FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visual Active Learning for News Stream Classification

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Michael Mazurek, BSc
Matrikelnummer 01126483

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Univ.Ass. Dr.techn. Manuela Waldner, MSc

Wien, 10. Oktober 2019

Michael Mazurek	Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Visual Active Learning for News Stream Classification

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Michael Mazurek, BSc
Registration Number 01126483

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Univ.Ass. Dr.techn. Manuela Waldner, MSc

Vienna, 10th October, 2019

Michael Mazurek

Eduard Gröller

Erklärung zur Verfassung der Arbeit

Michael Mazurek, BSc
Flurschützstraße 36/12/45, 1120 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. Oktober 2019

Michael Mazurek

Acknowledgements

I want to express my gratitude to my advisor Eduard Gröller and supervisor Manuela Waldner. Thank you for the motivation, guidance, and support throughout the whole Master's thesis. This applied not only to theoretical questions in the area of visualization but also to the software implementation that improved the quality of this work. Thanks to you, I have very much enjoyed the process of researching.

This work was partially supported by the FFG Innovationsschek 864303, in cooperation with PS Quant. I want to thank the PS Quant representatives Michael Pühringer and Sebastian Schrey for their domain-specific knowledge and patient support during this work.

Additionally, I would also like to thank Matthias Zeppelzauer for valuable discussions, as well as Dea Čizmić and Martin Śmiech for their implementation of the UI/framework that this work adapts.

My greatest thanks go to my parents that support me throughout every situation in life. Without your support throughout school, I would not be able to make a small contribution to the world of research. Thank you for being who you are.

Kurzfassung

In vielen Bereichen nimmt die Menge an relevanten Textinformationen täglich zu. Viel Zeit muss in diesen kontinuierlichen Strom an Information investiert werden, um sich auf dem neuesten Stand zu halten. Deshalb haben wir ein visuelles Klassifizierungsinterface für Text-Stream-Daten entwickelt. Das Interface lässt Benutzer Daten klassifizieren um benutzerspezifische Themengebiete zu lernen.

Aktuelle Ansätze, die große Mengen an unstrukturierten Daten kategorisieren, verwenden oft vortrainierte Modelle des Maschinellen Lernens zur Textklassifizierung. Diese Modelle ordnen Textdokumente basierend auf deren Inhalt vordefinierten Kategorien zu. Jedoch, abhängig vom Anwendungsfall, können die Interessen eines Anwenders nicht in den vorgegebenen Kategorien vertreten sein. Des Weiteren sind vortrainierte Modelle nicht in der Lage, sich an neue Terminologie anzupassen. Abgesehen von diesen Faktoren, vertrauen Anwender solchen Modellen oft nicht, weil sie die Entscheidung des Modells nicht nachvollziehen können.

Um dieses Problem zu lösen, lässt unsere Anwendung den Benutzer ein Klassifizierungsproblem definieren und ein Modell des Maschinellen Lernens durch Interaktion mit einer Star Coordinates Visualisierung trainieren. Das Konzept hinter unserer Anwendung ist eine Variante des aktiven Lernens, welches aussagt, dass ein Modell des Maschinellen Lernens eine höhere Genauigkeit mit weniger Trainingsdaten erreichen kann, wenn ein Benutzer zielgerichtet Daten klassifiziert, von welchem es lernen kann. Diese Strategie adaptieren wir für Text Stream Daten, indem wir die Zugehörigkeitswahrscheinlichkeit zu einem Themengebiet des Modells visualisieren und Interaktionswerkzeuge zur Verfügung stellen, welche es ermöglichen, das Modell iterativ zu verbessern.

Durch die Simulation von üblichen Selektionsstrategien des aktiven Lernens haben wir gezeigt, dass unsere Strategien, welche auf der Visualisierung basieren, den klassischen Strategien entsprechen. In unserer vorläufigen Nutzerstudie haben die von Anwender trainierten Modelle so gut wie die besten simulierten Selektionsstrategien abgeschnitten. Deshalb kommen wir zu dem Schluss, dass die Verbindung von aktiven Lernen mit Informationsvisualisierung vorteilhaft ist.

Abstract

In many domains, the sheer quantity of text documents that have to be parsed increases daily. To keep up with this continuous text stream, a considerable amount of time has to be invested. We developed a classification interface for text streams that learns user-specific topics from the user's labeling process and partitions the incoming data into these topics.

Current approaches that try to derive content categorization from a vast number of unstructured text documents use pre-trained learning models to perform text classification. These models assign predefined categories to the text according to its content. Depending on the use case, a user's interests might not coincide with the given categories. The model cannot adapt to changing terminology that was not present during training. Besides these factors, users often do not trust pre-trained models as they are a black box for them.

To solve this problem, our application lets users define a classification problem and train a learning model through interaction with a Star Coordinates visualization. The approach that makes this interaction efficient is a variant of active learning. This active learning variant states that a learning model can achieve greater accuracy with fewer labeled training instances, if a user provides data purposefully from which it learns. We adapted this strategy for text stream classification by visualizing the topic affiliation probabilities of the learning model and providing novel interaction tools to enhance the model's performance iteratively.

By simulating different selection strategies common in active learning, we found that our visual selection strategies correspond closely to the classic active learning selection strategies. Further, users performed on par with the best simulated selection strategies in the results from our preliminary user study. Our evaluation concludes that there are benefits from incorporating information visualization into the active learning process.

Contents

CHAPTER 1

Introduction

1.1 Motivation

In many domains, it is crucial to analyze a large amount of relevant news material. For instance, in the financial domain, trend analysts interpret current events to be able to foresee financial developments. Each analyst has a portfolio that holds assets for which the analyst makes business recommendations. According to our collaborators from the financial software development domain, trend analysts gather information by browsing various sources daily to give the best advice. They examine stocks, databases, and statistics to develop their predictions. Besides those information sources, daily publications like the Wall Street Journal, The New York Times, and The Economist can provide supplementary information as they cover the latest economic information. These newspapers can even contain relevant news in sections not strictly dedicated to financial news. An analyst might look for political and world news when investigating macroeconomics, which revolves around how aggregate economies behave. On the other hand, also opinion pieces can have an impact on the financial world when examined from the microeconomics side, where analysts consider the impact of singular individuals or businesses. The sheer number of potentially relevant news appears overwhelming, even when only examining single news sources.

According to an article by The Atlantic [Mey16], a newspaper like the Washington Post produces about 500 stories per day. This number, however, only includes stories written by the editorial staff. The overall number is closer to 1200. Newspaper websites provide sections, which group articles. When examining the sections, seven out of eleven categories seem relevant to the financial domain. Assuming a uniform distribution of news among categories, this still amounts to 750 articles published a day from a single newspaper. And besides keeping up with daily news, also manual filtering of financial reports is time-consuming, as information can be overlooked in a large set of textual

information. It should be obvious that much time is spent on such analysis tasks to keep up with changes in the field.

1.2 Problem Statement

With the ever-increasing amount of news and textual content in the form of news streams, a need for new tools arise. These tools should be able to pre-process, analyze, and classify raw text to enable more efficient interaction with textual data. Typical features of such tools are entity recognition, sentiment analysis, syntactic analysis, and content classification. As this trend is now existing for a long time, there are many solutions in the form of *natural language processing* (NLP) APIs available, which are computational techniques for the automatic analysis and representation of human language [YHPC18]. A vast number of companies offer APIs due to the easy availability and flexibility that cloud-based applications provide. In this set of companies, the major players of the software world, as well as smaller companies, are represented. To name a few APIs, Amazon Comprehend [Ama], Google Cloud [Goo], IBM Watson [IBM], and Microsoft Project Entity linking [Mic] are the products of the major companies. Smaller companies like Ambiverse [Amb], Geneea [Gen], Open Calais [Ope], and TextRazor [Tex] provide similar features for NLP while being more flexible and specialized. Besides the mentioned APIs, there are many more providers that deal with NLP. Robert Dale gives a thorough overview of the current market [Dal18a] [Dal18b].

As the motivation from Section 1.1 suggests, the problem, which we set out to address, is the sheer volume of news and documents that financial analysts are confronted with daily. We chose to solve this problem by semi-automatically categorizing the data, so analysts can focus on a subset of categories, which reduces the amount of data that has to be assessed for relevancy. One NLP approach that provides a direct solution is *text classification*, which is the process of assigning predefined categories to text according to its content. In our use case, every analyst might have diverse interests, given by their portfolios. These interests can be utilized as the categories of the classification, making the categorization itself user-specific. Besides the user-specific categorization, the changing terminology associated with these categories must be taken into account. News concerning a certain financial asset may introduce new terminology over time that was not present during training. For instance, in 2015 the emission scandal around the Volkswagen brand introduced the term "Dieselgate". Text classification can be adapted to deal with such changing feature spaces.

The previously introduced APIs implement text classification together with the other mentioned features as pre-trained *learning models* (see Section 2.1). Training models with available data beforehand, so they can subsequently predict new unseen data, is the state-of-the-art approach to solving text analysis tasks. These machine learning approaches generalize well on data not used in the training process. Due to the learning approach, pre-trained learning models have predefined classes and predefined feature spaces. For our use-case, we want to incorporate the personal interest of the users

(e.g., specialized assets) into the classification process and be able to adapt the feature space over time so that it can add new features. Another shortcoming of classic machine learning techniques is that users often do not trust pre-trained models as they are a black box, only providing a label as feedback.

1.3 Requirements

In consultation with our collaborators from the financial software development domain, it became clear that an interactive approach would be beneficial to classify the news streams that financial experts face daily. The categorization should be personalized to the financial analyst's portfolio and dynamic concerning the changing nature of news. Our collaborators provided the additional requirements of minimal effort to the users and transparency of information. It is essential for them that the additional expense of training a model is not overly complex and does not outweigh the benefits of a personalized and dynamic categorization. The application should communicate its results as clearly as possible, so the process of categorization is transparent to the users. We put these requirements into concrete terms in the following list:

- **R1:** Incorporating the users' prior knowledge and interests into the classification process should be possible.
- **R2:** As relevant aspects can change over time, the application needs to incorporate mechanisms to keep the classification relevant through incremental improvement procedures to the underlying model.
- **R3:** With sufficient training of the classifier, the application should enable users to determine for which of their defined classes (i.e., assets or topics) the document is most relevant.
- **R4:** Fine-tuning of machine learning models is complex. It is necessary to design interactions intuitively so that non-specialists can perform basic tasks and understand their impact. The users should have an understanding of the effects of the training and be able to quickly asses the state of the model.

1.4 Method Overview

To meet these requirements, we compared *unsupervised* and *supervised learning* approaches (see Sections 2.3 and 2.4) with respect to the given task. Unsupervised learning is a common approach to solve NLP problems as those methods provide general solutions without the need for interaction with the process of learning. We decided against the unsupervised approach of *topic modeling* (see Section 2.4) due to multiple drawbacks. One of these disadvantages is the representation of topics in topic modeling, which is in the form of a weighted list of terms. These lists reflect which terms found in the text are frequent in the same context and thus depict the common understanding of a topic in a simplified form. To derive descriptive topics from those lists, further interpretation and annotation is needed. Topic modeling is inherently unsupervised, therefore, limited guidance can be applied to skew the results into a direction of interest.

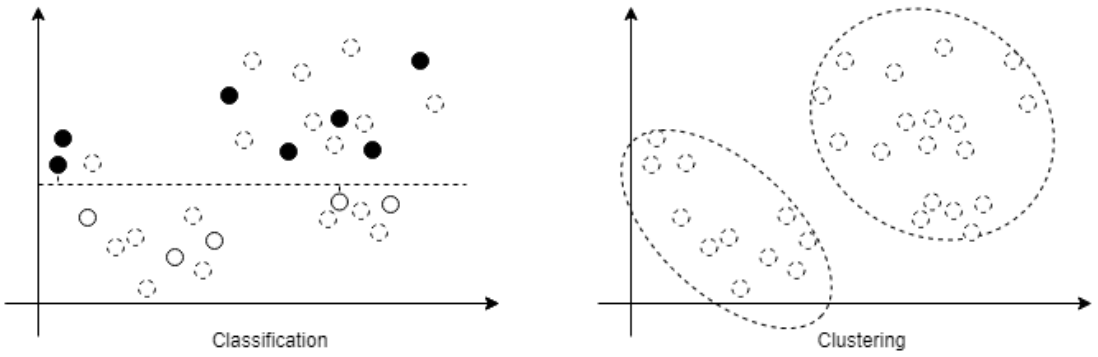


Figure 1.1: Comparison between clustering and classification. In this example, the black and white circles represent user-labeled samples, while dotted circles remain unlabeled. By utilizing this information, classification can find a subtle pattern that is indicated by users. Without the guidance of labels, a clustering algorithm might ignore this solution due to finding a more pronounced pattern.

Classification is a better fit to our personalization requirement (**R1**). It provides the possibility for users to incorporate their knowledge and interests. The financial analysts adequately define their interests through their assets, which do not have to necessarily align with patterns in the data that topic modeling picks up (see Figure 1.1). In this sense, the task is closer to grouping texts into predetermined groups than general exploration. By applying classification, the users' efforts shift to the start of the process.

Since iterative fine-tuning of a learning model is not feasible when working with a text stream, we decided to design a system, which can "tune itself" through minimal user interactions. Users can train the model with minimal interaction, incorporating their interests (**R1**), and the effects are communicated back to them through visualization. Therefore, we implemented a supervised text classification application, which learns incrementally by incorporating an *active learning* strategy into a visualization (see Section 2.5), similar

to *user-based active learning* by Seifert and Granitzer [SG10]. The model initializes the users' categories with sample documents and improves the representation by retraining iteratively. To keep the usability of the application high, we incorporated a visualization with a novel interaction design into the interface, which serves multiple purposes: First, the visualization encodes the model's results together with the confidence of each prediction, so the users can see how the model performs as a whole (**R3**, **R4**). Second, users can interact with single documents through the visualization (**R2**). By interpreting the interaction, the application derives a label for the document, which is used to improve the learning model's decision. With this approach, we aim to improve the current model's accuracy and keep the classification updated for continuous streams of text data and evolving user interests.

Our evaluation focuses on investigating the process of *visual active learning*, which is the process of applying active learning strategies iteratively with the help of our visualization to tune a learning model for a text stream. In Chapter 6, we compare our approach to classic active learning strategies (see Section 2.5) by using an evaluation tool that can simulate these strategies and their visual counterparts. We also examine how real users perform with this tool in a preliminary user study (see Chapter 7) to assess the benefits of incorporating them into the process. In these chapters, the following hypotheses are investigated:

- **H1:** Both classic and visual active learning strategies outperform a random strategy in training.
- **H2:** The visualization of document assignments to classes helps users to decide whether a news item is interesting for a particular class.
- **H3:** The direct interaction with the visualization is more efficient and effective for active learning than a classic interface.
- **H4:** Users outperform active learning strategies by adapting their strategy based on the learning model state.

1.5 Contribution

We implemented a web application for organizing streaming documents into multiple categories through interaction with a visual summary. The difference to already existing approaches is the visualization bundled with the interaction design that gives users an overview on the corpus, lets them see results of the classification, and - most importantly - enables interactive improvement of classification through interaction. The web application is accessible from every device that has a modern browser while shifting computationally expensive operations to a server. To evaluate the application, our tests examined the visualization's capability to support classic active learning strategies. We also conducted a preliminary user study comparing the visualization user interface to a list interface. The contributions of this thesis are summarized in the following list:

- An interactive visualization that facilitates efficient model training through visual active learning, and can be updated in an incremental learning fashion (see Chapter 4).
- The results of an evaluation of visual active learning strategies compared to classic active learning approaches and random selection through simulation (see Chapter 6) and the results of a preliminary user study showing that visual active learning can simulate classic active learning strategies (see Chapter 7).

CHAPTER 2

Topic Modeling and Text Classification

This chapter provides definitions and explanations from the field of machine learning that are necessary to understand the following contributions and implementation details. In particular, an overview of machine learning techniques is given at the beginning, which describes how algorithms learn to solve problems from data. Subsequently, explanations for feature engineering and classifiers follow, as these parts of the machine learning architecture play an essential role in the iterative classification process and the contribution of our application. Besides classification, the visualization community often applies topic modeling to similar problems. Therefore, topic modeling is introduced in this background chapter and compared to our classification-based approach. The chapter finishes with an introduction to active learning due to its central role in our application.

2.1 Machine Learning

A central aspect of the problem defined in Chapter 1 is the amount of data parsed daily. This is a good prerequisite for machine learning since these procedures benefit from big datasets [Bis06]. As so much data is available, our approach arranges data into useful groupings, so users have fewer documents to manually parse. The task of mapping input data to a set of categorical output variables is called *classification* [Mur12]. In this task, a rule is defined based on the present features to group the data. This rule is called the *learning model* or *model* and can be defined in different ways. One main criteria models can be categorized into is whether they are parametric or non-parametric. Shortly introduced, *parametric models*, like linear models [NKNW96], the perceptron [Ros58], or neural networks [MP43], define a finite number of parameters beforehand to fix the model complexity. On the contrary, *non-parametric models* grow in complexity with the available data. Examples include k-nearest neighbors (k-NN) [FHJ51], decision

trees [Bre17], and support vector machines [BGV92]. Due to the mentioned benefits, we chose to apply a parametric model in our application. These learning models define a parametric function on how to transform the present features in the dataset into a class prediction. At the initialization, the model only knows the family of parametric functions from which it selects a concrete function by estimating coefficients. During the learning process, the model learns the parameters of the particular function through the *training set*, which is a set of labeled data. This is accomplished through step-wise minimization of a *loss function* (i.e., *cost function*), which is a measure of loss incurred in choosing any of the available models [Bis06]. Finding the minimum of a function can be accomplished, for instance, by using the *gradient descent* algorithm [KW52]. This minimum defines the solution for the problem in the form of a parametric function. Subsequently, this parametric function is used to predict the class of new unseen data, usually called the *test set*.

When examining this process of finding a solution, it becomes apparent that the features present in the data are central. After training, the parametric model divides the feature space into distinct regions that represent the model's different choices. The features present in the data have a direct impact on the decision of the model. One might think that using as many features as possible is the right approach, so the model has all information available to find the correct answer. However, the *curse of dimensionality*, coined by Richard Bellman [Bel66] states: *"As the dimensionality of the feature space increases, the number of configurations grows exponentially, and thus the number of configurations covered by an observation decreases."*

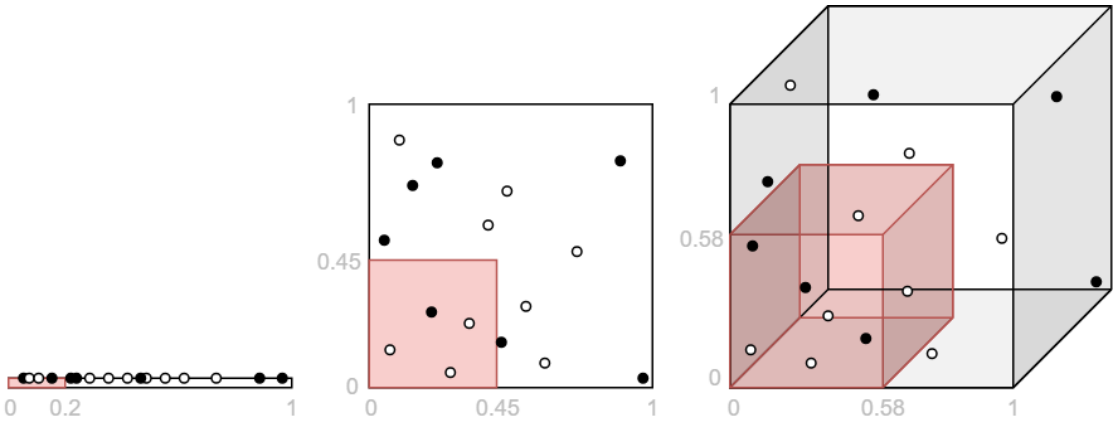


Figure 2.1: With increasing number of dimensions the amount of data needed to cover 20% of the feature space grows exponentially (Adapted from [Spr]).

Adding dimensions makes the present data fill less and less of the feature space as the data samples move apart along the new dimensions (see Figure 2.1). To maintain an exact representation of the underlying problem, the amount of data has to grow exponentially [Bis06]. This is important, as the model searches for a solution that separates the feature space perfectly. Due to the increasing sparsity, placing the decision boundary becomes

easier in higher dimensions (see Figure 2.2). A high-dimensional model will be able to fit a complex decision boundary on the data that separates the classes perfectly. This phenomenon is called *overfitting*. Such a model fits a hyperplane so well that it picks up exceptions that are specific to the training data and do not generalize well on new observations. There are many aspects to the architecture of a machine learning algorithm that can alleviate overfitting problems. In particular, this background overview will focus on feature selection techniques and choosing a learning model within the scope of text classification.

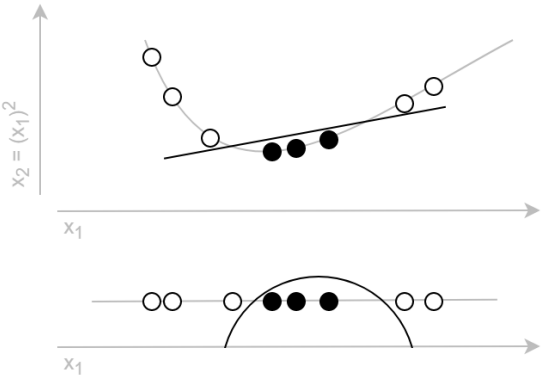


Figure 2.2: Adding dimensions increases the likelihood of separating classes successfully. Simple hyperplanes in high-dimensional spaces project to more complex hyperplanes in feature spaces with less dimensions (Adapted from [Ihl]).

2.2 Natural Language Features

It is necessary to reduce the number of features into a more manageable set for processing, to combat the curse of dimensionality. The task of finding informative and non-redundant features falls under the term *feature engineering*. Commonly, features are constructed manually by experts based on empirical tests and knowledge. Conventionally, machine learning approaches used sets of hand-crafted features as early algorithms did not implement procedures to automatically extract meaningful features [LBH15]. Besides feature engineering, also automatic techniques found application in the task of reducing the number of features. The approach of these *dimensionality-reduction* techniques is to project higher dimensional spaces into lower dimensions, while preserving structures present in the data. With the emergence of *deep learning*, the features themselves are learned alongside the machine learning model, by chaining two models after each other. These approaches encode additional information about the features into the architecture of the first model, the *feature extractor*, and connect it to a simple model to solve the problem with the features the feature extractor produced [Kim14].

Defining features in a hand-crafted manner is still a strategy widely practiced in text classification and text visualization [MP18]. Especially in cases where the available data is scarce, handcrafted features are advantageous as their quality is not as dependent on the dataset size as trained features [ZZL15]. A simple and efficient solution for text classification is to represent documents as *bag-of-words* [Har54]. This approach represents each sample of the dataset (i.e., document in the corpus) by a separate feature vector. Each of those vectors holds a set of predefined features (i.e., word frequencies) that are present in the document. This results in a word frequency vector whose length corresponds to the number of unique words present in the corpus (see Figure 2.3). These vectors, together with a label of the documents category, can then be used to train a model in a supervised fashion. Often, these methods are referred to as *traditional methods*, as these methods use handcrafted feature extractors. There are many different variants of neural networks that employ this technique [SPW+13, KGB14, Kim14, ZLR16, HQZ17].

Document d_1

Anna saw a german shepherd near a hill.

Document-Term Vector

	a	anna	german	hill	near	saw	shepherd
d_1	2	1	1	1	1	1	1

Figure 2.3: The bag-of-words procedure creates for each document a document-term vector, where all term-frequencies are saved.

Before text is transformed into a bag-of-words representation, *text normalization* operations are applied to transform all text into a standard format, so that texts from different sources can be compared consistently. The normalization process starts with the *tokenization* step, a step where the text is segmented into words. Usually, stop word removal follows after tokenization to reduce the dimensionality of the feature space, so the dataset populates the space densely. *Stop word removal* (see Figure 2.4) removes all words that are found in a lookup table. Typically, these lookup tables contain function words like articles, pronouns, prepositions, conjunctions, determiners, and lexical particles [JM14]. This approach is very effective as common words are removed that hold very limited semantic meaning for a topic. As the bag-of-words method disregards the order of words in a document, some major syntactic categories, like adjectives and adverbs, lose most of their semantics as it is not possible to determine what they refer to. Figure 2.3 highlights this problem. In the document-term vector representation of the document

it is not clear to which term an adjective refers to. An approach, therefore, can be to expand the removal process, so that not meaningful feature dimensions are removed as well. *Part-of-speech (pos) Selection* deals with these words by tagging each word of a text with the correspondent lexical category and keeping only words from those categories that preserve semantics well, e.g., nouns [Chu08].

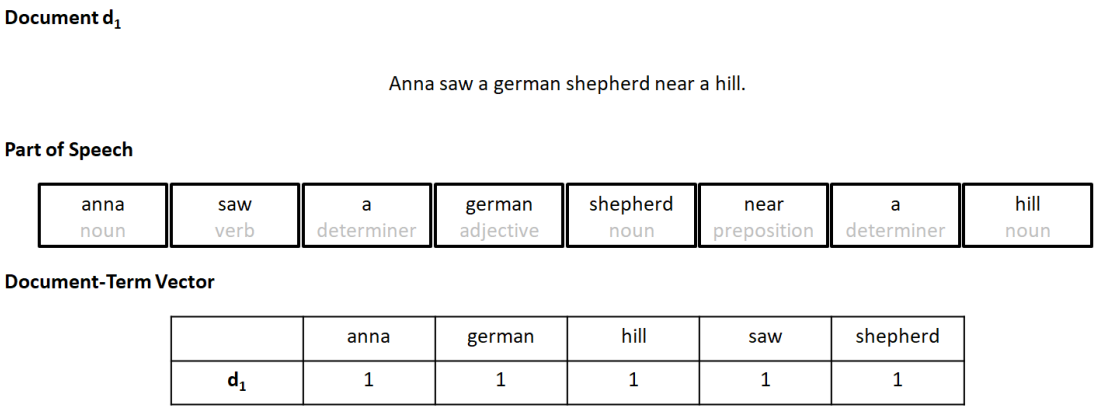


Figure 2.4: During this example of stopword removal, determiners and prepositions are removed from the text.

Another rather simple approach, that is included in the text normalization process, is to map semantically similar features into one. Multiple methods exploit different relationships between words to find such mappings. A very simple method of this kind is *stemming* [Por80]. It reduces the feature dimensionality by stripping words from their suffixes using simple rules. These rules primarily deal with conjugation rules of languages, reducing words to the word stem (see Figure 2.5). However, these word stems are not complete words and the procedure does not work with irregularities. To receive complete words, there is also an approach called *lemmatization*, which aims to map words to their lemma (i.e., the base form of the word) [JM14]. This process is more complex as it involves determining the part-of-speech of a word, and based on the lexical category, different normalization rules have to be applied. In comparison to stemming, lemmatization can grasp more information about the word being normalized, enabling it to more accurately apply normalization rules. Due to the sensitivity of the approach upon obtaining the correct lexical category, the benefits of using this approach over stemming is limited [BYRN99].

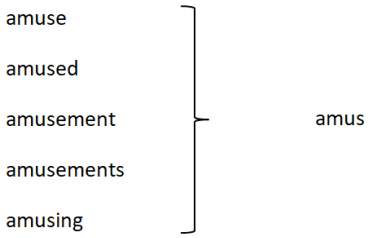


Figure 2.5: Examples of words that reduce to the same word stem by the stemming approach.

Besides text normalization itself, also the feature (i.e., word frequencies) in the bag-of-words approach can be adapted to improve performance. For instance, it is possible to incorporate *term-weighting* as not all words in texts contribute equally to the semantics. One popular weighting scheme is the *tf-idf score* by Spärk Jones [SJ72], which tries to reflect how important a word is. The score, formally:

$$tf * idf(w, d, D) = tf(w, d) * log \frac{n}{|\{d \in D : w \in d\}|} \quad , \tag{2.1}$$

weights a term w from a document d by multiplying its term-frequency $tf(w, d)$ with the inverse document frequency, a measure that determines how much information a word provides dependent on the corpus D ($|D| = n$). This weighting considers words as important if they are common in a text and also specific to it. The motivation behind this choice is that words that are frequent in a document but nowhere else hold relevant information for the document that is not available in the remainder of the corpus.

As transforming texts into word frequency vectors discards word order, it is possible to encode more complex text structures into feature vectors. An approach to preserve some of the textual order is to encode all occurring sequences of two words, called *bi-grams* [JM14], into a feature vector. Preserving the textual order of adjacent words might seem limited, however, it is potent enough to expand the range of lexical categories, which hold semantic meaning for the task. As adjectives and adverbs are found generally before or after the term they refer to, they can be incorporated more efficiently into the classification. The process of encoding sequences of words can also be expanded to longer sequences. These sequences are then called *n-grams* (see Figure 2.6).

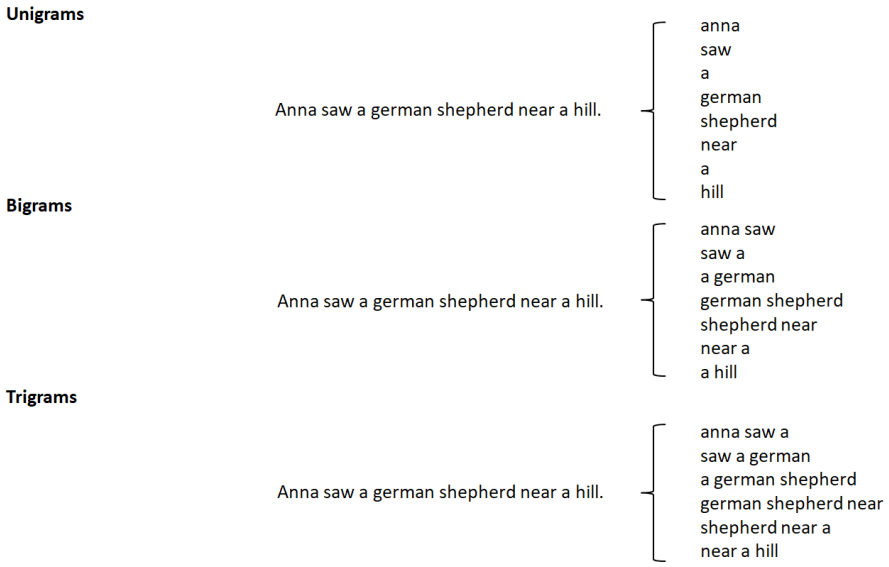


Figure 2.6: Unigram, bigram and trigram presentations generated from the same sentence.

This concept of capturing relations between words can be further expanded into the concept of *information extraction* by incorporating grammatical knowledge to derive structured information from the text. Information extraction turns unstructured information found in texts into a structured form. One central task in information extraction is *named entity recognition*, which revolves around finding all mentioned entities in a text by recognizing proper names and assigning them to a preselected group of categories. Generally, standard algorithms for named entity recognition start with labeling noun-phrases in the document with the *inside-outside-beginning (IOB) format*. In this *chunking* process, terms are grouped together into noun-phrases (see Figure 2.7). Noun-phrases are useful as they specify their entity more accurately than the noun on its own. These noun-phrases then can be tagged with specific types like person, organization, location to create named entities. In further steps, named entities can be utilized for *relation extraction*, which gives the relation between entities a structured form as an *entity relation*. There are multiple approaches to perform the labeling and classification found in these tasks [NS07]. These approaches can be split into learning algorithms [LBS+16] and rule/list-based approaches [CLR13]. Both, named entities and their entity relations, are a structured form of information that can be used to specify feature vectors for text classification.

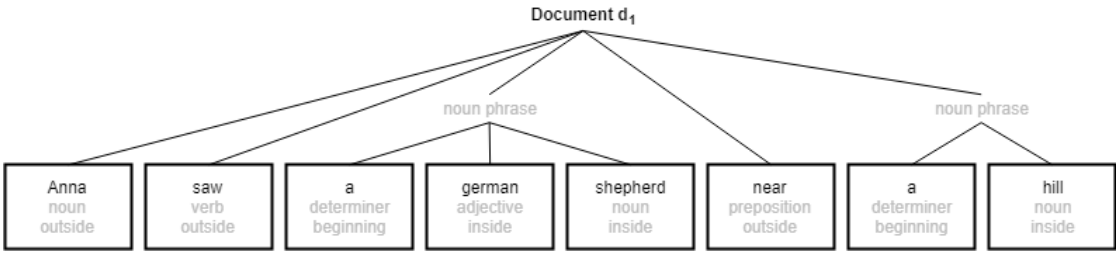


Figure 2.7: Depiction of the generated IOB structure that represents the present noun-phrases for an example sentence. The IOB tags that correspond to the drawn tree structure can be found under the part-of-speech tags.

The representation used in the bag-of-words approach has multiple drawbacks. It is invariant to word order, which leads to loss of relations between words. Representing a document by a word frequency vector creates a feature space where documents are only similar if they have a certain overlap in words. Encoding into such vector spaces provides limited information regarding relationships that may be present between individual documents [JM14]. To overcome some of these obstacles, a *vector space model* can be trained to provide a feature space. Vector space models represent text documents in a continuous vector space where semantically similar documents are mapped to nearby points. These techniques depend on the *distributional hypothesis*, which states that words appearing in the same contexts share semantic meaning [Har54], [RG65]. A commonly used vector space model is the vector space generated by topic modeling (see Section 2.4), where document association to topics is computed based on underlying structures in a collection of texts.

Predictive models try to directly estimate vectors called *word-embeddings*, where the weights in a word vector are set to maximize the probability of the contexts in which the word is observed [BDK14]. Such commonly used models are the word2vec models by Mikolov et al. [MSC+13, MCCD13] (see Figure 2.8) and the GloVe model by Pennington et al. [PSM14]. These models can be extended to go beyond the word-level to achieve a sentence-level or document-level representation. For instance, a document vector can be derived from all word vectors by weighted averaging or by combining word vectors in an order given by a sentence parse tree [SLMN11]. However, both of these approaches have their respective weaknesses. A more sophisticated approach of representing documents are the doc2vec models of Le and Mikolov [LM14]. In their work, they adapted word2vec to an algorithm that learns feature vectors, which are trained to predict words present in the document.

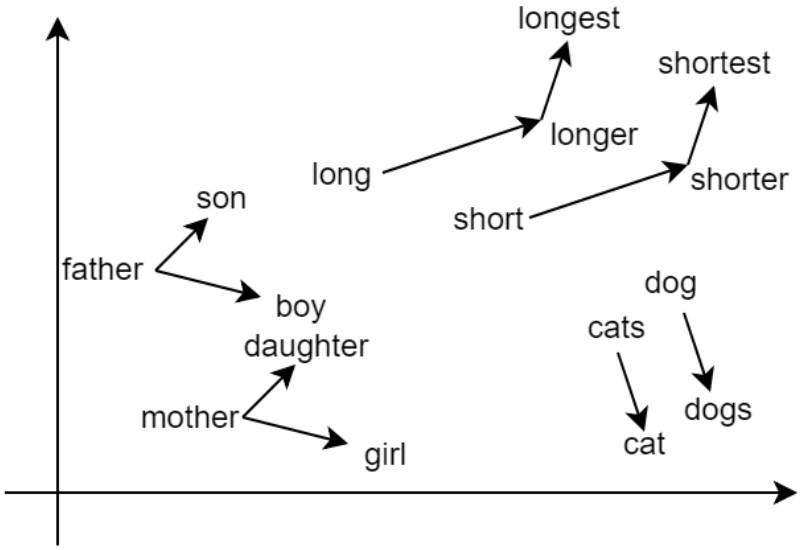


Figure 2.8: Depiction of a word2vec example space. In this example, similar terms lie in proximity of each other and similar associations are connected through the similar vectors (Adapted from [MSC+13]).

2.3 Supervised Learning / Text Classification

Besides feature engineering, we want to give a basic overview of the *classifiers*, which are the learning models used in text classification. For a broad overview, classifiers can be grouped into the following categories [AZ12], [MP18]: neural network classifiers, regression-based classifiers, and Bayesian (generative) classifiers that are parametric and decision trees, support vector machine (svm) classifiers, and proximity-based classifiers that are non-parametric. These categories of classifiers encompass a large part of classifiers in general, as text can be modeled as quantitative data with word frequencies or weighted word frequencies. Therefore, most classification methods for quantitative data can be applied directly to the text. For this overview, we will distinguish classifier in the broader categories of traditional methods and deep learning methods. In general, traditional methods differ from deep learning approaches based on how the feature engineering for classification is conducted. Traditional methods process data before classification so the learning algorithm receives a vector representation in which it can recognize important patterns. Deep learning methods contain additional network layers instead that learn to perform feature engineering themselves. Traditional methods outperform deep learning based approaches if only little data is available and the samples themselves are short [ZZL15]. In practice, traditional methods like support vector machines, logistic regression, and variants of naive Bayes classifiers are often used as baseline methods for text classification [WM12].

An early example of a traditional text classification method is the system presented by Joachims [Joa98]. Joachims used a basic variant of bag-of-words that normalizes for document length as a feature extractor and introduces the support vector machine as the classifier for this task. This work is a simple and efficient baseline for sentence classification. More recently, Joulin et al. [JGBM16] explored how to adapt traditional models to be on par with deep learning classifiers in terms of accuracy while speeding up the training time. They were able to decrease the training time by many orders of magnitude, through incorporating a hierarchical classifier and a low-dimensional text representation. The text representation they use is a combination of bag-of-words and bag-of-n-grams to alleviate the word order loss efficiently.

Besides traditional models, deep learning approaches, like the convolutional neural networks (CNN) by Kim [Kim14], have also shown competitive performances. This work demonstrates that CNNs with pre-trained word embeddings [MCCD13] can achieve excellent performance with little hyperparameter tuning. Another example is the recurrent CNN presented by Lai et al. [LXLZ15] that captures contextual information more efficiently than conventional CNNs. These two approaches from the Deep Learning field use basic linear regression for classification, as the features derived by these methods simplify the classification problem in their respective feature spaces. However, it should be noted that these models need large labeled datasets to perform best (see Table 2.1).

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Traditional Models:								
bag-of-words [ZZL15]	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
n-grams tf-idf [ZZL15]	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
fastText [JGBM16]	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6
Deep Learning Models:								
char-CNN [ZL15]	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN [XC16]	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN [CSBL16]	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7

Table 2.1: Accuracy comparison on sentiment datasets [ZZL15] from Joulin et al. [JGBM16]. In this comparison, datasets are ordered ascending in size from left to right, where the first half has a scale of several hundred thousand and the latter half goes to the scale of several million samples. This comparison illustrates how performance between both feature approaches varies based on training set size.

In our application, we chose to use a traditional model, by applying a handcrafted feature extractor together with a multinomial *naive Bayes classifier* (mnb). Shortly introduced, this probabilistic Bayes classifier works based on the bag-of-words model that uses Bayesian inference to transform known probabilities based on the present corpus and classes into the probability that a certain document belongs to a class [JM14]. Its formulation can be derived by following the *Bayes rule* (see Equation 2.2) that states how to factor a conditional probability $P(x|y)$ into three other probabilities:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \quad . \tag{2.2}$$

The naive Bayes classifier uses a simplified equation (see Equation 2.3) derived from the Bayes rule to calculate the estimate for the correct class ĉ based on the present training data. Compared to the Bayes rule, this classifier definition omits the denominator $P(y)$ as it does not change between classes. In the equation that models the classifier's decision, the likelihood $P(d|c)$ of document d being from class c is calculated as the product of the likelihoods $P(w_i|c)$ of the document's terms w_i. In practice, the likelihoods $P(w_i|c)$ are computed as the fractions of times the term w_i appears among all terms in all documents of a class c. Similarly, the prior $P(c)$ is computed as the fraction of documents of class c in the corpus:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c) \quad . \tag{2.3}$$

To compute these probabilities directly from the corpus as described, two simplifying assumptions must be made: First, the *bag-of-words assumption* is made, which states that the position of a term or feature in the document does not matter. Thus, the conditions of the probability only encode term identity. Second, the *naive Bayes assumption* is added, which states that the probabilities $P(w_i|c)$ are independent given the class and can be multiplied together. Therefore, the most probable class given a document can be computed by choosing the class with the highest product of the prior probability of the class and the likelihood of the document. This results in a simple linear classifier with solid performance, if the calculations are transformed into log space:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) + \sum_i \log P(w_i|c) \quad . \tag{2.4}$$

2.4 Unsupervised Learning / Topic Modeling

Besides classification, simplifying the process of sifting through corpora that expand regularly can also be addressed through clustering. In this section, we will explore the alternative solution of topic modeling, which is a widely used approach in the text visualization field. Rather than relying on users to tell the model what they are interested in, the model shows users the underlying structures found in the corpus. Topic modeling can supply this underlying information in the form of found topics, and users can then focus on particular parts of interest.

As shortly mentioned, *topic modeling* algorithms are a class of unsupervised machine learning algorithms, which categorize document collections based on an underlying distribution of topics discovered within [AA15]. In practice, these algorithms usually expect the data in the form of a document-term matrix $X \in \mathbb{R}^{nxm}$ together with a certain number of topics k. Related to representations mentioned in Section 2.2, this document-term matrix corresponds to a set of feature vectors constructed with the bag-of-words model. Upon finishing, they return the underlying distribution of topics in the form of two smaller matrices. The first being a document-topic matrix $H \in \mathbb{R}^{nxk}$ that holds for each document a row $\mathbf{h_i}$, which contains the soft assignment through weights to each topic. In the second output, the topic-term matrix $W \in \mathbb{R}^{kxm}$ characterizes each topic in a separate row $\mathbf{w_i}$ through term weights that signal how associated the word is with the topic.

There are two main classes of topic models: probabilistic models and non-probabilistic models. This list of presented algorithms is not exhaustive; there are multiple surveys [DLZM10], [AA15], [SLL+16], [JWY+17] for specific domains that offer a more complete overview.

Probabilistic models are more prominent as their statistical foundation gives them beneficial properties that make them simpler to interpret. Specifically, probabilistic frameworks produce representations for documents and topics that are all non-negative and sum up to one. From the class of probabilistic models, the probabilistic latent semantic analysis (p-LSA) [Hof99] and the latent Dirichlet allocation (LDA) [BNJ03] are the two models with the widest use. They use a generative model that returns the desired representation, and train the model with data from the document-term matrix through expectation maximization [Ble12]. Compared to each other, the p-LSA algorithm models a simpler generative process and therefore runs faster, while the LDA is a more complex model as it takes the priors for its parameters into account, which lends itself to better generalization. Probabilistic approaches suffer from several practical shortcomings in terms of consistency of outcome over multiple runs and empirical convergence [CLRP13].

From the group of *non-probabilistic models*, two factorization based approaches, the latent semantic analysis (LSA) [DDF+90] and the non-negative matrix factorization (NMF) [LS99], are the most prominent ones. Both approaches factorize the document-term matrix $X \in \mathbb{R}^{nxm}$ into the smaller matrices $W \in \mathbb{R}^{kxm}$ and $H \in \mathbb{R}^{nxk}$ (see Figure 2.9). The NMF factorization is often formulated in terms of the Frobenius norm, where the distance between the initial document-term matrix and the factorization is minimized. In the LSA approach, a singular value decomposition is used to compute the factorization, which introduces a third matrix $\mathbf{\Sigma}$. This matrix is a diagonal matrix whose values encode the strength of the correspondent topics in the corpus. In practice, NMF is more widely used as its non-negative constraint gives the approach many benefits of probabilistic topic models without introducing their shortcomings.

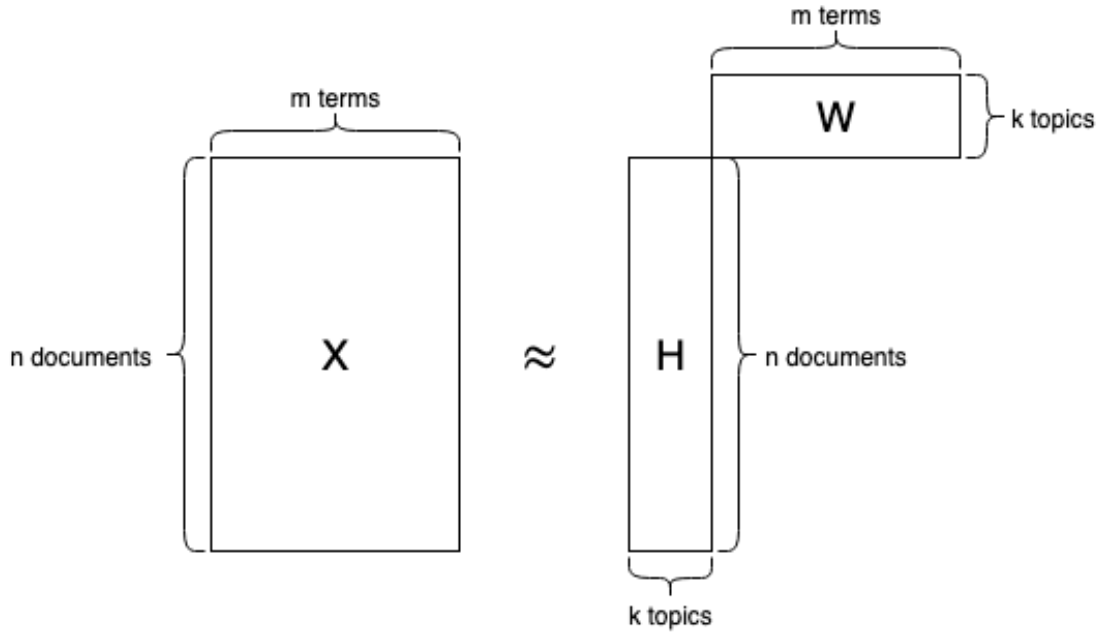


Figure 2.9: Depiction of the underlying topic representations in the form of smaller matrices generated during non-negative matrix factorization.

In the comparison between topic modeling and classification, multiple resemblances can be found. For this comparison, the process of topic modeling is generalized to clustering, as its process generates soft clusters of documents and terms concerning a latent variable called topic. In essence, Classification and clustering are both processes that follow the same goal of grouping input elements. The big difference between these methods is that classification is supervised. Clustering is an unsupervised approach. Classification needs a definition of the grouping it has to produce, which typically users provide through class examples. Therefore, it must be known beforehand how to group the data, making the classification process very dependent on a suitable definition. Clustering circumvents the problem by defining its grouping from the data, which can be advantageous. However, it leaves the process dependent on the given data features instead. In this sense, the suitability of these processes depends on user strategies or interactions to solve the given task. While classification corresponds to a more goal-oriented approach, clustering operates better in a data exploration setting. With respect to our application, incorporating the users' prior knowledge, as stated in **R1** (see Section 1.3), corresponds more closely to classification.

2.5 Active Learning

Commonly, in machine learning problems, training data is treated as a fixed and given part of the problem definition. In practice, it is frequently the case that abundant data that can be easily obtained is unlabeled. However, labels must be explicitly produced, which can be time-consuming. In this situation, where a large pool of unlabeled data is available, active learning is a possible solution. The term *active learning* refers to a method where the model itself has a role in determining what data it will be trained with [Coh17]. Active learning is often used in settings where the obtaining of labeled data is expensive and time-consuming. As this setting corresponds to our problem statement, we adapted active learning techniques as a central method of our application.

The idea behind these methods is to select training examples sequentially that enable the model to minimize its loss on future test cases (see Figure 2.10). By incorporating the model into the selection process, the model can make use of initial information to discard parts of the solution space that are sufficiently represented and focus data acquisition on uncertain regions. Active learning aims to greatly reduce the number of training samples needed and the computational effort required to achieve good generalization.

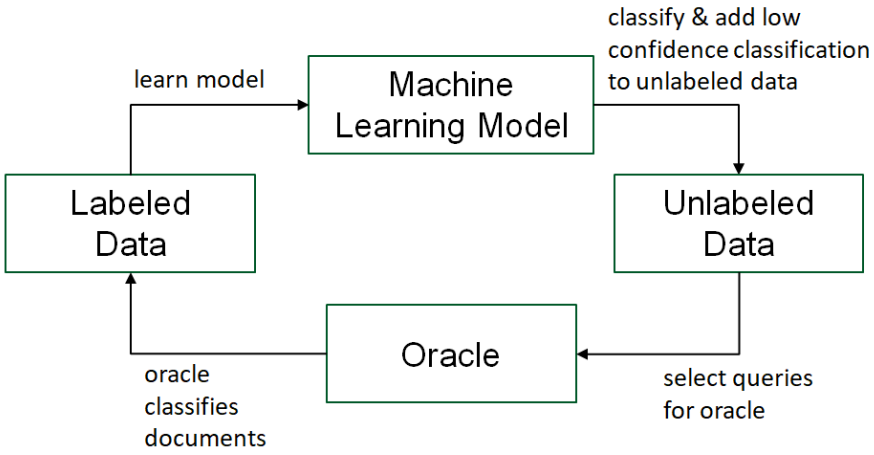


Figure 2.10: The sequential process of active learning (Adapted from [Set09]).

To incorporate active learning, a sequential selection process has to be established between the learning model and the oracle. *Oracle* refers in this case to the mechanism that provides labels for the selected data, which is usually manual labeling by a human. This sequential selection process between the model and the oracle is a loop, where the model selects a sample based on a strategy and updates itself upon receiving the label. Multiple active learning strategies define how to select a sample from the pool of unlabeled data. The earliest active learning work [RD89] relies on version space partitioning, which selects samples by computing explicit model hypotheses and choosing samples that are in maximal disagreement with these models. Query by committee [SOS92] relies on a

disagreement score from an ensemble of models. This strategy selects samples where the disagreement between models is maximal, as the disagreement signals an undetermined space in the training data. Besides these strategies, there is a selection of strategies based on statistics. Uncertainty sampling [LG94] is such a statistical strategy. For this strategy, the learning model maintains an explicit model of uncertainty from which it picks samples of least confidence. In case of particularly noisy data, the loss minimization strategy [CGJ96] is preferable. This strategy is suitable for active learning as it computes also the effect of the samples on the future uncertainty, besides the current learning model's uncertainty. However, this strategy is only feasible for models where this uncertainty can be computed explicitly.

Applying active learning sampling strategies, directly influences the data distribution the learning model is training on. So these approaches bear the risk that the dataset no longer reflects the underlying data distribution. However, the process of active learning has great potential to improve the generalization capabilities of learning models, while minimizing the computational effort. Multiple works from the field of active learning theory [Das17] illustrate these benefits for specific cases through a statistical analysis of active learning methods. In the general case, there is no clear answer on how much is gained through applying active learning methods, as the theoretical aspects of active learning are still mostly unexplored.

CHAPTER 3

Related Work

The core of this work is an interactive visualization interface for streaming text data. To get an overview of existing text visualization techniques, the Text Visualization Browser, which was developed alongside a text visualization survey by Kucher and Kerren [KK15], is a great place to start. For our overview, we adopt the coarser taxonomy of Cao and Cui [CC16], which partition existing text visualization techniques based on text data scope and analysis task groups. Existing text visualization techniques were mostly designed to deal with three major forms of text data: documents, corpus, or text streams. Based on these data types, most visualizations focus on particular analysis tasks they support in a distinct application domain. These analysis tasks can be grouped into showing similarity, showing content, showing opinions and emotions, and exploring the corpus. In the sense of this taxonomy, our visualization is a text stream based technique that visualizes document similarities.

3.1 Text Stream Visualization

Following the description from Babcock et al. [BBD+02], *text stream data* is continuous data potentially unbounded in size that arrives uncontrollably over time in no particular order. On arrival, this data has to be processed and incorporated, as the data has no particular starting point or end. Our discussion on text stream visualization focuses on how visualizations display a text corpus and capture its content in a manner that incorporates new information while preserving the connection to the previous state. Due to our application tasks, we confine this overview to visualizations that visualize the documents themselves. This excludes alternative approaches that illustrate temporal content of an entire corpus, like dynamic word clouds [CWL+10], as the contribution of particular documents is unclear.

Streamit by Alsakran et al. [ACZ+11] is a real-time text stream visualization that represents text streams as particle systems in a force-directed layout. In their layout, the position of a document particle is determined by a layout force based on pairwise keyword similarities between documents (see Figure 3.1). A two-dimensional particle-based visualization has many benefits for visualizing text stream data. It is straightforward to add particles for arriving data. Also, changes from the update can be incorporated by moving particles to their new position through an animation without any occlusion issues. To compute the similarity-based particle positioning, they adapt the cosine similarity (see Section 3.2) to dynamically change the text stream. In their adaptation of the cosine similarity, keywords are weighted with a dynamically computed importance that emphasizes frequent keywords and keywords with prolonged use. That way, they can incorporate additional temporal information, improving the layout for text stream visualization.

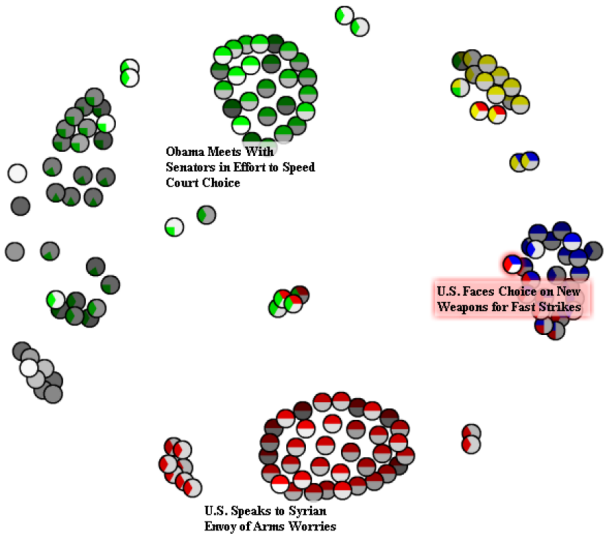


Figure 3.1: The Streamit visualization by Alsakran et al. [ACZ+11] utilizes a particle-based visualization based on keyword similarities to show how the content of a text stream evolves. In the visualization, greyscale is used to indicate the age of a document, and colored pies represent tracked keywords.

Gansner et al. [GHN13] designed a dynamic visualization that uses a node-link diagram to organize documents based on similarity and cluster them with an overlayed color-coded "country"-metaphor (see Figure 3.2). In this visualization, the similarity is defined with the cosine similarity of document tf-idf vectors (see Section 2.4) and applying multidimensional scaling (see Section 3.2). Additionally, modularity clustering [New06] based on graph edge weights is computed. It is used for cluster highlighting by color. To dynamically adapt to new data, their system includes strategies that locally apply multidimensional scaling to preserve the users' mental map of the data. To keep the unused space minimal, a packing algorithm is also utilized.

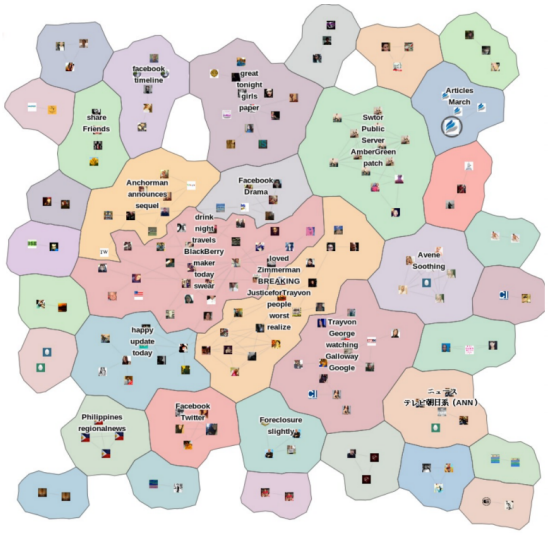


Figure 3.2: Gansner et al. [GHN13] developed a dynamic maps visualization to simplify analyzing information from text streams. They adapted node-link diagrams by overlaying cluster boundaries and developed algorithms to preserve the layout between updates.

In comparison to the discussed approaches, our application uses a similar visualization technique to capture the content of a text stream. As the text stream can add documents anytime, the discussed applications define visualizations that can be updated without losing the connection to the previous state. We also came to the conclusion that a point-based two-dimensional visualization handles this aspect well. However, our visualization captures the content of the stream by relating it to user-defined topics. As visualizing similarity between documents is an essential factor in our application, the following sections will explore concepts in this direction.

3.2 Visualizing Document Similarities

When visualizing a corpus based on document similarities, the generated visualization may contain clusters that represent given topics. This section explores similarity measures and dimensionality-reduction algorithms used to compute spaces where proximity encodes similarity. In particular, we will discuss radial visualizations, force-based visualizations, visualizations based on similarity measures, and visualizations based on machine learning.

There are multiple approaches to compute a radial visualization that juxtaposes multiple dimensions of a high-dimensional feature vector $\boldsymbol{a} = (f_{1,a}, f_{2,a}, ..., f_{n,a})$. One such visualization are Star Coordinates by Kandogan [Kan00] (see Figure 3.3). Star Coordinates are a radial visualization, which contain equiangular radial axes for each dimension of the data. The embedding, which places the data as points is computed by a linear combination of radial axes i and the corresponding feature values $f_{i,a}$ (see Section 4.5). Another algebraic approach is to normalize the n-dimensional vector representation and place documents in a regular n-sided polygon using an extension of the barycentric coordinates [HF06]. A variant of the TopicAssembly visualization by Riehmann et al. [RKKF18] uses this approach to convert their term vectors produced by LDA (see Section 2.4) into a visualization.

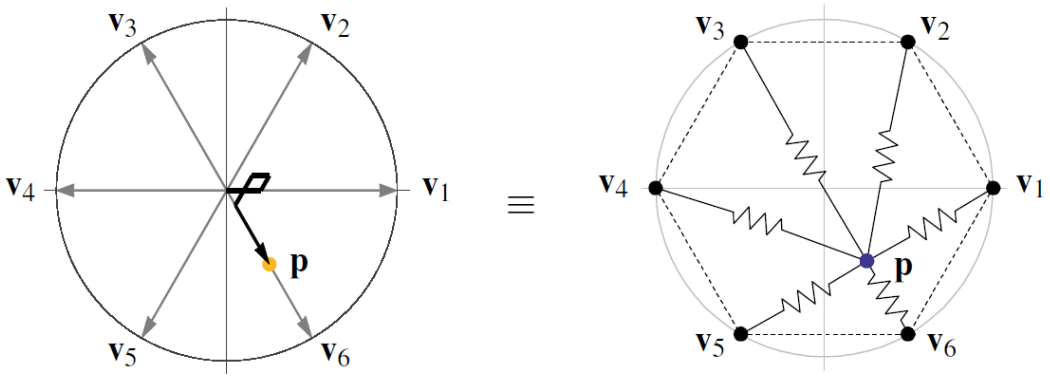


Figure 3.3: A comparison between the layout algorithms of Star Coordinates [Kan00] and RadViz [HGM+97] from a comparative study by Rubio-Sanchez et al. [RSRDS16]. Despite the different approaches both visualizations lead to a similar visual result.

Utilizing a force directed layout is another approach to create a two dimensional visualization from high-dimensional data $\boldsymbol{a} = (f_{1,a}, f_{2,a}, ..., f_{n,a})$. *Force-directed layouts* simulate different kinds of forces to generate visualizations that utilize the given space efficiently. If the data itself is taken to specify a combination of repulsive and attractive forces, the layout can reveal inherent properties found in the data. An example to create a space where proximity between documents encodes similarity is the RadViz visualization by Hoffmann et al. [HGM+97] (see Figure 3.3). This visualization implements a physical spring model metaphor. Similarly to the Star Coordinates visualization [Kan00], the n dimensions of the data are placed as radial axes equiangularly around a circle. Each document position p in the circle is determined by the equilibrium of n spring forces that connect the document's point representation to the radial axes. The equilibrium point of each document is dependent on its features $f_{i,a}$, as the spring's stiffness is dependent on the feature values.

Another option frequently used is to utilize similarity measures to compute pairwise similarities between all samples of a dataset and compute a similarity preserving space. Due to the popularity of the vector space model (see Section 2.2) the cosine similarity [MRS10] is the most popular measure, formally:

$$cos(\theta) = \frac{\mathbf{a} \bullet \mathbf{b}}{||\mathbf{a}||\,||\mathbf{b}||} = \frac{\sum_{i=1}^{N} f_{i,a} f_{i,b}}{\sqrt{\sum_{i=1}^{N} f_{i,a}^2}\sqrt{\sum_{i=1}^{N} f_{i,b}^2}} \quad , \tag{3.1}$$

where the cosine of the angle between two feature vectors $\boldsymbol{a} = (f_{1,a}, f_{2,a}, ..., f_{n,a})$ and $\boldsymbol{b} = (f_{1,b}, f_{2,b}, ..., f_{n,b})$ is computed. Both applications, covered in Section 3.1 [ACZ+11, GHN13], are examples that use a form of cosine similarity. Another measure that can be utilized is the Jaccard index that is defined as:

$$J(\mathbf{a}, \mathbf{b}) = \frac{|\mathbf{a} \cap \mathbf{b}|}{|\mathbf{a} \cup \mathbf{b}|} = \frac{\sum_{i=1}^{N} \min(f_{i,a}, f_{i,b})}{\sum_{i=1}^{N} \max(f_{i,a}, f_{i,b})} \quad , \tag{3.2}$$

where the intersection divided by the union between two feature vectors $\boldsymbol{a} = (w_{1,a}, w_{2,a}, ..., w_{n,a})$ and $\boldsymbol{b} = (w_{1,b}, w_{2,b}, ..., w_{n,b})$ is computed. Dependent on the space in which similarity is measured, even ordinary distance measures like L^1 or L^2 distances can define similarities.

To utilize pairwise similarities for visualization, a dimensionality-reduction technique is needed to define a 2D or 3D projection that preserves the computed similarities. The standard techniques encompass a principal component analysis (PCA) [Jol11], multidimensional scaling [Kru64], and the t-distributed stochastic neigborhood embedding (t-SNE) [MH08]. From this selection of techniques, t-SNE is commonly used for visualization due to its capabilities in revealing implicit groupings [CLRP13, PSPM15, KKP+17, LCL+19]. The algorithm works by calculating probabilities for similarities between documents in the high-dimensional input space and low-dimensional target space. The low-dimensional representation is found by minimizing the difference between both sets of probabilities. One notable work that utilizes a constrained version of t-SNE to contextualize uncertain instances is the instance visualization from Liu et al. [LCL+19] (see Figure 3.4). This visualization is a cicular-based constraint layout, with the outer arcs representing the classes while the instances are placed inside the circle. In their instance visualization, they enable users to improve annotations of uncertain cases by encoding the assignments of annotators in uncertainty glyphs and displaying them alongside the rest of the dataset that is encoded with simple dots. By examining the uncertainty glyphs and the surrounding instances in this space, users can resolve complex cases through comparison to similar examples.

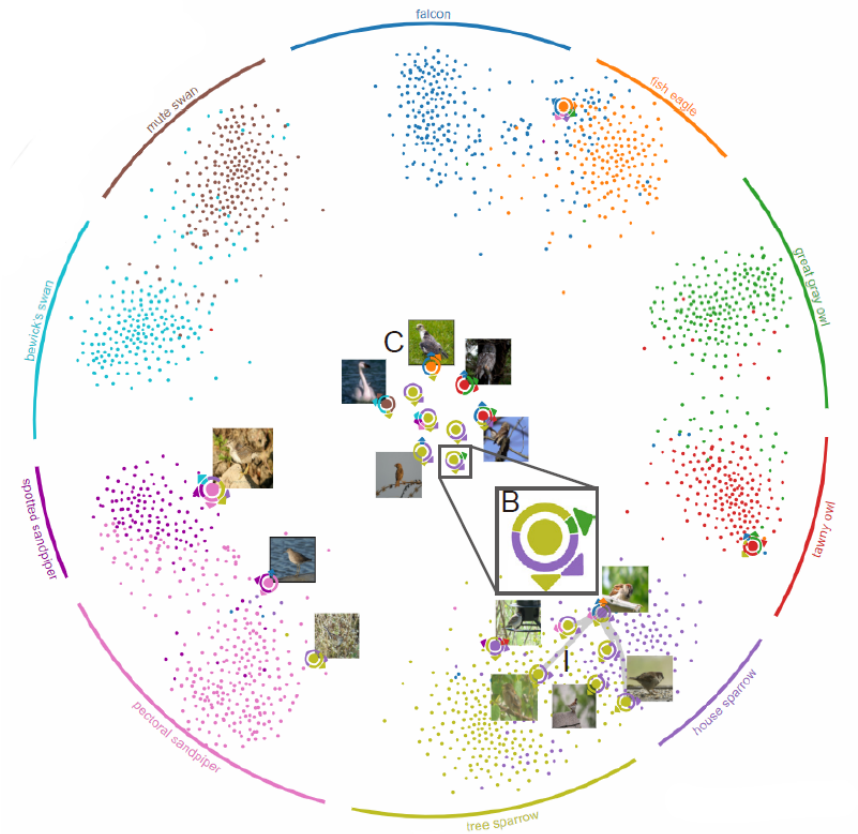


Figure 3.4: The instance visualization from Liu et al. [LCL+19] uses a constrained t-SNE algorithm to compute a layout for showing cases that need annotation in the context to the whole dataset.

Alternatively, machine learning approaches like topic modeling and classification can be an option to derive a low-dimensional representation of a corpus. Sections 2.3 and 2.4 introduced a selection of techniques from these fields that can be applied for this task. Dependent on the output dimensionality these learning models provide, they can also be combined with the previously mentioned dimensionality-reduction techniques or the radial embeddings to generate visualizations.

Compared to the above-mentioned techniques, our approach reduces the dimensionality of our high-dimensional corpus data by a user-specified classification. Since a focus lies on displaying the data in reference to this classification, we chose an encoding that is flexible with respect to the number of classes and can display all the dimensions at the same time. Therefore, we chose the Star Coordinates visualization, as it treats all of the classes equally. However, to generate a layout, where each document's position is relative to the users' interests, the classes have to be trained sufficiently. We decided to incorporate the model training into the visualization through direct interactions. Consequently, the next section will discuss visualizations, which interactively facilitate model training.

3.3 Steerable Model Visualization

Beyond interpreting a machine learning model, visualizations can be also used to steer a model to reflect user interests. Users can benefit from such interaction, as interacting with a visualization with which they are familiar may be simpler than performing formal updates to the model [EFN12]. In this section, we will take a closer look at approaches that explore how to design interaction tools for visualizations enabling users to change unsupervised and supervised models. Besides the mentioned focus on changing the learning model, there are multiple other techniques employed in the visual analytics field that integrate machine learning approaches. An overview of these techniques can be found in several survey papers [SZS+16, ERT+17, LGH+17, LWC+18].

An intuitive option to influence the results of unsupervised learning models is to change their parameters. In machine learning, commonly the parameter space of the learning model is searched to maximize performance in a process called *parameter tuning*. Brown et al. [BLBC12] adapted this process for the k-nearest neighbor algorithm. They developed an interactive node-link visualization for observing which instances of a dataset are considered to be similar based on a distance metric. They included functionality to directly manipulate the visualization to redefine similarities (see Figure 3.5). The user interest model from the StarSpire visualization [WBD+18] follows a similar idea. This model stores for each document an interest level that is computed as the sum of the weighted term-frequency vector-elements. While interacting with the node-link visualization, the user interest model updates the visualization through thresholding the visible documents based on the given interest level. The list of interpreted interactions includes opening, moving, pinning, overlapping, minimizing, and removing documents on the document level and searching, annotating, and highlighting text on the text level.

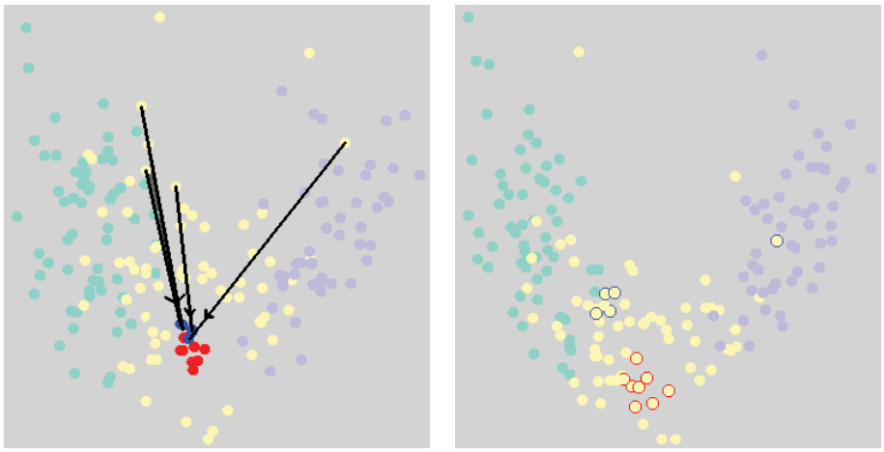


Figure 3.5: Example of an interaction that changes the similarity metric from Brown et al.'s Dis-Function visualization [BLBC12]. Dragging the blue documents closer to the red documents, causes the similarity metric to group both document sets closer together. In the updated metric both document sets are outlined with their corresponding colors.

Cavallo and Demiralp [CD18] provide another visual approach to improve exploratory data analysis of dimensionality-reduced data. The core of their work is *forward and backward projection*. These techniques enable to reason about the model by direct interaction with the input or output. On interaction, the modifications made to one set are projected onto the other one while respecting the dimensionality-reduction model between them. To facilitate the effective use of these interaction tools, they developed two visualization techniques that are used in combination with a scatter plot (see Figure 3.6). Forward projections enable users to interactively change feature values of a data sample and observe how these hypothesized changes in the data modify the current projected position of the sample. For an overview of forward projection, prolines show forward projection paths, which illustrate the path the sample would take if a certain feature dimension is adjusted. Forward projecting by changing the parameter in regular steps creates these prolines. By combining multiple prolines, a directional coordinate system based on original features is projected into the dimensionality-reduced space. Backward projection is a complementary interaction that enables the manipulation of the output, which in turn modifies the input of the dimensionality-reduction. During this projection, multiple points in the initial multidimensional space can project to the same position. Therefore, it makes sense to constrain backward projection, so users get tools to regulate the reverse mapping into the high-dimensional input space. The feasibility map is an area-based visualization that shows regions where the constraints are broken. This gives an overview which positions in the dimensionality-reduced space satisfy the desired constraints.

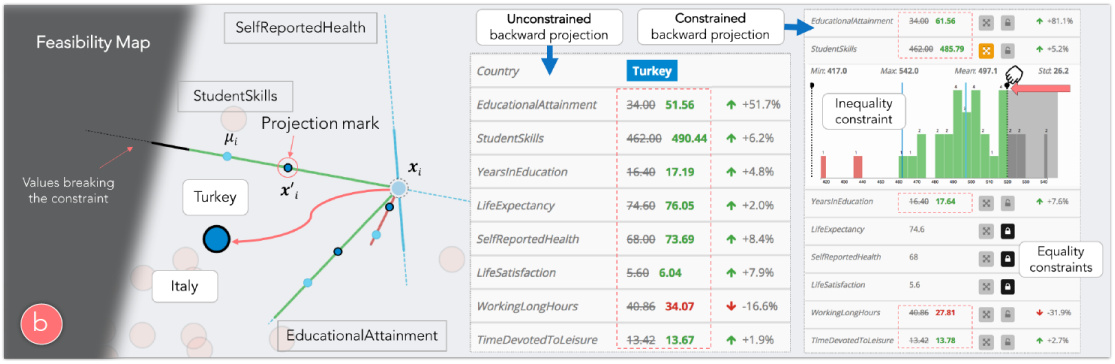
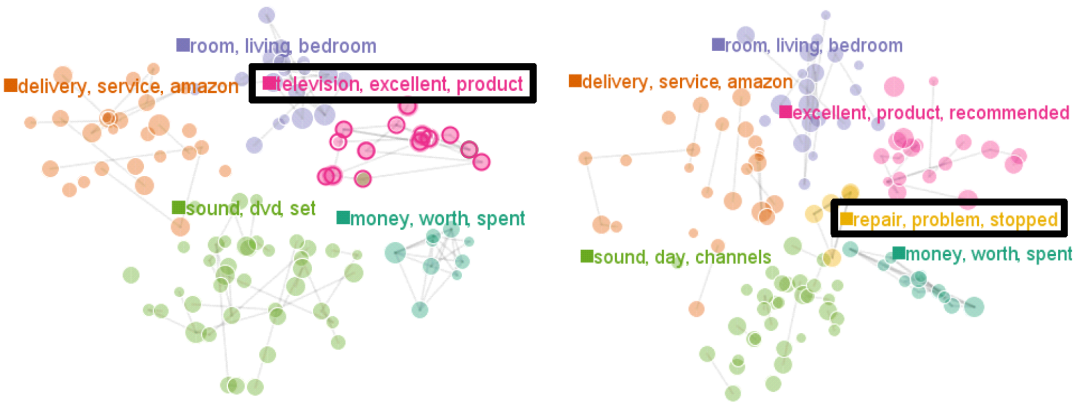


Figure 3.6: Example from the interaction framework by Cavallo and Demiralp [CD18]. Adding prolines to the dimensionality-reduced scatter plot provides a coordinate system in terms of the original features. The feasibility map shows where a document point could be placed by the dimensionality-reduction technique without breaking a set constraint.

To enable even more user interactions with topic models, some approaches interfere with the loss function (see Section 2.1) of the learning model directly. Choo et al. [CLRP13] developed such an approach that adapts NMF (see Section 2.4) for topic modeling into a semi-supervised variant. By adding some terms reminiscent of regularization terms in a minimization problem, the solution can be shifted so it takes the users' input into account while still approximating the input matrix $\mathbf{X}$. They utilize a semi-supervised NMF (SS-NMF) in an adapted node-link visualization (see Figure 3.7), and animate explicit topic cluster changes when users change the underlying model. The SS-NMF algorithm enables multiple interaction tools that provide the functionality of refining topic keywords, merging and splitting topics, inducing new topics through documents, and inducing new topics through keywords. Similarly, El-Assady et al. [EASS+18] add interaction tools into topic modeling approaches by applying reinforcement learning to an LDA model. Their reinforcement learning approach includes users into the learning process through an organized task structure that ends with a relevance feedback mechanism. There, users can decide to reward or punish the model by moving documents towards or away from certain topics.



(a) The initial visualization

(b) Document-induced topic creation

Figure 3.7: An example of document-induced topic creation from Choo et al.'s [CLRP13] UTOPIAN framework, enabled by a semi-supervised variant of the NMF.

Some supervised learning models include interaction tools for data exploration that are specific to them. One such technique that can be adapted to be interactive is active learning, in which classifiers are trained with user annotated data (see Section 2.5). Heimerl et al. [HKBE12] designed an interactive training framework based on active learning for binary classifiers. Their main view is a scatter plot, which depicts how the corpus relates to the decision boundary in terms of uncertainty (see Figure 3.8). This view provides interaction mechanisms for selecting documents, which can subsequently be classified in the labeling control view, updating the whole system. More recently, Huang et al. [HMdCM17] investigated the same concept by developing an interactive labeling interface that also uses active learning for binary classifiers. Their system provides node-link diagrams that create a layout where proximity encodes similarity and a chord diagram that shows how strongly a document is connected to both classes. These visualizations help users in the selection of documents to classify.

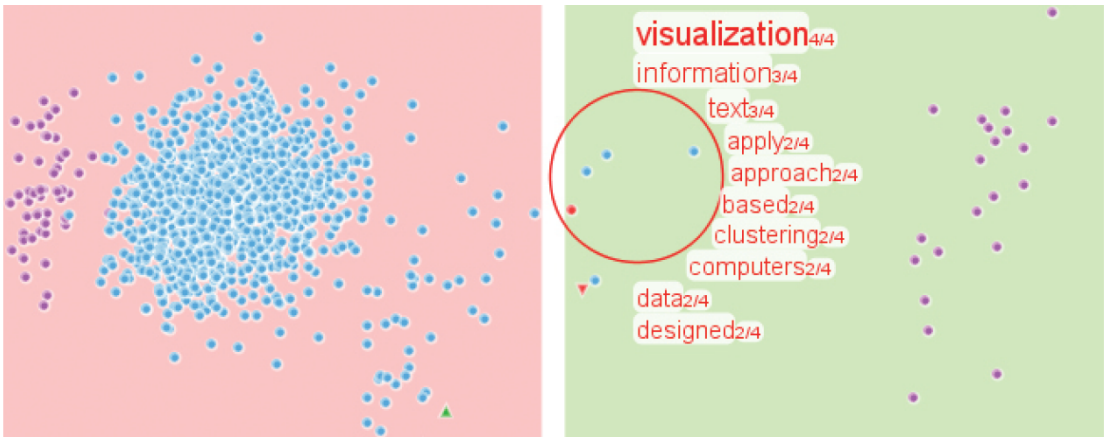


Figure 3.8: Heimerl et al.'s [HKBE12] scatter plot visualization that depicts the corpus in relation to the decision boundary based on uncertainty. The colored regions of the scatter plot split the dataset between both classes. Between the regions, the decision boundary is depicted as the white separation. Documents encode their uncertainty based on the distance to this decision boundary.

An approach that goes beyond binary classification is presented by Seifert and Granitzer [SG10]. They developed a radial visualization, which can be used to select and classify samples in an active learning approach. Their system includes a RadViz visualization [HGM+97], which projects unlabeled examples based on the classifier's a-posteriori output probabilities (see Figure 3.9). Paiva et al.'s [PSPM15] visual classification methodology is a more recent active learning approach, which focuses on the efficient application and rebuilding of the model. Beyond model updates, their methodology also supports model creation and classifier tuning. They integrate a neighbor-joining tree into the classification pipeline to support control over the whole classification pipeline. The neighbor-joining tree visualization preserves many of the benefits of the other point-based techniques mentioned, while depicting similarity relations more clearly through its tree structure.

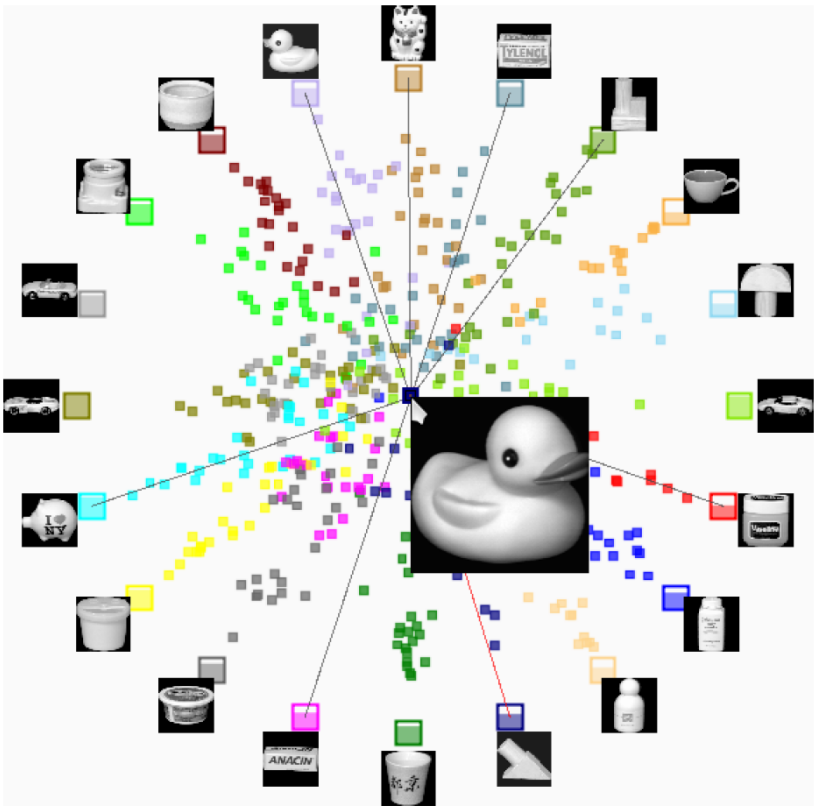


Figure 3.9: The RadViz visualization of a-posteriori classifier probabilities by Seifert and Granitzer [SG10].

As our learning model is supervised, we decided to implement an active learning approach in our application. Therefore, our application has many similarities to the supervised interactive frameworks that utilize active learning. Our augmented Star Coordinates visualization (see Section 4.5) bears some similarities with the visualization approach by Seifert and Granitzer [SG10]. However, our visualization focuses on supporting the classification process iteratively for text stream classification, emphasizing the changes in the test and training set through animation. In the sense of iterative model improvements, our application resembles Paiva et al.'s [PSPM15] visual classification methodology that aims to aid the whole classification process. In contrast to Seifert and Granitzer's visualization, we also show documents from the training set in our visualization for reclassification and reference purposes. Also, our application implements tools to manipulate the test set beyond classification for a model update. Our application implements basic addition and removal functionality for both test and training set, so the classification model can be updated over time and applied to text streams. This implementation allowed us to perform a first pilot study with actual users to get first evidence about the usefulness of such an approach.

CHAPTER 4

Visual Active Learning

This chapter describes our application with our visualization and its technical functionality. In the course of the chapter, the design of the classification model, the visualization, and the interaction design will be discussed. We will start by giving an overview of the application in Section 4.1 and going into detail in the following sections.

4.1 Overview

The requirements from Section 1.3 define a customizable classification pipeline that is dynamic in regard to incoming text streams. Our application defines a classification task during initialization that can be updated by users and the text stream during runtime. Figure 4.1 depicts crucial points in the application pipeline that will serve as a guideline for the following discussion. Additionally, these points are labeled with the respective section number. In the following paragraphs, we provide an overview of these pipeline sections and clarify how they relate to the requirements from Section 1.3.

Initially, our explanation revolves around the definition and initialization of the system. In principle, the application classifies text data, so the pipeline includes feature engineering (Section 4.3), a learning model (Section 4.4), and the Star Coordinates visualization to display results (Section 4.5). According to **R1**, users should be able to incorporate their knowledge into the classification. Therefore, during initialization users define the classification problem (Section 4.2). In this step, examples have to be provided on which the learning model can train to distinguish between different classes, which we refer to as *topics*. After the model training and prediction finishes, the Star Coordinates mapping for the visualization is finalized by optimizing the topic order with the help of a heuristic (see Section 4.5). The initialization finishes after the initial model training, when the visualization of the topics based on the classification definition is shown. This workflow is depicted on the left side of Figure 4.1, in the initialization column.

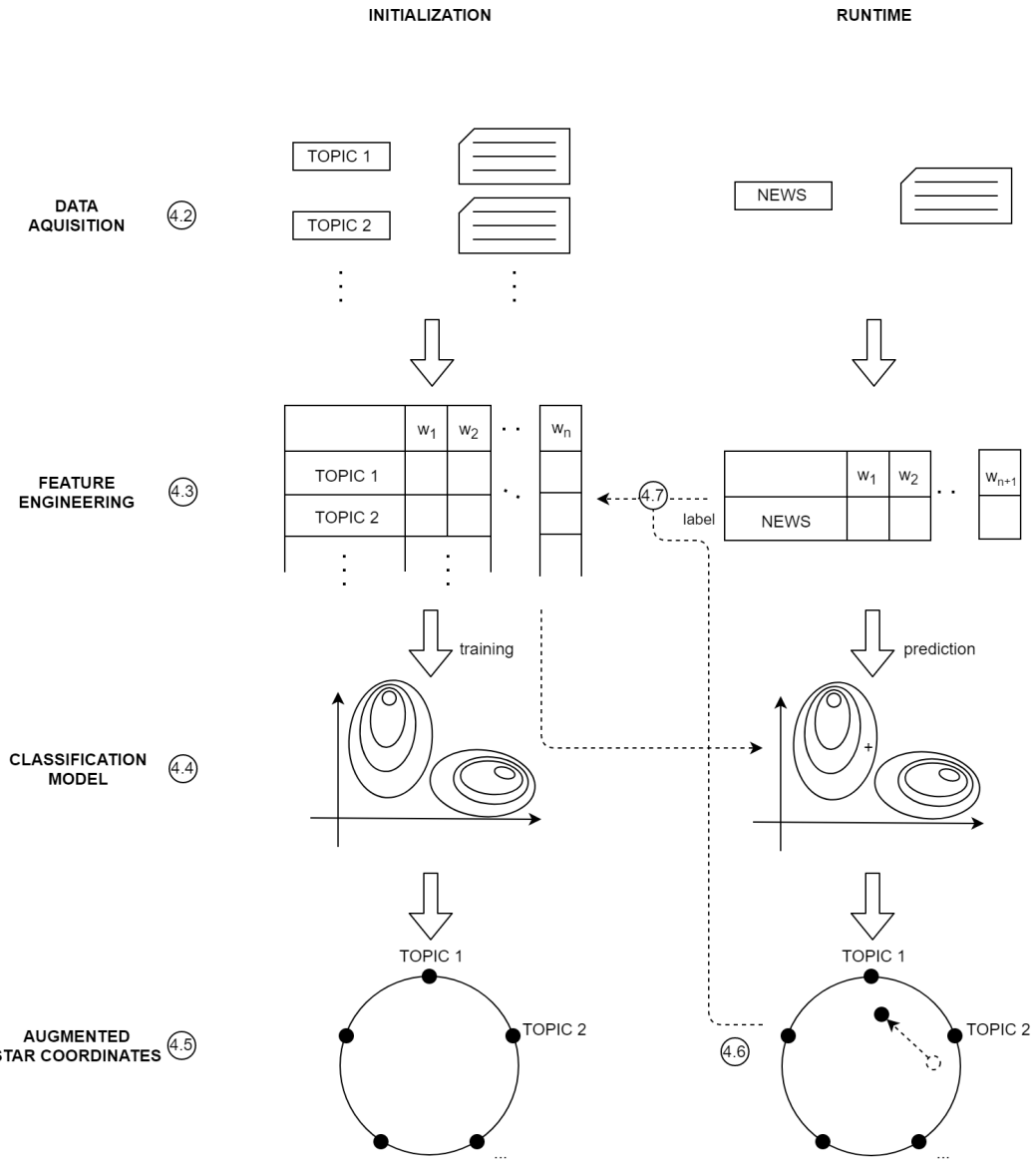


Figure 4.1: Overview of the application pipeline, emphasizing the definition and initialization of the method (4.2-4.5) in the left column and displaying the incremental update capabilities (4.6, 4.7) in the right column.

Following the initialization, documents are added from a text stream to perform classification during runtime. In a process similar to the initialization, documents pass through feature engineering (Section 4.3). The resulting document data is then passed to the learning model for prediction (see Section 4.4) and visualized in the Star Coordinates visualization (see Section 4.5). This process is depicted on the right side of Figure 4.1, in the runtime column.

Following **R2** and **R4**, our application needs an understandable mechanism to incrementally improve the learning model. To meet those requirements, our visualization is a direct manipulation tool (Section 4.6). Users can drag document points to a label, which triggers an iterative model retraining process (Section 4.7). On training completion, the model incorporates the new document and updates its prediction for all unlabeled documents, which subsequently triggers an update in the visualization. This means the model update is visualized through the updated document point positions. Our visualization provides a clear mechanism for improvement by letting both user and model communicate through changing document placements. Beyond incremental improvements, **R3** states that with sufficient training, the application shows whether a document is relevant for the user-defined topics. We fulfill this requirement through the visualization, where the mapping places documents with similar predictions close to each other. The distance of the document point to a topic point encodes the relative relevance for that particular topic. Together, these features create the customizable classification system described in the requirements. How the interactive functionality integrates into the system is depicted by the dotted lines in Figure 4.1. Moving a document in the visualization (Section 4.6) triggers a model update (Section 4.7) and a visualization update, forming an interaction loop.

4.2 Data Acquisition

Our application needs two types of text documents to define a classification task: labeled documents and documents to predict. In machine learning terms, the labeled documents that describe the topics are the training set and the documents in need of classification form the test set. First, users have to specify topics, which the application utilizes as the classes of the classification. These topics have to be defined by a set of labeled surrogate documents. As the goal of these surrogates is to create distinct term vectors for the topics, they can have multiple forms. These documents can be short textual descriptions or even keyword lists of the topic. In our tests, we constructed these document surrogates by selecting paragraphs from Wikipedia that describe the corresponding topic. Also, documents from the corpus could be used to initialize a topic of the classifier.

The second set contains text documents that need classification (i.e., the actual news items). These documents define the test set for classification. Our application simulates a text stream from different data sources (see Section 6.1) to mimic how news becomes available under realistic conditions. After the initial learning model has been created, the visualization displays the predictions concerning these documents.

4.3 Feature Engineering

After the users provide textual data, the application proceeds to a document processing step. To classify text data, they have to be transformed into a feature representation, which the classifier understands. Each document is represented by a term vector, where the features are terms that are present in the document. For our approach, we chose to create a traditional feature extractor (see Section 2.2). Our feature extractor operates by using bag-of-words to transform raw document data into a vector representation. An example is depicted in Figure 4.2. Feature extraction starts with processing text by using a tokenizer to break down documents into term vectors and labeling the term vectors with their correspondent part-of-speech. Subsequently, the part-of-speech tagged terms combine into composite tokens in a process called chunking. Applying only part-of-speech tags to determine how to combine tokens correctly is not always sufficient [BKL09]. In our application, a machine learning model creates chunks from the tokens. Those tokens are then passed to a classifier, which is trained to identify named entities. The named entities mark the end of our feature extraction process and are utilized as features for our learning model. To finalize the feature engineering step, the extracted term vectors are combined into document-term matrices that are passed to the classification model.

As the goal of this application is to provide classification by training on small size datasets with a few hundred documents, we decided to use a traditional model. We base this decision on multiple empirical comparisons between text classification models, which indicate that traditional models perform better on such training set scales [ZZL15, WM12]. During our active learning procedure, users increase the feature space iteratively by adding new documents, which in turn can grow the feature space rapidly. To mitigate the problems introduced by big feature spaces (see Section 2.1), we restrict it to a set of very distinct and descriptive features - namely, named entities.

4.4 Classification Model

After feature extraction, a multinomial naive Bayes classifier (see Section 2.3) uses the derived features to learn to discriminate between the defined topics. In this context, we refer to the classes of the classification as topics, as this corresponds to the anticipated task for the application to solve. However, this concept can be applied to a broad range of classification tasks as well. Therefore, besides the topic-labeling task, our topics can also represent other class types to perform tasks like sentiment analysis, language detection, and intent detection.

To start the training process, data is passed in the form of two document-term matrices: the *topic document-term matrix* holding the training set and the *input document-term matrix* holding the test set. Elements in these matrices indicate how often a certain term w_i is present in a document d_i or topic tp_i respectively. The classifier trains to predict the topic affiliation based on the observed data, using the observed terms of the training set. To predict the test set, the terms of the test set are aligned to the training set terms

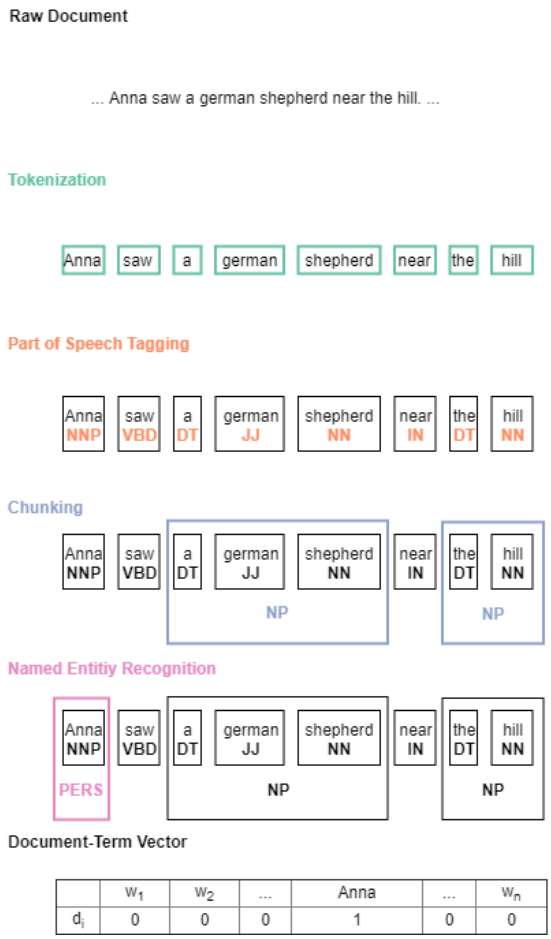


Figure 4.2: All steps of our feature extraction process demonstrated on a sentence.

Abbreviation	Meaning
DT	determiner
IN	preposition
JJ	adjective
NN	noun singular
NNP	proper noun singular
VBD	verb past tense
NP	noun-phrase
PERS	person entity
d_i	document i
w_i	term i

Table 4.1: Abbreviations of Figure 4.2.

(see Figure 4.3). Subsequently, the trained model computes topic affiliation probabilities for the current test set. We designed this procedure so our system can perform the classifier training and prediction step multiple times in an efficient manner. This choice is directly tied to the incremental model improvement approach of our system, as these operations are repeated, when users classify test set documents to improve performance gradually (see Section 4.7).

We decided for the multinomial naive Bayes classifier based on multiple factors. In the field of text classification, naive Bayes classifiers are particularly popular in commercial and open-source spam filters [MAP06, FSZ+16]. Generally, naive Bayes classifiers perform very well in comparison to other machine learning models despite their simplicity [NJ02, WM12, ZZL15]. Besides performance, the classifier has multiple properties benefiting our incremental improvement approach, where after each interaction, the system has to train a slightly adapted model. Naive Bayes classifiers can be trained quickly,

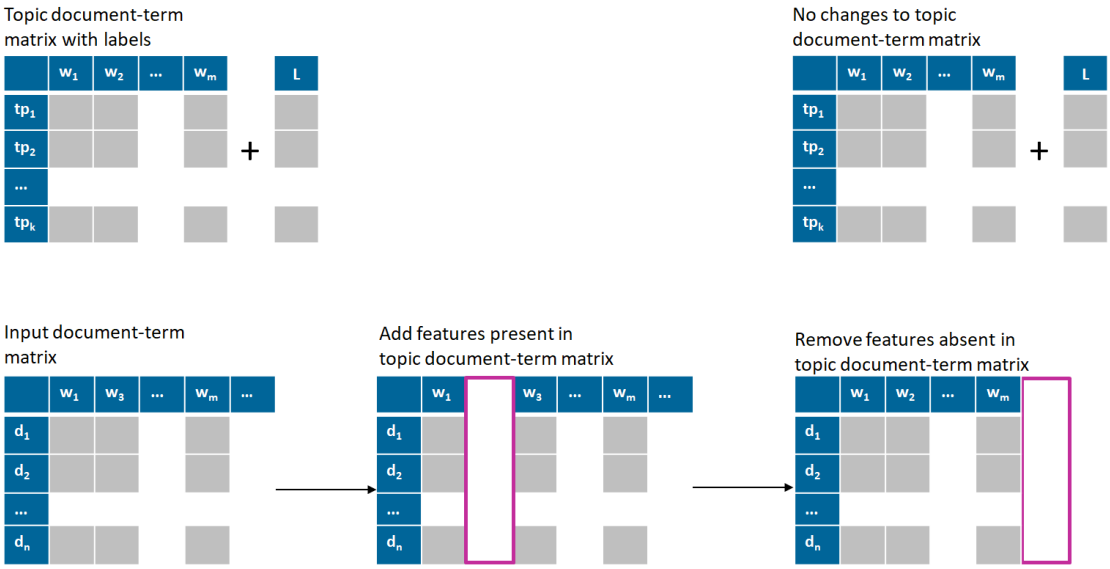


Figure 4.3: Depiction of the document-term matrix alignment process: Initially, the input document-term matrix (bottom) has different terms than the topic document-term matrix (top). During the allignment process, the input document-term matrix is changed by adding missing terms (w_2) and removing terms not present in the topic document-term matrix.

as they need very little explicit training compared to other classification methods. An update is performed by recomputing parameters for the features' probability distributions. During the interaction, users may add documents to the training set and thus features to the model, resulting in additional feature dimensions. Naive Bayes classifiers employ linear decision functions, which prevent them from overfitting to training data in high-dimensional spaces. In a system where the feature space is iteratively expanded upon, triggering a recomputation of the model every time, fast training times and linear decision functions are very beneficial.

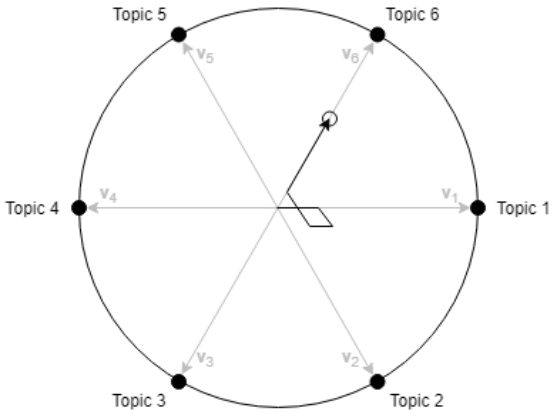
4.5 Augmented Star Coordinates

Before the process of incremental improvement starts, our system needs a way of communicating classification results and the state of the model. Both of these tasks fall under the responsibility of our visualization, a variant of the Star Coordinates visualization (see Section 3.2) by Kandogan [Kan00].

In our application, Star Coordinates visualize topic affiliation probabilities for each document provided by the classifier (see Section 4.4). To represent the topic associations of the documents graphically, the Star Coordinates generate a linear mapping from the n-dimensional affiliation space onto a two-dimensional space. In this space, the documents are then placed as points, where the positions of the points encode the topic affiliations.

For the purpose of computing document placements, the mapping places n vectors with a common origin that represent radial axes. Each vector $\boldsymbol{v}_i$ is associated with the i-th affiliation probability p_i (see Figure 4.4). At the endpoints of the axis vectors $\boldsymbol{v}_i$, topic points with topic labels are placed to label these axes with the user-assigned name. The two-dimensional embedding $\boldsymbol{x} \in \mathbb{R}^2$ of the document's topic affiliation $p = (p_1, p_2, ..., p_n)$ can be computed by a linear combination of the vectors $\boldsymbol{v}_i$, where the linear coefficients correspond to the single affiliation probabilities of $\boldsymbol{p}$, formally:

$$\boldsymbol{x} = p_1\boldsymbol{v}_1 + p_2\boldsymbol{v}_2 + ... + p_n\boldsymbol{v}_n \quad . \tag{4.1}$$

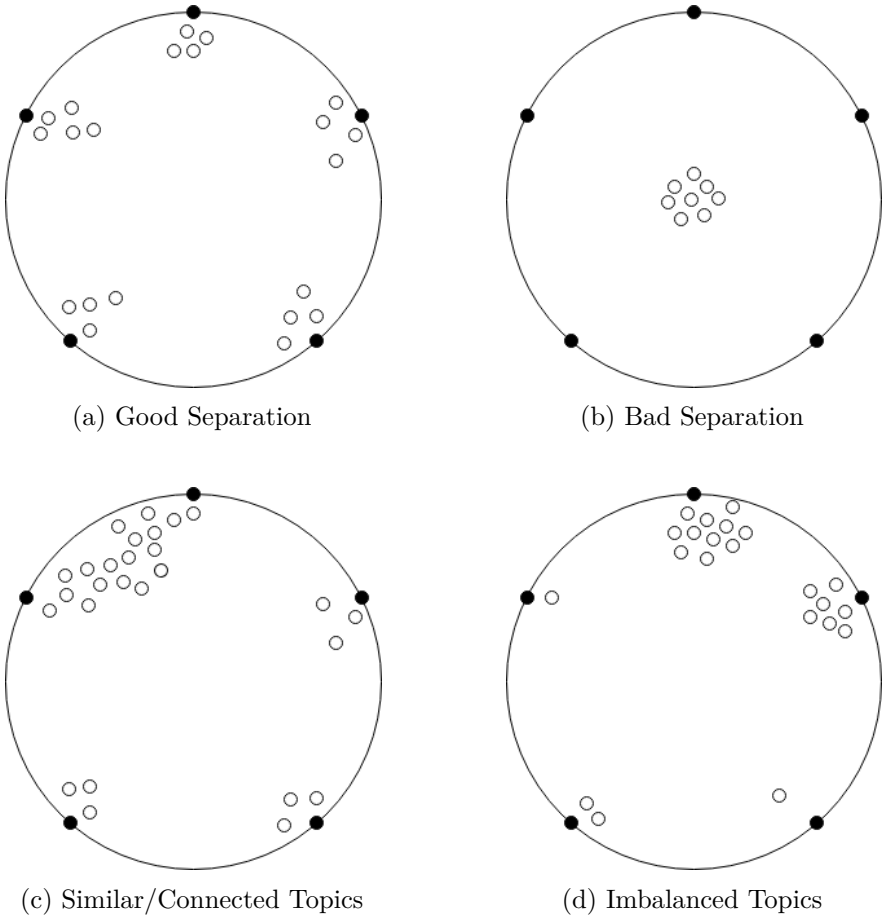


Topic	1	2	3	4	5	6
Probability	0.2	0.1	0.0	0.1	0.2	0.4

Figure 4.4: Document placement $\boldsymbol{x}$ in the visualization is determined by a linear combination of axis vectors $\boldsymbol{v}$ that are weighted by topic affiliations p_i (see Equation 4.1) (Adapted from Rubio-Sanchez et al. [RSRDS16]).

The resulting visualization communicates both, classification results and information about the model state to the users. This paragraph introduces a selection of patterns to demonstrate what information about the model is visible in the visualization and how it can be used to improve the model. For instance, a well-tuned model creates a visual pattern as depicted in Figure 4.5(a). In this result, every unclassified and incoming document is close to one of the topic points, indicating a clear topic affiliation. The pattern depicted in Figure 4.5(b) is in contrast to such a model. Here, all documents are placed in the center of the visualization where uncertain documents lie. Such patterns emerge, if data is in the system that does not fit to the model's decision criteria. In this case, either the data does not fit the user-defined topics or the model's decision criteria must be improved. If faced with this pattern, users have to investigate which of the two scenarios is present by browsing the data in the middle of the visualization. Subsequently, they can remove irrelevant data from the system, and uncertain documents can be classified.

Beyond signaling the certainty in classification, the visualization also can indicate connected topics through patterns (see Figure 4.5(c)). Similar topics can have an overlap in terms. In this case, the affiliation probabilities of both topics are high. The embedding places those documents between two topics, often connecting them in the process. Our implementation of Star Coordinates conveys topic similarities only implicitly through these patterns as the distance between topics is fixed. In comparison, multidimensional scaling techniques can encode similarity through topic distance. However, moving topic points introduces ambiguities, either documents moved due to a topic moving or because their topic affiliation changed. We decided against such a solution, as the emphasis of our visualization is on displaying affiliation to topics rather than topic similarities. Another interesting pattern is created, if the model's topics are imbalanced (see Figure 4.5(d)). The cause is an unbalance in either the corpus or the training data. If confronted with such patterns, users can react by changing the topics or re-balancing the training data with manual classification.



(a) Good Separation

(b) Bad Separation

(c) Similar/Connected Topics

(d) Imbalanced Topics

Figure 4.5: Various patterns conveying the condition of the underlying model through separation between topics and overall distribution of document points.

Making these patterns visible is dependent on multiple factors. Essential factors include the number of present axis vectors and their order. The arrangement of topics is crucial as the expressiveness of the visualization is dependent on the order of the axis vectors. Not all axis vectors can be mutually adjacent, if mapping models with four topic affiliation dimensions and more. Also, opposing axis vectors cancel each other out in this mapping (see Figure 4.6). Choosing opposing axis vectors for topics that positively correlate to each other, signals nonexistent classification uncertainty. This property is exemplified by document points with high topic affiliations for topics that are placed opposing each other. Instead of signaling that the documents are a mixture of both topics the document points are placed in the middle of the visualization where uncertain documents are expected. With increasing numbers, it gets harder to arrange topics that have relations to each other in a meaningful manner.

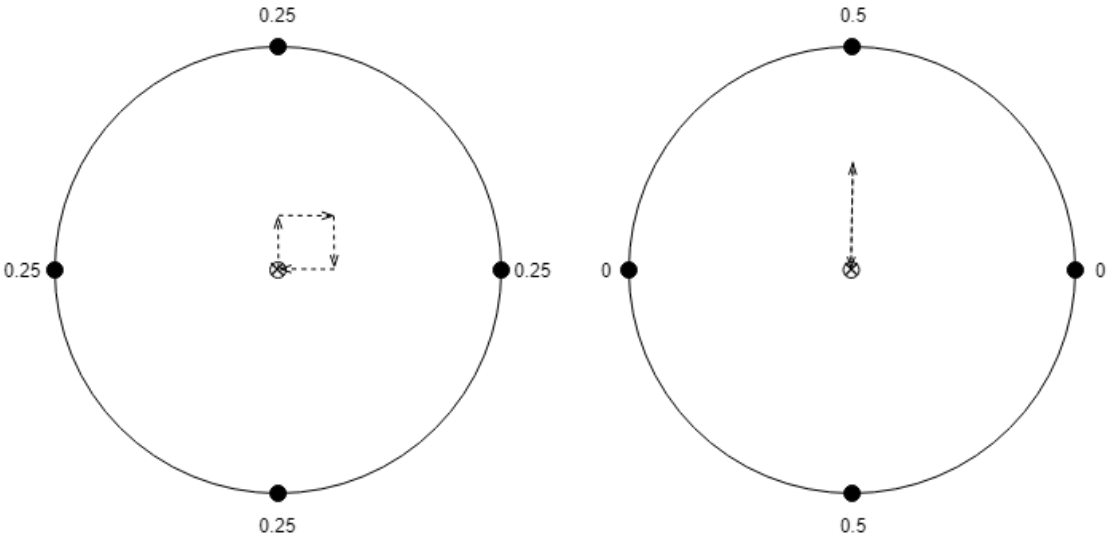


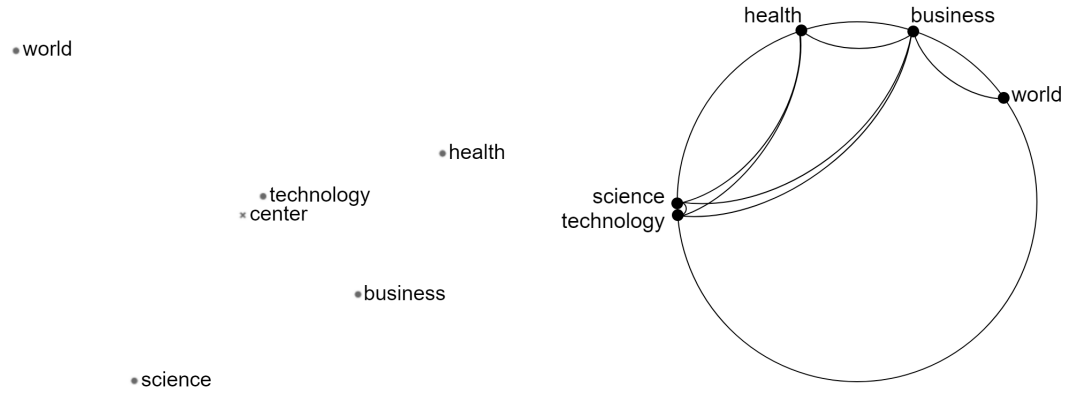
Figure 4.6: Ambiguities introduced by the Star Coordinates embedding: The Star Coordinates mapping algorithm can place different topic affiliation vectors onto the same point in the visualization. In this example, the topic associations p_i that lead to the ambiguity are displayed at their respective topic points.

Kandogan resolved this problem by visualizing the axis vectors and enabling user interaction to define the ordering by hand. In our application, an ordering is computed once at the start from the topic descriptions. We compared two automated solutions to generate an ordering that minimizes ambiguities. The first approach revolves around mapping the term vectors of the topic descriptions with multidimensional scaling [Kru64], based on Euclidean distance, into a two-dimensional space. In this mapping, a radial order of topic centroids from the overall center of centroids is computed with vector computations. We compare this approach against a circular barycentric heuristic approach [MS05], based on adjacencies derived from Jaccard-score similarities of documents in the corpus. The barycentric heuristic for circular layouts is an iterative technique that orders points by

computing average angles based on a given adjacency matrix of points. Multiple steps are necessary to correctly compute the average angle $\theta_{i\,avg}$ of a topic i in a wrapping circular layout. Similar to the Star Coordinates mapping, the heuristic computes a vector $\boldsymbol{n}_i$ for each topic that starts at the center of the visualization and points towards the direction of the corresponding topic point. Then, each vector is summed up with all vectors $\boldsymbol{n}_{i\,adj}$ of adjacent topics. The angle between the summed-up vectors and the x-axis vector $\boldsymbol{x}$ defines the average angle $\theta_{i\,avg}$, which is computed for each topic and sorted iteratively. This approach results in the formula:

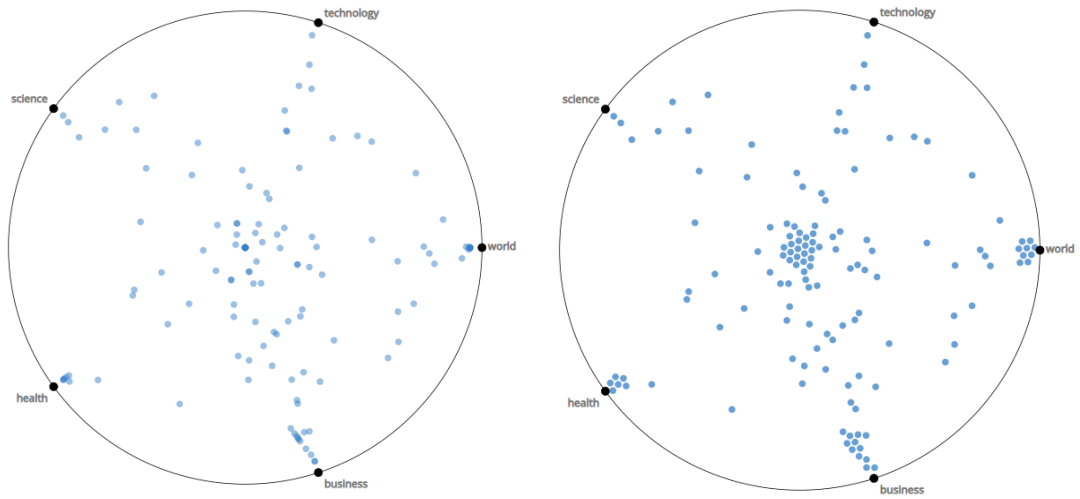
$$\theta_{i\,avg} = (\boldsymbol{n}_i + \sum_{\boldsymbol{n}_j \in \boldsymbol{n}_{i\,adj}} \boldsymbol{n}_j) \cdot \boldsymbol{x} \quad , \tag{4.2}$$

where the angle between the vectors is computed by the dot product. Ordering performance of the barycentric heuristic is strongly dependent on the definition of adjacency. In our system, adjacency is defined by computing the Jaccard similarity between topic descriptions, which are represented in a document-term matrix. As the results of this computation are continuous, and a binary decision for adjacency is needed, we apply an above-average similarity threshold, where average similarity is computed on a topic basis. This thresholding ensures that only topics with an above-average similarity score are treated by the barycentric heuristic as adjacent. In our testing, we observed that the barycentric heuristic creates more consistent orderings than the ordering based on multidimensional scaling, placing similar topics adjacent to each other. Figure 4.7 illustrates the difference between the ordering strategies derived from the same data. However, the performance of both approaches is very dependent on the initial topic description.



(a) Multidimensional scaling (b) Barycentric heuristic

Figure 4.7: Comparison of ordering strategies, using the New York Times study dataset (see Section 6.1): (a) mapping produced by multidimensional scaling of topic description term vectors; (b) ordering produced by a barycentric heuristic using the Jaccard similarity between topic descriptions for adjacency.



(a) Transparency (b) Force-Directed Layout

Figure 4.8: Comparison of overlap prevention strategies, on our New York Times study dataset (see Section 6.1).

As our application is supposed to support comparing neighboring documents, users should be able to directly interact with every document (e.g., click on it, hover it, drag it). With a large number of documents, a strategy for handling overlapping is needed. There are multiple viable strategies to resolve the overlap problem, from which we explored two options during development. Initially, transparency was added to document points to indicate positions where points accumulated (see Figure 4.8(a)). This technique enables users to see whether points have accumulated at a position. However, it does not encode accurately how many points overlap and does not enable unambiguous selection. This technique would require sophisticated selection tools to enable the selection of all documents in a pile. Instead, our visualization resolves overlaps by augmenting the mapping with force-directed layout techniques. A force-directed layout based on Verlet integration applies a collision force to all document points, which pushes them apart to resolve overlaps. To preserve the Star Coordinates mapping, an additional force keeps points near their original position. This force is defined as an α-scalable velocity vector $\boldsymbol{o}_v$ that pulls a point n from its current position $\boldsymbol{n}$ to its original position $\boldsymbol{o}$. Further, a noise vector $\boldsymbol{\epsilon}$ is added to the original position $\boldsymbol{o}$, resulting in the formula:

$$\boldsymbol{o}_v = ((\boldsymbol{o} \pm \boldsymbol{\epsilon}) - \boldsymbol{n}) * \alpha \quad . \tag{4.3}$$

This is necessary as, without such a noise vector, overlapping points tend to align along a straight line instead of a pile. Together these forces create a visualization that approximately preserves the Star Coordinates mapping, while all document points are accessible (see Figure 4.8(b)).

Due to the involved mapping process, Star Coordinates do not unambiguously convey the classifier's decision. Figure 4.9 shows this phenomenon, where not all points in the center cluster closest to the business topic are associated with that topic. As these transition areas are populated with documents that have small affiliation probabilities, this property is tolerable. To unambiguously communicate the classifier's decision, the visualization implements a topic filter. This filter can be accessed by hovering or clicking the topic labels. It has the effect of conveying the topic assignment through opacity in the visualization and removing documents from other topics temporarily from the document list (see Section 4.8). The filter has beneficial properties for one-against-all classification as documents that change topic affiliation are highlighted through an opacity change. Setting a filter for a topic of interest and observing the opacity changes can help to keep track of which documents that change topic affiliation.

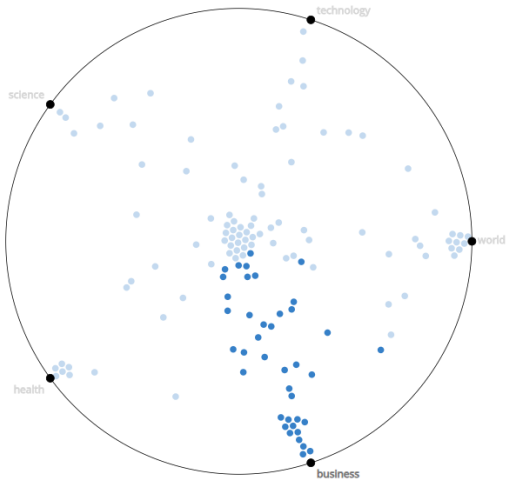


Figure 4.9: By clicking the topic label, users can filter the visualization. The filter mechanism highlights documents that are affiliated with the topic by increasing the opacity of their points in the visualization.

4.6 Active Learning with Star Coordinates

The application initialization part finishes with visualizing the first predictions. Beyond this static classification process, our application supports an incremental improvement process of the classifier through interaction with the visualization. This section describes in detail the interaction implementation, how to translate the user's interactions for the classifier, and how to accurately convey specifics of the underlying model to the user.

As described in the previous section, the document position in the visualization encodes topic associations. Our visualization supports the training of the classifier as a direct drag-interaction. Users can assign a single label to a document (see Figure 4.10) by dragging the corresponding document point into the vicinity of a topic label. After such an interaction occurs, the document's position has to be converted correctly into a topic affiliation. This topic affiliation has to be a concrete label for one of the available topics to add the moved document into the training set. More specifically, the classifier expects one label per training sample. Our application computes the distances l_i between all topic points in the visualization and the moved document's new position (see Figure 4.10) to derive a label for the classifier. The shortest distance defines the document's label.

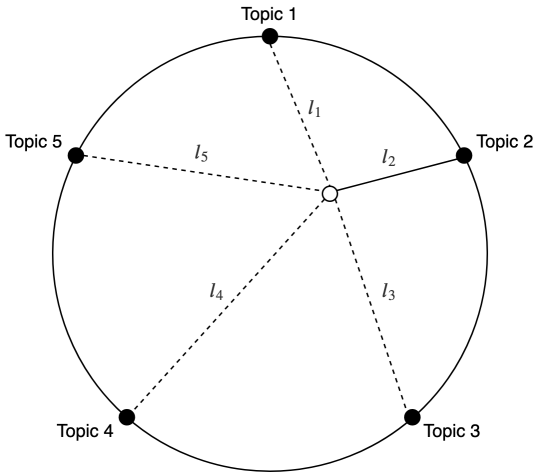


Figure 4.10: The updated label of a new document is determined by the nearest topic point. In this depiction, as the distance vector l_2 is the shortest, the moved document is assigned to Topic 2.

The described process leads to the challenge of communicating how the label is computed concretely. There is a discrepancy between the visualization and the interaction, as the visualization depicts a continuous space of topic affiliation probabilities, but the learning model does not take probabilities into account. Therefore, the interaction design adopts multiple measures to counteract misunderstandings that stem from this discrepancy. First, while users drag a document point, the nearest topic label is highlighted to convey the binary label decision more clearly. Second, if users are not able to clearly identify a topic affiliation, the document point can be hovered to highlight the corresponding topic point. To covey the training set more clearly, the classified document point is colored in black after the interaction. Marking manually labeled documents improves the user's perception of the model as it removes them visually from the test set documents that are marked as blue (see Figure 4.11).

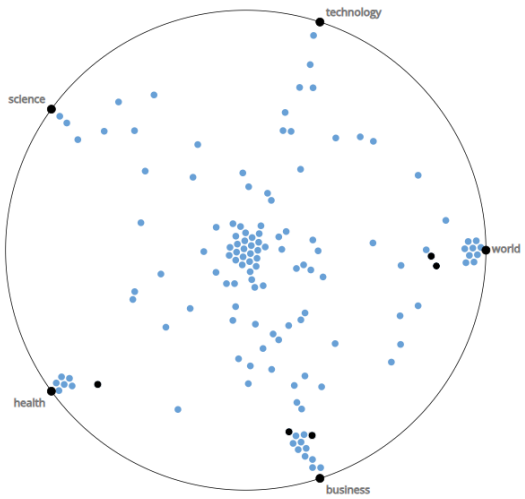


Figure 4.11: Documents that were labeled by users are colored in black. This distinguishes labeled documents from the test set that is marked in blue, while displaying how the labeled data for each topic relates to each other.

Indicating the connection between the user's input and the response of the classifier is also vital for understanding the impact of the interaction. To provide this connection, we decided to animate the classifier changes. Transitions help to track the changes on documents of interest. Likewise, animations highlight connections between points that were established by the classifier, as connected documents move into proximity of the user-labeled document during the update animation. As interactions and updates take a few seconds to compute, an update symbol is added to the visualization together with a notification system that keeps the user informed about ongoing processes.

4.7 Model Update

Our tool implements *visual active learning*, which is a variant of user-based active learning [SG10] for text stream data. *User-based active learning* expands on the concept of active learning by giving users agency over which documents to classify. In an interaction loop, the users act as an oracle and classify documents, which are subsequently added to the models training set to improve the prediction process. Our approach revolves around initializing a model with topic descriptions and applying an improvement process iteratively. Initially, the model is expected to perform poorly as a single document per topic is insufficient to describe it adequately. Over time the model should be able to increase its prediction performance with the user's help. After being trained, the model can still be further updated to adapt to changes in a text stream.

The process of incorporating users for classification to extend the training set is called active learning (see Section 2.5). Our application expands on the interactions possible within this concept of giving users access to the training set to change the model's classification hypothesis. It also supports reclassification and removal interaction. In the context of text stream data, concepts like topics can change over time. Documents that have fitted a concept earlier can become irrelevant. Further, within an explorative application, it makes sense to reverse earlier interactions. Our visual active learning approach supports interaction with the test set in addition to the user-based active learning interactions.

Our application supports the following selection of interactions:

- Interactions that manipulate the training set (user-based active learning interactions):
  - **labeling**, moving a test sample to the training set with a specified label,
  - **reclassification**, changing a label on a training sample, and
  - **removal**, removing a training sample.
- Interactions that manipulate the test set:
  - **addition**, adding a new test sample and
  - **removal**, removing a test sample.

Core to the active learning procedure is the labeling interaction, where documents are labeled to update the model. In the labeling interaction, an unlabeled document d_o from the test set enters the training set (see Figure 4.12). This process is implemented by manipulating both document-term matrices that represent the training and test set respectively. Here, a term vector d_o from the term vectors $d_1, ..., d_o$ of the input document-term matrix is moved to the term vectors $tp_1, ..., tp_k$ that comprise the topic document-term matrix. New terms from a user labeled document are added to the topic document-term matrix to facilitate the learning process. To extend the feature space of the training set the complete term vectors of all documents are preserved during feature extraction (see Section 4.3) so terms that are exclusive to the new document can be added for the model training (see Section 4.4). Adding a new term w_n to the topic document-term matrix is accomplished by adding a zero column, which encodes a non-occurrence in the current dataset. After the feature space update, the new term vector d_o is added through concatenation to the topic document-term matrix. As terms are used as features in our application, it is necessary to train a new model, so the new features can be added to the architecture. A new prediction of the test set also takes place due to the changes in the model.

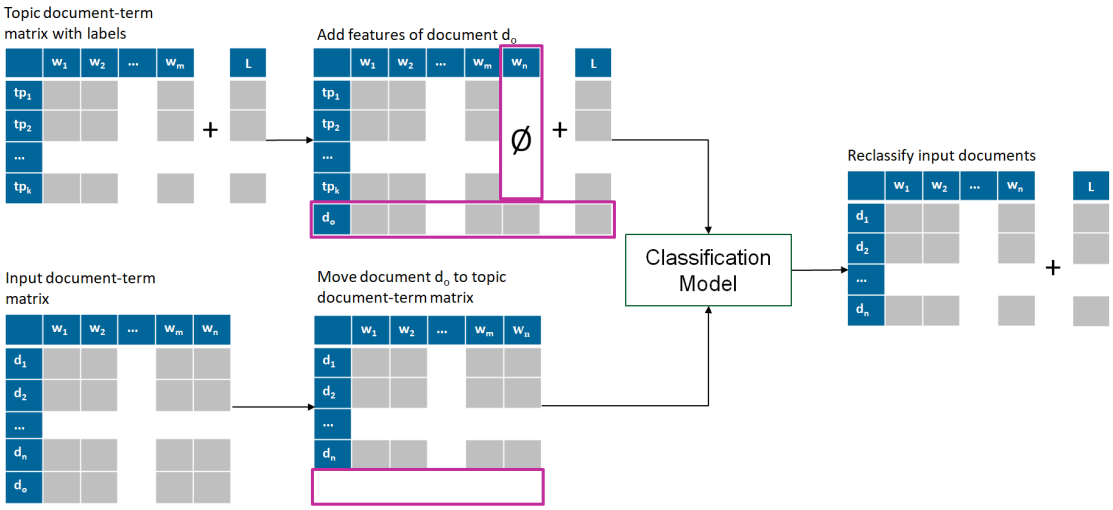


Figure 4.12: Labeling of document d_o. The document d_o is moved from the test set to the training set by adding its new term w_n to the training set and concatenating the document's vector to the topic document-term matrix.

In comparison, the reclassification interaction just changes the label of a document d_o (see Figure 4.13). As document d_o is already in the training set, the feature set of the model does not change through this interaction. Nonetheless, this change demands training a new model (retraining) as the label, which the previous model has trained on has changed. Therefore, also a reclassification entails retraining of the underlying model and new predictions for the test set.

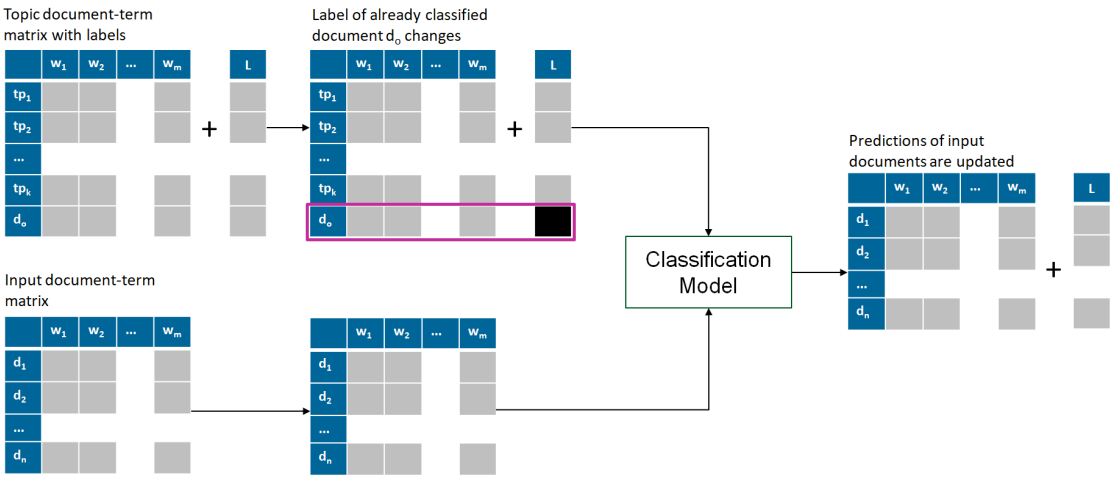


Figure 4.13: Reclassification of document d_o. The label in the topic document-term matrix is changed to reflect the updated label.

Lastly, our application implements a removal interaction (see Figure 4.14). This interaction is essential as it enables users to remove documents that do not fit into their task. If document d_o has to be removed from the training set, both retraining and new predictions are needed. In the training set document-term matrix, the removed document might contain a terms that are exclusive to it. If this is the case, this term w_n is removed from the document-term matrix to simplify the training process. Even if the document d_o does not contain exclusive terms, retraining is necessary, as the model should not incorporate information from document d_o into its decisions.

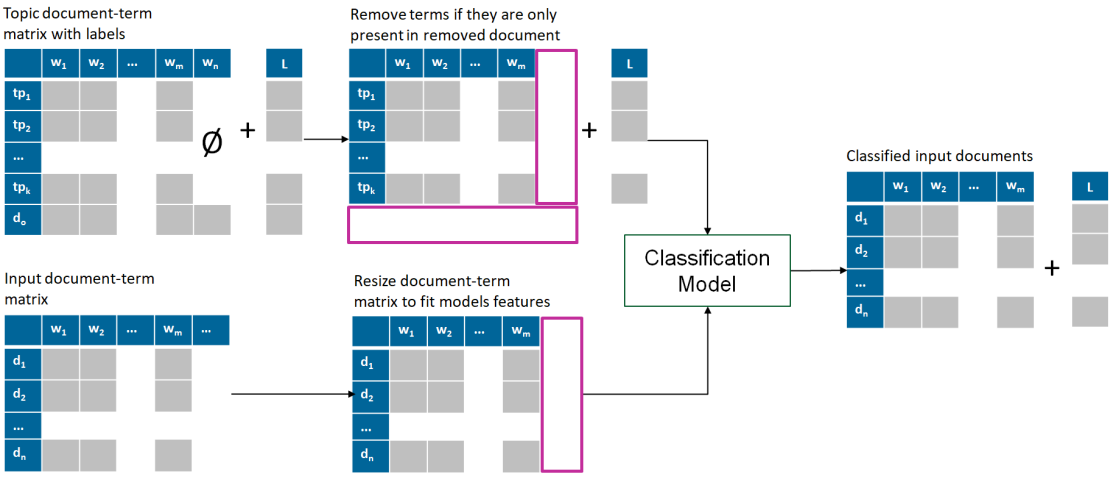


Figure 4.14: Removal of document d_o. Alongside the document vector d_o, also terms, unique to this document, are removed from the topic document-term matrix. In this example term w_n is removed.

Besides active learning interactions, our application supports adding and removing documents from the test set. If dealing with streams of text data, the model has to add new documents to the data structure. Removing uninteresting or old documents is useful as well. These interactions change the initial input document-term matrix before the alignment process. Just like the addition and removal from the training set, these changes trigger the same operations on the test set instead, which is represented by the input document-term matrix. Subsequently, documents added to the test set are treated as the rest of the test set, by fitting their term vector to match the model's features (see Figure 4.3). These operations require no retraining of the learning model as the training set data is not affected. Adding new documents only requires a prediction for the labels of new documents.

Learning concepts from new data through additional features is one central requirement our active learning routine has to satisfy. However, as our classifier is based on bag-of-words features, this requirement comes with a price. If a document introduces a new feature, the architecture of the learning model has to change to accommodate it. This in turn excludes more efficient online learning approaches as they cannot change the set of features after the initial training. Therefore, our approach retrains the model on each

interaction that changes the feature space. Rebuilding the models gives the flexibility of changing the feature space and avoids *catastrophic interference* problems, which is due to the tendency of online learning models to forget previously learned information upon learning new information [MC89].

4.8 Dashboard

The visual active learning process is central to the presented application. Therefore, our dashboard reserves almost half of the screen space for the visualization that includes the interaction tool for the visual active learning process (see Figure 4.15 (a)). Most of the remaining part of the screen is occupied by a scrollable list, called document list (see Figure 4.15 (b)), that provides document details.

The document list holds a document card for each document sorted from newest at the top to oldest at the bottom. Additionally, the list supports pinning one selected item to the top. Selecting documents is a linked interaction between the list and the visualization. The interaction can be triggered by clicking the document in either representation. A blue border around the document card, marks the pinned document in the list. In the visualization, a black ring around the correspondent document point is placed. This interaction serves two purposes. It keeps the focus on a single item, when switching between the list and the visual depiction of the data and enables easier comparisons between documents, which are not adjacent in the list. The described functionality is also available through hovering in both representations, which has the same effect as a click temporarily. If a document is hovered in the visualization, a tool-tip with the document's title is displayed alongside the document point, so users do not have to consult the document list.

Details of documents are accessible through the list's document cards (see Figure 4.15 (b)). A card holds typical text information such as the documents title and a short description. The full text of the document card expands into a pop up over the base interface. Further, the document cards provide the possibility of removing documents from the application and learning model by clicking the close button on the correspondent card. As a whole, the document list with its cards provides the core functionality of a document viewer, so users can browse documents in a familiar manner.

The document list and visualization are linked together, so users can look at document details, while keeping an overview. In particular, users can hover and click documents in either representation, which causes an analog interaction in the other component. The visualization is fixed in place, decoupled from the document list on the right side that can be scrolled through. This interaction is designed to let users keep the overview in sight, while moving through the list. To supply initial topic documents, some space below the visualization is available to place the document loader and options (see Figure 4.15 (c) and (d)). The dashboard also supports updates of the document structure. Adding new documents, displays them as red document points in the visualization and includes document cards with red borders at the top of the document list (see Figure 4.15 (e)).

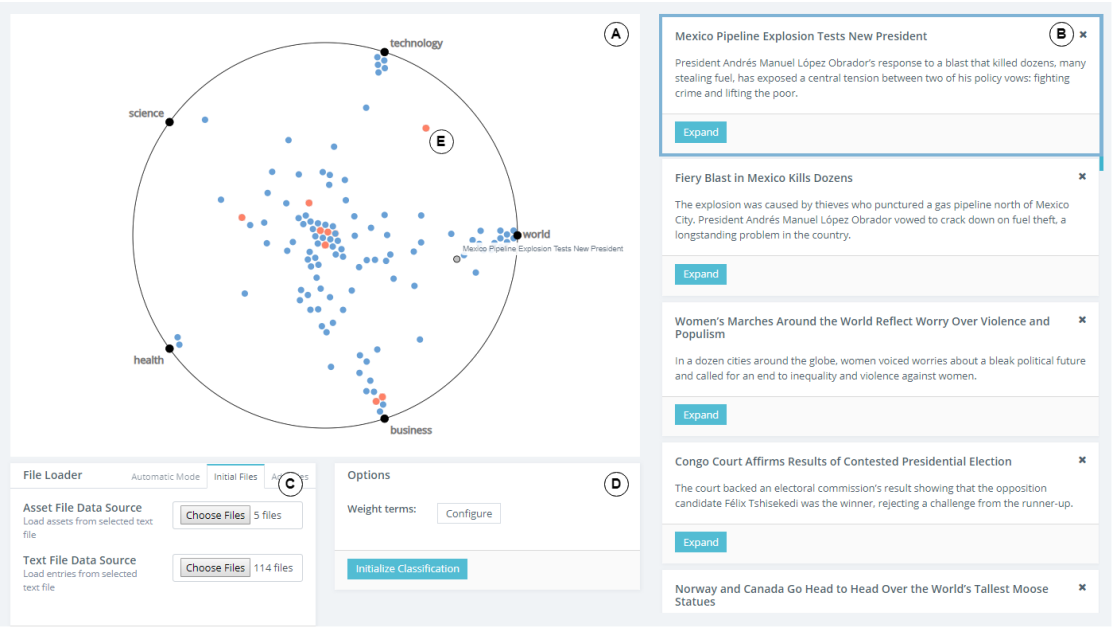


Figure 4.15: The dashboard of our application: (a) the Star Coordinates visualization to illustrate prediction results and classify data; (b) the document list to inspect document details; (c) the file loader card to input topics and text stream data; (d) the options card to change term weights of the model; (e) new document, highlighted in red.

CHAPTER 5

Implementation

Our application is a customizable classification pipeline for text streams with a dashboard that is integrated into the framework by Čizmić [Ciz18] and Śmiech [Smi18]. The sections below discuss the software structure, the frameworks and libraries, and further task-specific details of the server and the client our application is split into. In the closing section of the chapter, we will also discuss the simulation server that replaces the client in automated tests (see Chapter 6).

During the design stage, we decided to split the application into a client-server architecture. This decision was made to outsource expensive classification tasks from the client, which is dependent on the user's browser and system. Therefore, the client's tasks encompass parsing user input and computing the visualization, leaving the expensive machine learning tasks to the server. The server tasks include natural language processing, active learning, and classification. This distribution of tasks leads to a data structure with two distinct parts (see Figure 5.1), which is called the *application state* and is passed between server and client. The server's part of the data structure includes the current operation (state) and the corpus (test set, training set). In the application state, the corpus is represented as three arrays of documents, where each document contains the textual content as a string together with further organizational information. As the server implements our active learning procedure, the data structure contains the operation (state) and the documents on which it should be applied (changed set). In the client's part of the application state, document and topic points are included. These two sets of points are the data structure of the visualization that holds the topic affiliations for each document and the order of the topics respectively. A document point primarily stores topic affiliations, which is defined by users or the result of the learning model prediction. The topic points encode the order of topics through their order in the array.

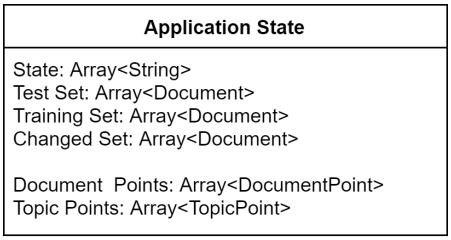


Figure 5.1: The data structure that is passed between the client and the server. It holds the raw text data for the server's classification task as well as the topic affiliations for each document and the topic order for the Star Coordinates visualization of the client.

5.1 Server

The *Flask*-based [Fla] server backend is programmed in *Python* [Pyt] and implements the custom classification functionality of the application. *Flask* is a micro web-framework that simplifies the set up of a server's routing functionality. As most of the server's computations deal with transforming document-term matrices, *Numpy* [Num] is especially useful with its efficient processing of large, multi-dimensional arrays and matrices. The server's functionality is realized as a modular pipeline, where individual components are easily replaceable and reusable. Each of the components solves one of the server's four tasks (see Figure 5.2): 1) the feature extraction of documents, 2) the classification of documents, 3) the visual active learning interactions, and 4) the visualization order optimization. The functionality of the feature extractor and classifier is based on *nltk* [Nat], Python's natural language toolkit and *scikit-learn* [PVG+11], a Python library for data mining and data analysis. All these tasks define separate components, which are chained sequentially to implement the visual active learning interactions described in Section 4.7.

Nltk provides the functionality for the server's *feature extraction component*. Feature extraction starts with the conversion of the test set and training set from text to word tokens with *nltk.word_tokenize()* that splits on punctuation in the text. This process yields an array of tokens on which the NLP functions of the application are applied. Our application includes the following NLP functions that can be added into the pipeline: stemming (*nltk.PorterStemmer()*), lemmatization (*nltk.WordNetLemmatizer()*), ngram computation (*nltk.ngrams()*), pos selection (*nltk.pos_tag()*), and named entity recognition (*nltk.ne_chunk()* and *spaCy.nlp()*). In summary, our feature extraction pipeline takes a set of documents, creates tokens, and computes named entities with *nltk.ne_chunk()*. The resulting tokens are then transformed into a term vector that is combined with the term vectors of other documents to form a document-term matrix. Using this process, both input and topic document-term matrix are generated from the test and training set respectively and subsequently utilized for classification. The resulting matrices are saved in the form of a *numpy.matrix*. Due to the iterative training procedure, the input and topic document-term matrix have deliberately different features (see Section 4.4).

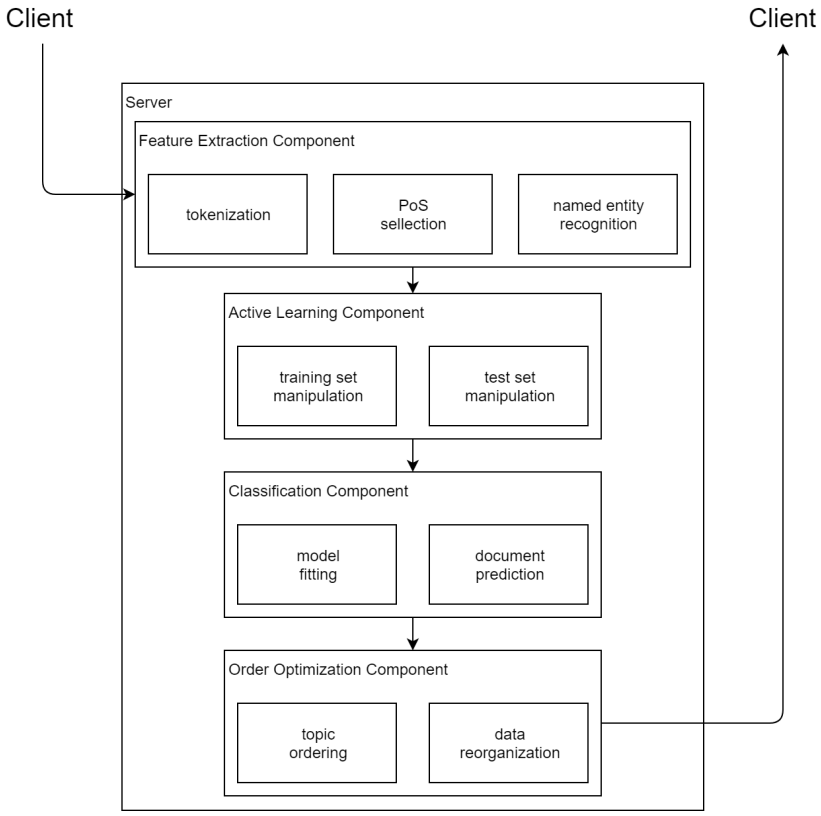


Figure 5.2: The application's server implements feature extraction, active learning, classification, and order optimization that are chained sequentially.

In our application, the *active learning component* (see Section 4.7) is placed before the classifier. The visual active learning interactions affect the classification by moving documents between the test and training set. A detailed description of all interactions can be found in Section 4.7. We chose to implement these changes on the document-term matrix level, as *Numpy* provides tools to efficiently manipulate matrices. For instance, during the *add_document()* operation Numpy is utilized to add new features to the training set. To add a new feature, the operation inserts a column at the appropriate position into the document-term matrix with the function *numpy.hstack()*. When the feature dimensions match between the current document-term matrix and the new document, it is added as a new row with the *numpy.vstack()* function. In a similar manner, the *remove_document()* operation deletes entries by applying column and row operations. For deletion, the *numpy.delete()* function can remove documents by removing the corresponding row. The same Numpy function can also remove a column, which corresponds to removing a term from the model. After a document is removed, the server checks for features to remove by computing all column sums of the document-term matrix with the *numpy.sum()* function. Subsequently, the application removes all columns that sum to zero.

Following the transformations caused by active learning, the document-term matrices are passed to the *classification component*, which is based on *sklearn.MultinomialNB()*, a multinomial naive Bayes model from the *scikit-learn* library. Besides this model, the server also can apply different models for test purposes. Our classification component also implements the following models: support vector machine (*sklearn.SVC()*), k-nearest neighbors (*sklearn.KNeighborsClassifier()*), multilayer perceptron (*sklearn.MLPClassifier()*) and complement naive Bayes (*sklearn.ComplementNB()*). For classifier training and prediction, the feature dimensions of the training and test sets have to match. First, the model is trained by passing the input document-term matrix to the *fit()* function. Subsequently, the pipeline aligns the input document-term matrix to the topic document-term matrix (see Section 4.4). After training, the model predicts the test set. For this functionality, the *predict_proba()* function from the naive Bayes classifier is used. This function returns the topic affiliations, which are based on Bayesian inference and are computed during predictions (see Section 2.3).

After the model classifies all documents in the test set, the predictions are passed to the *order optimization component*. To create an expressive display of the results, the visualization needs a meaningful topic order. This order is computed by applying the barycentric heuristic from Makinen and Siirtola [MS05] (see Section 4.5), which is utilized to sort the topic affiliations in the application state data structure. The resulting order together with the sorted topic affiliation data is subsequently passed to the client.

5.2 Client

The client contains all interaction and visualization functionalities in the form of a dashboard. The implementation is written with *TypeScript* [Typ], a typed superset of *JavaScript*. Within the client, the UI is based on *Angular* [Ang], *Bootstrap* [Boo], *jQuery* [jQu], and *d3* [Bos]. The *Angular* framework enforces an architecture that separates views and related logic. Beyond architectural restrictions, the framework provides a collection of libraries and features that simplifies an asynchronous programming approach. *Boostrap* adds a streamlined collection of UI elements to the application that follow general design principles to create appealing user interfaces. In particular, *Boostrap* offers grid-based alignment systems to create responsive and resolution independent user interfaces that can be displayed on different devices. To manipulate basic *HTML* elements and send data through *POST* requests, *jQuery* is utilized, as it simplifies interacting with these concepts. The client's responsibilities can be split up into three tasks: 1) the translation of user interactions into server operations, 2) the visualization computation (see Figure 5.3), and 3) the management of asynchronous actors.

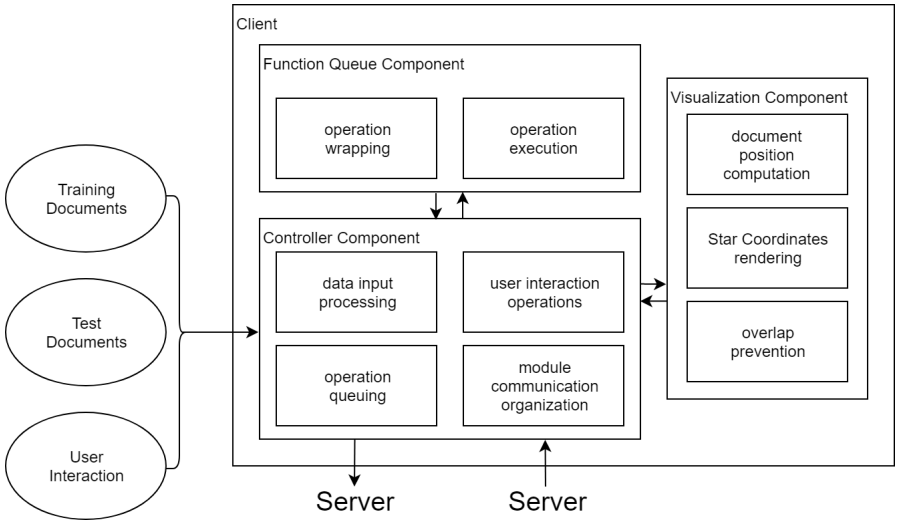


Figure 5.3: A representation of the clients architecture. The client is mainly concerned with tasks of user input processing, operation execution and visualizing the server's classification results, which define the split into the different components.

Translating user interactions is concerned with utilizing *EventListeners* and updating data structures within the application's *controller component* accordingly. *RxJS* is the library included in the *Angular* framework that enables a reactive and asynchronous design of the application's client. Central to the asynchronous design are the *rxjs.Observables* wrapping the application's data structures, which provide different components with access to the data through a subscriber model. The *Observables* notify subscribed components, when data structures change, so they can subsequently perform operations on the new data. By subscribing to the other components of the client and providing *Observables* of its own, the controller component manages the client's input and the communication to the server.

To explain how the controller component processes user interactions, we take the interaction of labeling as an example. When users drag a document point in the *d3*-based [Bos] visualization, the movement triggers the *d3 EventListener* of the corresponding Scalable Vector Graphics (SVG) circle in the separate visualization component. The *EventListener* triggers an algorithm to interpret the user's classification confidence (see Section 4.6) and the visualization component changes the label in the underlying *Observable*. These changes in the underlying data structure trigger the subscribed controller component to queue a visual active learning interaction, preparing data structures to be sent to the server, which leads to an update of the underlying model based on the user's interaction. Similarly, when the model passes the results back, the controller component incorporates the changes into its data structure. These can include changes in the topic affiliations of certain documents and the order of topics. Subsequently, the controller component triggers the visualization component to compute a transition of the visualization into the

new state. In case of an update, the changed topic affiliations are utilized to compute new positions for the document points, which are then displayed to users through a *d3.transition()* from the old position to the new one.

To test our applications update functionality for text stream data, continuous updates from a text stream were simulated. For this purpose, the *controller component* implements an automatic mode that is based on *Timer Observables* provided by the *RxJS* library. These *Observables* space out the updates of the text stream. Configuration files are used that hold the information for the timing between iterations and the contents of the updates. As all learning model related operations are handled by the function queue component, the automatic mode can simulate all other basic interactions with the application as well.

Our application has to handle two actors that can trigger changes. Besides the obvious actor (i.e., the user), the text stream itself represents the second actor that interacts with the application. It is necessary to manage the asynchronous interactions of both actors, so the applications shared data structure, the *application state* (see Figure 5.1), is kept consistent. For this purpose, our application implements the *function queue component*, that receives all asynchronous operations from the actors and stores them together in a sequential queue structure. The component wraps the operations of the application, together with their context and own callbacks, in a function and chains the exterior functions through callbacks together. After an operation with its callbacks finishes, the exterior function wrapping it is executed that triggers the function queue component to process the next operation in the queue. In this manner, the queue is stringed together and resolves itself sequentially. This way, users can interact with the client without having to wait for the server to finish the operation first and without interruptions from updates of the text stream.

The Star Coordinates visualization (see Section 4.5) that presents results and enables interactions is computed in the *visualization component*. This visualization is implemented as an interactive, scalable vector graphic (SVG) with the help of the *d3.js* library. *D3.js* enables a data-focused approach in the process of creating interactive visualizations, through selection and data join operations. In principle, the Star Coordinates visualization is implemented as a set of SVG circles that are dynamically updated based on their correspondent documents. Through *enter* and *exit* selections, the displayed SVG circles are appended to or removed from the SVG. Similarily, selections attach *EventListeners* onto *HTML* elements that are used to create various interactions in our application. *EventListeners* are also utilized to add hover, click, and drag interactions to the document points that correspond to selecting a document temporarily, selecting it until deselection, and labeling a document respectively. In the same manner, the filter interactions of the topic labels are implemented. Animations, like our update animations, are created by using the *d3.transtion()* interface that interpolates the data smoothly from the old state to the current one.

5.3 Simulation Server

For the tests in Chapter 6, our client side is replaced by the *Node.js*-based [Nod] simulation server that can perform automated tests on datasets while simulating selection strategies. The set of selection strategies includes strategies based on classic active learning as well as strategies that are based on our Star Coordinates visualization. The simulation server automates multiple tasks needed for *cross-validation based evaluation*, which is a form of evaluation for assessing how the results of a statistical analysis will generalize to an independent dataset. These tasks include: 1) cross-validation dataset preparation, 2) execution of cross-validation tests, 3) simulation of active learning selection strategies, 4) runtime estimation, 5) the computation of evaluation measures, and 6) computing Star Coordinates visualization (see Figure 5.4).

Regarding cross-validation, the *controller component* of the simulation server can prepare datasets in multiple ways. The controller component can split the available data into multiple sets of the same size or create a test and training set split. These splits can be saved and loaded so additional tests can be run under the same conditions. With a specific split, the simulation server then can perform an automated test where it initializes a new learning model for each split in the dataset following a preset selection strategy. In total, the simulation server has five different selection strategies that it can simulate, which are explained in Section 6.2. Besides these strategies, the simulation server can also simulate how a learning model performs with only the initialization data and with all documents available from a specific split. The last two strategies are especially helpful for *hyperparameter tuning*, which is the process of finding the optimal hyperparameters for a learning model. After each interaction between the simulation server and the server, multiple measures can be computed (see Section 6.3). In summary, the simulation server can compute accuracy, precision, recall, the receiver operating characteristic curve (ROC) curve. To provide an overview of how the model develops over iterations, the simulation server includes a simplified *visualization component* that can compute the Star Coordinates visualization of the current learning model. As some tests can be extensive and run for long periods, the simulation server also implements runtime estimation, in a separate *timer component*, to give an overview of the progress. To estimate the runtime the component computes for each split an average time and uses the estimation to project to the whole runtime.

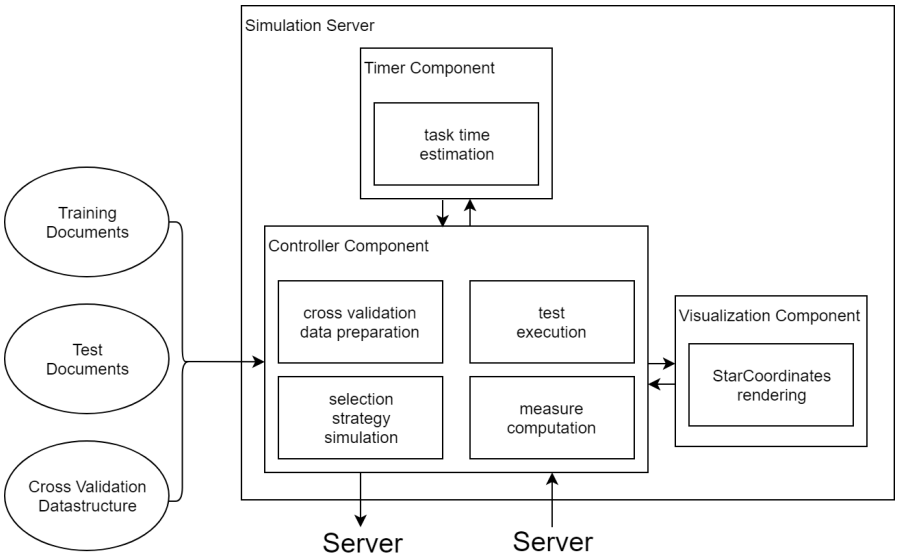


Figure 5.4: The structure of the simulation server. This server provides multiple tools automating processes performed in cross-validation based evaluation.

CHAPTER 6

Experiments

This chapter presents the results of several experiments simulating the active learning process for news streams. These experiments were performed to evaluate choices made during the development process and the final application. There are two sets of tests discussed in this chapter. First, tests are presented that were performed during development for tuning the learning models' hyperparameters. Second, we compared our visual active learning approach to classic active learning strategies. In the second set of tests, the goal is to find evidence that visual active learning strategies are comparable to their classic active learning counterparts.

6.1 Datasets

We chose to conduct our experiments on a self-generated dataset, called *NYT dataset*. This dataset is generated from news articles from the New York Times [The] that were extracted from a selection of their RSS feeds. The RSS feeds were parsed on seven days in the timespan between December 2018 and February 2019. Besides using the data provided in the RSS feeds (title, description) the dataset also contains the full articles found on the corresponding website. From the selection of RSS feeds, we chose a subset of categories that would minimize news article overlap between the categories. In particular, we chose the categories: world, technology, science, health, and business. For testing purposes, we removed all duplicates, which are articles associated with multiple categories from the dataset, which results in a total of 607 documents with an average length of 6378 characters (see Table 6.1). There are multiple reasons we decided to generate a dataset ourselves. To fit the use case, the dataset for testing should resemble real news articles. In our case, the real data consists of financial reports, which are documents that usually are longer than the snippets and descriptions provided in open datasets. Further, we chose to use the dataset dealing with basic news categories, as it makes it easier for non-experts to engage with the data. This argument was a decisive factor in the

dataset choice, as we intended to conduct a preliminary user study after the experiments (see Section 7). Another important aspect is the possibility to generate a dataset for cross-validation that is reasonably balanced. For our testing, we used a balanced subset of the NYT dataset for three times cross-validation, where we picked 60 documents per category in a random fashion (see Section 6.2).

To compare the classification capabilities of our system, we also ran some tests on the *Reuters R8 dataset* [Reu]. Following Debole and Sebastini's convention [DS05], this dataset is a subset of the Reuters-21578 dataset where only ten classes with the highest number of training documents were considered. Subsequently, the dataset was also filtered to contain only documents with a single label, which reduced the set to eight classes. The final set contains 7674 documents with an average length of 892 characters, which is split into predefined training and test set called Mod Apté (see Table 6.2).

NYT	
Class	**total # docs**
world	230
technology	78
science	64
health	80
business	155
Total	**607**

Table 6.1: NYT dataset.

Reuters R8			
Class	**# train docs**	**# test docs**	**total # docs**
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
Total	**5485**	**2189**	**7674**

Table 6.2: Reuters R8 dataset.

6.2 Procedure

To evaluate our approach, we set up multiple tests and simulated active learning procedures. In this procedure, each run of a test was initialized with the same selection of initialization documents. After initialization, the learning model was trained by labeling and adding one document from the test set at a time to the training set. How the document is chosen is dependent on the selection strategy that mimics an active learning strategy (see Section 2.5). After a document is selected and labeled, the model is retrained and the test set is reclassified. This procedure was repeated until the whole test set is added to the training set. After each iteration, we evaluate the state of the model on a separate validation set. To avoid random biases due to initialization or within different selection strategies, we use k-fold cross-validation (k=3) where it is feasible. In *k-fold cross-validation* the data is partitioned into k subsets, performing model training on $k-1$ of the subsets (training set) and withholding one subset (test set) to evaluate the performance. This procedure is repeated k times while switching the subset used for evaluation. By averaging the results of all iterations, cross-validation derives a more accurate estimate of the model's prediction performance.

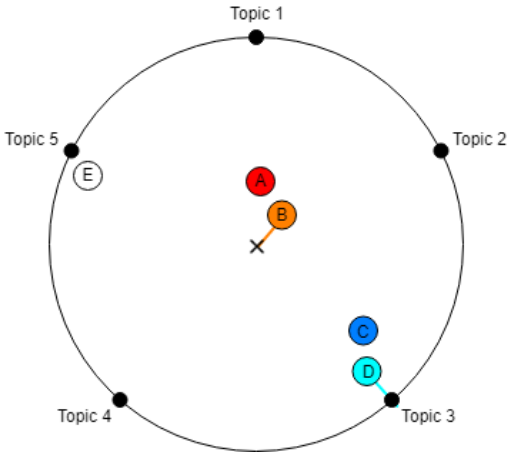
The selection strategies are core to our evaluation. Our initial evaluation aims to find evidence that strategies based on visual measures correspond directly to classic active learning strategies. It is necessary to examine the correspondence, as our Star Coordinate mapping can create ambiguities in the process of displaying topic affiliations (see Figure 4.6).

We formulated a hypothesis to test this property of our application: **H1:** *Both classic and visual active learning strategies outperform a random strategy in training.*
We assume that our visual active learning strategies should perform similarly to their classic counterparts when run on the same data. The crucial difference between both strategies is that in the visual measure is dependent on the Star Coordinates mapping. To evaluate **H1**, we compared how the strategies influence the performance of the classifier over time. We examine the accuracy in particular as it gives some insight into performance while being a straightforward measure to visualize over time.

To evaluate this hypothesis, we implement strategies for comparison in their classic form and visual form based on our Star Coordinate mapping. In particular, our evaluation incorporates two classic strategies [Coh17]: uncertainty sampling [LG94], and query by committee [SOS92] (see Section 2.5). For clarity, Figure 6.1 illustrates all selection strategies in a small example case.

As our learning model inherently keeps track of the certainty of classification in the form of the topic affiliation, we use this measure to implement the uncertainty sampling strategy. Our *uncertainty sampling* strategy thus picks the first document with the lowest maximum association probability that decided the association of a document. The corresponding *visual uncertainty sampling* strategy uses the distance between the center of our visualization to each document point. As uncertain points are placed in the middle of the Star Coordinates visualization (see document A in Figure 6.1), this measure corresponds closely to the classic uncertainty sampling strategy. This strategy also relates to a real-life scenario where users examine documents with uncertain labels and decide these cases.

Secondly, we implemented a strategy we call *correction* that corresponds to the query by committee strategy. The query by committee strategy tries to find underdetermined spaces in the training data by using an ensemble of learning models and choosing documents with the highest disagreement between the models. As we have the ground truth for the data, our *correction strategy* follows the same goal as the query by committee strategy by searching for the document with the highest confidence (i.e., topic affiliation) which is classified incorrectly (see document C in Figure 6.1). The corresponding *visual correction* strategy uses the distance between the topic points and document points instead of the topic affiliation. There is also a corresponding real-life scenario for this strategy. In this scenario, users hover the title of documents trying to find incorrect labels which they then correct.



	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Ground truth
Document A	0.25	0.25	0.125	0.125	0.25	2
Document B	0.5	0	0.3	0.2	0	3
Document C	0.05	0	0.85	0	0.2	1
Document D	0	0.1	0.8	0.1	0	2
Document E	0	0	0	0.05	0.95	5

Color Legend:
Uncertainty Sampling
Visual Uncertainty Sampling
Correction
Visual Correction

Figure 6.1: Example demonstrating how all described selection strategies and their visual counterparts are computed. For the visual strategies (visual uncertainty sampling and visual correction) the deciding distances are drawn in the Star Coordinates visualization as colored lines. The deciding probabilities for the classic strategies (uncertainty sampling and correction) are colored in the probability table.

6.3 Measures

Many measures used in the evaluation of a classification are defined for a binary classification. To apply them to our application, these measures must be adapted for multi-class classification. Usually, to accomplish this adaptation, the binary measures are combined across labels. For averaging binary measures, two techniques that can be applied are micro-averaging and macro-averaging [VA13]. The *micro-average* aggregates the contributions of all classes to compute the average metric, treating each document equally. In comparison, *macro-averaging* works in reverse, by computing the metric independently for each class and then taking the average, treating classes equally in the process.

In our evaluations, we used *accuracy* as our basic evaluation metric to compare different variables over the active learning process. This measure is defined as the fraction of documents correctly classified as positive (true positive rate). We decided to compute multi-class accuracy through micro-averaging as this approach captures class imbalances more adequately. The micro-averaged score Acc_{micro}, formally:

$$Acc_{micro} = tpr_{micro} = \frac{\sum_{c \in C} tpos_c}{\sum_{c \in C} d_c} \quad , \tag{6.1}$$

can be computed based on the classified documents d with the fraction of correctly labeled documents $tpos$ across all classes $c \in C$. Accordingly, the definition of the macro-averaged score Acc_{macro} is very similar, formally:

$$Acc_{macro} = tpr_{macro} = \frac{\sum_{c \in C} \frac{tp_c}{s_c}}{|C|} \quad , \tag{6.2}$$

where the fraction of correctly labeled documents *tpos* is computed independently per class $c \in C$ and averaged afterward.

Beyond accuracy, our evaluation also utilizes the *receiver operating characteristic* (ROC) curve, for evaluation. This graphical plot illustrates the performance of a binary classifier as its discrimination threshold varies (see Figure 6.2). For classifiers that return class probabilities, this discrimination threshold determines how high the probability has to be so that the document is assigned to the class. In this manner, the graph shows the model's trade-off between the true positive rate and false positive rate. These two measures answer the question, how likely the model is to identify documents correctly or incorrectly that are actually belonging to a certain topic. Depending on the task, one of these measures might be more critical. By plotting the changes between true positive rate and false positive rate along the discrimination threshold, the model can be tuned to the desired trade-off. The ROC curve provides a very simple graphical view of the possible trade-offs [BD06]. These factors make the ROC curve a good measure for comparing multiple classifier models. Models can be compared by interpreting each model's capability to separate classes and choosing discrimination thresholds for specific tasks. The plot is created by plotting the true positive rate *tpr* against the false positive rate fpr at various discrimination threshold settings. The false positive rate fpr for both averaging techniques is defined in the following manner:

$$fpr_{micro} = \frac{\sum_{c \in C} fpos_c}{\sum_{c \in C} s_c} \quad , \tag{6.3}$$

$$fpr_{macro} = \frac{\sum_{c \in C} \frac{fpos_c}{s_c}}{|C|} \quad , \tag{6.4}$$

where the fraction of incorrectly labeled documents $fpos$ is either computed across all classes $c \in C$ or independently per class $c \in C$ and averaged afterward. To compute the ROC curve, we use the micro-averaging approach.

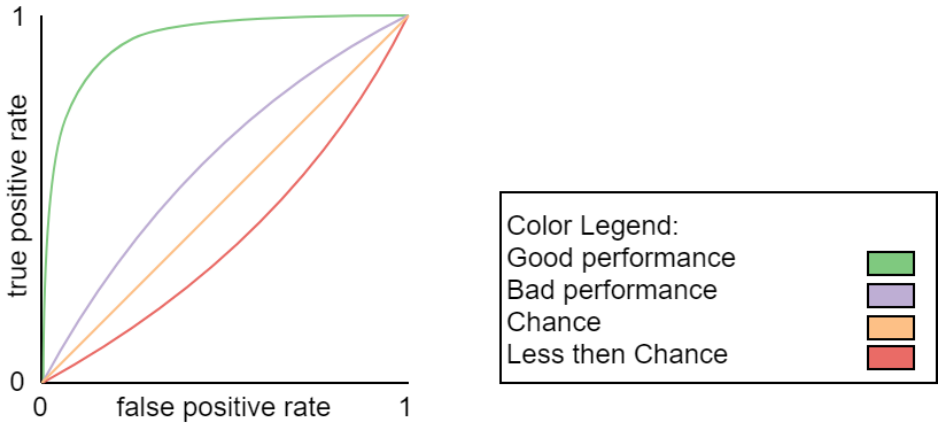


Figure 6.2: Simplified example of a ROC curve, displaying how different classifier performances look on the curve. The orange straight line is often displayed in ROC curves as it represents the performance of a classifier that is equal to selection by chance. Preferably, a classifier should be able to achieve a high true positive rate without many false positives. In this simplified, example the green curve exemplifies such a classifier. By comparison, the violet curve shows a classifier with bad performance. Lowering the discrimination threshold on this classifier adds many false positives to the result.

6.4 Parameter Tuning

During development, we performed multiple tests to tune our classification pipeline (see Section 4.3) to the problem. In these tests, we simulate active learning strategies, as introduced in Section 6.2, to compare different variants of the pipeline. This hyperparameter tuning was performed on our NYT dataset (see Section 6.1) in a three-fold cross-validation procedure. Table 6.3 summarizes, which dimensions of the parameter space are examined during the tuning process. The search strategy applied in the tests was grid search. However, due to parameters performing similarly (e.g., part of speech and named entities with part of speech combined), some parameter combinations were skipped.

Test Parameter Space				
Features	pos	ne (nltk)	ne (spaCy)	ne (nltk) + pos
Model	mlp	svm	mnb	cnb
Stage	0	200		

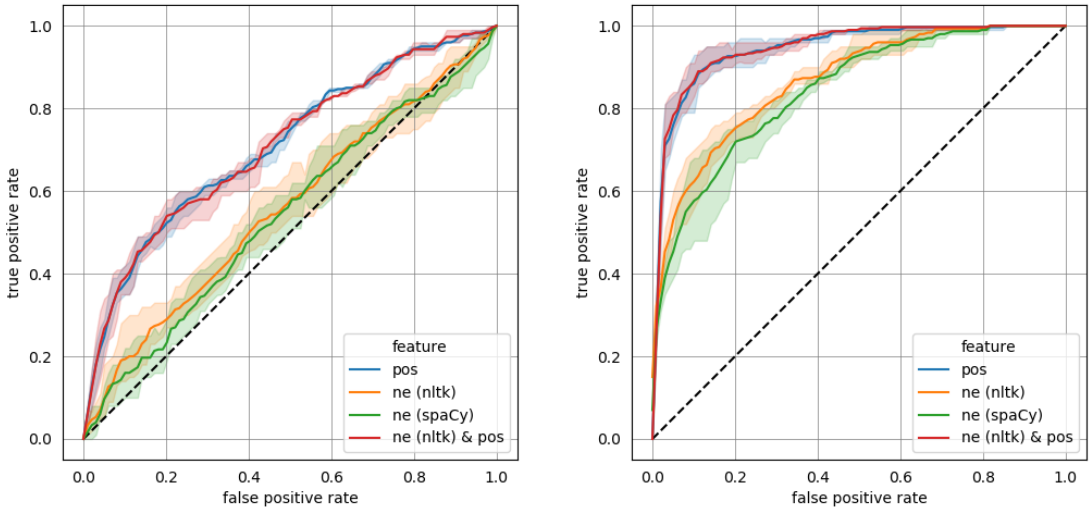
Table 6.3: Overview of the set of hyperparameters examined in the parameter tuning tests.

Abbreviation	Meaning
pos	part-of-speech
ne	named entities
mlp	multilayer perceptron
svm	support vector machine
mnb	multinomial naive Bayes
cnb	complement naive Bayes

Table 6.4: Parameter space abbreviations.

The first test deals with comparing part-of-speech (pos) and named entity (ne) features (see Section 2.2) that are available for classification. Regarding part-of-speech features, only terms that belong to the major part-of-speech classes [JM14], i.e., nouns, verbs, adjectives or adverbs, were utilized as features. For this feature comparison test, we compare the following sets of features: only part-of-speech features, only named entities from nltk, only named entities from spaCy - another NLP library [spa] - and part-of-speech with nltk named entities combined.

Figure 6.3 compares the ROC curves of classifiers based on the features mentioned above. Most noteworthy is the clear separation between part-of-speech trained and named entity based classifiers. This separation is present in both the initialization stage as well as after 200 labeled documents were added to the learning models. We hypothesize that this separation comes from the low number of features the named entity based classifiers have at this stage.



(a): Model at initialization stage.

(b): Model trained with 200 documents.

Figure 6.3: Comparison of all features with mnb model between stage 0 (a) and 200 (b). Tests run on the NYT dataset, using a three fold cross-validation procedure.

To investigate this hypothesis we picked two representative conditions, the nltk named entities and the nltk named entities with part-of-speech combined, and computed feature density and cross-document occurrence for both conditions. *Feature density* is the percent of terms that are chosen as features (see Table 6.5). The measured feature density shows that the combined feature set has more than twice the density compared to nltk named entities. Also, the feature density gets lower during training, indicating that each document introduces more new terms than named entities. Probably, the features representing the classes occur so infrequently that very few model features are present in the test data, complicating classification. We inspected how frequent features are

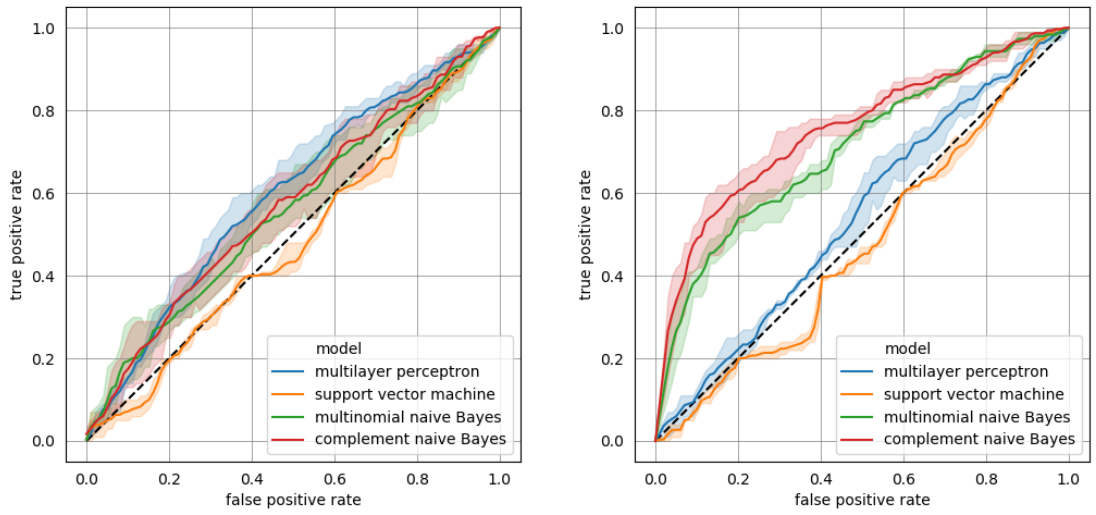
present in more than one document, which we denoted as the *cross-document occurrence*. The results show drastic differences between both feature variants (see Table 6.5). In this comparison, the combined feature set has occurrences of shared features between documents that are higher by a factor of ten.

NYT Feature Test			
Dataset	**Features**	**Feature Density**	**Cross-Document Occurrence**
Initial	ne (nltk)	16.545%	6.224%
Initial	ne (nltk) + pos	39.848%	76.493%
Full	ne (nltk)	8.140%	6.643%
Full	ne (nltk) + pos	23.538%	76.824%

Table 6.5: Comparison between feature sets at different model training stages based on feature density and cross-document occurrence. Feature density denotes the percent of terms in the documents that are utilized as features. The cross-document occurrence stands for the percent of features that are present in more than one document.

Due to concerns for large feature spaces, caused by a choice for part-of-speech features, we continued to investigate the discrimination performance during the active learning process. A performance comparison in the final stages of active learning can be seen in Figure 6.3(b). Interestingly, the performance ranking is similar to the initialization stage test, even when the number of features is substantially bigger. Multiple causes can lead to a better performance of part-of-speech features compared to the named entity approach. One factor could be the size of the dataset and thus the limited number of part-of-speech features present in the data. It seems that the testing dataset is not big enough to run into problems of big feature spaces. Another cause might be the sparseness in the named entity features (see Table 6.5), which might also cause an underperformance of this feature set.

As our visual active learning procedure is flexible regarding the classifier used, we compared a selection of classifiers with two variants of model features: named entities from nltk and part-of-speech with nltk named entities combined. In the test, we compared two variants of naive Bayes classifiers (multinomial naive Bayes (mnb), complement naive Bayes (cnb)) with a multilayer perceptron (mlp) and a support vector machine (svm). Figure 6.4 shows the results of both tests with different features at the classifier initialization stage. As already mentioned in the last test, it seems that the small number of named entity features is not descriptive enough for a effective classification. Figure 6.4(a) exemplifies this problem. All models perform similarly with performances that are close to classifying by chance, except for the support vector machine that performs even worse than chance. The test with features based on part-of-speech and named entities combined showed quite different behaviors (see Figure 6.4(b)). With the larger feature set, the naive Bayes methods perform significantly better, with complement naive Bayes outperforming the multinomial variant.

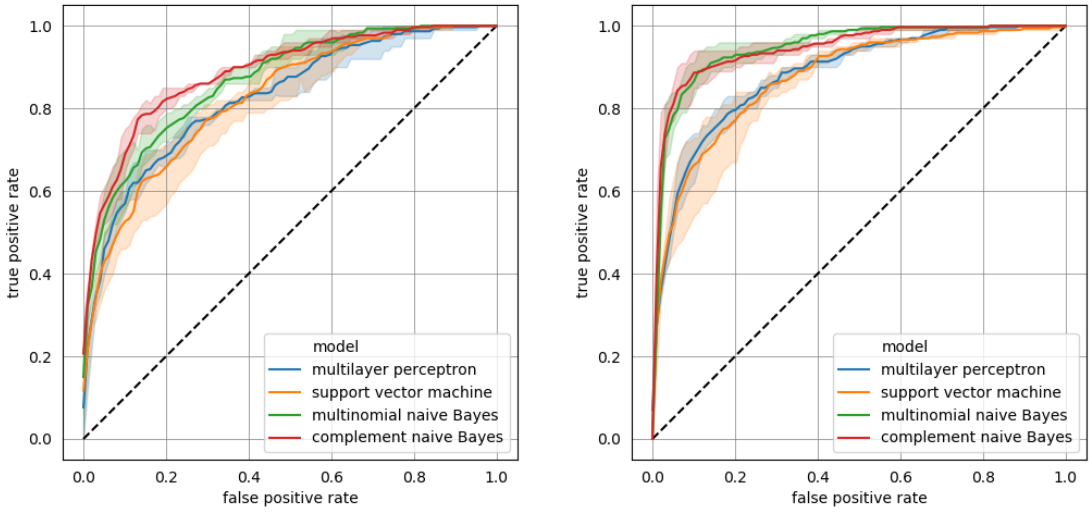


(a): Models based on named entities.

(b): Models based on named entities and part-of-speech.

Figure 6.4: Comparison of all classifiers during the initialization stage between (a) ne and (b) ne + pos. Tests were run on the NYT dataset, using a three fold cross-validation procedure.

Figure 6.5 explores the discrimination capabilities of the different models with 200 documents classified. This comparison shows that the performance between the models is stable concerning features and iterations. With the combined feature setting (ne + pos) (see Figure 6.4(b)), the naive Bayes models perform significantly better. In this test case, it can be noticed that the separation between the naive Bayes models and the other models is larger than the confidence intervals. From the parameter tuning tests we conclude that a combination of named entities and part-of-speech (ne + pos) with a complement naive Bayes (cnb) classifier yields the best classification performance for our data characteristics.



(a): Models based on named entities.

(b): Models based on named entities and part-of-speech.

Figure 6.5: Comparison of all classifiers during stage 200 between ne (a) and ne + pos (b). Tests were run on the NYT dataset, using a three fold cross-validation procedure.

6.5 Active Learning Strategy Evaluation

Our application needs to represent different active learning strategies visually. To evaluate this functionality, we compared our visual active learning strategies approach to classic active learning strategies by simulating the process. We performed these experiments on our balanced NYT dataset and the R8 subset of the Reuters-21578 dataset (see Section 6.1).

Figure 6.6 shows the accuracy multiple labeling iterations on our NYT dataset. In general, if the confidence intervals are taken into account, no single strategy outperforms any other significantly. This observation fits very well with the observation made by Seifert and Granitzer [SG10] and Settles [Set09]. For a clearer comparison between corresponding strategies, Figure 6.7 shows the strategies on separate graphs. In Figure 6.6, it is noteworthy that the correction strategies perform worse the random strategy at the start of the active learning process. The reason for this is an initial imbalance of classes. The selection strategies create a big imbalance of classes in the first twenty iterations.

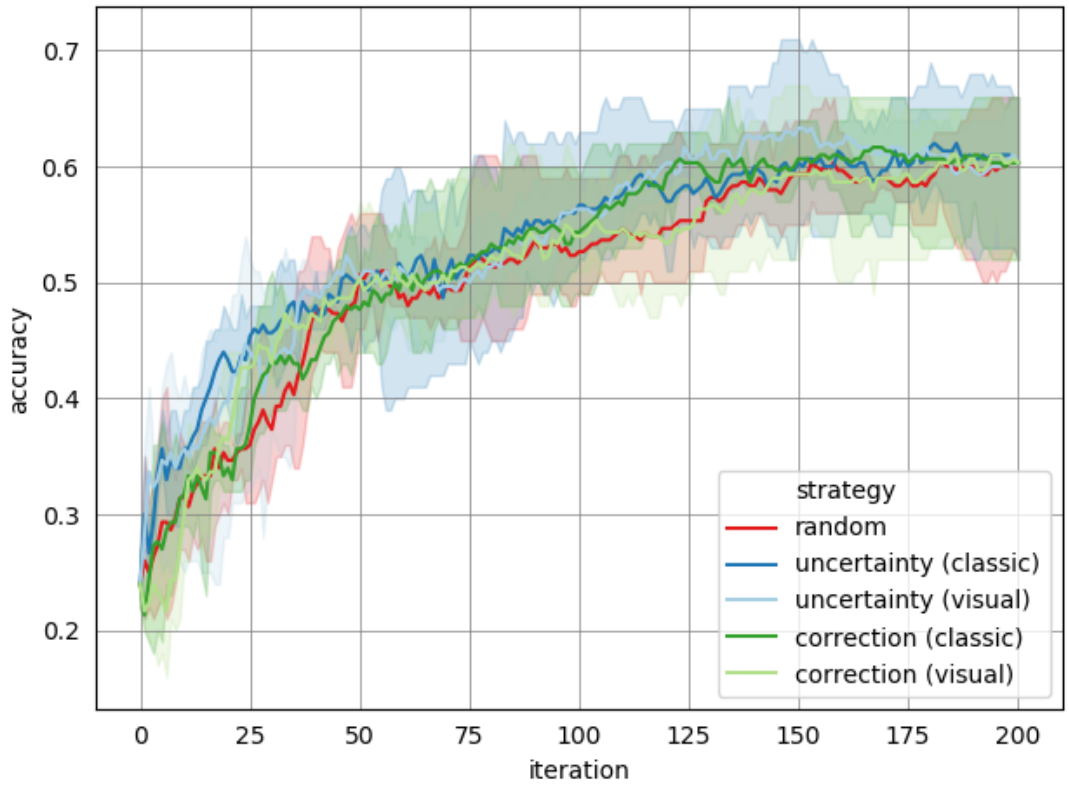
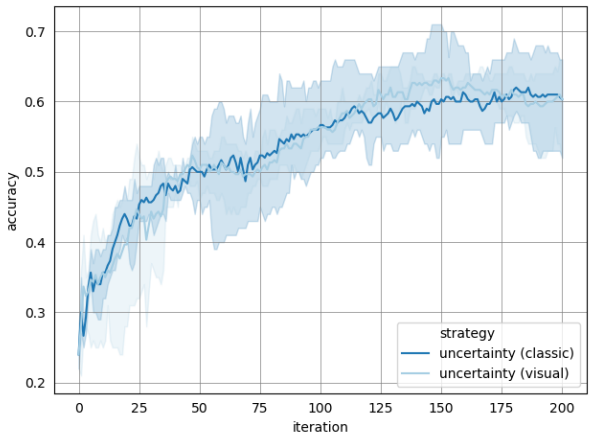
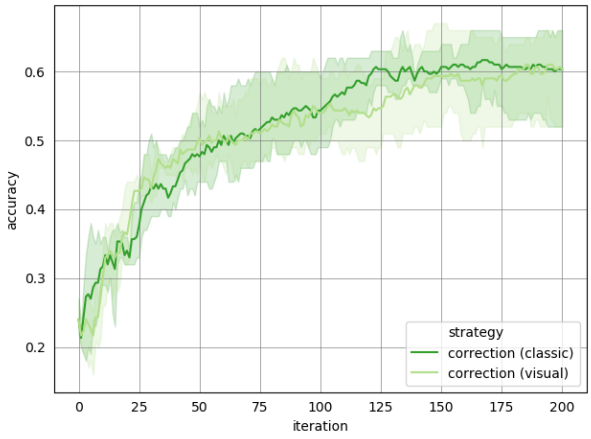


Figure 6.6: Comparison of labeling strategies in terms of accuracy over iterations. Simulated strategies were tested by using a three fold cross-validation procedure on the balanced NYT dataset (see Section 6.1).

We performed this test also on the Reuter R8 dataset. The results of this test are depicted in Figure 6.8. Due to the size of the dataset, this test was run in a fixed training/test set configuration. Nonetheless, the results indicate some interesting connection between visual and classic strategies. Between iteration 1500 and 3000, both correction strategies show the same oscillation artifact, which is for both strategies the point where they reach their peak accuracy. In comparison, the uncertainty strategies can steadily increase their performance longer than the correction strategies up to iteration 3500, where both strategies drop similarly in performance. These two artifacts give some evidence that the visual approaches select documents in a similar manner to the classic active learning strategies.



(a) Uncertainty strategy comparison.



(b) Correction strategy comparison.

Figure 6.7: Comparison of corresponding visual and active learning strategies in terms of accuracy over iterations. The simulated strategies were tested by using a three fold cross-validation procedure on the balanced NYT dataset (see Section 6.1).

Due to the high number of iterations, the performance of the strategies in the early phases of the experiment is obstructed. Therefore, we plotted the starting section separately in Figure 6.9. This figure highlights an interesting aspect of active learning, as it is visible that the random strategy outperforms the selection strategies in the first hundred iterations. Due to the targeted selection strategies, the early training sets have the potential to get very unbalanced, which does not happen with a random selection. With an increasing number of classified documents, this effect gets smaller as the imbalances become proportionally smaller. The results from the tests with the R8 dataset support our hypothesis **H1**. In this comparison, the uncertainty and correction strategies were able to outperform a random labeling strategy in both their visual and classic variants.

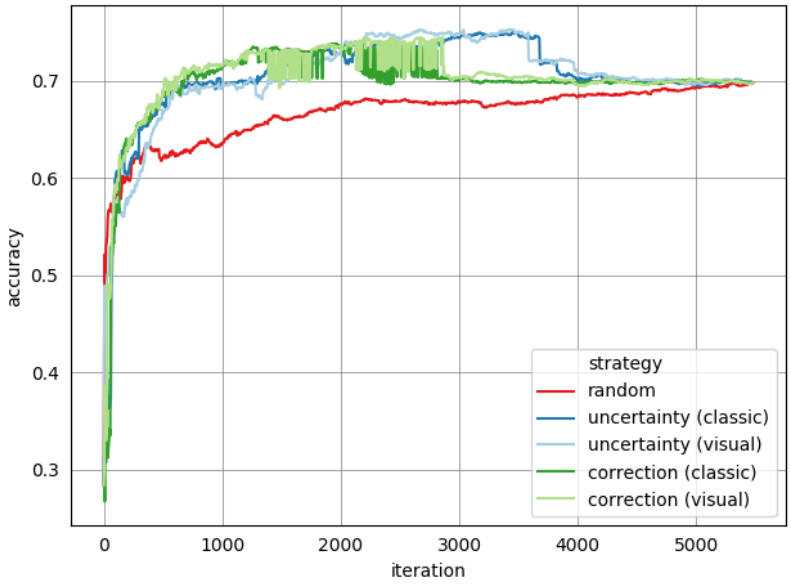


Figure 6.8: Comparison of learning strategies in terms of accuracy over iterations. Simulated strategies were tested by using a training/test set split on the Reuters R8 dataset (see Section 6.1).

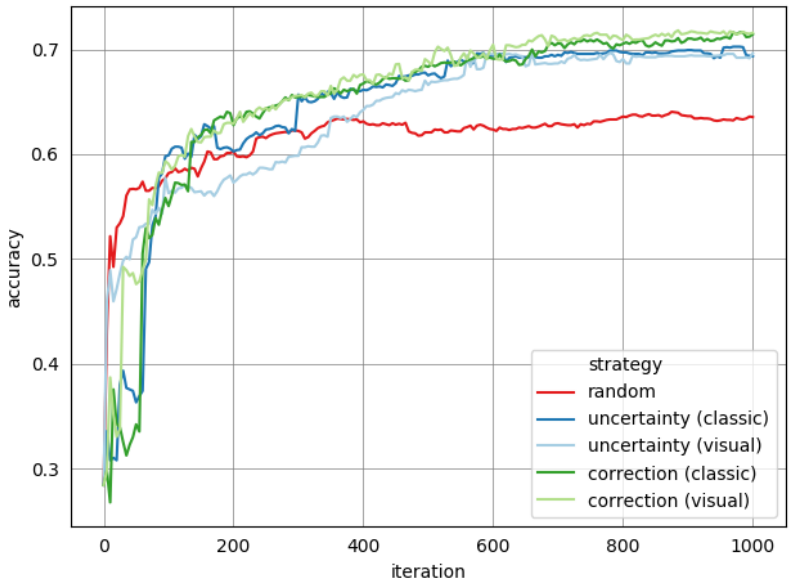
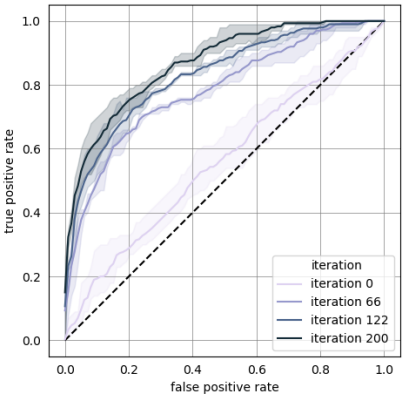
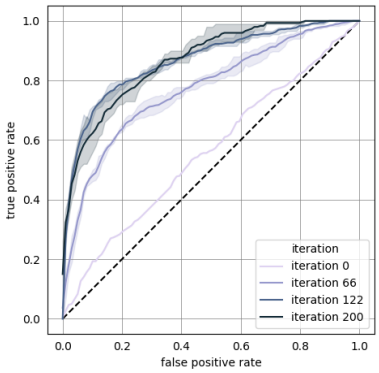


Figure 6.9: Starting section of the comparison of learning strategies in terms of accuracy over iterations on the R8 dataset (see Section 6.1).

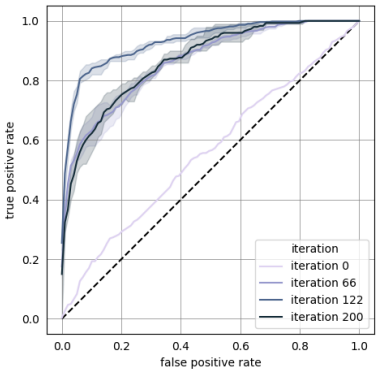
We also investigated the performance of the different strategies on the NYT dataset based on the ROC curve measure. As the ROC curve is already a two-dimensional feature, we chose to plot all strategies side by side in Figure 6.10. Each of these figures shows the development of a strategy at discrete iterations. In this comparison, the random strategy performs worse than all other strategies. All strategies besides the random strategy reach peak performance before the last iteration and drop to the random performance when the whole training set is used. This performance change might signal overfitting. It seems plausible that the increasing feature set can lead to such problems.



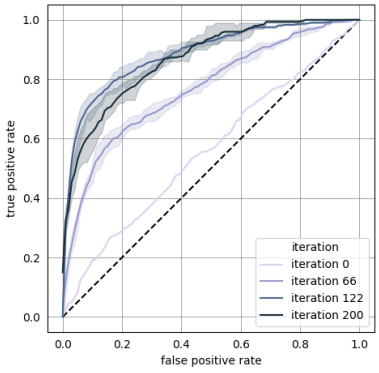
(a) random strategy



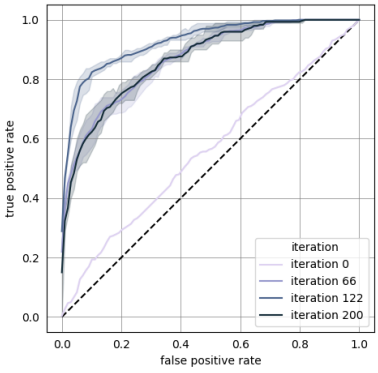
(b) uncertainty (classic) strategy



(c) correction (classic) strategy



(d) uncertainty (visual) strategy



(e) correction (visual) strategy

Figure 6.10: Comparison of different learning strategies in terms of discriminatory capabilities (ROC curve) over iterations. The simulated strategies were tested by using a three fold cross-validation procedure on the balanced NYT dataset (see Section 6.1).

CHAPTER 7

Evaluation

We conducted a preliminary user study to evaluate the benefits and limitations of incorporating users into the active learning process through visualization. For this preliminary study, we recruited four participants (one female, three males, age 25 to 28) with different backgrounds, including one computer scientist, but all experienced in using computers. We used the NYT dataset, introduced in Section 6.1, for the study and initialized the classes of the dataset with snippets of fitting Wikipedia articles. The choice for the custom NYT dataset is motivated through multiple factors. As the dataset's samples are current full articles from the New York Times, they are moderately simple to classify for the participants. Further, the categorization offered by newspapers is a classification that is comprehensible for participants as well. To compare the visualization's advantages, we implemented the list condition as a baseline.

7.1 List Condition

The list condition moves the functionality of the visualization into the document list. In particular, the classification, topic filtering, and indication of new items are performed by interacting with the document list. To enable classification, a set of buttons (see Figure 7.1(a)), where each button represents a topic, is added to each document card in the document list. Through this approach, the document cards can express their topic affiliation and enable classification, as the button of the current affiliation has a different style. For filtering, a set of topic buttons (see Figure 7.1(b)) is also added to the header of the application. Similarly to the classification buttons, these buttons also change style if active, filtering the document list beneath. Finally, new documents are added at the top of the list and are marked with a red border so users can track the application's actions better.

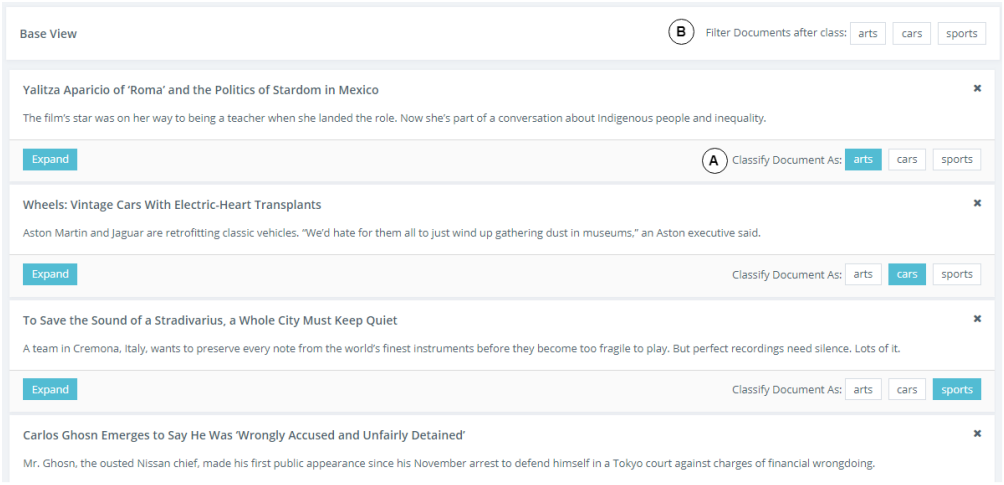


Figure 7.1: In the list view the documents can be classified with (a) the topic buttons. To filter documents, (b) filter buttons are added to the view in the top right.

7.2 Hypotheses

In this evaluation, we focus primarily on the benefits and limitations of the visual active learning approach of our application compared to a simpler learning approach of assigning class labels to documents. Particularly, we investigate the benefits of the visualization concerning the training process of the learning model. Our assumption has been that visualizing the model's prediction beyond showing the final label can have multiple benefits. Displaying the classification probability based on the model can be interpreted as the confidence of the classifier. In this probability space, documents are placed in proximity to each other if they are similar with regard to the model's decision boundary. By visualizing this space, users can find similar documents, interpret the decision boundary of the model by exploring the proximity of documents and how the model's classification reacts to change. We, therefore, formulated three additional hypotheses:

Our assumption has been that the visual information improves the user's ability to identify a particular class, even if the classification is not perfect. **H2:** *The visualization of document assignments to classes helps users to decide, whether a news item is interesting or not for a particular class.*
We expect that at a certain accuracy of the underlying model, users can find relevant news more efficiently. As documents enter the visualization, they move in the direction of their classification. Then users can disregard a subset of entering documents, which do not move to the class of interest, as they signal that they are not relevant for the category of interest. Therefore, we hypothesize that users select fewer irrelevant news items, which leads to a higher selection precision (**H2.1**). Also, filtering documents from the list of documents to parse, should accelerate the process of finding relevant documents, i.e., increase the selection recall (**H2.2**).

H3: *The direct interaction with the visualization is more efficient and effective for active learning than a classic interface.*
Due to the more informative arrangement of the documents in the two-dimensional visualization, we expect users to improve the classifier with fewer re-classifications to a useful accuracy than with the list condition (**H3.1**). Encoding additional information in a visualization yields multiple benefits for the classifier training task. Changes in the model are more apparent with the visualization during training, as users can track changes of document classifications over the whole dataset by following the movement during an update. Further, if documents move together, this indicates that they contain the same features that were deciding the classification. Through investigation of these subsets of documents, users can more easily interpret the decision boundary of the model and adjust it to their interests. These factors should influence the effectiveness of the training as well, which lets us expect that the overall accuracy will increase more consistently over multiple reclassifications than with the list condition (**H3.2**).

For the final hypothesis, we inspect the users' performance in relation to the simulated learning strategies examined in Chapter 6. **H4:** *Users outperform active learning strategies by adapting their strategy based on the learning model state.*
Compared to the simulated strategies, users are more flexible in the selection of their labeling. We expect users to interpret the current state of the learning model and to adapt their strategy based on their observations. Therefore, the user-based training models should outperform the simulated learning strategies as well as the random strategies in terms of accuracy over iterations (**H4.1**).

7.3 Study Design

For this study, we employed a within-subjects design with the visualization as the independent variable. We designed study so the independent variable has two conditions. The first condition contains the interface with the visualization (see Section 4.8) and is called visualization condition. The second condition is called list condition, which uses the interface without the visualization (see Section 7.1). Our study comprises of two time-limited tasks, one document selection task to verify **H2** and a second model-training task to verify **H3** and **H4**. Due to the within-subjects design, users performed each task once per variable, resulting in four counterbalanced tasks. We split our NYT dataset (see Section 6.1) into four subsets for the evaluation. All subsets have similar size and distribution of classes as can be seen in Table 7.1. The study was conducted in the Google Chrome web browser on a 32" monitor.

	NYT dataset split			
Class	**# set 1**	**# set 2**	**# set 3**	**# set 4**
business	34	27	25	31
health	19	18	18	15
science	26	24	20	9
technology	13	13	15	29
world	40	36	36	43
Total	**132**	**118**	**114**	**127**

Table 7.1: Sizes of the NYT subsets, used for the preliminary user study.

Initially, participants filled out a consent form and a short demographics questionnaire. During the study, each task was preceded by a task description (see Table 7.2) in the browser together with a test task using a test dataset. When the participants felt comfortable with the task, the study proceeded to the measured study task. Both tasks were limited in time to five minutes, where users started with thirty documents and the application added documents periodically during this time. In the selection task, participants had to identify incoming documents of a particular class and select them. During this task, the learning model was not trained beyond the initialization. For the training task, participants trained the model to increase the model's classification accuracy.

Study Tasks	
Select	Your task is to forward any **health**-related news to the fictional management by pressing the button "select" in the user interface.
Classify	Your task is to improve the systems classification process by classifying examples that fit the systems categories yourself. Keep in mind that you can classify every document shown in the interface to every category. You don't have to focus on one category.

Table 7.2: Every task in the study was proceeded by a website containing a scenario, task description, and explanations to the user interface utilized in the task. This table contains the task description shown on that website.

7.4 Analysis

To evaluate our hypotheses, multiple scores were measured during the study. For hypothesis **H.2**, we investigated whether the visualization benefits users in selecting relevant documents for the task. The number of selected documents should show whether the visualization helps to find documents faster. Beyond the sheer quantity, we were also interested in the visualization's capability to guide users to relevant documents. In the

context of selecting health-related documents, a document d from corpus D is relevant if the ground truth coincides with the selection of the user for the given document. Formally:

$$d \in G_h \wedge d \in S_h \quad , \tag{7.1}$$

where the G_h is the set of health-related documents according to the ground truth $G = \{G_b, G_h, G_s, G_t, G_w\}$ and S_h is the set of health-related documents from the users entire selection $S = \{S_b, S_h, S_s, S_t, S_w\}$. Based on this relevancy definition we evaluated the precision and recall for hypothesis **H.2**. In particular, *precision* gives insight into the fraction of relevant documents that were retrieved by the participant, resulting in the formula:

$$precision = \frac{|S_h \cap G_h|}{|S|} \quad . \tag{7.2}$$

Recall measures the fraction of relevant documents that were successfully found by the participant, which leads to this formula:

$$recall = \frac{|S_h \cap G_h|}{|G_h|} \quad . \tag{7.3}$$

We evaluated hypotheses **H.3** and **H.4** by measuring the model's accuracy over iterations on an independent test set. The accuracy measure gives a basic overview of the visualization's capability to guide users to documents that improve the classification efficiently. As with the measures introduced in Section 6.3, we have to adapt accuracy for multi-class classification. We chose to use micro-averaging in this instance as well, so that the measure can capture class imbalances [VA13]. Adapting *accuracy* to micro-average over the classes leads to the following formula:

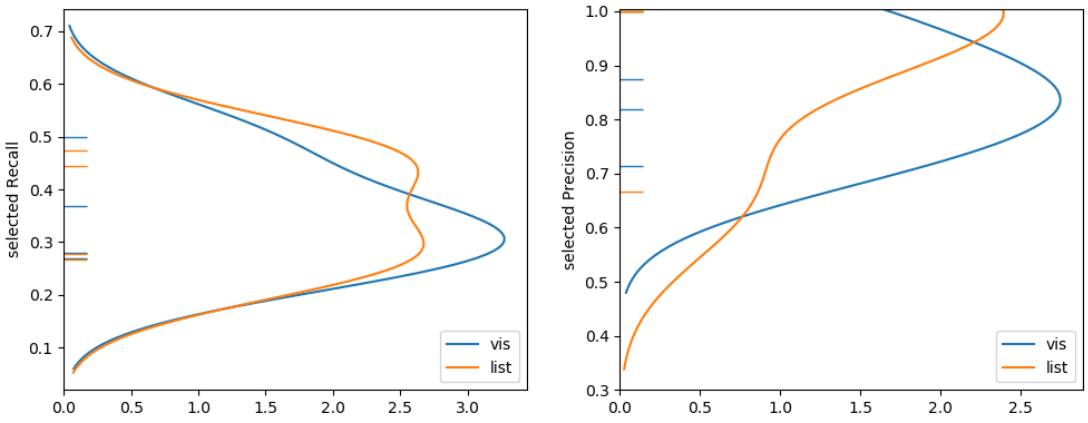
$$accuracy = \frac{\sum_i |P_i \cap G_i|}{|D|} \quad , \tag{7.4}$$

where P_i and G_i are the model's prediction and the ground truth for a certain topic i respectively.

7.5 Results

We evaluated hypothesis **H2** by examining the distribution of the measured precision and recall. In general, participants selected slightly more documents, on average, with the visualization condition (eight documents on average) than with the list condition (seven documents on average). To test hypothesis **H2.1**, we compared the recall measured during the selection task. This measure is taken at the end of each task from the participant's final models. Using the visualization, the participants' fraction of relevant documents found is slightly less than with the list condition (see Figure 7.2(a)). In both conditions, users were only able to identify around half of the health-related items during the study.

For hypothesis **H2.2**, we calculated the precision of both conditions from the final models of the selection task data. Similarily to the recall, the precision of the visualization condition is slightly less than the list condition (see Figure 7.2(b)). Using the list condition, three out of four participants picked solely health-related items, while the visualization condition only reaches an average recall of 90%. We inspected the seven selections from other classes (see Table 7.3). From this set of documents, four documents are health-related despite being labeled as belonging to another class in the ground truth. On closer examination, the remaining three documents are not related. However, the list condition still outperforms the visualization condition slightly. *This disproves hypothesis **H2**: In the tested circumstances, the visualization of document assignments to classes does not help users to decide the relevance of a news item for a particular class.*



(a) Recall

(b) Precision

Figure 7.2: Precision and recall of the participants' final models from the study displayed in a kernel density estimation (KDE) plot. The KDE plot estimates the probability density function of both measures by smoothing the observations that are displayed on the y-axis with a gaussian kernel.

	Selected Documents
Vis	Aaron Klug, 92, Dies; His 3-D Images of Bodily Molecules Won a Nobel
	In China, Gene-Edited Babies Are the Latest in a String of Ethical Dilemmas
	New Diet Guidelines to Benefit People and the Planet: More Greens for All, Less Meat for Some
	New Popeye Videos Show What 90 Years of Spinach Can Do for a Guy
	April Bloomfield Closes Her Los Angeles Restaurant Hearth & Hound
Base	Retiring: If You Do Medicare Sign-Up Wrong, It Will Cost You
	Wealth Matters: Taxing the Wealthy Sounds Easy. It's Not.

Table 7.3: Table showing the titles of the non-health related documents that were selected by the participants.

To verify hypothesis **H3**, we look again at the clearly defined sub-hypotheses **H3.1** - **H3.2**. Hypothesis **H3.1** involved investigating the final accuracy of the participants' models during the model-training task. In this task, the visualization condition performed slightly better than the list condition (see Figure 7.3). Additionally, we compared the accuracy of the participants models over iterations to evaluate hypothesis **H3.2**. As with the final model, a similar trend is visible along all iterations: the visualization condition outperforms the list condition slightly (see Figure 7.4). *We therefore confirm partially hypothesis **H.3**: The direct interaction with the visualization is slightly more efficient for active learning than a classic interface.*

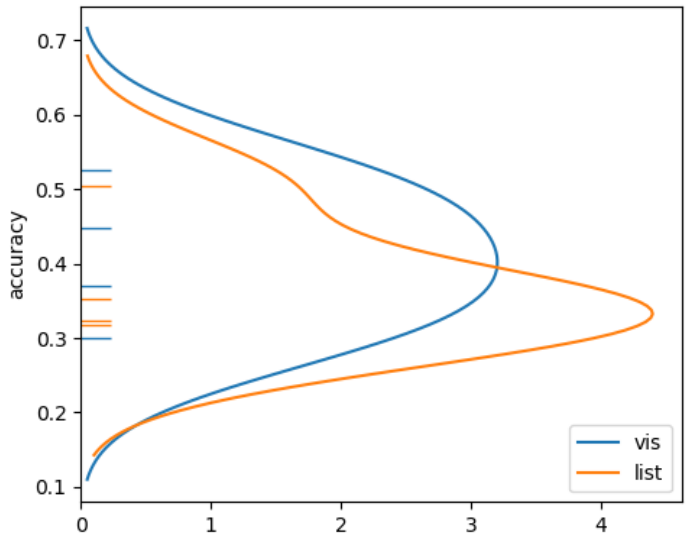


Figure 7.3: Accuracy of the participants final models from the study displayed in a KDE plot.

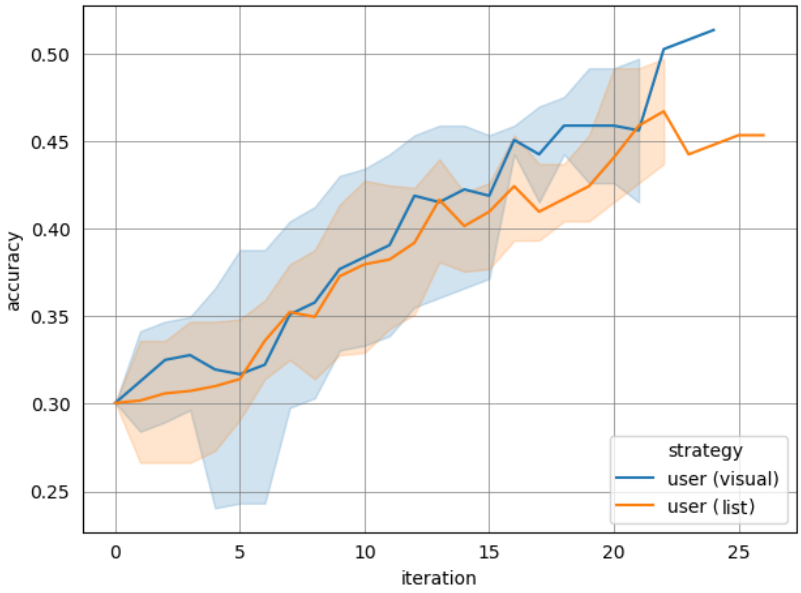


Figure 7.4: Comparison user accuracy between test conditions.

As our experiments in Chapter 6 applied some of the same measures, we also created a comparison between users and the automated techniques and defined Hypothesis **H4**. For this test, the study participant's choices and all automated learning variants were simulated again with a consistent test set so the results are comparable. Figure 7.5 shows this comparison, which is run on the NYT dataset with a fixed training/test set configuration (70% training, 30% test).

To get a better insight into the early perfomance, Figure 7.5 shows the first 100 iterations separately. In this figure, it becomes apparent that the user-trained models perform comparably to the best performing automated strategies. Especially the visual condition performs quite well. The model reaches the classification accuracy of 51% in 24 iterations, which is not outperformed until the random strategy reaches its 37th iteration. In this test, the user-trained model outperformed the other strategies with 35% - 48% less data, which indicates that involving further human decisions into the learning process is beneficial. However, the participants' data does only cover the initial twenty iterations due to the time constraint of the test. *Therefore, we cannot confirm the hypothesis **H4**: The initial iterations of the visual active learning strategy indicate performance on par with the best performing strategies in the test. However, the visual active learning strategy does not outperform other strategies. Also, the data just covers the initial learning stage, to confirm the hypothesis models need to be trained over a longer period.*

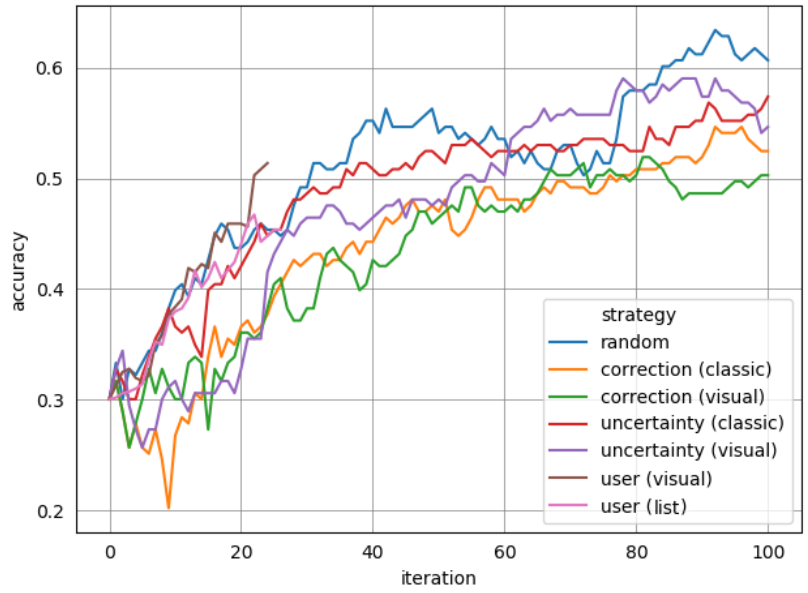


Figure 7.5: Comparison between users and automated strategies on same independent test set.

7.6 Limitations

There are multiple limitations present in our study. It is important to consider that this was a pilot study, no real conclusions can be drawn from the results. Due to the small sample size, it was not possible to show significant differences between conditions. However, the pilot study was helpful to test the design of the study to determine which aspects work and what could be improved. Besides the low number of participants, the time limit proved to be too restrictive for the task. In both conditions, study participants defaulted to only reading article headlines to maximize the number of documents they inspect for classification and selection. This strategy proved very efficient, as the classes in our study are based on simple news categories. For a follow-up study, it might be beneficial to increase the time limit and define harder tasks for the users, so that they have to engage beyond headlines with the application. Another limitation that became apparent after the study is the limited benefits the visualization provides at the initialization stage. In this configuration, the affiliations to topics and proximity to other news documents did not encode helpful information. This might be the explanation for the worse recall and precision observed in the selection task. To test the efficiency of selecting relevant documents, the model has to be sufficiently trained so the visualization can provide actual benefits. For future studies, it is necessary to pre-train a model or redesign the classification task so participants can train a model and select documents.

CHAPTER 8

Discussion

With the simulated experiments and the pilot study, there are multiple insights found within the studies results. Therefore, this chapter will mainly focus on investigating the hypotheses defined in Section 1.4. Subsequently, we will discuss qualitative observations that were made during the conduction of the study.

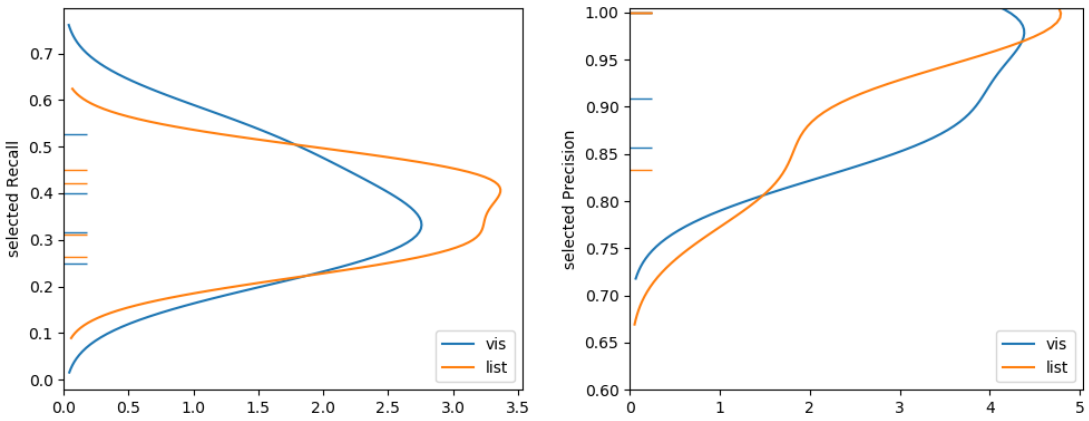
8.1 Discussion of Hypotheses

The results from the investigation of **H1** in Section 6.5, already indicated some promising aspects about how the classic active learning strategies compared to their visual counterparts. While in our tests there was no clear strategy that outperformed the others, there were some noteworthy trends. The results of the accuracy comparison on the Reuters R8 dataset (see Figure 6.8) indicates that the visual active learning strategies correspond very closely to their classic active learning counterparts as they exhibit very similar patterns in performance. With increasing iterations, all active learning strategies were able to perform similarly or better than the random strategy on average. One crucial observation we have made from these results is that the other strategies outperform the random strategy after an initial phase of rapid improvement in the active learning process (see Figure 6.9). In our testing, that phase encompasses crudely the first two-hundred labeled documents in a classification task with five to eight classes. Based on these observations, we speculate that it is necessary to label at least twenty documents per class to start getting benefits from the active learning process. Therefore, in our use case of a personalized classifier, the benefits of using such an application may not show after a single use of our application.

Multiple aspects may shorten this initial phase, improving the usability of the application. The initial performance of the learning model could be negatively affected by the imbalances the active learning strategies cause. Often, learning models are not incentivized enough to optimize for underrepresented classes, as these classes have a very low impact on the loss that is minimized. One possible solution to this problem is to make imbalances in the visualization more salient. We therefore color-coded the documents that are in the training set as black in the visualization. A recommender system could be added to the application, which marks documents that it deems to be representative of an underrepresented class. This could shift the users' attention towards labeling these documents. Improving the learning models' features might also be an option to find a suitable solution for the classification problem faster. Descriptive, high-level features could improve the training of the learning model drastically as they simplify the feature space in which a solution is searched. In the tests that were run for this hypothesis, named entities were utilized, which on closer inspection turned out to be very sparse. It might be more reliable to improve on the named entity algorithm used or to fall back to simpler features such as n-grams or a set of descriptive lexical categories.

Inspecting the results of hypothesis **H2** highlighted flaws in our study design and usability problems in our application. The results (see Figure 7.2) show a considerable difference between the conditions in both measures. Also, the recall in both conditions was generally low. We suspect that these results are strongly influenced by the study design. Due to the restrictive time frame, the average number of selection per user is with around seven documents very low. In our case, single mislabeled items caused this considerable difference between conditions. This is exemplified in Figure 8.1(b), where the difference in precision across participants shrinks significantly by only changing a few ground truth labels. Similarly, the low recall is mostly caused due to a flaw in the study design. The task was designed in a manner so that users had not enough time to check every document in the dataset during the test. We expected that the grouping and overview the visualization provides in an initialized state, would let users prioritize relevant documents, thus making the task easier within the time frame. However, we overestimated the visualizations' capabilities to guide users to documents, when the learning model is just initialized. In addition to this misjudgment, users struggled with the size of the tooltip present in the visualization, making exploring the data even harder.

Following the study, we increased the size of the document points as well as the tooltip to improve the direct interaction design of the visualization. To improve upon the study design for the following studies, it is crucial to improve the initial training set of the topics for the selection task beforehand, so that the benefits of the application can be captured more accurately. Further, the selection task would benefit from a longer task time with a bigger dataset so that the time limit of the task is preserved. In this manner, users would still be time-constrained as in the original design. However, the longer test time would result in a greater average number of selections the participant can make, potentially showing the differences between the conditions more clearly.



(a) Recall

(b) Precision

Figure 8.1: Precision and recall of the participants final models, if documents from other classes that are health-related are taken into account.

Similarly to the observation made previously, the limited scope of the study limited the insights gained in the classification task for both hypotheses **H3** and **H4**. Our earlier tests on this dataset indicate that the learning model needs around two hundred iterations to reach a state where the iterative improvements slow down. In the study's time limit of five minutes, users were not able to label this amount of documents. Giving users access to the software to use it over multiple sessions or designing a task that runs over one longer session might yield more insightful results. To evaluate if the participants learning strategies outperform the classic active learning strategies, more iterations are needed so they could potentially reach their performance ceiling.

8.2 Qualitative Observations

In summary, the results indicate that visual active learning has some advantages over the classic counterpart. However, as participants were not able to sufficiently train a learning model in the time constraint of the study the results are not conclusive. We, therefore, examined the participants' strategies individually and will discuss the qualitative observations made. Besides the measured results, the preliminary study was able to show how slightly trained users interact with both conditions.

In the visualization condition, participants tended to explore in a region of choice rather than following updates. During the selection task, one participant found that many of the relevant documents were positioned in the center of the visualization and decided to focus on this region. As the learning model was only initialized, this proved to be an efficient strategy. Further, during the classification task, participants tended towards starting by classifying documents positioned in the center of the visualization and moving on to classifying and correcting, when a prominent bigger cluster formed around a topic. It seems that the pattern of one prominent bigger cluster that forms if the model or the dataset is imbalanced captures the participants' attention. By exploring the bigger cluster, participants move from an uncertainty based strategy in the start to a correction strategy, which is a behavioral pattern very beneficial for the model. On the other hand, some participants did not notice the imbalance pattern of the visualization during classification, leading to poor results. The visualization could benefit from a mechanism that enables users' to get an initial assessment of the model's classification results without interaction with the points.

Compared to the visualization condition, users interacted with the document list in a very different way. In general, participants tended towards examining the list from top to bottom or vice versa. When they reached the bottom of the document list, they moved towards the top and investigated the latest update. Due to this process, multiple participants missed multiple updates as they happened while they were working through entries down the list.

The interactions of the participants with the list made two drawbacks of this approach very apparent. First, users tend to miss changes, if they are not directly on the screen when they happen. To remedy this flaw, it would be necessary to encode changes in a document in a permanent manner until users interact with it. Second, the list itself does not inherently promote any selection strategy. To support active learning it would be necessary to order elements in a way that promotes a certain selection strategy. This way, users that are working through the list from top to bottom can follow a more efficient strategy than random selection.

CHAPTER 9

Conclusions and Future Work

This final chapter summarizes our contribution to classification supported by active learning. Additionally, we address the current limitations of our application with options to improve these factors in the context of future work.

9.1 Conclusion

We presented a new active learning interface for the classification of news streams, providing guidance for labeling, by visualizing the topic affiliation probabilities of the learning model and providing novel interaction tools to enhance the model's performance sequentially. Our approach adapts the active learning approach by extending the users' involvement in the process by letting them decide on the selection strategy. Selecting samples is aided by our visualization of topic affiliation probabilities, that gives the users an overview of the whole dataset. In this visualization, users can interactively explore the classification results and deduce which terms in the samples are deciding for the learning model by comparing nearby samples. Besides conveying the learning model's result and decision boundaries, the visualization also serves as a direct interaction tool for classification and reclassification. These interactions initiate an update of the learning model, which in turn updates the visualization, creating an interaction loop. Further, our application contains interaction tools to add and remove samples, which allow for incrementally training on text streams or curating a dataset in general.

From the evaluation of our hypotheses, we conclude that the selection strategies present in our visual representation closely resemble classic active learning strategies. Our active learning strategies performed similarly or better than classic active learning strategies and showed very similar performance patterns to their corresponding strategies in our selection strategy comparisons. Further, our pilot study indicates that users perform better with the visual representation than greedy active learning strategies. The experiments on different datasets show a varying performance of the strategies indicating that the benefit

of a certain strategy is dependent on the dataset and the model state. It seems that adopting the selection strategy to the current model state is very important in the active learning process. Finally, testing on our dataset has shown that the learning model of our application needs more data to achieve sufficient accuracy. Initially, we anticipated that the application would show its benefits after an initial set of around twenty labeled documents. After evaluation, the results indicate that beneficial performance is reached with around twenty documents per model class. This number could increase if content characteristics change over time. Based on these observations, we conclude that this form of visual active learning has potential to enable users to determine whether a document is relevant for any of their defined classes (i.e., assets or topics). Beyond text classification, the concept of visual active learning can also be applied to a broad range of classification tasks. These include text related classification problems like sentiment analysis, language detection, and intent detection and even classification tasks on different data types, like image classification.

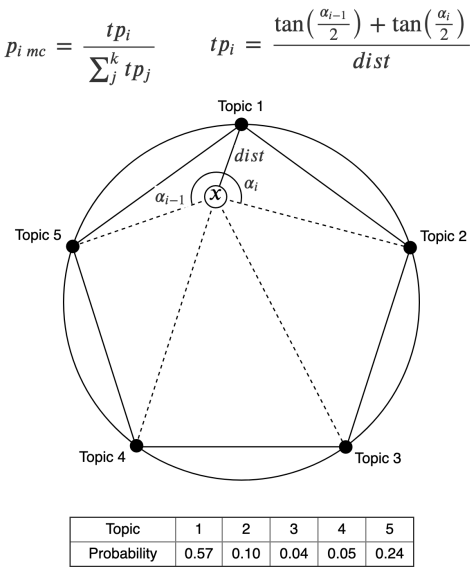
9.2 Future Work

Multiple of the following ideas come from observations made and feedback gathered during our preliminary study. There are multiple aspects that users struggled with when employing our application that can be improved upon. Our collaborators from the financial domain also provided valuable feedback and suggestions to us, which shows that there is interest in this type of classification tool.

The results of our evaluation have outlined multiple aspects, our learning model could be improved upon. In our examination of feature density and cross-document occurrence (see Table 6.5), it became apparent that the named entities utilized in our tests are quite sparse and often occur only in singular documents. Based on this observation the learning model would greatly benefit from improving on the utilized features. The performance could be improved by incorporating other NLP features such as a selection of descriptive lexical categories, n-grams, named entity relations, or pre-trained word embeddings. Another possibility is to change the learning model to a model like the CNN [Kim14] that can learn representative features.

Examining the design of the application, there is a mismatch between the encoding of the visualization and the interpretations of the users' classification interaction. When the model is updated, the documents points are placed in proximity to topics points based on the model's certainty that a document is affiliated with a topic. Classification is included in the visualization as a direct interaction tool. Ideally, the classification interaction would capture proximity information to the topic points as well, to incorporate additional information into the classification. However, the multi-class learning models used are not able to utilize additional information beyond a singular label. Therefore, the classification interaction signals users a singular label by highlighting the corresponding topic point during classification.

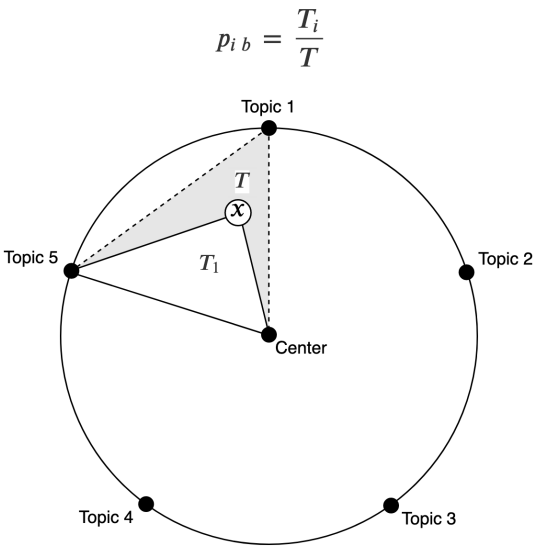
During development, we explored solutions to translate the document placement into assignment probabilities. Concerning document placement, in the case of this visualization, the class labels form a convex planar polygon in which documents are placed (see Figure 9.1). One solution to translate the distance relations between the document point and the labels is to compute a coordinate based on the class labels. Hormann and Floater [HF06] present such a solution by extending the concept of the barycentric coordinates to arbitrary polygons in a plane. These coordinates can provide relations between each class label and a position. However, these mean value coordinates may not reflect the intentions of a user. Users may not take the distances of all labels to the point they are manipulating into account.



Topic	1	2	3	4	5
Probability	0.57	0.10	0.04	0.05	0.24

Figure 9.1: The mean value coordinates from Hormann and Floater [HF06] can be utilized to interpret a documents probability from the position of a document. Here, the space in the Star Coordinates visualization is mapped onto the convex polygon that is produced by connecting all topic points. The mean value coordinates provide a representation of the document in relation to all topics.

A more appropriate approach could be to apply barycentric coordinates for triangles. By placing a triangle through the center point and the nearest two class labels and computing the coordinates based on this triangle, the resulting probabilities may fit closer to the interpretation of a user (see Figure 9.2). Here, the coordinate corresponding to the center of the visualization holds the users' certainty while the other two coordinates hold information for multi-class documents.



Topic	1	2	3	4	5	center
Probability	0.55	0.00	0.00	0.00	0.15	0.29

Figure 9.2: Barycentric coordinates can provide another solution for translating a document's position into a probability. For this approach, a triangle is placed in the Star Coordinates visualization between the nearest two topic points and the center. The resulting probabilities represent the document as a mixture between the nearest two topics while also encoding a sense of user certainty based on the distance to the center of the visualization.

With probability-based labels, there is still a need for a learning model that can utilize the information. Multiple learning models can accept labels in other forms than a hard assignment to a certain class. The field of *multi-label classification* is concerned with developing classifiers for problems where the given label is in the form of affiliation probabilities to all classes. Often, the multi-label problem is transformed into multiple single-label problems. One such popular approach is binary relevance [ZLLG18], where an ensemble of single-label binary classifiers that has one classifier per class is trained. The union of all classes that were predicted is taken as multi-label output. However, this method is not able to take dependencies between labels into account. The label-powerset [TK07] method can take dependencies between labels into account by expanding the number of single-label classifiers with a selection of possible combinations of labels. Adding the probabilistic information that reflects the users' confidence to their classification is another way how our additional information could be used. Nguyen et al. [NVH11] approached this problem of learning binary classification models from soft labels by replacing the probability assessments with the pairwise ordering of data samples in the training data. These pairwise orderings are computed from the probabilistic soft labels, which are then used to train a classifier.

In general, utilizing the Star Coordinates visualization to generate topic affiliation probabilities has the potential to simplify multi-label classification. Using the described procedure to generate labels is beneficial as it limits the size of the label powerset to the neighboring classes found in the visualization. However, to generate a useful simplification of the classification, the reduced label powerset still has to capture the major correlations between class labels. This approach is therefore very sensitive to the topic order.

Another aspect that could be improved is the application's ability to signal the state of the model and the result of the classification itself. Through the restrictive time frame of the preliminary study, it became apparent that users struggle to identify documents that are relevant for a topic. This is the result of hiding the titles of the documents, when the document points are not interacted with. In a similar matter, it is hard to assess whether the learning model in the background has captured a meaningful representation, as this judgement also needs interaction with the document points. Our current application only enables users to asses the model indirectly by comparing the terms of documents that are placed in the same region of the Star Coordinates visualization. A solution for both of these problems could be achieved by utilizing more information from the learning model itself. In case of our naive Bayes classifier, the empirical probabilities of features given for a class are very useful information, that could be utilized.

The empirical posterior probabilities are a basis on which our application can extend its functionality on the term level. These probabilities can be displayed in lists for each class separately with a filtering mechanism, so only term probabilities are shown that are very high or low. In this manner, users could assess how the model's understanding of a particular topic is evolving in time, based on the terms it deems as informative. Adding manual filtering for term probabilities would enable further exploration of the learning model. With more insight into how terms are utilized in the learning model, an interaction tool for manipulating individual terms becomes more valuable. Therefore, the application's term-weighting capabilities could be enhanced, so that the weights of the terms can be changed sequentially throughout the learning process. Finally, with available term probabilities, the highlighting of features based on probabilities in the documents is another use of this data that is very beneficial to the task of model training and assessing documents. By highlighting terms that are deemed decisive for the model, users can find phrases in documents that contain these terms. This guides users to parts of the document that might hold relevant information. As the meaning of longer text snippets is easier to interpret than single words, this information might help users refine the set of terms that is decisive for the classifier.

Besides a basic display, the term probabilities can be incorporated into a visualization. A visualization like a word cloud [CF01] for the term probabilities, where the font size encodes the probability, might ease the exploration of terms. Dynamic changes within the word cloud could give a helpful overview of how the classifier's understanding of a topic changes through an update. Incorporating the terms into our Star Coordinates visualization could improve its capability to convey the result of the classification without requiring interaction. Similarly to the FacetAtlas by Cao et al. [CSL+10], our visualization could incorporate a density map of terms that contains high-class probabilities or terms derived from a topic modeling approach. In this manner, the visualization could convey where certain content is placed within the visual depiction, allowing for a fast initial assessment of the results.

This overview of future work shows that visual active learning still has many open questions to be investigated in the future. Our work provided the first evidence that visual active learning through direct interaction has the potential to efficiently classify text stream data. We, therefore, believe that it will be worthwhile to address the open questions in the future.

List of Figures

List of Tables

Bibliography

[AA15] Rubayyi Alghamdi and Khalid Alfalqi. A survey of topic modeling in text mining. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 6(1), 2015.

[ACZ+11] Jamal Alsakran, Yang Chen, Ye Zhao, Jing Yang, and Dongning Luo. Streamit: Dynamic visualization and interactive exploration of text streams. In *2011 IEEE Pacific Visualization Symposium (PacificVis)*, pages 131–138. IEEE, 2011.

[Ama] Amazon Comprehend. `https://aws.amazon.com/de/comprehend/`. [Accessed 7-October-2019].

[Amb] Ambiverse, Text to Knowledge. `https://www.ambiverse.com`. [Accessed 7-October-2019].

[Ang] Angular. `https://angular.io`. [Accessed 7-October-2019].

[AZ12] Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM, 2002.

[BD06] Christopher D. Brown and Herbert T. Davis. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 80(1):24–38, 2006.

[BDK14] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247, 2014.

[Bel66] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[BGV92] Bernhard E. Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[BLBC12] Eli T. Brown, Jingjing Liu, Carla E. Brodley, and Remco Chang. Dis-function: Learning distance functions interactively. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92. IEEE, 2012.

[Ble12] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, April 2012.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[Boo] Bootstrap. `https://getbootstrap.com`. [Accessed 7-October-2019].

[Bos] Mike Bostock. D3.js, Data-Driven Documents. `https://d3js.org`. [Accessed 7-October-2019].

[Bre17] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*, volume 463. ACM, 1999.

[CC16] Nan Cao and Weiwei Cui. *Introduction to text visualization*. Springer, 2016.

[CD18] Marco Cavallo and Çağatay Demiralp. A visual interaction framework for dimensionality reduction based data exploration. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 635. ACM, 2018.

[CF01] Douglas Coupland and Jefferson Faye. Microserfs. *The American Review of Canadian Studies*, 31(3):501, 2001.

[CGJ96] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[Chu08] Stephanie Chua. The role of parts-of-speech in feature selection. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2008.

[Ciz18] Dea Cizmic. Exploratory data visualization dashboard for technical analysis of commodity market indicators. Bachelor's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, April 2018.

[CLR13] Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 827–832, 2013.

[CLRP13] Jaegul Choo, Changhyun Lee, Chandan K. Reddy, and Haesun Park. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, 2013.

[Coh17] David Cohn. Active learning. *Encyclopedia of Machine Learning and Data Mining*, pages 9–14, 2017.

[CSBL16] Alexis Conneau, Holger Schwenk, Loıc Barrault, and Yann Lecun. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*, 2, 2016.

[CSL+10] Nan Cao, Jimeng Sun, Yu-Ru Lin, David Gotz, Shixia Liu, and Huamin Qu. FacetAtlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181, 2010.

[CWL+10] Weiwei Cui, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X. Zhou, and Huamin Qu. Context preserving dynamic word cloud visualization. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 121–128. IEEE, 2010.

[Dal18a] Robert Dale. Text analytics apis, part 1: The bigger players. *Natural Language Engineering*, 24(2):317–324, 2018.

[Dal18b] Robert Dale. Text analytics apis, part 2: The smaller players. *Natural Language Engineering*, 24(5):797–803, 2018.

[Das17] Sanjoy Dasgupta. *Active Learning Theory*, pages 14–19. Springer, 2017.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology*, 41(6):391–407, 1990.

[DLZM10] Ali Daud, Juanzi Li, Lizhu Zhou, and Faqir Muhammad. Knowledge discovery through directed probabilistic topic models: a survey. *Frontiers of Computer Science*, 4(2):280–301, 2010.

[DS05] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of Reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 56(6):584–596, 2005.

[EASS+18] Mennatallah El-Assady, Rita Sevastjanova, Fabian Sperrle, Daniel Keim, and Christopher Collins. Progressive learning of topic modeling parameters: a visual analytics framework. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):382–391, 2018.

[EFN12] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482. ACM, 2012.

[ERT+17] Alex Endert, William Ribarsky, Cagatay Turkay, William Wong, Ian Nabney, I Díaz Blanco, and Fabrice Rossi. The state of the art in integrating machine learning into visual analytics. In *Computer Graphics Forum*, volume 36, pages 458–486. Wiley Online Library, 2017.

[FHJ51] Evelyn Fix and Joseph Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, University of California Berkeley, 1951.

[Fla] Flask | The Pallet Projects. `https://palletsprojects.com/p/flask/`. [Accessed 7-October-2019].

[FSZ+16] Weimiao Feng, Jianguo Sun, Liguo Zhang, Cuiling Cao, and Qing Yang. A support vector machine based naive Bayes algorithm for spam filtering. In *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2016.

[Gen] Geneea, Intelligent Interpretation. `https://www.geneea.com`. [Accessed 7-October-2019].

[GHN13] Emden R. Gansner, Yifan Hu, and Stephen C. North. Interactive visualization of streaming text data with dynamic maps. *Journal of Graph Algorithms and Applications*, 17(4):515–540, 2013.

[Goo] Google Cloud, Natural Language. `https://cloud.google.com/natural-language/`. [Accessed 7-October-2019].

[Har54] Zellig S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[HF06] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics (TOG)*, 25(4):1424–1441, 2006.

[HGM+97] Patrick Hoffman, Georges Grinstein, Kenneth Marx, Ivo Grosse, and Eugene Stanley. Dna visual and analytic data mining. In *Proceedings. Visualization'97*, pages 437–441. IEEE, 1997.

[HKBE12] Florian Heimerl, Steffen Koch, Harald Bosch, and Thomas Ertl. Visual classifier training for text document retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2839–2848, 2012.

[HMdCM17] Lulu Huang, Stan Matwin, Eder J. de Carvalho, and Rosane Minghim. Active learning with visualization for text data. In *Proceedings of the 2017 ACM Workshop on Exploratory Search and Interactive Data Analytics*, pages 69–74. ACM, 2017.

[Hof99] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[HQZ17] Minlie Huang, Qiao Qian, and Xiaoyan Zhu. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)*, 35(3):26, 2017.

[IBM] IBM Watson, Natural Language Understanding. `https://www.ibm.com/watson/services/natural-language-understanding/`. [Accessed 7-October-2019].

[Ihl] Alexander Ihler. Support Vector Machines (3): Kernels. `https://www.youtube.com/watch?v=OmTu0fqUsQk`. [Accessed 7-October-2019].

[JGBM16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[JM14] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, pages 137–142. Springer, 1998.

[Jol11] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[jQu] jQuery. `https://jquery.com`. [Accessed 7-October-2019].

[JWY+17] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent Dirichlet Allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, pages 1–43, 2017.

[Kan00] Eser Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proceedings of the IEEE Information Visualization Symposium*, volume 650, page 22. Citeseer, 2000.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[KK15] Kostiantyn Kucher and Andreas Kerren. Text visualization techniques: Taxonomy, visual survey, and community insights. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pages 117–121. IEEE, 2015.

[KKP+17] Minjeong Kim, Kyeongpil Kang, Deokgun Park, Jaegul Choo, and Niklas Elmqvist. Topiclens: Efficient multi-level visual topic exploration of large-scale document collections. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):151–160, 2017.

[Kru64] Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[KW52] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[LBS+16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[LCL+19] Shixia Liu, Changjian Chen, Yafeng Lu, Fangxin Ouyang, and Bin Wang. An interactive method to improve crowdsourced annotations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):235–245, 2019.

[LG94] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[LGH+17] Yafeng Lu, Rolando Garcia, Brett Hansen, Michael Gleicher, and Ross Maciejewski. The state-of-the-art in predictive visual analytics. In *Computer Graphics Forum*, volume 36, pages 539–562. Wiley Online Library, 2017.

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning - Volume 32*, pages 1188–1196, 2014.

[LS99] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[LWC+18] Shixia Liu, Xiting Wang, Christopher Collins, Wenwen Dou, Fangxin Ouyang, Mennatallah El-Assady, Liu Jiang, and Daniel Keim. Bridging text visualization and mining: A task-driven survey. *IEEE Transactions on Visualization and Computer Graphics*, 25(7):2482–2504, 2018.

[LXLZ15] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[MAP06] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive Bayes-which naive Bayes? In *CEAS 2006 - The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA*, volume 17, pages 28–69. Mountain View, CA, 2006.

[MC89] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

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

[Mey16] Robinson Meyer. How Many Stories Do Newspapers Publish Per Day? `https://www.theatlantic.com/technology/archive/2016/05/how-many-stories-do-newspapers-publish-per-day/483845/`, 2016. [Accessed 7-October-2019].

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[Mic] Microsoft, Project Entity Linking. `https://labs.cognitive.microsoft.com/en-us/project-entity-linking`. [Accessed 7-October-2019].

[MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[MP18] Marcin M. Mirończuk and Jarosław Protasiewicz. A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36–54, 2018.

[MRS10] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[MS05] Erkki Makinen and Harri Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica*, 29(3):357–364, 2005.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[Mur12] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[Nat] Natural Language Toolkit. `https://www.nltk.org/#natural-language-toolkit`. [Accessed 7-October-2019].

[New06] Mark E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[NJ02] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[NKNW96] John Neter, Michael H. Kutner, Christopher J. Nachtsheim, and William Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

[Nod] Node.js. `https://nodejs.org/en/`. [Accessed 7-October-2019].

[NS07] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[Num] NumPy: a fundamental package for scientific computing with Python. `https://www.numpy.org`. [Accessed 7-October-2019].

[NVH11] Quang Nguyen, Hamed Valizadegan, and Milos Hauskrecht. Learning classification with auxiliary probabilistic information. In *2011 IEEE 11th International Conference on Data Mining*, pages 477–486. IEEE, 2011.

[Ope] Thomson Reuters, Open Calais. `http://www.opencalais.com`. [Accessed 7-October-2019].

[Por80] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[PSPM15] Jose Gustavo Paiva, William Robson Schwartz, Helio Pedrini, and Rosane Minghim. An approach to supporting incremental visual data classification. *IEEE Transactions on Visualization and Computer Graphics*, 21(1):4–17, 2015.

[PVG+11] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Eduard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Pyt] Python Software Foundation. Python programming language. `https://www.python.org`. [Accessed 7-October-2019].

[RD89] Ritchey A. Ruff and Thomas G. Dietterich. What good are experiments? In *Proceedings of the sixth international workshop on Machine learning*, pages 109–112. Elsevier, 1989.

[Reu] Reuters-21578 Text Categorization Collection. `https://www.cs.umb.edu/~smimarog/textmining/datasets/`. [Accessed 7-October-2019].

[RG65] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[RKKF18] Patrick Riehmann, Dora Kiesel, Martin Kohlhaas, and Bernd Froehlich. Visualizing a thinker's life. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1803–1816, 2018.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[RSRDS16] Manuel Rubio-Sánchez, Laura Raya, Francisco Diaz, and Alberto Sanchez. A comparative study between radviz and star coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):619–628, 2016.

[Set09] Burr Settles. Active learning literature survey. Technical report, Department of Computer Sciences, University of Wisconsin-Madison, 2009.

[SG10] Christin Seifert and Michael Granitzer. User-based active learning. In *2010 IEEE International Conference on Data Mining Workshops*, pages 418–425. IEEE, 2010.

[SJ72] Karen Spärk Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[SLL+16] Xiaobing Sun, Xiangyue Liu, Bin Li, Yucong Duan, Hui Yang, and Jiajun Hu. Exploring topic models in software engineering data analysis: A survey. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 357–362. IEEE, 2016.

[SLMN11] Richard Socher, Cliff C. Lin, Chris Manning, and Andrew Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 129–136, 2011.

[Smi18] Martin Smiech. Configurable text exploration interface with NLP for decision support. Bachelor's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, April 2018.

[SOS92] Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[spa] spaCy, Industrial-strength Natural Language Processing in Python. `https://spacy.io`. [Accessed 7-October-2019].

[Spr] Vincent Spruyt. The Curse of Dimensionality in classification. `https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/`. [Accessed 7-October-2019].

[SPW+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[SZS+16] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A. Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, 2016.

[Tex] TextRazor, The Natural Language Processing API. `https://www.textrazor.com`. [Accessed 7-October-2019].

[The] The New York Times. `https://www.nytimes.com`. [Accessed 7-October-2019].

[TK07] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[Typ] TypeScript, JavaScript that scales. `https://www.typescriptlang.org`. [Accessed 7-October-2019].

[VA13] Vincent Van Asch. Macro-and micro-averaged evaluation measures. *Belgium: CLiPS*, pages 1–27, 2013.

[WBD+18] John Wenskovitch, Lauren Bradel, Michelle Dowling, Leanna House, and Chris North. The effect of semantic interaction on foraging in text analysis. *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–24, 2018.

[WM12] Sida Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[XC16] Yijun Xiao and Kyunghyun Cho. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*, 2016.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

[ZL15] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[ZLLG18] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.

[ZLR16] Rui Zhang, Honglak Lee, and Dragomir Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*, 2016.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657, 2015.