconditions	None
Preconditions GUI	W2: Start Concept Detection View
Test Steps	1. Choose a method 2. Delete default parameter 3. Select dataset 4. Attempt run
Expected Result GUI	Invalid parameter fields are marked red, with parameter hint. Tooltip message "Please validate your parameter inputs!" is shown.
Expected Exception	Run is not started
Postcondition System	None
ID	TCS8.4
Preconditions	Service of selected method is offline
Preconditions GUI	W2: Start Concept Detection View
Test Steps	1. Choose a method 2. Select dataset 3. Attempt run
Expected Result GUI	Tooltip message "Could not contact backend!", Status field shows "offline"
Expected Exception	Run is not started
Postcondition System	None

SF9: Filter run results

The tests in Table 6.11 test correct behavior of the result filter toolbar, which is present in workspaces W3 and W5.

Table 6.11: System Tests SF9: Filter run results

Field	**Value**
ID	TCS9.1
Preconditions	At least two methods exist, at least one result for each method exists
Preconditions GUI	W3: Detection Results View or W5: Document View
Test Steps	1. Select one of the methods from the method dropdown 2. Open the result dropdown
Expected Result GUI	The result dropdown only shows the results from the selected method
Expected Exception	None
Postcondition System	None
ID	TCS9.2
Preconditions	At least one run that has failed exists
Preconditions GUI	W3: Detection Results View or W5: Document View
Test Steps	1. Open the result dropdown
Expected Result GUI	The result dropdown does not show any failed run
Expected Exception	None
Postcondition System	None

SF10: Display run result

SF 10 is tested by the test in Table 6.12, which checks correct UI display of the results.

Table 6.12: System Tests SF10: Display run result

Field	Value
ID	TCS10.1
Preconditions	At least one result exists
Preconditions GUI	W2: Start Concept Detection View or W3: Detection Results View
Test Steps	1. Choose method of result 2. Choose run result (if in W2 -> select "show")
Expected Result GUI	W3, correct result displayed, correct method selected in method dropdown, correct result selected in result dropdown
Expected Exception	None
Postcondition System	None

SF11: Match concepts with documents

Table 6.13 lists the test for SF11, there is only one test as there are no variances in data or behavior expected.

Table 6.13: System Tests SF11: Match concepts with documents

Field	Value
ID	TCS11.1
Preconditions	At least one result exists
Preconditions GUI	W5: Document View
Test Steps	1. Choose result in result dropdown
Expected Result GUI	The dataset from the result is shown along with the detected concepts
Expected Exception	None
Postcondition System	None

SF12: Rename run result

The test for SF12, listed in Table 6.14 is executed for both workspaces listed in the precondition GUI.

Table 6.14: System Tests SF12: Rename run result

Field	Value
ID	TCS12.1
Preconditions	At least one result exists
Preconditions GUI	W2: Start Concept Detection View or W3: Detection Results View, result is selected
Test Steps	1. Click "Edit Name" Icon (W2) or click "Rename" Button (W3) 2. Enter a new name 3. Click "Edit" button
Expected Result GUI	Tooltip indicating that result has been renamed. Displayed name of result has changed.
Expected Exception	None
Postcondition System	Name of result changed in storage

SF13: Download run result

Table 6.15 lists the test for SF13. There is only one test as there are no variances in data or behavior expected.

Table 6.15: System Tests SF13: Download run result

Field	Value
ID	TCS13.1
Preconditions	At least one result exists
Preconditions GUI	W3: Detection Results View
Test Steps	1. Select result from dropdown 2. Click "Download" Button
Expected Result GUI	A download is shown in the browser, the downloaded file is in json format and contains the correct result data
Expected Exception	None
Postcondition System	None

SF14: Delete run result

Table 6.16 lists the test for SF14. There is only one test as there are no variances in data or behavior expected. However, the test is executed for both workspaces mentioned in precondition GUI.

Table 6.16: System Tests SF14: Delete run result

Field	**Value**
ID	TCS14.1
Preconditions	At least one result exists
Preconditions GUI	W2: Start Concept Detection View or W3: Detection Results View, result is selected
Test Steps	1. Click "Delete Result" Icon (W2) or "Delete" Button (W3) 2. Confirm deletion
Expected Result GUI	Tooltip indicating that result has been deleted. Result is not displayed anymore in W2 or W3.
Expected Exception	None
Postcondition System	Result is removed from storage

7 Evaluation

This chapter describes the evaluation process of the implemented topic detection methods LDA and SeaNMF on the task of concept detection. First, the evaluation concept is presented in Section 7.1 and relevant metrics are explained. Section 7.2 displays the results of the parameter analysis. Afterwards, the comparison of topic coherence and F1-score can be seen in 7.3. Lastly, the evaluation results are discussed and summarized in Section 7.4.

7.1 Evaluation Methodology

The evaluation process has multiple steps that are performed. Both methods are tested with a given dataset and the outputs are compared to a ground truth. The test dataset contains 34 documents, of which 24 documents are about the Moodle e-learning platform and the remaining 10 are about the Stud.IP platform. Each ground truth element is a natural language element of any length manually annotated in the dataset. Typically, it consists of 1-3 words. There are around 900 unique ground truth elements for this dataset. The topic detection methods return topic words which describe a topic (10 words per topic). This means, for 10 topics, there are 100 topic words returned. The same word can appear as a topic word for multiple topics, so the number of unique words returned may be lower. For this calculation, the topic words are regarded as concept words. A match (true positive) is found, when for a ground truth element, there is at least one concept word that is contained in the segment. This is because ground truth elements can have more than one word and a direct match cannot be made then, even though its meaning matches. As performance metric, precision, recall and F1-Score [36] are calculated as is standard in most algorithm evaluation. Precision is the ratio of correctly detected concepts among all detected concepts, while recall is the ratio of correctly detected concepts among all correct concepts. Their calculation is based on true positives (TP), false positives (FP) and false negatives (FN).

The formulas are as follows:

$$Precision = TP/(TP + FP)$$

$$Recall = TP/(TP + FN)$$

The F1-score is the weighted average of precision and recall:

$$F1\text{-}score = 2 * (recall * precision)/(recall + precision)$$

Parameters are considered optimal, when the F1-score is the highest among all tested combinations. A search grid is created for one parameter by choosing different values among the valid range as dictated by the authors. A single parameter is changed at a time, while the other parameters remain fixed and the F1-score is measured for those runs. The parameter that yielded the best score is used when other parameters are checked.
Both methods have a metric used in their original publication, which is used for evaluating the performance on topic modeling: Topic coherence [33]. This metric and the F1-score are checked for a correlation of their values for both methods. Correlation measures the linear relationship of two variables. It can be expressed with the Pearson correlation coefficient r and is calculated as follows [11]:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2(y_i - \bar{y})^2}}$$

where x_i and y_i are the values for variables x and y, and where $\bar{x}$ and $\bar{y}$ are the mean of the values for x and y respectively. A value > 0 indicates a positive correlation, while $r = 1$ indicates a perfect linear correlation. On the other side, a negative value stands for an inverse correlation and $r = -1$ means a perfect inverse correlation. Values close to zero are indicating weak to no relationship between two variables.

7.2 Parameter Analysis

This section is divided into two subsections, of which Section 7.2.1 is for the LDA parameters and Section 7.2.2 is for SeaNMF parameters. The random seed will be fixed for both methods, as this makes the results reproducible. For details about the individual parameters, see Chapter 5.

7.2.1 LDA Parameter

For LDA, there is no direct learning rate parameter, instead it is auto-learned during the run. Stemming will be disabled, as the method output then contains the stemmed words and those words cannot be directly compared to the ground truth anymore, which will lead to a lower score.
The *chunksize* parameter declares how many documents are used at one time during processing. This does not affect the results of LDA, only the speed of processing. Since the dataset only has 34 documents, setting the *chunksize* to any higher value will make no change. After training, the topic distribution is inferred over *iterations* times. When enough iterations have passed, the result does not change anymore. Setting *iterations* to 5,000 is sufficient for this. *Passes* determines how many times the dataset is processed. Increasing it will change the result up to a certain point, where all documents are fully processed.
For the first analysis runs, *chunksize* is set to 200 and *iterations* is set to 5,000 as described before. This leaves the number of topics and *passes* as the two main parameters of interest. *Passes* is left to the default value of 1. The number of topics is increased and the F1-score is reported in Figure 7.1.

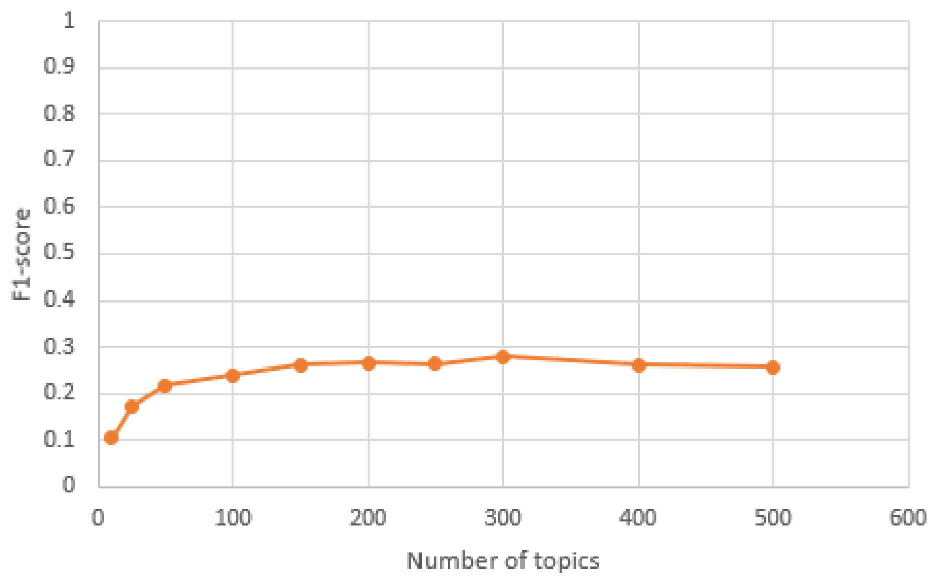


Fig. 7.1: Relation of F1-score and Number of Topics for LDA

The F1-score first rises with the number of topics, but after 300 topics, it drops again slightly. Increasing the number can lead to more words that are taken from the dataset as topic words, but there seems to be a limit, which words will be considered. For the further analysis, the number of topics is set to 300 as it yields the best result with a F1-score of 0.2797.
Next, the number of passes is analyzed and different values are tested. Figure 7.2 shows the relation of passes and F1-score.

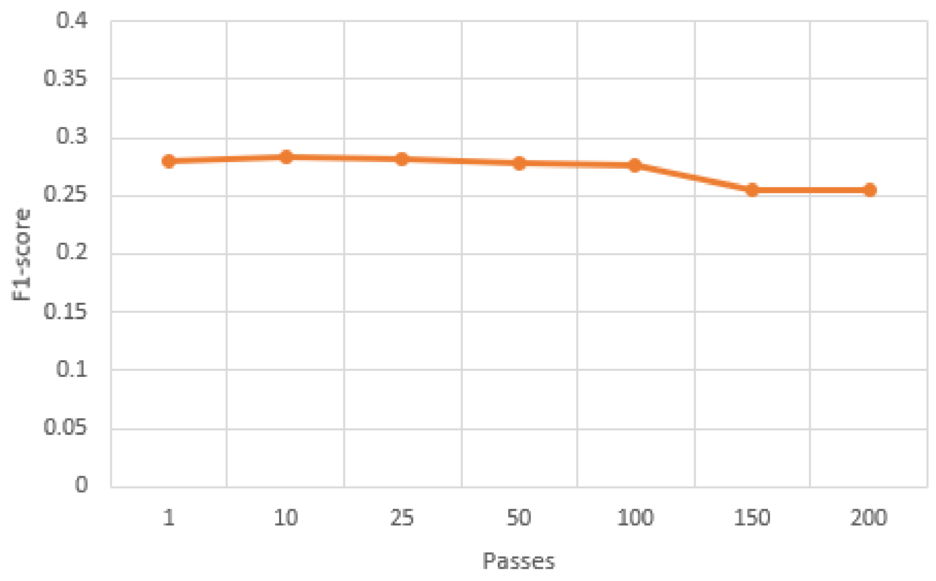


Fig. 7.2: Relation of F1-score and Passes for LDA

The F1-score remains stable at around 0.28, but seems to lower at around 150 passes. Interestingly, after 150 passes, the F1-score and result (as listed in Table 7.1) stays the same, indicating some form of convergence. The full list of parameters that were applied, and their results can be seen in Table 7.1. Best values are marked.
For parameter optimization, *passes* and the number of topics seem to have an intermediate peak, which means that it may be necessary to test different values to find the optimal one. *Chunksize* can be adjusted to contain the dataset documents or to a value that is possible for the computer memory. *Iterations* affects the processing time, which may be an issue with big datasets. Otherwise, it can be increased until the result does not change anymore.

Table 7.1: Parameter and Scores for LDA

Passes	Topics	Coherence	Concepts	Precision	Recall	F1-score
10	1	0.6591	56	**0.9107**	0.0556	0.1049
25	1	1.3043	105	0.838	0.096	0.1723
50	1	1.56	134	0.8507	0.1244	0.2171
100	1	1.8364	144	0.8819	0.1386	0.2396
150	1	1.9567	165	0.8606	0.155	0.2627
200	1	2.0584	167	0.8622	0.1572	0.2659
250	1	2.0347	175	0.8228	0.1572	0.2639
300	1	2.0922	185	0.8324	0.1681	0.2797
400	1	2.1522	177	0.8079	0.1561	0.2616
500	1	2.1492	171	0.8187	0.1528	0.2575
300	1	2.0922	185	0.8324	0.1681	0.2797
300	10	2.095	**188**	0.8297	**0.1703**	**0.2826**
300	25	2.0928	185	0.8378	0.1692	0.2815
300	50	2.09	179	0.8491	0.1659	0.2776
300	100	2.0949	178	0.8483	0.1648	0.276
300	150	2.1343	165	0.8363	0.1506	0.2553
300	200	2.1343	165	0.8363	0.1506	0.2553
10	10	0.7609	57	0.8947	0.0556	0.1048
25	10	1.3533	107	0.8317	0.0971	0.1739
50	10	1.5738	135	0.8518	0.1255	0.2188
100	10	1.8465	144	0.8958	0.1408	0.2433
150	10	1.9638	170	0.8588	0.1593	0.2688
200	10	2.0581	166	0.8614	0.1561	0.2643
250	10	2.0369	176	0.8352	0.1604	0.2692
350	10	2.1155	183	0.8196	0.1637	0.2739
400	10	2.1511	176	0.8181	0.1572	0.2637
450	10	**2.1642**	160	0.8437	0.1473	0.2509
500	10	2.1506	175	0.8171	0.1561	0.2621

7.2.2 SeaNMF Parameter

SeaNMF has a weight factorization parameter *alpha*, which can be tuned. It can influence the topic coherence score, but the effect on concept detection is not known. The *beta* parameter refers to sparse SeaNMF, which is a variant of SeaNMF. Therefore, beta is not considered relevant.

Iterations and *stopping criterion* are training related parameters. The training will stop either when the number of iterations reaches its value or when the internal error difference is below the threshold of *stopping criterion*. As the test dataset is small, *stopping criterion* is set below the standard value to 0.001 and *iterations* is set to 50,000. This may increase the analysis time but may yield a bit more accurate result. For first runs, *alpha* is set to the default value and the number of topics is increased step wise to see its effect on F1-score. Figure 7.3 shows the relation of F1-score and number of topics.

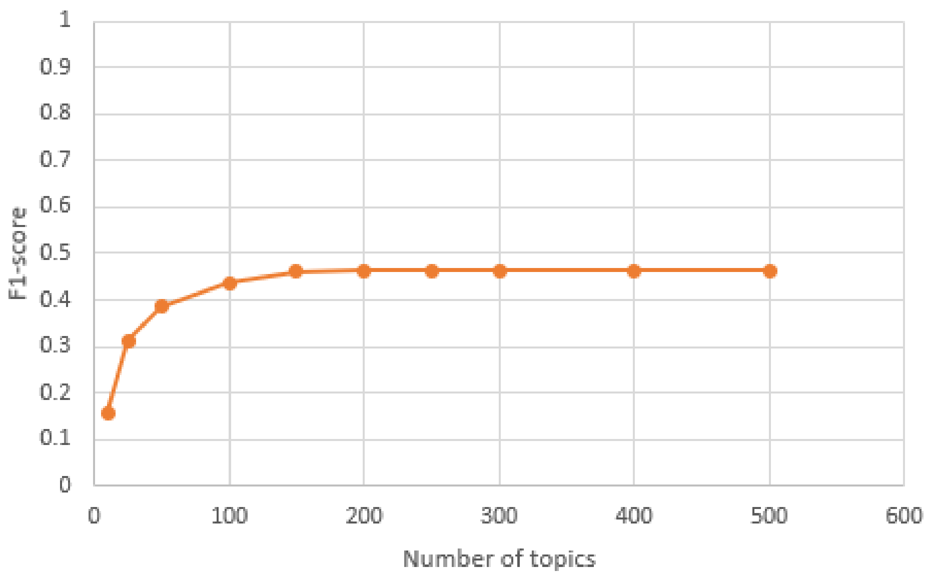


Fig. 7.3: Relation of F1-score and Number of Topics for SeaNMF

The F1-score rises as the number of topics rises until around 150 topics, where it levels off at around 0.46. This is due to an increase of detected concepts, which cause the recall to increase, while precision rests at about 0.83 as seen in Table 7.2. For further runs, the number of topics is set to 200, which is sufficiently high to get the best results encountered so far.

The *alpha* parameter is now set to different values, and the F1-score is reported for those runs. Figure 7.4 shows the relation of alpha and F1-score.

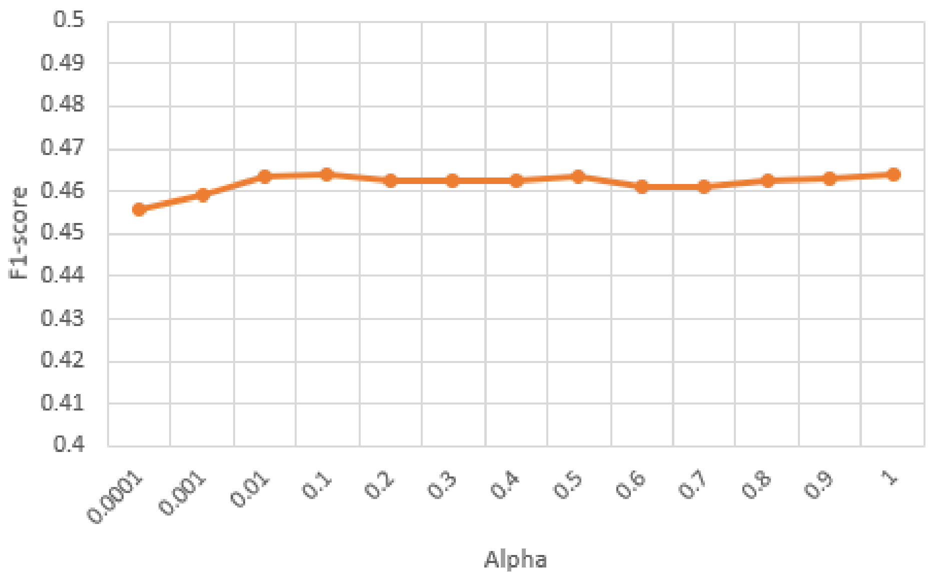


Fig. 7.4: Relation of F1-score and Alpha for SeaNMF

For *alpha* <= 0.001 the reported F1-score is below 0.46 while for *alpha* >= 0.1 the F1-score is slightly above 0.46, with the highest values at *alpha* = 0.01 and *alpha* = 1. Overall, the score difference is small. This seems to be the maximum that is possible for SeaNMF. Table 7.2 lists the relevant parameters and the reported scores for all performed SeaNMF analysis runs, the best values are marked.
For big datasets with more than 1,000 documents, it may not be possible to set *iterations* very high and *stop criterion* below default, as this will increase processing time. For *alpha*, there is not much difference in the resulting score, so it can be left with the default of 0.1. The number of topics can be increased until it levels off.

Table 7.2: Parameter and Scores for SeaNMF

Alpha	Topics	Coherence	Concepts	Precision	Recall	F1-score
0.1	10	1.9879	97	0.8247	0.0873	0.1579
0.1	25	**1.9964**	216	0.8194	0.1932	0.3127
0.1	50	1.4762	278	0.8273	0.251	0.3852
0.1	100	1.4443	327	0.8318	0.2969	0.4376
0.1	150	1.4124	350	0.8342	0.3187	0.4612
0.1	200	1.3669	351	**0.8376**	**0.3209**	**0.464**
0.1	250	1.1345	350	0.8371	0.3198	0.4628
0.1	300	0.9569	351	0.8347	0.3198	0.4625
0.1	350	0.8915	350	0.8342	0.3187	0.4612
0.1	400	0.8061	351	0.8347	0.3198	0.4625
0.1	450	0.6917	350	0.8342	0.3187	0.4612
0.1	500	0.7207	**352**	0.8347	**0.3209**	0.4637
0.0001	200	1.1454	344	0.8343	0.3133	0.4555
0.001	200	1.1247	347	0.8357	0.3165	0.4592
0.01	200	1.242	**352**	0.8352	**0.3209**	0.4637
0.2	200	1.429	351	0.8347	0.3198	0.4625
0.3	200	1.4072	351	0.8347	0.3198	0.4625
0.4	200	1.4726	351	0.8347	0.3198	0.4625
0.5	200	1.4397	**352**	0.8352	**0.3209**	0.4637
0.6	200	1.414	350	0.8342	0.3187	0.4612
0.7	200	1.4686	350	0.8342	0.3187	0.4612
0.8	200	1.4376	351	0.8347	0.3198	0.4625
0.9	200	1.4411	350	0.8371	0.3198	0.4628
1	200	1.4457	351	**0.8376**	**0.3209**	**0.464**

7.3 Metric Comparison

In this section, the relation of F1-score and topic coherence is analyzed. Figure 7.5 shows the before mentioned relation for LDA. The number of passes was set to 10 for this comparison, as it yielded the best performance for F1-score. Interestingly, the topic coherence score is slightly increasing with the number of topic, although it seems unlikely that a dataset with only 34 documents contains about 450 topics. The F1-score first rises with the number of topics and has a peak with a score of 0.2826 at 300 topics, but stays about the same level. The Pearson correlation coefficient (Section 7.1) for those results is ≈ 0.97, which indicates a very strong correlation of those metrics for these results.

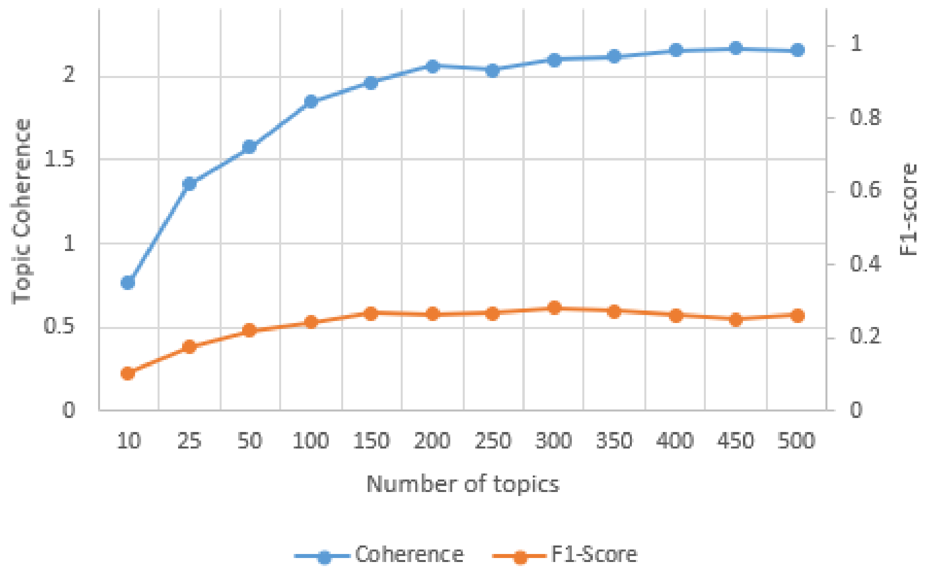


Fig. 7.5: Relation of F1-score and Topic Coherence for LDA

The relation of F1-score and topic coherence for the SeaNMF method can be seen in Figure 7.6. The topic coherence score is relatively high at 10 and 25 topics, which indicates that the optimal number of topics for topic detection is around this value. Afterwards, the topic coherence score drops, while the F1-score rises. However, topic coherence continues dropping after 150 topics, while F1-score keeps its level. The coherence score is falling, as a higher number of topics means moving away from the optimal number of topics. For a bigger dataset, the optimal number of topics may be higher, resulting in a different topic coherence curve. The correlation coefficient r is ≈ -0.78 for these results, indicating a strong inverse correlation of those metrics.

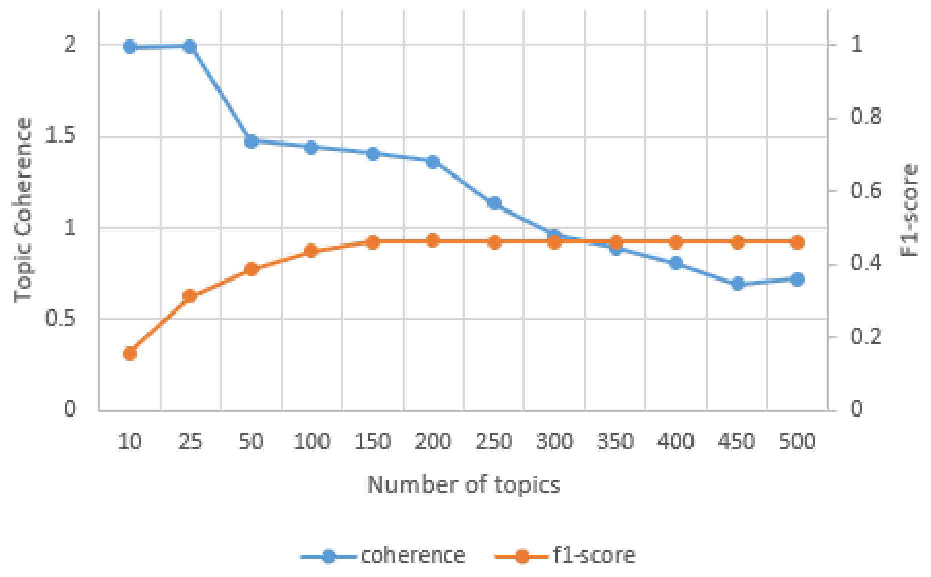


Fig. 7.6: Relation of F1-score and Topic Coherence for SeaNMF

7.4 Method Comparison and Discussion

The F1-score for both methods with their best parameters and for a different number of topics can be seen in Figure 7.7.
LDA has a lower F1-score for each number of topics, which is related to a lower overall recall (as listed in Table 7.1), while precision is on average nearly the same (≈ 0.84 for LDA and ≈ 0.83 for SeaNMF). This means that LDA takes a lesser number of unique words of the dataset into account. Overall, SeaNMF shows a better performance and parameter tuning is easier, as increasing the number of topics should be sufficient to find the best results.

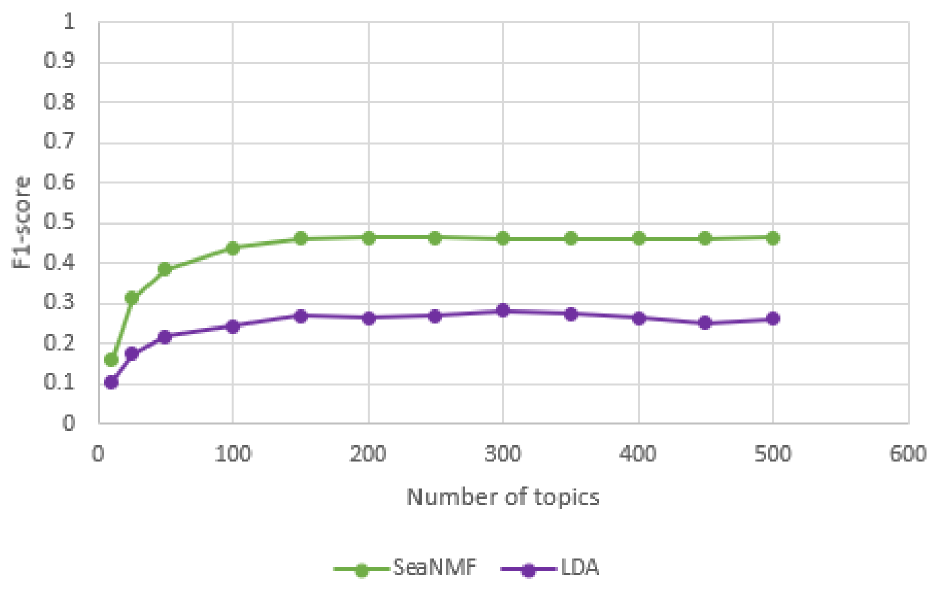


Fig. 7.7: F1-score for SeaNMF and LDA

While SeaNMF performs better than LDA, both methods seem to have a limit, which words are taken into account as topic words, and thus concept words. In the preprocessing of both methods, stop-word removal is performed, which reduces the number of words in the dataset, but it only removes words as "the" or "I", which are very common in the English language and may not indicate a special topic or concept. Both methods have a F1-score below 0.5, which is a considerably low performance for concept detection. The main issue is, that with the limited amount of unique concept words returned, the recall is very low in both cases. Precision is above 0.8 in for both methods, which means that the amount of false positive is relatively low and that sorting out false positives can be done fairly fast manually afterwards.
The topic coherence score shows a very strong positive correlation with the F1-score for LDA and a strong inverse correlation for SeaNMF. As it is the opposite for both methods, this means that the topic coherence metric cannot be used to determine which topic modeling method may be suited for concept detection in general. The correlation should be analyzed further with different datasets until conclusions can be made. As topic modeling methods have a different goal than concept detection, the topic coherence metric also measures different result properties. As a conclusion, it cannot be transferred for concept detection, and topic modeling methods with a good topic coherence score in benchmarks might not essentially be good at concept detection tasks.

8 Conclusion

In this chapter, Section 8.1 summarizes the thesis and Section 8.2 will give a discussion and a lookout for future improvements.

8.1 Summary

Coding of natural language is usually still performed manually, which requires a lot of work. Automatic processing would simplify this process, especially for big datasets. This thesis provides a software tool that can be used as a starting point for automatic processing of natural language. Different methods can be selected for analysis and the tool can be easily extended with more methods. The method results can then be visualized and reviewed to select an appropriate method for the task at hand. For concept extraction, a literature review of topic modeling methods on forum data has been performed and the two most suitable methods have been implemented and evaluated. LDA is a very widely used topic modeling method that can be applied to many use cases. SeaNMF is an improvement over NMF, which showed increased performance over LDA for short documents. Many other methods are very specific to forums and take forum metadata into account for improving their results. The implemented methods can be configured and analyses can be started via a web interface, which is integrated into the feed.ai platform. After a method finishes work, the results can also be seen in the web interface. SeaNMF and LDA share a single result view, as both are topic modeling methods and share the same output. In the result view, the applied parameters can be reviewed. The topic modeling output is interpreted for concept detection. A word cloud gives an overview of words identified by those methods. Concept words are then listed along with the ground truth, if available. A color coding indicates, which elements are matching and which are not. The performance of the applied method is evaluated with the ground truth comparison widget, which provides scores as precision, recall and F1-score. Lastly, a heat map indicates, which of the found words are contained in which document. An evaluation of the implemented methods on the task of concept detection has shown that both methods have a good precision score (≈ 0.84 for LDA and ≈ 0.83 for SeaNMF on average). Recall, however, is rather low (≈ 0.17 and ≈ 0.31 for SeaNMF). The topic coherence and F1-score metrics have been checked for a correlation. There is a very strong correlation between these metrics for LDA and a strong inverse correlation for SeaNMF. However, this should be analyzed further with different datasets until conclusions can be made. The tool has been tested thoroughly and is fully integrated into feed.ai, sharing the same software architecture and user interface design, which provides a unified look and feel throughout the tool.

8.2 Discussion and Future Work

The evaluation has shown that the topic modeling algorithms have a moderate to low performance for concept detection. While precision is high, the amount of concept words is low, which results in a low recall and thus in a low F1-score. However, the design of the tool makes it easy to implement new methods, which then can be evaluated. The implementation of more methods could be one of the future tasks for the tool. These methods should possibly not be topic modeling methods, as they might report a similar low recall as LDA and SeaNMF.

While a word-to-document relation heat map can provide an overview of concepts and their distribution in documents, the information gain is low. Other visualizations, like a concept relationship graph, could provide more insights for concepts in documents, or concepts in general, which is a research area that could be part of a further literature review.

Currently, the ground truth can be uploaded via the user interface, which means, it has to be created outside the tool. Coding the documents of a dataset inside the tool for the creation of a ground truth could be a future functionality, as well as integrating the concept detection results as suggestions for annotations to the user.

On the technical side, pre-processing is performed in each method micro-service, which will lead to redundancy when more methods are added. A pre-processing micro-service can reduce this redundancy, as long as it has all pre-processing capabilities required by the methods in use.

Furthermore, the tooltip system can be improved in the future. Currently, when the tool user deletes multiple results or datasets in a row, the tooltips are displayed on top of each other, blocking the tooltip below.

In the future, a progress indicator of a run could be useful, because right now, there is only information that the run has been started or has been finished. It is not known to the tool user, how long processing will take. However, this feature may not be possible for all methods, as not every method may return a progress indicator.

Additionally, there could be a load indicator for method services. This can show the user how many runs there are currently running on a method service. With this knowledge, the tool user can decide to start another run later, when other runs have finished. This might be especially useful, when there are multiple users working with the tool at the same time.

9 Bibliography

[1] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, & Y. Yang, “Topic detection and tracking pilot study final report,” *Journal Contribution*, Jun. 2018. DOI: 10.1184/R1/6626252.v1.

[2] T. Atapattu & K. Falkner, “A Framework for Topic Generation and Labeling from MOOC Discussions,” in *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, New York, NY, USA: ACM, April 2016, pp. 201–204. DOI: 10.1145/2876034.2893414.

[3] B. Athira, S. M. Idicula, & J. Jones, “Multi-label Topic Classification of Patient Generated Content in a Breast-cancer Community Forum,” in *Proceedings of the 4th International Conference on Medical and Health Informatics*, New York, NY, USA: ACM, August 2020, pp. 266–274. DOI: 10.1145/3418094.3418132.

[4] V. G. Bertalan & E. E. S. Ruiz, “Using topic modeling to find main discussion topics in brazilian political websites,” in *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, New York, NY, USA: ACM, October 2019, pp. 245–248. DOI: 10.1145/3323503.3360644.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[6] D. Blei, A. Ng, & M. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, May 2003. DOI: 10.1162/jmlr.2003.3.4-5.993.

[7] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, & D. Blei, “Reading tea leaves: How humans interpret topic models,” *Neural Information Processing Systems*, vol. 32, pp. 288–296, January 2009.

[8] F. Chen, J. Du, W. Qian, & A. Zhou, “Topic Detection over Online Forum,” in *2012 Ninth Web Information Systems and Applications Conference*, IEEE, November 2012, pp. 235–240. DOI: 10.1109/WISA.2012.15.

[9] V. Cheng & C. H. Li, “Linked Topic and Interest Model for Web Forums,” in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, IEEE, December 2008, pp. 279–284. DOI: 10.1109/WIIAT.2008.227.

[10] D. Ganguly & G. J. Jones, "Partially Labeled Supervised Topic Models for RetrievingSimilar Questions in CQA Forums," in *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, New York, NY, USA: ACM, Sep. 2015, pp. 161–170. DOI: 10.1145/2808194.2809460.

[11] A. Garcia Asuero, A. Sayago, & G. González, "The correlation coefficient: An overview," *Critical Reviews in Analytical Chemistry - CRIT REV ANAL CHEM*, vol. 36, pp. 41–59, January 2006. DOI: 10.1080/10408340500526766.

[12] K. Halder, M.-Y. Kan, & K. Sugiyama, "Health Forum Thread Recommendation Using an Interest Aware Topic Model," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, New York, NY, USA: ACM, November 2017, pp. 1589–1598. DOI: 10.1145/3132847.3132946.

[13] X. Hao & Y. Hu, "Topic detection and tracking oriented to BBS," in *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, IEEE, August 2010, pp. 154–157. DOI: 10.1109/CMCE.2010.5610205.

[14] I.-H. Hsiao & P. Awasthi, "Topic facet modeling," in *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, New York, NY, USA: ACM, March 2015, pp. 231–235. DOI: 10.1145/2723576.2723613.

[15] ISO / IEC, "Iso / iec 25010 system and software quality models," Tech. Rep., 2010. DOI: 10.3403/30215101.

[16] H. Jelodar, Y. Wang, C. Yuan, X. Feng, X. Jiang, Y. Li, & L. Zhao, "Latent dirichlet allocation (lda) and topic modeling: Models, applications, a survey," *Multimedia Tools Appl.*, vol. 78, no. 11, pp. 15 169–15 211, Jun. 2019. DOI: 10.1007/s11042-018-6894-4.

[17] N. Kahani, M. Bagherzadeh, J. Dingel, & J. R. Cordy, "The problems with eclipse modeling tools," in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, New York, NY, USA: ACM, October 2016, pp. 227–237. DOI: 10.1145/2976767.2976773.

[18] S.-H. Kim, H. Tak, & H.-G. Cho, "Polarized Topic Modeling for User Characteristics in Online Discussion Community," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, February 2019, pp. 1–4. DOI: 10.1109/BIGCOMP.2019.8679489.

[19] J. Krishnamani, Y. Zhao, & R. Sunderraman, "Forum Summarization Using Topic Models and Content-Metadata Sensitive Clustering," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, IEEE, November 2013, pp. 195–198. DOI: 10.1109/WI-IAT.2013.182.

[20] U. Kuckartz, "Qualitative text analysis: A systematic approach," in *Compendium for Early Career Researchers in Mathematics Education*. Cham: Springer International Publishing, 2019, pp. 181–197. DOI: 10.1007/978-3-030-15636-7_8.

[21] H. Li & Q. Li, "Forum topic detection based on hierarchical clustering," in *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, IEEE, Jul. 2016, pp. 529–533. DOI: 10.1109/ICALIP.2016.7846583.

[22] X. Li, G. Dai, S. Lai, & H. Dai, "Hot topic detection in Chinese web forum using statistics approach," in *2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, IEEE, Sep. 2011, pp. 1–4. DOI: 10.1109/ICSPCC.2011.6061621.

[23] H. Liu, Q. Li, R. Yao, & D. D. Zeng, "Analyzing Topics of JUUL Discussions on Social Media Using a Semantics-assisted NMF model," in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI 2019)*, Piscataway, NJ: IEEE, 2019, pp. 212–214. DOI: 10.1109/ISI.2019.8823541.

[24] Z. Liu, S. Rudian, C. Yang, J. Sun, & S. Liu, "Tracking the Dynamics of SPOC Discussion Forums: A Temporal Emotion-Topic Modeling Approach," in *2018 Seventh International Conference of Educational Innovation through Technology (EITT)*, IEEE, December 2018, pp. 174–179. DOI: 10.1109/EITT.2018.00042.

[25] Z. Liu, T. Wang, N. Pinkwart, S. Liu, & L. Kang, "An Emotion Oriented Topic Modeling Approach to Discover What Students are Concerned about in Course Forums," in *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*, IEEE, Jul. 2018, pp. 170–172. DOI: 10.1109/ICALT.2018.00047.

[26] M. Zhu, W. Hu, & O. Wu, "Topic Detection for Discussion Threads with Domain Knowledge," in *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, IEEE, August 2010, pp. 545–548. DOI: 10.1109/WI-IAT.2010.68.

[27] R. Mehrotra, S. Sanner, W. Buntine, & L. Xie, "Improving lda topic models for microblogs via tweet pooling and automatic labeling," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '13, Dublin, Ireland: Association for Computing Machinery, 2013, pp. 889–892. DOI: 10.1145/2484028.2484166.

[28] L. K. Nelson, D. Burk, M. L. Knudsen, & L. McCall, "The future of coding: A comparison of hand-coding and three types of computer-assisted text analysis methods," *Sociological Methods & Research*, vol. 50, pp. 202–237, May 2018. DOI: 10.1177/0049124118769114.

[29] Z. Nie, “Research on algorithm of Chinese BBS topic detection based on content analysis,” in *2011 IEEE International Conference on Computer Science and Automation Engineering*, IEEE, Jun. 2011, pp. 512–516. DOI: 10.1109/CSAE.2011.5952730.

[30] M. Paul & R. Girju, “Cross-cultural analysis of blogs and forums with mixed-collection topic models,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing Volume 3 - EMNLP ’09*, Morristown, NJ, USA: Association for Computational Linguistics, 2009, p. 1408. DOI: 10.3115/1699648.1699687.

[31] J. Reich, D. Tingley, J. Leder-Luis, M. E. Roberts, & B. Stewart, “Computer-assisted reading and discovery for student generated text in massive open online courses,” *Journal of Learning Analytics*, vol. 2, no. 1, pp. 156–184, November 2014. DOI: 10.18608/jla.2015.21.8.

[32] M. E. Roberts, B. M. Stewart, D. Tingley, E. M. Airoldi, *et al.*, “The structural topic model and applied social science,” in *Advances in neural information processing systems workshop on topic models: computation, application, and evaluation*, Harrahs & Harveys, Lake Tahoe, vol. 4, 2013, pp. 1–20.

[33] M. Röder, A. Both, & A. Hinneburg, “Exploring the space of topic coherence measures,” in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ser. WSDM ’15, Shanghai, China: Association for Computing Machinery, 2015, pp. 399–408. DOI: 10.1145/2684822.2685324.

[34] V. Rolim, R. Ferreira Leite de Mello, M. Ferreira, A. Pinheiro Cavalcanti, & R. Lima, “Identifying Students’ Weaknesses and Strengths Based on Online Discussion using Topic Modeling,” in *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, IEEE, Jul. 2019, pp. 63–65. DOI: 10.1109/ICALT.2019.00020.

[35] T. Shi, K. Kang, J. Choo, & C. K. Reddy, “Short-text topic modeling via non-negative matrix factorization enriched with local word-context correlations,” in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW ’18, Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 1105–1114. DOI: 10.1145/3178876.3186009.

[36] M. Sokolova, N. Japkowicz, & S. Szpakowicz, “Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation,” vol. 4304, January 2006, pp. 1015–1021. DOI: 10.1007/11941439_114.

[37] I. Sommerville, *Modernes Software-Engineering: Entwurf und Entwicklung von Softwareprodukten*, ser. Pearson. Hallbergmoos: Pearson, 2020, vol. 4396.

[38] C. Stanik, "Requirements intelligence: On the analysis of user feedback," Ph.D. dissertation, Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky, 2020.

[39] Y. Sun, K. Loparo, & R. Kolacinski, "Conversational Structure Aware and Context Sensitive Topic Model for Online Discussions," in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, IEEE, February 2020, pp. 85–92. DOI: 10.1109/ICSC.2020.00019.

[40] W. Tang, X. Wu, Y. Li, & J. Xu, "A Topic Label Extraction Method for the University BBS," in *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, IEEE, Jun. 2016, pp. 678–682. DOI: 10.1109/DSC.2016.16.

[41] J. M. Vytasek, A. F. Wise, & S. Woloshen, "Topic models to support instructors in MOOC forums," in *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, New York, NY, USA: ACM, March 2017, pp. 610–611. DOI: 10.1145/3027385.3029486.

[42] G. K. W. Wong, S. Y. K. Li, & E. W. Y. Wong, "Analyzing academic discussion forum data with topic detection and data visualization," in *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, IEEE, December 2016, pp. 109–115. DOI: 10.1109/TALE.2016.7851779.

[43] Z.-L. Wu & C.-h. Li, "Topic Detection in Online Discussion Using Non-negative Matrix Factorization," in *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, IEEE, November 2007, pp. 272–275. DOI: 10.1109/WI-IATW.2007.117.

[44] C. Yang, H. Zhang, & D. Shi, "An on-line adaptive topic evolution model in web discussions," in *2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, IEEE, Jul. 2013, pp. 847–852. DOI: 10.1109/FSKD.2013.6816312.

[45] S. Yang, T. Fan, & I. Chen, "Adding semantic retrieving concept model into the discussion board of learning community by using topic maps," in *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, IEEE, 2005, pp. 122–126. DOI: 10.1109/ICALT.2005.43.

[46] T. Zarra, R. Chiheb, R. Faizi, & A. El Afia, "Student Interactions in Online Discussion Forums," in *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications - LOPAL '18*, New York, New York, USA: ACM Press, 2018, pp. 1–5. DOI: 10.1145/3230905.3230920.

[47] Y. Zhang & H. Zhang, "Social Topic Detection for Web Forum," in *2012 International Conference on Computer Science and Service System*, IEEE, August 2012, pp. 955–959. DOI: 10.1109/CSSS.2012.242.

[48] Z. Zhang & B. Wu, “Document similarity measure for topic detection in BBS,” in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, August 2010, pp. 2354–2357. DOI: 10.1109/FSKD.2010.5569864.

[49] Y. Zhao & J. Xu, “A novel method of topic detection and tracking for BBS,” in *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN 2011)*, Piscataway, NJ: IEEE, 2011, pp. 453–457. DOI: 10.1109/ICCSN.2011.6014309.

[50] M. Zhu, W. Hu, & O. Wu, “Topic Detection and Tracking for Threaded Discussion Communities,” in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, IEEE, December 2008, pp. 77–83. DOI: 10.1109/WIIAT.2008.50.

Glossary

Backend Aspects of the software that concern calculations and server-side processes. 39, 44, 51, 54, 72, 74

Backward Snowballing Using the references of a given paper to find older, related articles. 14

Bug Defect within the code, causing unwanted behavior. 73, 74

Code Smell Characteristic within source code indicating a deeper problem or flaw. 72

Containerization The packaging of software code with just the operating system libraries and dependencies required to run the code to create a single lightweight executable — called a container — that runs consistently on any infrastructure.. 4, 6

Continuous Deployment A software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment.. 52

Continuous Integration Refers to a process of agile software development in which the system is continuously assembled from individual components. 52

Forward Snowballing Identifying related articles that cite a reviewed paper in their references. 14

Frontend Aspects of the software that concern elements seen and used by the user directly, like the user interface. 51, 58

Full Stack Refers to all parts of a software application, from frontend to the backend and database. 7

Ground Truth Information that is known to be real or true. 2, 30, 32–34, 46, 87, 98

Monolithic Software A single-tiered software application in which the user interface and data access code are combined into a single program from a single platform.. 5

Open-Source Designates software with public source code. 7, 51

Acronyms

ACM	Association for Computing Machinery. 13
AHC	Agglomerative Hierarchical Clustering. 19
API	application programming interface. 5, 7, 52, 54, 55
BBS	Bulletin Board System. 14
GUI	Graphical User Interface. 2, 58, 74
IEEE	Institute of Electrical and Electronics Engineers. 13
LDA	Latent Dirichlet Allocation. 10, 16, 51, 87, 98
MOOC	Massive Open Online Course. 16, 17
NFR	Non-functional requirement. 42, 51, 72
NLP	Natural Language Processing. 2, 16
NMF	Non-negative Matrix Factorization. 11, 18, 98
SeaNMF	Semantics Assisted Non-negative Matrix Factorization. 18, 51, 87, 98
SF	System Function. 35, 61, 75
SLDA	Sequential Latent Dirichlet Allocation. 16
STM	Structural Topic Model. 20, 21
SVM	Support Vector Machine. 18

TCS	Test Case System. 74
TDT	Topic Detection and Tracking. 9
TFM	Topic Facet Model. 16
UI	User Interface. 2, 46, 75
W	workspace. 46

List of Figures

List of Tables

Eidesstaatliche Erklärung zur Masterarbeit

Ich versichere, dass ich diese Master-Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Ich habe die Grundsätze und Empfehlungen “Verantwortung in der Wissenschaft” der Universität Heidelberg gelesen und befolgt.

Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

Abgabedatum: 17. September 2021