

---

# Collaborative Web-Based Short Text Annotation with Online Label Suggestion

---

**Kollaborative webbasierte Annotation von kurzen Texten mit Online-Labelvorschlägen**  
Master-Thesis von Kai Steinert aus Frankfurt am Main  
März 2017

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Collaborative Web-Based Short Text Annotation with Online Label Suggestion  
Kollaborative webbasierte Annotation von kurzen Texten mit Online-Labelvorschlägen

Vorgelegte Master-Thesis von Kai Steinert aus Frankfurt am Main

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Eugen Ruppert, MSc.

Tag der Einreichung:

---

# Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 3. März 2017

---

(Kai Steinert)

## Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, March 3, 2017

---

(Kai Steinert)

---

---

# Abstract

Social media has an increasing influence in our daily interaction and political opinion formation. Social media creates vast amounts of textual information that is desired to be processed automatically. Machine learning algorithms that are capable of automatically processing natural language require labeled training data. Creating labeled training data is commonly done by human annotators. To support the annotator during an annotation project a collaborative web-based annotation tool is developed. Suggesting a label to an annotator decreases the time needed for annotation, is the hypothesis that is investigated. Literature of short text classification is reviewed, online learning is motivated and the Naïve Bayes classifier described. Then, active learning is described and examined as enhancement for online learning. The developed application is presented. The hypothesis is evaluated based on data gathered from an annotation project. The results show an decrease in annotation time if labels are suggested.

# Zusammenfassung

Soziale Medien haben einen immer größeren Einfluss auf unsere alltägliche Interaktion und die politische Meinungsbildung. Es ist wünschenswert die großen Mengen an Text automatisch zu verarbeiten. Algorithmen des Maschinellen Lernens sind in der Lage automatisiert geschriebene Sprache zu verarbeiten. Damit diese Algorithmen funktionieren, werden Trainingsbeispiele benötigt. Trainingsbeispielen werden in der Regel von menschlichen Annotatoren erstellt. Um die Annotatoren bei ihrer Arbeit zu unterstützen, wird ein kollaboratives webbasiertes System entwickelt. Die Hypothese der Arbeit lautet, dass die Annotationszeit verringert wird, wenn dem Annotator ein automatisch generiertes Label vorgeschlagen wird. Dazu wird die Literatur zur Klassifizierung von kurzen Texten beschrieben, Online-Lernen motiviert und der Naïve Bayes Klassifizierer beschrieben. Aktives Lernen wird als Erweiterung von Online-Lernen vorgeschlagen. Danach wird das entwickelte System vorgestellt. Ein Annotationsprojektes wird durchgeführten und mit den gewonnen Daten wird die Hypothese evaluiert. Die Ergebnisse zeigen, dass das vorgeschlagen von Labeln zu einer Reduktion der Annotationszeit führen.

---

# Contents

<b>1. Introduction</b>	<b>2</b>
1.1. Motivation . . . . .	2
1.2. Hypothesis . . . . .	3
1.3. Overview . . . . .	3
<b>2. Annotation of Text Documents</b>	<b>4</b>
<b>3. Classifying Short Text Documents in an Online Learning Setting</b>	<b>6</b>
3.1. Short Text Classification . . . . .	6
3.2. Domain Adaptation and Online Learning . . . . .	8
3.3. Naïve Bayes Classifier . . . . .	9
<b>4. Document Selection with Active Learning</b>	<b>10</b>
4.1. Active Learning Scenarios . . . . .	10
4.1.1. Membership Query Synthesis . . . . .	10
4.1.2. Stream-Based Selective Sampling . . . . .	11
4.1.3. Pool-Based Sampling . . . . .	11
4.2. Querying Documents with Uncertainty Sampling . . . . .	12
<b>5. Assisted Annotation with a Collaborative Web-Based Tool</b>	<b>13</b>
5.1. Installation and Project Setup in <i>Annotate!</i> . . . . .	13
5.2. Application Screens . . . . .	17
5.3. Database Model . . . . .	20
5.4. Monitoring an Annotation Project . . . . .	21
5.5. Exporting Annotations . . . . .	22
<b>6. Evaluation</b>	<b>23</b>
6.1. Project Description and Results . . . . .	23
6.2. Annotation Quality and Annotator Agreement . . . . .	25
6.2.1. Cohen’s Kappa . . . . .	26
6.2.2. Fleiss’ Kappa . . . . .	27
6.3. Determining Final Labels with Majority Vote . . . . .	28
6.4. Evaluating the Advantage of Suggesting Labels . . . . .	29
6.4.1. Temporal Difference in Annotation Time When Suggesting Labels . . . . .	30
6.4.2. System Overhead of Suggesting Labels . . . . .	32
6.4.3. Classifier Performance . . . . .	33
<b>7. Conclusion</b>	<b>35</b>
7.1. Future Work . . . . .	36
<b>Bibliography</b>	<b>37</b>

---

<b>A. Annotate! Documentation</b>	<b>41</b>
A.1. Configuration	41
A.1.1. MySQL Configuration	41
A.2. Database Model	41
A.2.1. Annotation	41
A.2.2. AnnotationQueue	42
A.2.3. Document	42
A.2.4. Label	43
A.2.5. NBC_class_count	43
A.2.6. NBC_vocabulary	43
A.2.7. NBC_word_count_given_class	43
A.2.8. QueueElement	43
A.2.9. Entity Relationship Model	44
A.3. Commandline Tools	45
A.3.1. addDocument	45
A.3.2. addLabels	45
A.3.3. annotationQueueLoop	45
A.3.4. batchTrain	46
A.3.5. createAnnotationQueue	46
A.3.6. createCurve	47
A.3.7. createGroup	47
A.3.8. createUser	47
A.3.9. evaluate	47
A.3.10. exportAnnotation	48
A.3.11. predictLabel	48
A.3.12. status	48
A.3.13. wipeDB	48

---

## List of Figures

2.1. Annotation Project Workflow . . . . .	5
4.1. Pool-Based Active Learning Cycle . . . . .	11
5.1. Login Screen . . . . .	17
5.2. Annotation Screen . . . . .	18
5.3. Training Screen . . . . .	19
5.4. Admin Screen . . . . .	20
5.5. Entity Relationship Model . . . . .	20
5.6. Status Command Results . . . . .	21
6.1. DB Annotation Project Overview . . . . .	24
6.2. Agreement Overview . . . . .	26
6.3. Final Label Distribution . . . . .	29
6.4. Estimating the Distribution of Durations . . . . .	29
6.5. Beta-prime distributions fitted with MLE. . . . .	31
6.6. System Time vs. Annotation Duration . . . . .	32
6.7. Prediction Results . . . . .	33
6.8. Learning Curve of Random Order vs. Minimal Margin Sampling: . . . . .	34
A.1. Entity Relationship Model Extended . . . . .	44

## List of Tables

6.1. Inter Annotator Agreement . . . . .	27
6.2. Strength of Agreement for Different Kappa Ranges . . . . .	27
6.4. Comparison of Bayesian Information Criterion Results . . . . .	31

## List of Codes

1. Apache Configuration . . . . .	14
2. my.cnf . . . . .	15
3. Django Configuration . . . . .	15
4. JSON Export Format . . . . .	22
5. MySQL Configuration . . . . .	41

---

# Glossary

---

## Abbreviations

---

API application programming interface

BIC Bayesian information criterion

BOW bag-of-words

BRAT brat rapid annotation tool

DAL Dictionary of Affect in Language

DB Deutsche Bahn AG

EM Expectation Maximization

ERM Entity Relationship Model

GUI graphical user interface

IAA inter annotator agreement

KDE kernel density estimation

MaxEnt maximum entropy

MLE maximum likelihood estimation

NLP Natural Language Processing

NLTK Natural Language Toolkit

ORM Object Relational Mapper

POS Part-Of-Speech

SCL structural correspondence learning

SSH Secure Shell

SVM support vector machine

tf-idf term frequency-inverse document frequency

WSGI Web Server Gateway Interface

---



---

# 1 Introduction

Natural Language Processing (NLP) is a field of study concerned with the interpretation and understanding of human language. This includes processing speech, generating speech and processing text. In recent years NLP predominantly uses statistical machine learning. Machine learning is a set of algorithms that automatically discover patterns in data. These patterns are used to make predictions about new data. The goal is to make good predictions even though the new data possibly was never seen before. For the course of this thesis the data we are interested in is natural language in the form of text created by a human author.

---

## 1.1 Motivation

---

In the last decade, social media and customer review web sites had a huge influence on people's interaction on the internet. People create content and share it with potentially large groups at an enormous rate. The content has a vast variety, covers all of life's subjects and carries information about preferences, attitudes and opinions. This information in turn is of interest to companies, politicians, and scientists. Companies are interested in customer satisfaction. Politicians are interested in people's opinions about their or the opposition's agenda. Scientists are interested in all kinds of topics.

Social media sites such as Twitter<sup>1</sup> and Reddit<sup>2</sup> or customer review sites such as Yelp<sup>3</sup> and TripAdvisor<sup>4</sup> offer the possibility to gather large quantities of texts that are related to some topic. Because the amount of text is too big to be processed by humans entirely computer programs are needed instead. But understanding and interpretation of human language is a challenging task for a computer program because of several factors. Human language is often ambiguous, depends on context and can be fuzzy. To cope with these obstacles a program has to handle patterns in natural language. This pattern handling is done by machine learning algorithms. But in order for these algorithms to work one needs to have access to training data. The algorithm learns the patterns from the training data. In this context, training data are text documents together with their corresponding labels. A label is the output of the algorithm that we are trying to deduce. Examples for labels are the overall sentiment of a document, its relevance or the objects being discussed in the document.

To utilize a text as a training example, a human has to manually annotate the text with a label. This process is tedious, expensive and time consuming. The motivation for this thesis is to support the annotator during the annotation process. Annotators need access to a software program that allows them to review a text section and stores their annotation. In the past software was specifically developed for a pending annotation project. Yet, many tasks are redundant across projects. This draws interest to more general annotation tools. Biemann et al. (2017) give a detailed motivation for a general-purpose collaborative web-based annotation tool:

*Flexibility:* Configurable options allow more flexibility, instead of build-in annotation schemes and limited label sets.

---

<sup>1</sup> <https://twitter.com>

<sup>2</sup> <https://www.reddit.com>

<sup>3</sup> <https://www.yelp.com>

<sup>4</sup> <https://www.tripadvisor.com>

- 
- Maintainability:* Every client uses the same and most recent software version. The application does not have to be installed on a client machine.
- Reusability:* The infrastructure needed to conduct an annotation project is the same across many projects. This includes annotator management, text management, and result evaluation. This functionality is reused instead of having it re-implemented.
- Usability:* Users who are familiar with web applications do not have to be trained specifically if a common interface is implemented. Given an internet connection, users can participate from different locations and with different devices.
- Scalability:* With a web-based architecture it is easy to distribute annotation tasks between annotators and retrieve their collective effort without risking data integrity.
- Open source:* Freely available source code adds value to the community of people who want to conduct annotation projects. Additionally development of new features and reviewing is done by more than just the initial developers.

---

## 1.2 Hypothesis

---

The strategy we choose to examine to support the annotator is to suggest a label during annotation. The hypothesis is as follows:

**Suggesting possibly correct labels to an annotator decreases the average time needed to submit labels compared to the case when no labels are suggested while annotating short text documents from social media websites.**

To test this hypothesis, a collaborative web-based annotation system is developed. The system is used in a real annotation project. To evaluate the hypothesis, meta data such as the time it took an annotator to submit an annotation is recorded. Additional factors such as the temporal performance of the system as well as the performance of the algorithm that suggests the labels are considered. A suggestion is carried out in the form of pre-selecting a label in the system's user interface. The user can then directly submit the pre-selected label or correct it if needed.

---

## 1.3 Overview

---

This thesis is structured as follows. The next chapter gives an introduction to annotation systems with label suggestion and an overview of the most prominent systems. In Chapter 3 the challenges of short text classification are described. The literature is reviewed for existing approaches. Then the domain adaption problem is described and online learning is suggested as circumvention. The last section of Chapter 3 describes the Naïve Bayes classifier, which is used to suggest labels to the annotator. Active learning is described in Chapter 4. When using online learning for every annotation project it is reasonable to combine it with active learning to improve the classification model as fast as possible. Chapter 5 describes the application that is built for this thesis. The different features, screens and database internals are elucidated. The developed application was used in a real annotation project. Chapter 6 describes the project and the consequential evaluation. Finally, the conclusion and an outlook is given in Chapter 7.

---

## 2 Annotation of Text Documents

In general machine learning is divided into two. The first type is *unsupervised learning* where only the inputs for an algorithm are given and the objective is to find interesting structures e.g. find clusters of data. The second type is *supervised learning*. Here, not only the inputs but also the outputs are given. The goal is to learn a mapping from the inputs to the outputs.

Supervised learning itself is divided into *regression* and *classification*. In regression, the outputs are continuously valued, while in classification the outputs are discrete. There is additional differentiation for classification. If every input is assigned only one class then it is called *multi-class classification*. Whereas, if every input is assigned one or more classes then it is called *multi-label classification*.

A common problem in NLP is classification of text documents. That is assigning a label  $l \in \mathcal{L}$  to a document  $d \in \mathcal{D}$ . Here  $\mathcal{D}$  is the set of all documents and  $\mathcal{L}$  is the set of all possible labels. Examples of labels are the category a document belongs to or the sentiment expressed in a document.

The premise for any supervised approach in NLP is access to training data. The set of pairs  $\mathcal{C} = \{(d_i, l_i)\}_{i=1}^N$  is called a *training set* and  $N$  is the number of training examples. A training set is created by human annotators. An annotator is either a layperson, someone who has been trained to do a particular annotation task or a professional providing her expertise.

The web can be utilized as a source to find agglomerations of documents. Sites such as Amazon<sup>1</sup>, Yelp or TripAdvisor offer their users to write reviews about products and services. But not only review sites are suitable for acquisition. On sites such as Facebook<sup>2</sup>, Twitter or Reddit, users are eager to express their opinions and participate in discussions. Either sites are scraped with a web crawler or documents are accessed through an application programming interface (API). After documents are gathered, typically they are transformed into a common format for ease of further processing. The workflow of an annotation project is described in Figure 2.1.

In the past, this process was carried out with software specifically developed for a pending annotation task. Functionality such as the graphical user interface or the data export is often similar across different projects. Therefore, systems were developed that provide functionality that is common in annotation projects. In the following, we survey the most prominent examples.

Bontcheva et al. (2013) present GATE Teamware a an open-source, collaborative, web-based, text annotation framework. GATE Teamware provides user management with multi-role support and task distribution. Corpora can be stored in a network transparently. An automatic pre-annotation service is included. This service trains a machine learning model with newly obtained annotations. A new document is then partially pre-annotated. The annotator only corrects the pre-annotation. This service is similar to the approach presented in this thesis.

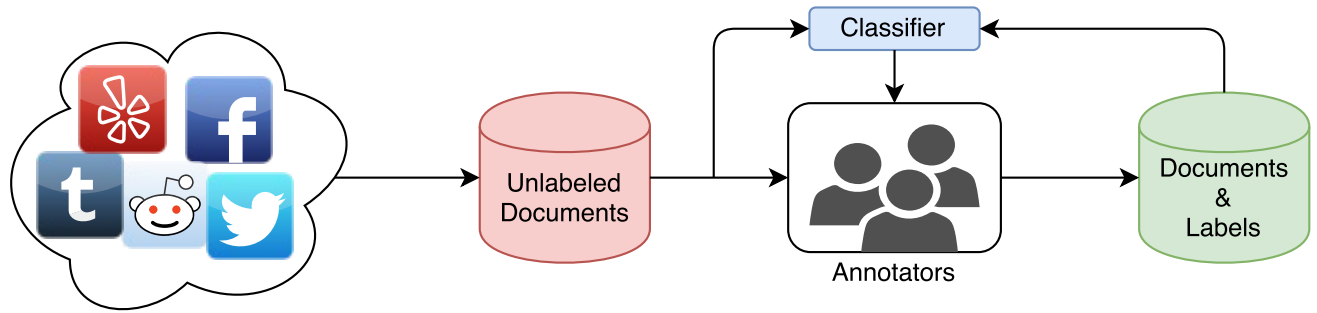
Stenetorp et al. (2012) created the brat rapid annotation tool (BRAT) a web-based text annotation system. BRAT supports diverse annotation schemes and offers automatic annotation support. They conducted an annotation project for semantic class disambiguation. Their results show a reduction of 15.4% in total annotation time for their 'rapid mode'.

Yimam et al. (2013) created a general purpose web-based annotation tool for a variety of tasks, such as part-of-speech, named entity, dependency parsing and co-reference chain annotations. It also supports

---

<sup>1</sup> <https://www.amazon.com/>

<sup>2</sup> <https://www.facebook.com/>



**Figure 2.1.:** Annotation Project Workflow: The cloud on the left side represents the Internet with its different social media and review websites. These websites serve as a source for documents we want to gain information from and therefore annotate. Usually documents from different sources are transformed into a common format for further processing. This is represented by the red cylinder. Then unlabeled documents have to be distributed between annotators and their selection of one of the predefined labels has to be recorded. Finally, a strategy such as majority voting is used to determine the definitive labels. The result is a labeled training set on the right. In order to test our hypothesis and suggest labels to the annotators a classifier is used. The classifier is trained with the training data that was already created.

annotation project management, freely configurable tagsets and user management. Web-Anno offers an extensible system architecture organized in users, front-end, back-end and data storage. Yimam et al. (2014) wrote an extension to WebAnno with automatic annotation suggestion. Suggestions are supplied for text sequences. For their case study they report an increase of 21% in annotation speed.

An alternative to using stand-alone annotation systems is to use a crowdsourcing platform. Instead of employing annotators directly, laypeople create annotations through the crowdsourcing platform. This was done by Biemann (2013) who used Amazon Mechanical Turk to create a word sense inventory. A difference to employing expert annotators oneself is that an annotation task has to be designed with larger overlap between annotations on the same item. This is necessary because there is less control over the annotators. The benefit of crowdsourcing is a reduced cost and scalability, while the quality of annotation was found to be comparable (Snow et al. 2008).

---

## 3 Classifying Short Text Documents in an Online Learning Setting

To test our hypothesis and suggest a label to an annotator a text classifier is needed. A text classifier is an algorithm that takes a text section as input and returns one label in case of multi-class classification and multiple labels in case of multi-label classification. Document classification as the name suggests is concerned with predicting a label for an entire document. On the contrary is classification that operates on a sub-document level e.g. part-of-speech tagging (Toutanova et al. 2003) or named entity recognition (Nadeau & Sekine 2007). Most classifiers can only process numeric values but not character strings or symbols. To be able to input documents into these classifiers the text has to be transformed into feature space. A data point in feature space is the numeric representation of a document. One of the simplest representations is the bag-of-words (BOW) model. BOW represents each document as the frequency of words it contains.

This chapter examines the characteristics of short text documents and gives an overview of the literature. Online learning, which is a form of training a classifier, is discussed. Finally, the Naïve Bayes classifier is described.

---

### 3.1 Short Text Classification

---

Various classifiers perform differently depending on the type of text that is passed to them as input. The motivation states social media sites as primary source of documents that we intend to use. Characteristic for tweets and comments is their short length, which ranges from a few words to a couple of sentences compared to other text documents commonly found on the internet such as news or blog articles. These short texts impose special challenges on the classifier. Their short length introduces sparsity in the input domain as they provide only few word occurrences. Furthermore, they are noisier because their authors often disregard spelling or grammar rules. This makes it harder to measure similarity between tweets and comments and therefore harder to classify them. Also, text artifacts might be present, for example links, smileys or special character strings such as hash tags. However, it is inconvenient to invent completely new algorithms because of these properties. Thus, the commonly used approach is to create and test different feature sets. In the following we survey over the literature regarding short texts obtained from social media and their different feature sets.

A common methodology in order to handle the problem of sparsity is to use additional sources to enrich short text documents. Banerjee et al. (2007) used Wikipedia<sup>1</sup> to cluster news feed headlines. Based on their findings Phan et al. (2008) propose a general framework for classification. This framework is based on the idea of using a large-scale external data collection to extract a topic model. They use latent Dirichlet allocation (Blei et al. 2003) to build the topic model. Then, the topic model is used to infer topics from the text snippets they want to classify. The inferred topics serve as the enrichment features. This approach leverages the information contained in large text collections that are easily accessible. Phan et al. (2008) use Wikipedia and Ohsumed/MEDLINE<sup>2</sup> as external sources and note them as being appropriate because of the bright range of domains covered. To train a classifier a much smaller labeled

---

<sup>1</sup> <https://en.wikipedia.org>

<sup>2</sup> Ohsumed: <ftp://medir.ohsu.edu/pub>

---

training set, compared to the text collection, of short texts is needed. The documents in the training set are represented as the concatenation of their BOW model and their topics. With this representation they train a maximum entropy (MaxEnt) classifier (Berger et al. 1996). To classify unseen text its topics have to be inferred and together with its BOW representation inputted into the classifier. The framework is evaluated with two tasks. The first is domain disambiguation of web snippets and the second is disease classification of medical abstracts. They find significant improvement over not using topics as enhancement.

Chen et al. (2011) extended the approach from Phan et al. (2008) with topics of multi-granularity. They propose an algorithm to select the best subset from the set of all generated topics. Their selection of a topic subset is based on two qualities. The first is the ability of topics to discriminate short text into different classes. The second is the distance between a subset and another subset with different granularity. To measure the distance between subsets they use the Kullback-Leibler (KL) divergence.

Sriram et al. (2010) recognize the problem of users being overwhelmed with the amount of information they are exposed to on Twitter. Their proposed solution is to classify tweets into one of the categories {News, Events, Opinions, Deals}. They define the author, presence of shortening of words and slangs, time-event phrases, opinioned words, emphasis on words, currency and percentage signs, “@username” at the beginning of the tweet, and “@username” within the tweet as features. These features address the problem of sparsity with utilizing available meta information such as the author’s identifier and special directives such as “@username”. They compare their feature set to BOW on a hand labeled data set of 5407 tweets from 684 randomly selected authors. The Naïve Bayes classifier and 5-fold cross-validation is used for the evaluation. In their experiments, their feature set outperforms BOW by 32.1% accuracy.

Asur & Huberman (2010) predict the box-office revenue in the opening week of movies in advance of their release based on Twitter messages. They investigate the frequency of promotional material in the form of URLs and retweets (sharing already published tweets with others). Even though some correlation with revenue is found, its predictive quality is marginal. They define the *tweet-rate* as the number of tweets in which a particular movie is mentioned per hour. The tweet-rate has a strong positive correlation with the box-office gross. The tweet-rate time series is utilized as a feature and used together with linear least squares regression to acquire a coefficient of determination value of 0.973. Also they investigate the importance of sentiment in tweets to predict the movie revenue. Sentiment analysis is a well-known problem in NLP (Pang & Lee 2008). Generally speaking, sentiment analysis is a supervised learning problem in which given a document the task is to assign commonly one of the labels  $\mathcal{L} = \{\text{Positive, Neutral, Negative}\}$  to the document based on the document’s sentiment. Asur & Huberman (2010) use Amazon Mechanical Turk to obtain a labeled training set for sentiment analysis of their tweets. They train a classifier with their acquired data and achieve an accuracy of 98% with cross-validation. Then, this classifier is used to enhance the prediction of movie revenues. Even though they find that it improves the performance they note, that the tweet-rate is the more important feature.

Agarwal et al. (2011) analyze the sentiment of Twitter data. The proposed feature set is a combination of Part-Of-Speech (POS) tags and prior polarity of words. To determine the prior polarity of a word they use the Dictionary of Affect in Language (DAL) (Whissell & Charuk 1985) and extend it with WordNet (Fellbaum 1998). The used approach is described by Agarwal et al. (2009). DAL contains 8742 words and for each word three values that represent pleasantness, activeness and imagery of the word on a scale of 1 to 3, low to high. WordNet is a database that contains lexical and semantic relationships between words including sets of synonyms. The prior polarity of a word is computed as follows. For every word in a short text the normalized pleasantness score is looked up. If the word is not found in DAL its set of synonyms is looked in WordNet. Every synonym is searched in DAL and the first match’s normalized pleasantness score is used. If also no synonym is found no score is assigned. With this method for 88.9% of English words prior polarity scores are found. For their experiments they use a support vector machine (SVM) and 5-fold cross-validation on a hand-labeled data set of 5127 tweets with equally many texts for

---

each class {Positive, Neutral, Negative}. A SVM (Cortes & Vapnik 1995) is a binary classification algorithm that tries to maximize the margin between the training examples and the separating hyperplane. An emoticon dictionary and an acronym dictionary are created and used to pre-process the training set. They represent a document as a tree by combining different features in the tree representation. To use a tree with a SVM the partial tree kernel is used (Moschitti 2006). Their model is compared to the BOW model. They report an overall gain of over 4%.

This section surveyed over the literature with an emphasize on the different features used to handle the challenges imposed by short texts. The next section discusses the problem of domain adaptation and online learning possible solution.

---

## 3.2 Domain Adaptation and Online Learning

---

The previous section introduced the specific challenges with classifying short texts. To handle sparseness and noise different feature sets from the literature are introduced. This section examines the problem of domain adaptation and online learning as possible solution in the particular context of an annotation system with generated suggestions.

The implicit precondition for classification is that training data, test data and future data is drawn from the same distribution. A classifier trained with data from a particular domain but evaluated or used with data of another domain generally performs worse than if trained and used in the same domain. This problem is known as *domain adaptation* and is a special case of *transfer learning* (Pan & Yang 2010). Transfer learning is concerned with any form of learning across different domains, while in domain adaptation labeled training data is only available in the source domain but not in the target domain. An example in the context of NLP is to train a classifier to discriminate sentiment of customer reviews of different product types (Blitzer et al. 2007).

Domain adaptation is of relevance because annotation projects are carried out in many different domains. A naive solution to this problem is to train a new classifier for every domain. But this approach generally reaches its limits fast because of the lack of labeled training data for novel domains. Thus, there were more elaborated approaches proposed in the literature.

Daumé & Marcu (2006) formalize the problem of domain adaptation and recognize labeled in-domain data is scarce but labeled out-of-domain data is more easily available. Based on this observation they propose a maximum entropy model with conditional expectation maximization. Blitzer et al. (2006) propose structural correspondence learning (SCL). They differentiate between *source* domain and *target* domain. The idea is to use unlabeled data from both the source and target domain to find correspondence features that perform well in both domains. Another approach is instance weighting (Jiang & Zhai 2007). The key finding is that identifying representative instances from the target domain is much more valuable than excluding misleading instances from the source domain.

Even though training a classifier in advance of an annotation project and using one of the different methodologies is an option, we choose a different approach. By creating an annotation system access to in-domain labeled data is granted. Thus, to circumvent domain adaptation we train a classifier during each annotation project. One can either periodically retrain the classifier with the currently available data or train continuously with every newly obtained labeled instance. Training a classifier from a sequence of examples rather than from a batch is referred to as online learning. This approach is beneficial from a usability point of view. The previous methods require to retrain the classifier for the target domain upfront. This is inconvenient for users little or no programming experience. With online learning this process is fully automated during the annotation project without additional effort for the user. There is an extended literature with in depths theoretical considerations. For an overview see Shalev-Shwartz

(2012). In the context of NLP, online learning is used in a variety of tasks, e.g. categorization of news articles (Ng et al. 1997) or sentiment analysis of product reviews (Cui et al. 2006).

---

### 3.3 Naïve Bayes Classifier

---

The last section motivated the use of online training based on the problem of domain adaptation. In this section the multinomial Naïve Bayes classifier is introduced.

The *multinomial Naïve Bayes classifier* is a generative probabilistic model that is based on the assumption that each of the features are conditionally independent given some label (Manning et al. 2008). This assumption is a simplification. In reality the words in a documents are not independent of each other. But this simplification enables us to compute the probability of a label  $l$  to belong to document  $d$  as:

$$P(l|d) \propto P(l) \prod_{i=1}^n P(t_i|l), \quad (3.1)$$

where  $P(l)$  is the prior probability of the label,  $t_1, t_2, \dots, t_n$  are the terms of the document and  $P(t_i|l)$  is the conditional probability of a term given the label. The distributions  $P(l)$  and  $P(t_i|l)$  are unknown. To estimate them a training set is used and the empirical distributions  $\hat{P}(l)$  and  $\hat{P}(t_i|l)$ . They are estimated with

$$\hat{P}(l) = \frac{N_l}{N} \quad \text{and} \quad \hat{P}(t|l) = \frac{T_{lt} + 1}{\sum_{t \in V} (T_{lt'} + 1)}, \quad (3.2)$$

where  $N$  is the number of training examples,  $N_l$  is the number of times labels  $l$  is present in the training set and  $T_{lt}$  is the number of times  $t$  occurs in the training document with label  $l$ . The Naïve Bayes classifier selects the label that is *maximum a posteriori*(MAP).

$$\hat{l} = \operatorname{argmax}_{l \in \mathcal{L}} \hat{P}(l|d) = \operatorname{argmax}_{l \in \mathcal{L}} \hat{P}(l) \prod_{i=1}^N \hat{P}(t_i|l) \quad (3.3)$$

Multiplying many fractions can exceed machine precision of floating point numbers. For numerical stability the natural logarithm is applied. Applying the logarithm does not change the prediction. Thus,

$$\hat{l} = \operatorname{argmax}_{l \in \mathcal{L}} \left[ \ln \hat{P}(l) \sum_{i=1}^N \ln \hat{P}(t_i|l) \right] \quad (3.4)$$



---

## 4 Document Selection with Active Learning

The previous chapter motivated the use of online learning by stating the domain adaption problem. Online learning algorithms enable continues training during the annotation project, instead of periodical retraining. When learning a new model for every new project the desire to increase the classifier's performance as fast as possible is natural. To implement this desire it is important to note that not all documents are equally relevant to the classifier in terms of information they provide. The question arises which documents teach the classifier the most. The field of active learning is concerned with methods that try to answer this question.

As a prerequisite it is assumed that there is some kind of oracle that is capable of providing the correct label to an unlabeled instance if asked. In the context of document annotation the role of the oracle is taken by the annotators. Here, an instance or query is a document that is selected based on some strategy to be labeled by the annotators. The newly labeled document is used to train the classifier or learner as it is typically called in the literature We intend to use online learning, but this is not mandatory. Instead, the classifier might be batch trained with all the available training data. Then, the updated classifier is used to make predictions about the remaining unlabeled instances in the selection strategy. Thus, closing a cycle as a new unlabeled document is selected to be annotated. A visualization of this cycle for pool-based active learning is depicted in Figure 4.1. For a comprehensive overview of active learning see Settles (2012). For a literature survey that is more related to the context of NLP see Olsson (2009). A more diverse view with a connection to other methods is given in Aggarwal et al. (2014).

---

### 4.1 Active Learning Scenarios

---

According to Settles (2009) there are three distinct scenarios which are described in the following.

---

#### 4.1.1 Membership Query Synthesis

---

The first scenario to be examined are membership queries. The idea is that an instance or query is not chosen from an empirical distribution but rather created such that the learner is uncertain about it. Then, the oracle is consulted to decide whether or not the created query is a member of a possible class or not. Angluin (1988) gives an example for the game of poker. Creating queries in this context comes natural. A query is a collection of five cards and their class membership is easily decided by someone familiar with the rules of the game. But Settles (2009) points out based on Lang & Baum (1992) findings regarding handwritten character recognition, that membership query synthesis might not be suitable for NLP tasks. The problem is that a human annotator might not be able to assign a sensible label to a computer generated document. Consider the BOW model which represents a document as its word frequencies. The order of words is not captured in this model and the machine does not distinguish between two documents with the same words but eventually very different meaning. It is clear that the word order is very important for a human to understand a document but a generated document might not comply this need. The two remaining scenarios are concerned with queries drawn from an empirical distribution and presented in the following.

---

## 4.1.2 Stream-Based Selective Sampling

---

In this scenario unlabeled instances are obtained sequentially from a data source. It is assumed they are drawn one at a time. The learner then decides whether or not to request a label from the oracle. The preceding assumption is that instances from the stream are obtained without significant cost, but acquiring a label is expensive (Atlas et al. 1990). While in the previous scenario a learner created an instance according to its needs, in the stream-based data acquisition a learner has to wait until an instance is presented that it is sufficiently uncertain about (Cohn et al. 1994). Stream-based selective sampling is adequate to use if the source of instances is very large and continuously changing.

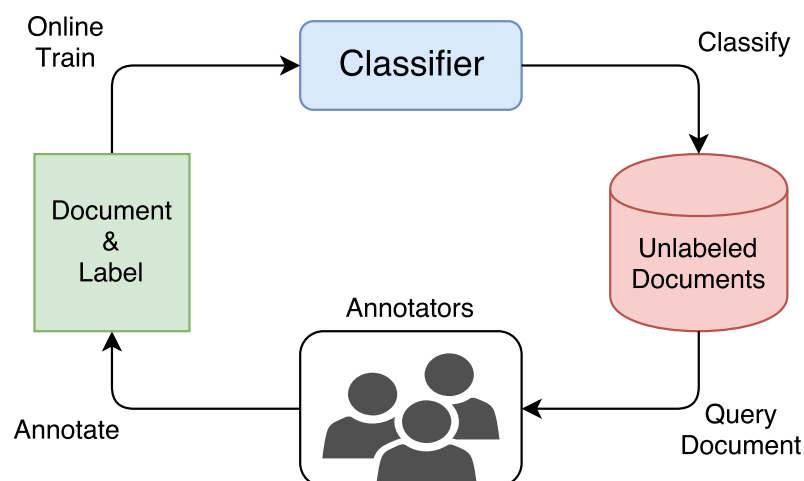
Smailović et al. (2014) used stream-based selective sampling for sentiment classification of Twitter messages to predict the movement of stock prices in advance. They faced the problem of not having a publicly available hand-labeled data set of Twitter data for sentiment analysis in the financial domain. Because of that a general purpose sentiment classifier is trained on data provided by (Go et al. 2009). Then, tweets are recorded over a period of nine month from March 11 to December 9, 2011. As tweets stream in they are filtered according to company names mentioned. Batches of 100 tweets are sentiment classified and respectively 5 positive, neutral and negative tweets are selected to label their financial relevance. They collected a total of 152,570 tweets from eight different companies and labeled 11,389 tweets discussing relevant financial information about the Chinese company Baidu.

Even though during an annotation project labels are created sequentially, stream-based active learning is not appropriate. Usually, the unlabeled documents are known upfront and also their entirety is supposed to be labeled not a selected fraction. This leads to the third form of active learning *pool-based sampling*.

---

## 4.1.3 Pool-Based Sampling

---



**Figure 4.1.:** Pool-Based Active Learning Cycle based on Settles (2009) Figure: 1: This cycle must be initialized with a randomly selected document if no classifier model exists. A document is selected from the pool of unlabeled documents with an active learning strategy and presented to an annotator. The annotators chooses a label. The label and the corresponding document are used to train the classifier. The updated classifier and active learning method are used to find the next unlabeled document that teaches the classifier the most.

---

In pool-based sampling an instance is selected from a collection of documents. A prerequisite is either a small number of labeled documents or a premature classification model. The procedure of pool-based sampling is depicted in Figure 4.1.

Pool-based sampling is used in multiple text classification tasks. Lewis & Gale (1994) used it to classify AP newswire texts. McCallum & Nigam (1998b) combine active learning with Expectation Maximization (EM). Tong & Koller (2002) present an algorithm for performing pool-based sampling with a SVM to classify news documents.

Lewis & Catlett (1994) show that it is not inevitable to use the same classification algorithm for document selection and for training. They use a Naïve Bayes model to select documents to train the rule induction algorithm C4.5 (Quinlan 1993).

The next section examines uncertainty sampling as a possible selection method.

---

## 4.2 Querying Documents with Uncertainty Sampling

---

Throughout this chapter active learning was outlined without elaborating on the document selection method. This section describes minimal margin sampling a method to select a document based on the uncertainty of the learner. The goal is to find the document  $\hat{d}$  for which the learner is the most uncertain about.

Lewis & Gale (1994) used a Naïve Bayes model to classify AP newswire texts. In each selection phase all remaining documents are classified with the current model. Documents for which the probability is closest to 0.5 are selected.

An extension is proposed by Scheffer et al. (2001).

$$\hat{d} = \operatorname{argmin}_d P(\hat{l}_1|d) - P(\hat{l}_2|d) \quad (4.1)$$

This approach is termed minimal margin sampling. It selects the document for which the uncertainty between the first and second most probable labels is the greatest.

Baldrige & Palmer (2009) investigated the task of language documentation, more specifically morpheme glossing. For this task two annotators with different expertise levels are employed. They find that the expert annotator performs best with uncertainty sampling but suggesting labels has only a marginal advantage. The novice annotator performs best with random selection of documents and with label suggestions.

The presuming assumptions from the introduction of the chapter may be too strict in a real world scenario. Donmez & Carbonell (2010) propose proactive learning, a generalization of active learning, where prerequisites are relaxed. In proactive learning there is more than one oracle, that do not always return an answer, that do not always answer correctly and charge different costs.

---

## 5 Assisted Annotation with a Collaborative Web-Based Tool

The concepts presented in the previous chapters are implemented in the collaborative web-based prototype *Annotate!*. The application is written in Python 2.7<sup>1</sup> and developed with Django 1.10<sup>2</sup>. Django is an open source high-level web framework that prescribes a client-server architecture.

The application was developed and tested on Ubuntu 16.04.1 LTS<sup>3</sup>. Enables freely configurable label-sets.

Most of the admin functionality is currently only available with physical or via Secure Shell (SSH) access to the host machine. But the graphical user interface (GUI) is easily extensible to support more functionality in the future.

---

### 5.1 Installation and Project Setup in *Annotate!*

---

The source code for the prototype is obtainable from GitHub *Annotate!*<sup>4</sup>.

---

#### Prerequisites

---

A database system has to present for *Annotate!* to store data in it. A list of supported databases<sup>5</sup> can be obtained from the Django documentation. *Annotate!* was developed and tested with MySQL 5.7.16<sup>6</sup> and deployed on Apache 2.4.18<sup>7</sup>. Moreover, the Natural Language Toolkit (NLTK)<sup>8</sup> has to be installed for full functionality. Version 3.2.1 is tested.

---

#### MySQL Configuration

---

For *Annotate!* to be able to use MySQL a database API driver is needed. In particular `mysqlclient` 1.3.9<sup>9</sup> is used to develop and test the prototype. In MySQL a database with the name 'annotationApp' has to be created. This is done with calling the command `create database annotationApp`; within the commandline tool of MySQL. For a more detailed description on the configuration of MySQL see A.1.1.

---

<sup>1</sup> <https://www.python.org/>

<sup>2</sup> <https://www.djangoproject.com/>

<sup>3</sup> <https://www.ubuntu.com/>

<sup>4</sup> <https://github.com/tudarmstadt-lt/annotationApp>

<sup>5</sup> <https://docs.djangoproject.com/en/1.10/ref/databases/>

<sup>6</sup> <https://www.mysql.com/>

<sup>7</sup> <https://httpd.apache.org/>

<sup>8</sup> <http://www.nltk.org/>

<sup>9</sup> <https://github.com/PyMySQL/mysqlclient-python>

---

## Apache Configuration

---

To expose a web application to the internet a web application server is needed. The Apache HTTP Server is used for *Annotate!*. To bridge a connection between a web server and a python application some middleware is needed. The `mod_wsgi` package is a Apache module which can host any Python web application that implements the Web Server Gateway Interface (WSGI) specification. Django enables deployment with WSGI. The `mod_wsgi` serves as middleware to connect the Django application with Apache. For more information on how to install `mod_wsgi` and its usage see the documentation<sup>10</sup>.

After `mod_wsgi` is installed Apache's `httpd.conf`<sup>11</sup> file needs to be augmented with the lines in Code 1. Line 12 specifies the root URL path to serve the application and Line 12 points to the location of the WSGI file. The `<Directory>` tag in Lines 15–19 ensures that Apache can access the file `annotationApp/annotationApp/wsgi.py`. Lines 1–10 create aliases and permissions for both static and media file that are served by the server. For a more detailed explanation on how to deploy Django with `mod_wsgi` on Apache see the Django documentation<sup>12</sup>.

---

```
1 Alias /static/ /var/www/annotationApp/static/
2 Alias /media/ /var/www/annotationApp/media/
3
4 <Directory /var/www/annotationApp/static>
5 Require all granted
6 </Directory>
7
8 <Directory /var/www/annotationApp/media>
9 Require all granted
10 </Directory>
11
12 WSGIScriptAlias / /Path/To/annotationApp/annotationApp/wsgi.py
13 WSGIPythonPath /Path/To/annotationApp
14
15 <Directory /Path/To/annotationApp/annotationApp>
16 <Files wsgi.py>
17 Require all granted
18 </Files>
19 </Directory>
```

---

**Code 1: Apache Configuration**

---

## Annotate! Configuration

---

To grant the Django application access to MySQL, a `.cnf` is created and filled with the content in Code 2. Line 1 specifies that client side configuration follows. In Line 2–3 the user credentials are provided. Line 5 assign the character-set to `utf8mb4` which is the same as in Code 5. Line 6–7 set the database cache equal to the configuration in Code 5.

The last file to be edited before the configuration is complete is the `annotationApp/annotationApp/settings.py` file. This files configures the Django framework. In Code 3 line 3 the database is specified.

---

<sup>10</sup> <https://modwsgi.readthedocs.io/en/develop/>

<sup>11</sup> <https://wiki.apache.org/httpd/DistrosDefaultLayout>

<sup>12</sup> <https://docs.djangoproject.com/en/1.10/howto/deployment/wsgi/modwsgi/>

---

```
1 [client]
2 database=annotationApp
3 user=your_username
4 password=your_password
5 default-character-set=utf8mb4
6 query_cache_type=1
7 query_cache_limit=8000000
```

---

**Code 2: my.cnf**

In line 5 the path to the my.cnf file from Code 2 is added. If the *Annotate!* is not supposed to run directly on the root URL a sub site has to be listed. This is done in line 10 and ensured that all links are generated accordingly. Not only a possible sub site has to be specified but also a base URL which can be seen in line 11.

---

```
1 DATABASES = {
2     'default': {
3         'ENGINE': 'django.db.backends.mysql',
4         'OPTIONS': {
5             'read_default_file': '/Path/To/annotationApp/my.cnf',
6         },
7     }
8 }
9 TIME_ZONE = 'Europe/Berlin'
10 SUB_SITE = '/annotation'
11 BASE_URL = 'http://my-web-site.com' + SUB_SITE + '/'
```

---

**Code 3: Django Configuration**

Finally, after Apache is installed and configured and Django is configured the command `python manage.py collectstatic` is called from within the `annotationApp/` directory. This operation collects all static files (e.g. CSS, JS, images) and all media files (e.g. mp3, mp4) and copies them into the directory specified in the Apache configuration Code 1. The final step in the database setup is to create the tables used by *Annotate!* in the database, which is done with the command `python manage.py migrate`.

This section outlined the prerequisites needed for *Annotate!* to work. It describes how to configure MySQL, Apache and the application itself. The next section explains the initialization of an annotation project.

---

## Initializing an Annotation Project

---

In the previous section the requirements for *Annotate!* are laid out. This section describes how to initialize an annotation project.

`python manage.py createsuperuser`

The first step is to create a superuser. A superuser has all permissions. When called username and password have to be entered.

---

```
python manage.py createGroup
```

A group is a set of users that hold certain privileges or are restricted to specific operations. In the annotationApp the group 'Annotators' is such a group. Members of this group are considered during the distribution of annotation tasks in a later step. They are not automatically permitted to access the 'Admin' screens. On the contrary a superuser that is not a member of the 'Annotators' group is not considered during task distribution. The group has to be created in the database once with this command. For more information see A.3.7.

```
python manage.py createUser
```

This command adds annotators to the database. It expects the username as first argument and a password as second argument. Users created with this command are automatically members of the Annotators group. For more information see A.3.8.

```
python manage.py addLabels
```

The next step is to import a set of labels. The imported labels are the ones to be displayed in screens such as Figure 5.2 and Figure 5.3. This command expects one argument that is a file with the following format. Each line is considered one label. A line is separated with an '@' symbol. Everything to the left of the symbol is considered the label itself. To the right an identifier is expected, either 'check' for check box or 'radio' for radio button. For more information see A.3.2.

```
python manage.py addDocuments
```

This command imports documents into the database. It has three mandatory and one optional argument. Documents can either be imported from a single .csv file or from a directory with exclusively .csv files inside. It is possible to specify an upper limit for the number of tokens a document is supposed to have. For more information see A.3.1.

```
python manage.py createAnnotationQueue
```

Finally, annotators have to be assigned tasks. With this command one can specify how many annotations each document receives. Though, no document is annotated more than once from one annotator. Furthermore, it is possible to suggest labels but also to leave out suggestions and to suggest wrong labels. A detailed explanation can be found at A.3.5.

Now an annotation project is set up and ready for the annotators to begin. The next section presents the different screens of the application.

---

## 5.2 Application Screens

---

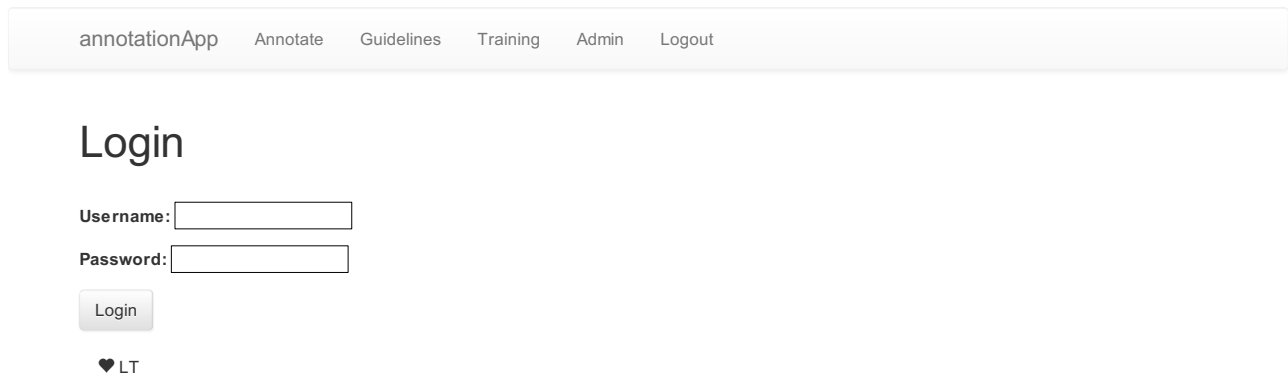
This section introduces the different application screens, their functionality and structure. The layout was created with Bootstrap<sup>13</sup> a popular open source front-end web framework. Bootstrap has a clear design and automatically adjust websites to different devices. This feature enables *Annotate!* to be used not only on desktop machines, but also in tablets and smartphones.

---

### Login Screen

---

The login screen is the first screen a user is presented when visiting the base URL specified in Code 3. The user has to enter her username and password that were supplied during the initialization of the project (see Section 5.1).



The screenshot shows a web application interface. At the top is a light gray navigation bar with the following links: annotationApp, Annotate, Guidelines, Training, Admin, and Logout. Below the navigation bar is the main content area. The word "Login" is displayed in a large, bold, black font. Underneath "Login" are two input fields: "Username:" followed by a text input box, and "Password:" followed by a text input box. Below the password field is a button labeled "Login". At the bottom of the page, there is a small heart icon followed by the text "LT".

**Figure 5.1.:** Login Screen

---

<sup>13</sup> <http://getbootstrap.com/>

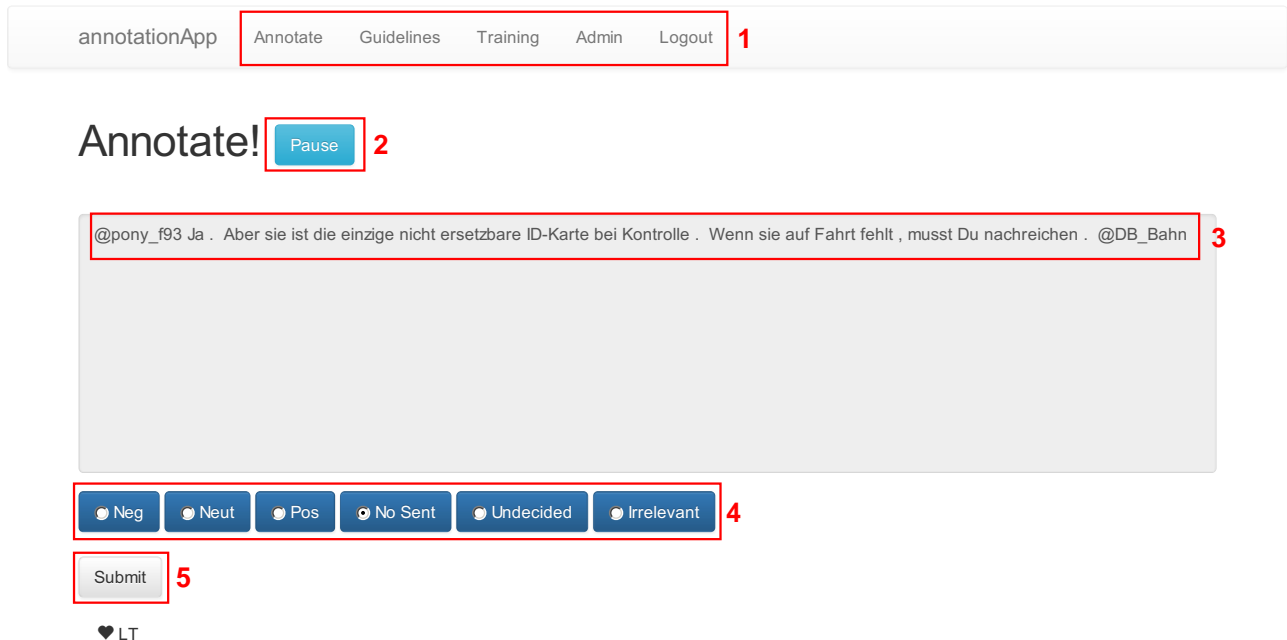


---

## Annotation Screen

---

The annotation screen is the main screen for annotators to work with. A simple layout is used to not distract user from their task at hand. In Figure 5.2 an image of the annotation screen is depicted. The different parts of the screen are explained below.



**Figure 5.2.:** Annotation Screen

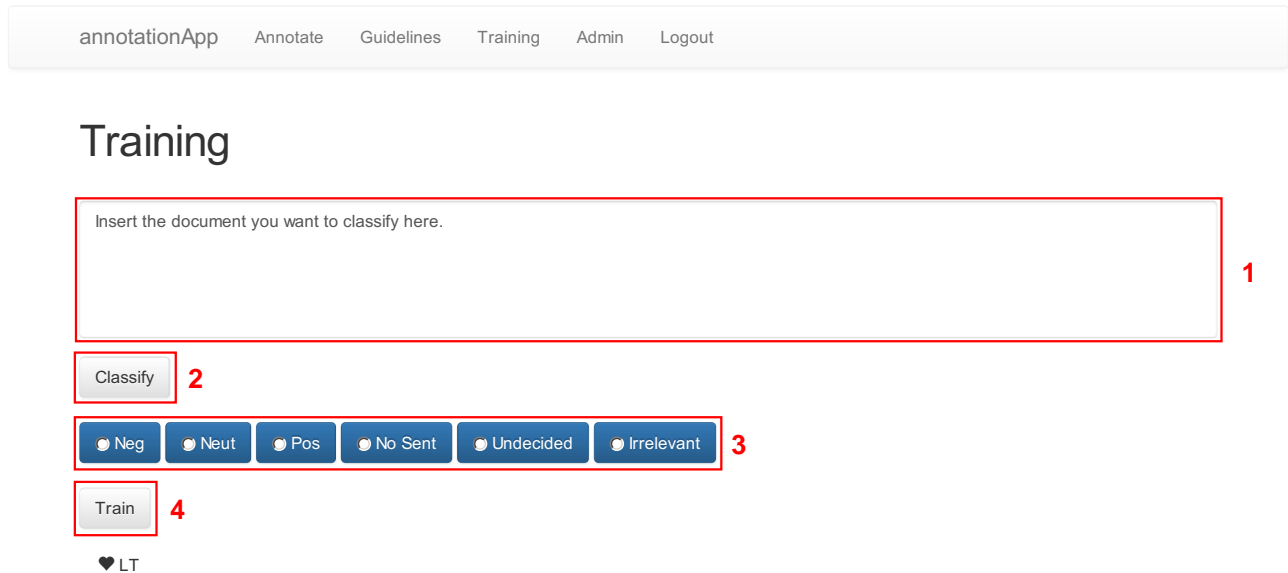
- 1 Navigation Bar** The buttons in the bar are used by users to navigate through the application. For a user with the role of an 'Annotator' mostly the buttons 'Annotate' and 'Guidelines' are relevant. A user with the role of a 'Staff Member' mostly uses the buttons 'Training' and 'Admin'.
- 2 Pause Button** This button pauses the currently active annotation session of a user. As a result the document's text is hidden, the label buttons are disabled and the submit button is disabled. This functionality is purely relevant for the evaluation.
- 3 Text Document** The document to be annotated is display in text area. The size of the text area can be adjusted to the documents size and is not editable.
- 4 Labels** All in the database available labels are displayed in the order they were added to the database. Their appearance changes from radio buttons to check boxes according to what was specified in their import file. Label buttons can be activated and deactivated with the numbers keys on the keyboard.
- 5 Submit Button** After a label is selected, hitting the submit button sends the selected label together with a time measure back to the server application. If no label was selected an alert message appears informing the user to select a label. This button is also triggered with the enter key.

---

## Training Screen

---

The training screen is used for evaluation and manual training. Single documents can be entered, classified and used as training examples. This screen may be used by project supervisors to initialize a classification model. See Appendix A.3.4 for the command-line tool for automatic training. An image of the training screen can be seen in Figure 5.3. In the following the different parts of the screen are explained.



**Figure 5.3.:** Training Screen

- 1 Input Area** In this editable text area documents are added for training.
- 2 Classify Button** With this button added documents are classified with the current classifier model. The result is presented as a pre-selection of one of the label buttons. Additionally, a table is displayed which lists the probability for each label.
- 3 Labels** This list of labels is equivalent to the one the annotation screen 5.2
- 4 Train Button** This button sends the current document and the selected label to the server. The document label pair is treated as a training instance and used to update the classifier's model. Yet, this instance is kept separate from the documents that are supposed to be annotated and the annotations created by the annotators.

---

## Admin Screen

---

The admin screen is auto-generated from Django see Figure 5.4. It offers a graphical interface to interact with the database model 5.3. All entries in the database can be viewed, edited and deleted through the admin screens. Also new entries can be added. The entries are not being aggregated, for an overview of the current annotation project see 5.4

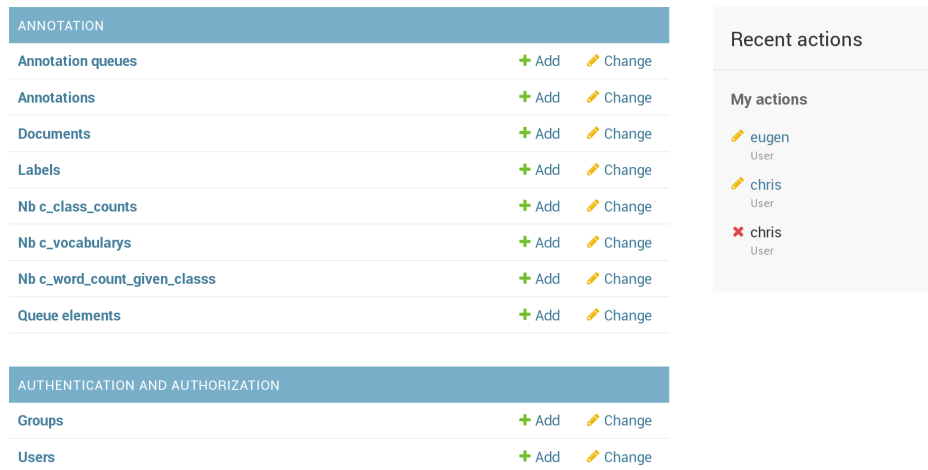


Figure 5.4.: Admin Screen

### 5.3 Database Model

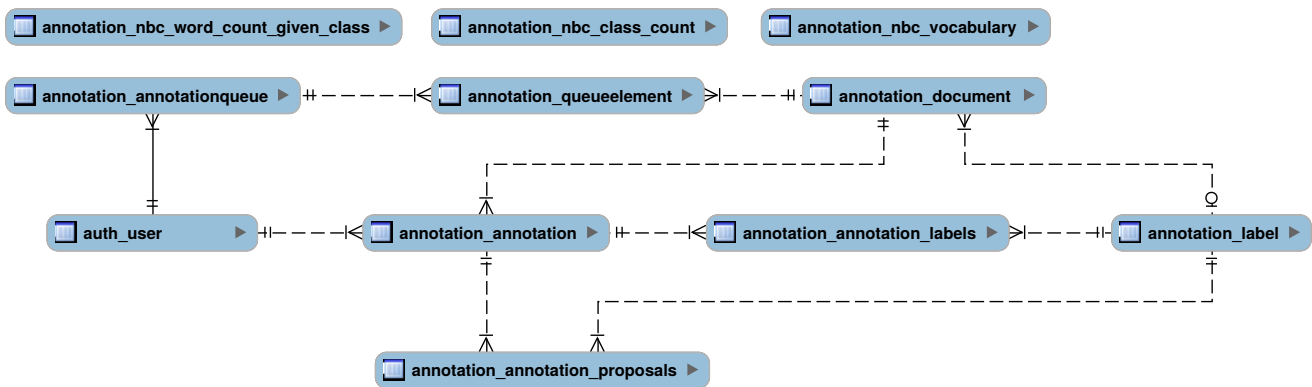


Figure 5.5.: Entity Relationship Model

The database model describes relations between different tables in the database. The model emulates real world relationships between entities that must be persisted. Certain constraints are imposed onto inserting, deleting and altering entries in tables. The Entity Relationship Model (ERM) is a tool in form of a diagram to visualize the different relations between tables. A simplified version of the ERM used in *Annotate!* can be seen in Figure 5.5. This ERM was generated from Django's Object Relational Mapper (ORM). The figure was created with MySQL Workbench 6.3<sup>14</sup>. The ORM prefixes user defined tables with 'annotation\_' this prefix will not be mentioned in the following again.

The tables `NBC_word_count_given_class`, `NBC_class_count` and `NBC_vocabulary` persist the classifier's model in the database. `document` and `annotation` are the core tables of an annotation project. An annotation belongs to exactly one document but a document can have several annotations. The same relation exists between a user and an annotation. An annotation can only belong to one user. To distribute tasks between user an `annotationqueue` is assigned to every user. The `queueelement` is an auxiliary table to realize that a document can be in many `annotationqueues` but also an `annotationqueue` holds many documents. Labels are separated from annotations because they carry information not every annotation has to store (e.g. whether to use a check box or radio button). To be able to have more

<sup>14</sup> <https://www.mysql.de/products/workbench/>

---

than one label per annotation their relation is many to many and realized with the `annotation_labels` table. The same is true for the `annotation_proposals` table to store label suggestions. For a more detailed description of the database model see A.2.

---

## 5.4 Monitoring an Annotation Project

---

To get an overview over the current annotation project the `python manage.py status` command was created. It allows the project lead to get a summary of some key statistics. Figure 5.6 shows an example output.

The result of calling the `status` command is divided into four parts. Firstly, documents and annotations are summarized. The number of documents with 0, 1, 2, etc. annotations are printed. The second part lists the number of annotations every annotator created. The third part outlines agreement upon annotators. The number of documents with 1, 2, 3, etc. distinct labels is listed together with Fleiss' Kappa. For more information about Fleiss' Kappa see 6.2.2. Lastly, an overview of the current classification model is presented together with a confusion matrix.

```
→ annotationApp git:(master) x python manage.py status 3
Documents:
There are currently a total of 7225 documents and 9586 annotations
4052 documents with 0 annotation
30 documents with 1 annotation
41 documents with 2 annotations
2953 documents with 3 annotations
141 documents with 4 annotations
5 documents with 5 annotations
2 documents with 6 annotations
1 document with 14 annotations

Annotators:
stina created 2641 annotations.
mara created 1172 annotations.
falko created 1676 annotations.
jan created 2050 annotations.
marlene created 484 annotations.
moritz created 1548 annotations.
chris created 1 annotations.
eugen created 14 annotations.

Agreement:
1902 documents are annotated with 1 distinct label
950 documents are annotated with 2 distinct labels
101 documents are annotated with 3 distinct labels

Fleiss' Kappa: 0.615318507206

Naive Bayes Model:
The vocabulary consists of 21086 words.
The label No Sent occurred 4874 times with a total of 187813 words
The label Pos occurred 376 times with a total of 16400 words
The label Neg occurred 2601 times with a total of 115911 words
The label Irrelevant occurred 1680 times with a total of 126409 words
The label Undecided occurred 167 times with a total of 9919 words
The label Neut occurred 141 times with a total of 6666 words

Confusion Matrix
Neg Neut Pos No Sent Undecided Irrelevant
Neg 1486 0 6 390 2 17
Neut 39 0 0 25 0 1
Pos 83 0 28 106 0 7
No Sent 1089 0 58 2267 13 71
Undecided 54 1 0 27 0 1
Irrelevant 587 3 36 161 16 317
```

Figure 5.6.: Status Command Results

---

## 5.5 Exporting Annotations

---

To allow further processing annotations are exported with the command `python manage.py exportAnnotation`. Per default annotations are exported in a JSON format. It is also possible to export to a `.tsv` format. For more information about the export command see A.3.10. The JSON format is outlined in Code 4 and explained below.

```
1  {
2    "document id": {
3      "active_prediction": "No Sent",
4      "dateTime": "2016-11-04 14:21:04",
5      "document": "document text",
6      "margin": 0.00148114129970439,
7      "annotations": [
8        {
9          "labels": ["No Sent"],
10         "proposals": ["Neg"],
11         "dateTime": "2016-12-01 18:09:12",
12         "proposalFlag": "proposal",
13         "user": "A",
14         "duration": "15804"
15       },
16       // more annotations
17     ]
18   },
19   // more documents
20 }
```

**Code 4:** JSON Export Format

2	document id	The unique identifier of a document.
3	active_prediction	Predicted label from the last time the last time the document was classified during ranking for active learning.
4	dateTime	Date and time of when the document was added to the database.
5	margin	The margin between the most and second most probable label during the last ranking.
6	annotations	Array with the annotations for this document.
8	labels	Array with the annotated labels.
9	proposals	The suggested labels by the classifier. (Possibly empty)
10	dateTime	Date and time the annotation was added to the database.
11	proposalFlag	A flag indicating the which suggestion mode was used. Possible values are 'proposal', 'no proposal', 'wrong proposal'.
12	user	User name of the annotator.
13	duration	Duration in milliseconds it took the annotator to label the document.

---

## 6 Evaluation

During the course of this thesis an annotation project was carried out with *Annotate!*. This chapter analysis the obtained annotations and measurements. Therefore, a variety of monitoring functionality is included in *Annotate!*. Most notably, the time it took an annotator to submit an annotation is recorded, which is essential to evaluate the hypothesis of the thesis. Also different modes of suggesting a label are distributed across annotation tasks. The modes are *suggestion*, *no suggestion* and *wrong suggestion*. In suggestion mode the Naïve Bayes classifier predicts the label of the document to be presented to the annotator. The suggestion is given in form of a pre-selection of one of the labels in the GUI (see 5.2). No suggestion offers a comparative measure to evaluate the temporal advantage of suggestion. Wrong suggestion predicts a label equally to suggestion, but intentionally one of the other labels is pre-selected. The reason for wrong suggestion is to verify the annotator’s focus on their task. Suggesting labels bears the risk of annotators relying on the pre-selection and not correcting the classifier’s prediction if necessary (see Subsection 6.4.3 for the result). After each annotation is submitted the classification model is online trained with the obtained document label pair.

This chapter is organized as follows. First, the annotation project and its results are described. Then, the quality of the obtained annotations is discussed in terms of the annotator’s agreement. The final label of a document is determined with majority voting. Documents with ambiguous annotations are decided with a method based on annotator agreement. Finally, the proposed hypothesis is tested in detail. Therefore, we analyze the annotation time, estimate the computational overhead caused by suggesting a label and consider the classifiers performance.

---

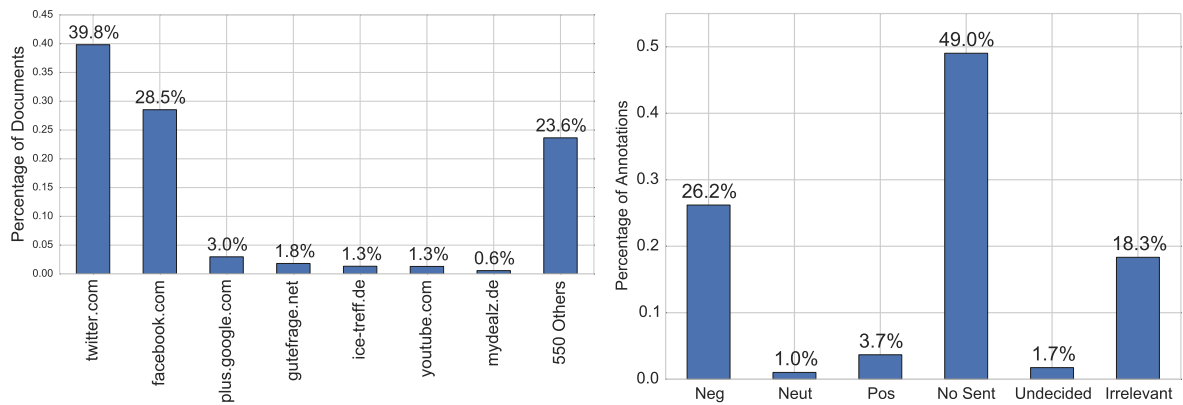
### 6.1 Project Description and Results

---

The goal of the project is to annotate a document’s overall sentiment regarding products and services of Deutsche Bahn AG (DB). Documents are in german and were gathered mostly from social media websites by a third party company. A total number of 557 different websites were crawled. The most common websites can be seen in Figure 6.1a. Even though the number of different sources is big, over one third of the documents were acquired from a single website.

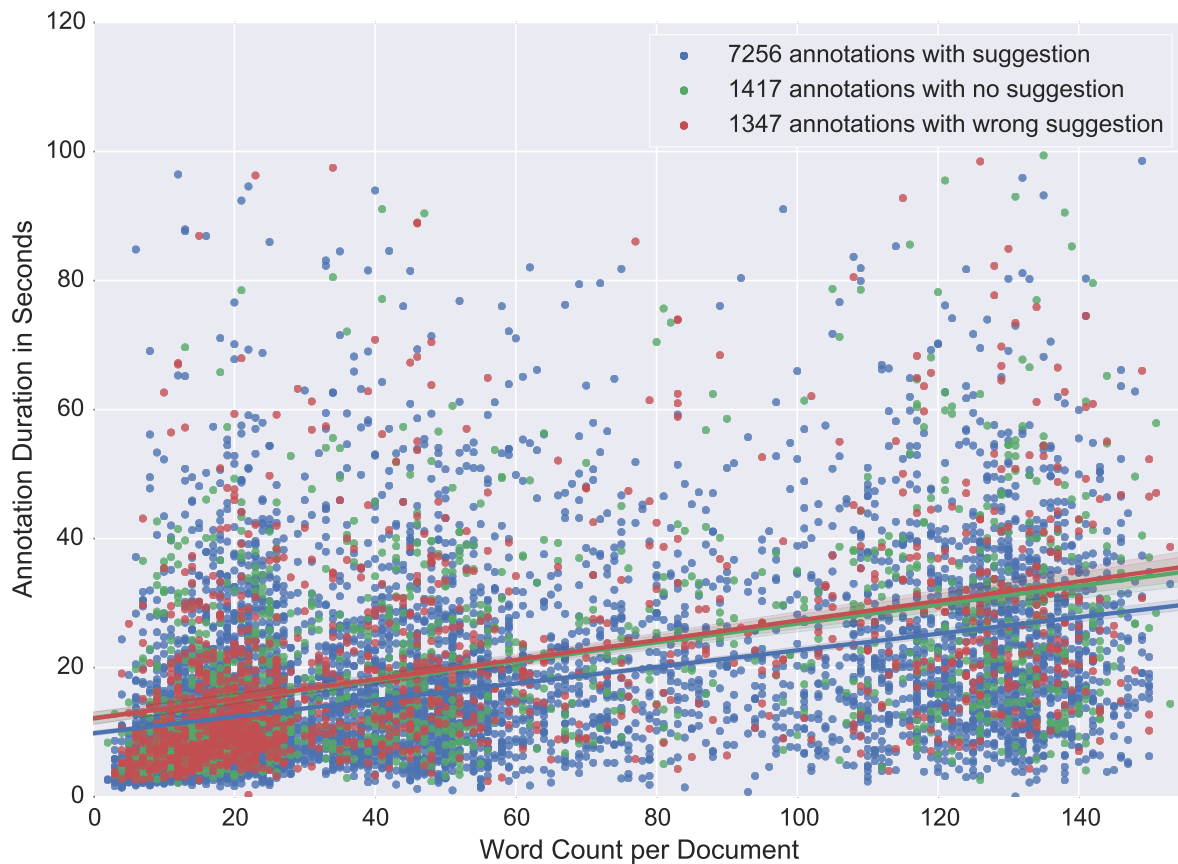
Annotators are instructed to choose one of the labels  $\mathcal{L} = \{\text{Neg, Neut, Pos, No Sent, Undecided, Irrelevant}\}$ . Documents in which a negative, neutral or positive sentiment is expressed are supposed to be annotated with Neg, Neut or Pos respectively. Because annotations are made on a document level it is possible that a document has a sentiment with more than one polarity. For example “My train was in time, but the seats were dirty.” has positive and negative elements. Those documents are labeled with Undecided. On the contrary, a document whose subject is regarding DB but expresses no sentiment is annotated with No Sent. Lastly, Irrelevant is used for documents whose content is not about Deutsche Bahn AG.

To be able to make objective statement about a document’s label and be less dependent on a single annotators expertise, each document is annotated by three different annotators. The final label is determined afterwards. Six annotators actively participated in the project. For privacy reasons they are referred to with alphabetic characters.



(a)

(b)



(c)

**Figure 6.1.:** DB Annotation Project Overview: **(a)** shows the distribution of documents over the most common source websites. Documents were gathered from a total of 557 different websites. **(b)** plots the distribution of annotations over the different labels. **(c)** is a comparison between the annotation times on the vertical axis and the number of words of the corresponding documents on the horizontal axis. A total of 10020 annotations were created during the measurement period. Different annotation modes are differentiated by color. The regression lines indicate a reduced annotation time if suggestions were made as well as an increase of this effect for documents with a larger number of words.

Annotators were free to schedule their annotation tasks throughout a given time period. While creating annotations they were not monitored in a controlled environment. This opens up the possibility of

---

measurement errors, even though they were instructed to use the pause button (see Figure 5.2) when taking a break or being distracted elsewhere.

A total number of 10060 annotations have been created. Their distribution over the different labels is depicted in Figure 6.1b. Almost half of all annotations belong to No Sent which makes them irrelevant for further processing. A quarter of annotations are Neg, about seven times as many as Pos. This result is expected because of a psychological bias towards negative events. We are more likely to remember negative events, emotions, persons, etc. than positive ones Baumeister et al. (2001), Rozin & Royzman (2001). Therefore, it can be expected to find more reports about negative customer experiences online. Neut and Undecided are both represented with below two percent.

The different modes are used with different volume. 7282 of annotations get a label suggested, 1423 of annotations get no label suggested and 1355 of annotations get suggested a wrong label.

In total 3255 documents are fully annotated. The maximum number of words per document is capped at 150 words. Additional words are ignored on document import (see Appendix A.3.1). The average number of words per document is 50.3. Interestingly enough in Figure 6.1c it appears that there are at least three clusters of documents with different word length. The first cluster is in the range of zero to 30 words, the second cluster is in the range of 30 to 90 words and the last cluster is in the range of 90 to 150 words. Further investigation of this phenomenon reveals its cause to be the different document sources. The primary source of documents in the first cluster is Twitter. Twitter has a policy to restrict message length to 140 characters. The documents in the second cluster are mostly obtained from Facebook. Facebook comments are longer than tweet but generally do not exceed the length of a paragraph. The third cluster is composed from different sources, containing longer articles.

This section gives an overview of the annotation project. The next section discusses the obtained annotations and their quality.

---

## 6.2 Annotation Quality and Annotator Agreement

---

Classifying a text document into several classes is a challenging task not only for a computer program but may also be debatable among humans. In this section the inter annotator agreement (IAA) among annotators about labels is analyzed. The premise is that there is no criterion to correct annotation. There are neither gold labels (meaning verified correct labels) nor an supervising expert to ask for clarification but rather annotators and their annotations is all there is. Furthermore, all annotators are equally competent and there are no restrictions placed upon them Cohen (1960). Therefore, agreement upon annotations is an important measure to make statements about the quality of annotations. It is an attempt at an objective measure that is independent of the single annotator. A high agreement is interpreted as a high quality of labels, whereas low agreement is considered low quality. Figure 6.2 shows the number of documents over the number of annotations per document and the agreement upon annotators.

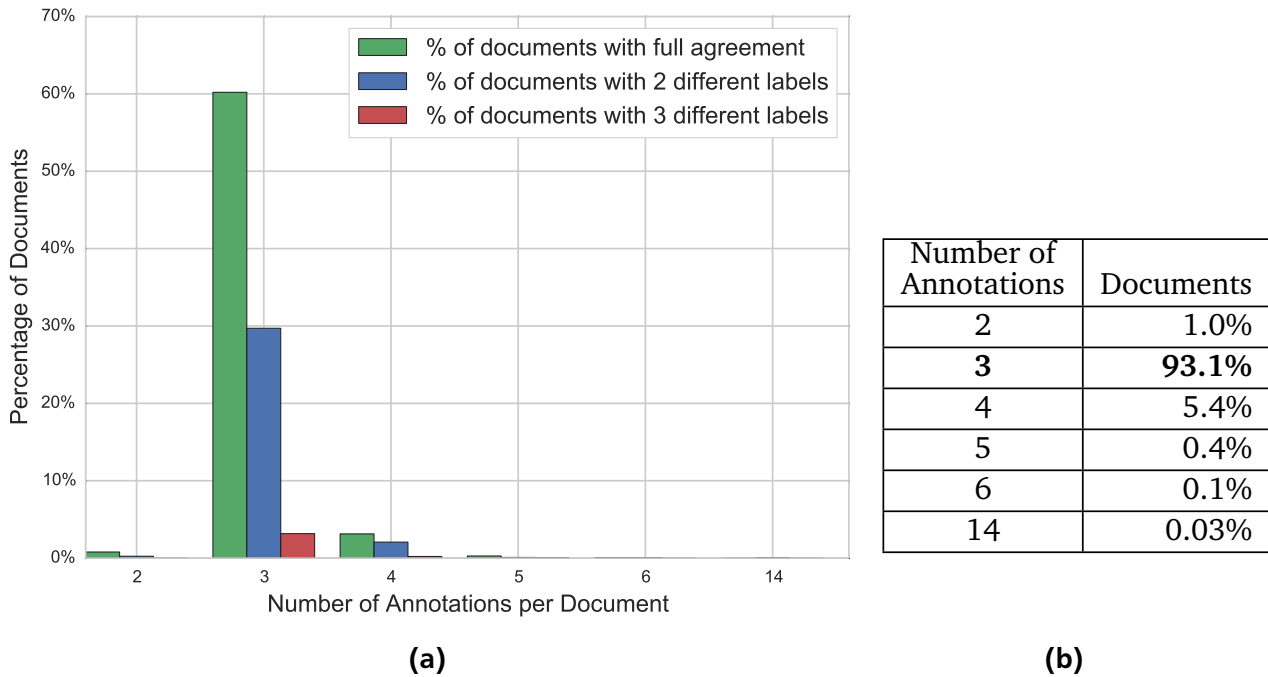
The observed agreement can be defined as the percentage of annotations on which two annotators agree when they judge the same document Scott (1955). Let  $N$  be the total number of fully annotated documents and  $a$  be the number of documents on which two annotators agree, then the observed agreement is defined as,

$$p_0 = \frac{a}{N} \quad (6.1)$$

Yet, the observed agreement and Figure 6.2 are no more than an overview because they ignore an important factor.



Two annotators might agree simply by coincidence. With six possible labels two annotators agree with  $\approx 8.3\%$  purely by chance. This is especially relevant for documents that are controversially annotated. In the following sections two statistics are presented that compensate for random picks.



**Figure 6.2.:** Agreement Overview: A total of 3289 documents have two or more annotations. **(a)** shows for different number of annotations the percentages of documents for which annotators either fully agree, partially agree or do not agree at all. **(b)** shows the distribution of documents over the number of annotations. As expected the large majority of documents has three annotations. Yet, there is a considerable amount of documents with 4 or more annotations — even one document with fourteen annotations. This behavior was thought to be caused by users and their interaction with the system. A closer investigation revealed a pattern that this phenomenon seems to follow. The exceeding annotations are almost always created by the user that created the third annotation. One would expect a certain user to be the originator more than others if it was caused by users. Instead it appears this is caused by the system. Though it is not clear what causes the additional annotations.

## 6.2.1 Cohen’s Kappa

Cohen’s Kappa is a statistic to calculate the IAA for a nominal scale that takes random agreement into account Cohen (1960). It is applicable for exactly two annotators. Either two different ones or the same annotator twice for different measurements. For our discussion only the first is relevant. Furthermore, it processes multi-class classification but not multi-label classification. Cohen’s Kappa is defined as

$$\kappa = \frac{p_0 - p_e}{1 - p_e}, \quad (6.2)$$

where  $p_e$  is the expected agreement by chance. The denominator of Equation 6.2 measures how much agreement above chance is achievable, while the numerator indicates the agreement found beyond chance Artstein & Poesio (2008).

The calculated value is between -1 and 1. Intuitively 1 means strong agreement, 0 is agreement purely by chance and -1 is active disagreement. A more differentiated scale was proposed by Landis & Koch (1977) who assigned the ranges to agreement qualities as seen in Table 6.2. As they point out this partitioning is arbitrary but serves as useful in practice. An alternative to the scale of Landis et al. was proposed by Fleiss (1981). Their scale consists of three sections which are listed in Table 6.3b.

The results for all pairs of annotators are listed in Table 6.1. They range from 0.48 to 0.747 and are therefore moderate to substantial. Cohen (1968) introduced a weighted variant of this statistic.

**Table 6.1.: Inter Annotator Agreement**

	A	B	C	D	E	F
A		0.711	0.57	0.499	0.65	0.498
B			0.595	0.589	0.747	0.625
C				0.536	0.68	0.48
D					0.586	0.483
E						0.611
F						

**Table 6.2.: Strength of Agreement for Different Kappa Ranges**

**(a) Kappa Scale by Landis & Koch (1977)**

Kappa Statistic	Streng of Agreement
< 0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substantial
0.81-1.00	Almost Perfect

**(b) Kappa Scale by Fleiss (1981)**

Kappa Statistic	Streng of Agreement
< 0.40	Poor
0.40-0.75	Fair to Good
> 0.75	Excellent

## 6.2.2 Fleiss' Kappa

An extension of Cohen's unweighted Kappa for any constant number of annotators is Fleiss' Kappa Fleiss et al. (1971). The basic form of the Fleiss' Kappa statistic is still the same

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}. \quad (6.3)$$

But the observed agreement is now defined as

$$\bar{P} = \frac{1}{N} \sum_{i=1}^N P_i, \quad \text{where} \quad P_i = \frac{1}{n(n-1)} \sum_{j=1}^N n_{ij}(n_{ij} - 1) \quad (6.4)$$

with  $n_{ij}$  the number of annotators who assigned the  $j$ th label to the  $i$ th document and  $n$  the number of annotations per document. The expected agreement becomes,

$$\bar{P}_e = \sum_{j=1}^k p_j^2, \quad \text{where} \quad p_j = \frac{1}{Nn} \sum_{i=1}^N n_{ij} \quad (6.5)$$

---

$p_j$  is the proportion of annotations of the  $j$ th label.

Fleiss' kappa for the DB project is 0.621. For Fleiss' kappa the ranking in Table 6.2 can also be applied. Therefore, the overall agreement is considered to be substantial.

---

### 6.3 Determining Final Labels with Majority Vote

---

Each document is annotated once by three different annotators. To create a labeled training set the final labels of the documents have to be determined from their annotations. The annotations are not created simultaneously, therefore they have a chronological order. The following cases are distinguished

**Case A** All three annotators agree upon a document.

**Case B** Two of the three annotators agree and the third annotator picked a different label. This case can be split up into three sub-cases.

**Case B.1** The first two annotators agree and the third disagrees.

**Case B.2** The first and the last annotators agree and the second disagrees.

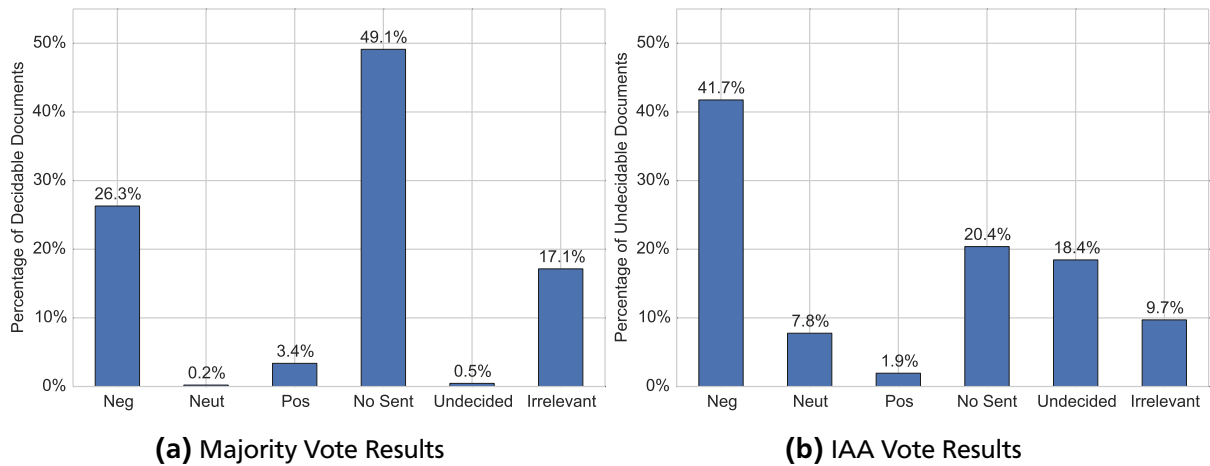
**Case B.3** The second and the last annotators agree and the first disagrees.

**Case C** All three annotators disagree.

Case A is unambiguous and leaves no room for interpretation. To determine the label in Case B majority voting is used. Majority voting returns the label which has the highest frequency in the set of annotations. Thus it is evident that Case C is not decidable with majority voting. We propose to leverage the insights gained from the IAA in Table 6.1 to determine a label for these ambiguous annotations. This proposal is motivated by considering that an annotator with high agreement is interpreted to be more trustworthy of creating qualitative annotations. The prerequisite is that we have no information about the expertise of any annotator. To determine the label of an ambiguously annotated document the following method is proposed.

- For each annotator take the mean of the IAA scores that annotator has with any other annotator that created an annotation for the document in question.
- Select the label of the annotator with the highest mean.

The results of the majority vote and the IAA vote are shown in Figure 6.3. In the next section the temporal advantage of suggestion labels is analyzed.

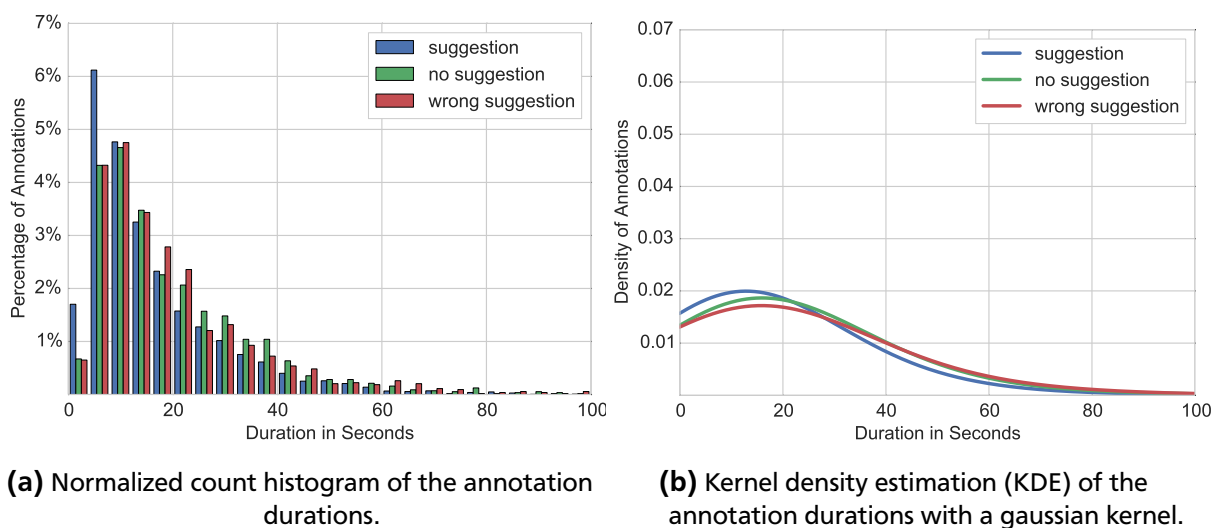


**Figure 6.3.:** Final Label Distribution: Of the 3255 fully annotated documents, 3145 are decidable with majority vote and 110 are ambiguous.

## 6.4 Evaluating the Advantage of Suggesting Labels

This section examines whether or not suggesting a label during annotation is advantageous. Therefore, we consider the difference between the three modes suggestion, no suggestion and wrong suggestion.

For an overview over the different modes consider Figure 6.4. Figure 6.4a shows a histogram with 25 normalized bins for each mode. A trend in favor of suggesting labels is noticeable. But a disadvantage of a histogram is that it is discrete. We much rather would like to have a continuous representation because time measurements are continuous as well. Figure 6.4b shows a kernel density estimation (KDE) of the same data. KDE is continuous but it does not represent the data adequately. The extreme values around 5–10 seconds are not as concise as in the histogram.



**Figure 6.4.:** Estimating the Distribution of Durations

---

## 6.4.1 Temporal Difference in Annotation Time When Suggesting Labels

---

In the introduction of this section the general shape of time measurement distributions for each mode is indicated. This leads us to choose four distributions which shape resembles that seen in Figure 6.4. The four distributions are the beta prime, chi-squared, exponential and gamma distribution. They all have in common that their support is the intervall  $[0, \text{inf})$  and the majority of their probability mass is near zero given adequate parameters. This specific selection is somewhat arbitrary as there are many distributions that share these properties.

To get an appropriate representation of the different time measurements our goal is to select a distribution that best describes the data, while being as simple as possible. In this case simplicity is characterized by the number of parameters a distribution has. In general the more parameters a distribution has the more expressive it is but also more complex.

To estimate the parameter values of a distribution that best describe the data the maximum likelihood estimation (MLE) (Thisted 1988) is used. MLE tries to maximize the likelihood function given the time measurements. Let  $f$  be the density function whose parameter  $\theta$  we want to estimate. Then, the joint density function for all measurements is

$$f(x_1, x_2, \dots, x_N | \theta) = \prod_{i=1}^N f(x_i | \theta), \quad (6.6)$$

where  $x_1, x_2, \dots, x_N$  are the time measurements and  $N$  is the number of measurements. For Eq. 6.6 to hold true the prerequisite is that the measurements are sampled *independently and identically distributed* (iid). The likelihood function takes the parameters as input given the measurements. Therefore, the likelihood function is

$$\mathbb{L}(\theta | x_1, x_2, \dots, x_N) = \prod_{i=1}^N f(x_i | \theta) \quad (6.7)$$

The maximum likelihood estimate  $\hat{\theta}$  is

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \mathbb{L}(\theta | x_1, x_2, \dots, x_N) \quad (6.8)$$

The likelihood function is a prerequisite to use the Bayesian information criterion (BIC) (Schwarz 1978). With the BIC one of the proposed distributions is selected to model the measurements. The BIC is defined as

$$\text{BIC} = \ln(N)k - 2 \ln(L), \quad (6.9)$$

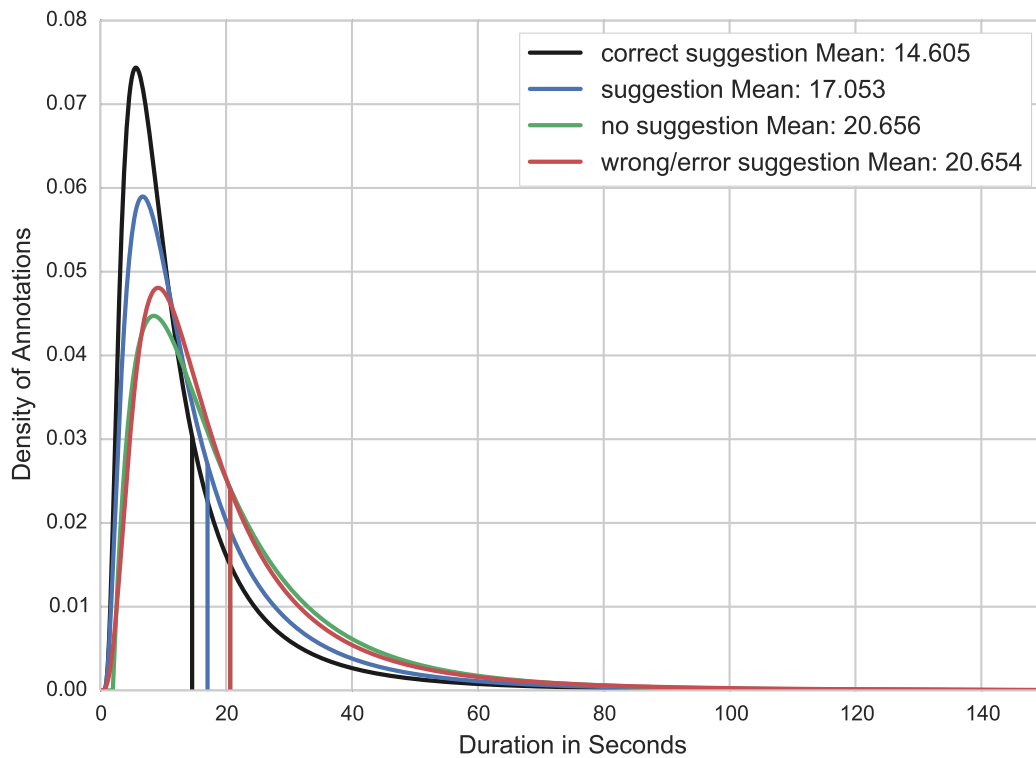
where  $L$  is the maximized value of the likelihood function and  $k$  the number of parameters of each distribution. Table 6.4 lists the BIC results for the different distributions. The beta-prime distribution is selected to model the time measurements as it has the lowest values for all modes.

Figure 6.5 shows a different beta-prime distribution for each mode. The parameters for these distributions are fitted with MLE. Note that the suggestion mode is split into to *correct suggestion* and *error*

*suggestion*. Correct suggestions are those for which the classifier predicted the label returned by majority vote. A total of 4105 suggestions are correct. On the other, side error suggestion are those, for which the classifier predicted a label different from that obtained by majority vote. A total of 2971 suggestions are erroneous. Wrong suggestion and error suggestion are grouped together for clarity. This is sensible because their BIC and MLE are identical up to three decimal digits. In Figure 6.5 we see a clear temporal advantage in suggesting labels. This advantage is larger in case only the correct suggestions are considered.

**Table 6.4.:** Comparison of Bayesian Information Criterion Results

BIC	correct suggestion	suggestion	no suggestion	wrong/error suggestion
betaprime	<b>28727.0</b>	<b>52098.95</b>	<b>10695.74</b>	<b>33016.66</b>
chi2	29167.99	52725.5	10726.45	33471.81
expon	29583.82	53487.3	10828.7	34294.49
gamma	29167.99	52725.5	10726.45	33471.81



**Figure 6.5.:** Beta-prime distributions fitted with MLE.

## 6.4.2 System Overhead of Suggesting Labels

In the previous section we saw an improvement in average annotation time by  $\approx 3.37$  seconds when suggesting a label. But this improvement comes at a cost. Training the classifier and predicting a document's label costs additional processing time and noticeably influences the performance of the system. To account for this overhead the annotation's timestamps are used to estimate the temporal difference between suggestion and no suggestion. As notes in Section 6.1 annotators were not monitored while working, which introduces a possible bias. Also network latency is a possible source of distortion. Additionally, programming related challenges add to the complexity of interpreting the measurements. Thus, the measurements have to be seen as estimates. Figure 6.6 shows the estimated results.

The total time it took to create an annotation is calculated by taking the difference between timestamps of two consecutive annotations. To account for inactivity, periods of more than five minutes between annotations are discharged. The accumulated total time in hours of a mode is written at the top of each bar. The time it took an annotator to submit an annotation is also recorded. Its proportional amount of the total time is visualized in blue. The temporal overhead created by the system is given in percent. During the annotation project the database access was optimized. Figure 6.6a depicts the initial measurements while Figure 6.6b shows the measurements after the optimization.

Unfortunately, there was a error in the code that was undiscovered until after the measurements were made. When no suggestion was given an unnecessary prediction was made in some circumstances. In case of a wrong suggestion no on demand prediction was made but rather the field `active_prediction` was used (see Code 4 line 3). When a suggestion was made the program worked as intended. As a consequence the following percentages can be assumed. Wrong suggestion should take as much time as suggestion and therefore  $\approx 8.5\%$ . When no suggestion is made it can reasonably be assumed to take  $\approx 3.6\%$  of the total time. The results imply an increase of  $\approx 4.9\%$  of computation time for suggestion compared to no suggestion. However, this drawback may be further reduced as the prototype matures.

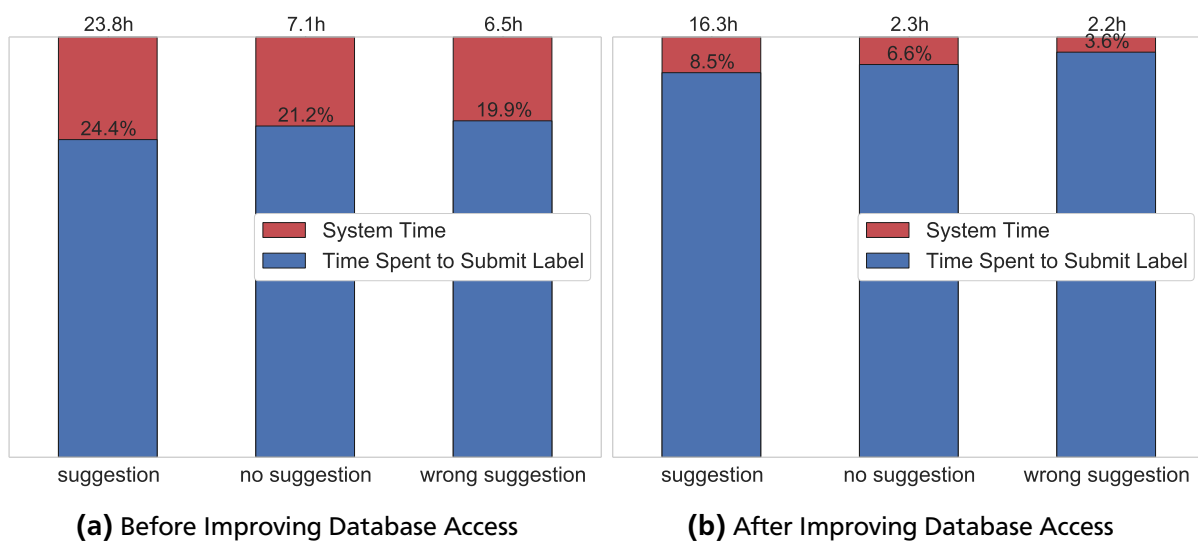


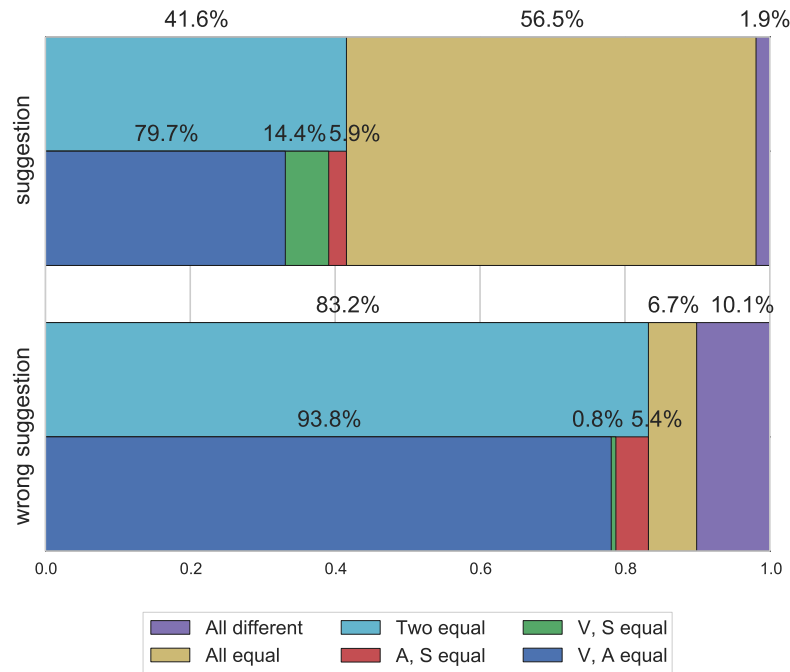
Figure 6.6.: System Time vs. Annotation Duration:

### 6.4.3 Classifier Performance

Considering the results from Figure 6.5 it is evident that suggestion the correct label contains a great potential to reduce annotation time. As a consequence the classifier's performance directly influences the time saving gained from suggesting labels. Considering all predictions made during the annotation project the classifier has an overall accuracy of 62.5%. For a differentiated listing of the prediction results see Figure 6.7. This figure not only depicts the performance of the classifier but also provides information about the annotators.

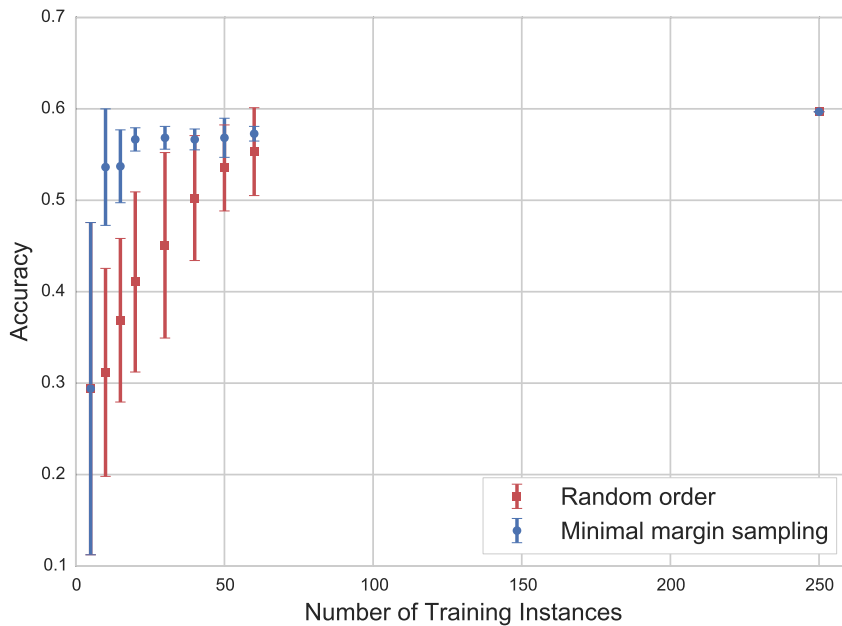
As described in the introduction to this chapter, the intension of the wrong suggestion mode is to test their focus on the task. In 4.5% of wrong suggestions the annotator submitted the pre-selected label, that is not equal to the majority vote.

The decision to online train the classifier for each annotation project is also affected by the results from Figure 6.5. Active learning was examined and implemented with the prospect of reduced training data needed. To evaluate this prospect a learning curve is created from the obtained data. A comparison of the learning curves of random selection and minimal margin sampling is shown in Figure 6.8.



**Figure 6.7.:** Prediction Results: Consider all documents with three or more annotations, then ignore all fourth and more annotations (see Figure 6.2). Then use majority vote to determine the label of these documents and ignore undecidable documents. For these documents there were 8112 annotations created of which 84.6% got a suggestion and 15.4% got a wrong suggestion. The letters in the legend stand for annotation (A), suggestion (S) or wrong suggestion and result of majority vote (V). 'All equal' describes instances in which all three are the same label while 'All different' are instances in which the majority vote, annotation and suggestion are all different. The case that only two match is subdivided into its possible combinations.





Number of Training Examples	Accuracy Random Order	Accuracy Minimal Margin
5	0.294	0.294
10	0.312	0.536
15	0.369	0.537
20	0.411	0.567
30	0.451	0.568
40	0.502	0.567
50	0.535	0.568
60	0.553	0.573
250	0.597	0.597

(a)

(b)

**Figure 6.8.:** Learning Curve of Random Order vs. Minimal Margin Sampling: Consider all documents with three or more annotations, then ignore all fourth and more annotations (see Figure 6.2). Then use majority vote to determine the label of these documents and ignore undecidable documents. Of those randomly select 250 documents and their respective majority votes as training set and 500 documents and their respective majority votes as test set. Train two separate classification models stepwise. For one model use minimal margin sampling to select the documents for the next step from the training set. For the other model keep the random order. After each step evaluate both models on the test set. Repeat this process 10 times and ensure that the training sets are disjoint. Also keep the step size constant across repetitions. **(a)** shows the mean accuracy and two standard deviation for each training step across the different repetitions. **(b)** shows the number of training examples used for each step and the respective mean accuracy for both models.

---

## 7 Conclusion

This thesis was motivated by the need for a general-purpose collaborative web-based annotation tool for document wise annotation. To support annotators further, generated label suggestions were introduced. The hypothesis was proposed that suggesting labels decreases the annotation time.

BRAT, GATE Teamware and Web-Anno were described as representatives of general-purpose annotation tools.

Reviewing the literature of short text classification revealed different feature strategies to cope with the challenges, imposed on a classifier by short text. The biggest issue is sparseness caused by the few word occurrences. As possible approaches we reviewed feature augmentation with topic models, special character strings, meta information such as author and publication frequency and lexical features.

Starting out from the practical consideration that annotation projects are carried out in very different domains, awareness of the domain adaptation problem was raised. Approaches such as a discriminate maximum entropy framework, instance weighting and correspondence features were mentioned. Ultimately, a different approach was pursued with online learning. Online learning has the benefit of full automatized domain specific training. To suggest labels to the annotators we presented the Naïve Bayes classifier.

As further enhancement to online learning, active learning was introduced. The different application scenarios of active learning in the literature were presented. For the usage in an annotation project pool-based sampling was recognized. Minimal margin sampling was used to select documents to be annotated first.

A prototypical application was realized with the introduced concepts. The application screens were developed with a focus on simplicity and familiarity to the user. A client-server architecture allows task distribution between the annotators. The database model was conceptualized to enable effortless extensibility. A feature-rich set of command line tools allow a project supervisor monitor annotators, configure projects and extract common statistics. Commented source code and a comprehensive documentation add further value.

The developed application was deployed in an annotation project. The obtained data was used to evaluate the application and the hypothesis. Agreement between pairs of annotators and overall agreement was analyzed. The Fleiss' kappa is 0.621 which is considered to be substantial agreement. To create a labeled training set from the acquired annotations, majority vote was used to determine the final label of a document. For documents whose annotations were ambiguous and not decidable by majority vote, a methodology was proposed that leverages the IAA to find a label. The time it took annotators to submit annotations was recorded. Different modes of assisting the annotator were implemented to analyze the effectiveness of suggesting labels. The time distributions of the different modes were modeled as beta-prime distributions based on a MLE. Without any suggestions it took annotators a mean of 20.656 seconds to submit a label, with suggestions 17.053 seconds, if the suggestion was correct 14.605 seconds and if the suggestion was wrong 20.654 seconds (intentionally wrong and misclassified suggestions). Based on timestamps it was estimated that suggesting a label increases the systems computation time by 4.9%. The classifier was evaluated to have an overall accuracy of 62.5% when considering all predictions made during the annotation project. A learning curve was created to evaluate the active learning component. A mean accuracy of 0.567% was reached after 20 training documents.

---

## 7.1 Future Work

---

The results obtained in this thesis offer a variety of future directions to be pursued. First, it is worth noted that the developed application currently is in a prototypical stage. The application benefits from additional features and more stability. Feature proposals include multiple project management, additional monitoring tools and a graphical realization of a hierarchical label structure.

A more sophisticated feature transformation and an advanced classifier are most promising to further decrease the average annotation time. Both changes are easily incorporated in the current code base. If full agreement upon annotators is not mandatory, the observations from Section 6.3 can be leveraged to further reduce required resources. Case A and Case B.1 are unambiguously determined by majority vote. To reduce the total number of annotations created, their chronological creation can be exploited. After the first two annotations are submitted, they are tested for equality. If their labels are equal, the corresponding document is removed from the pool of unlabeled documents. For the executed annotation project this method would have reduced the total number of annotations by 2425 annotations, which is a reduction of 24.3%.

Case B.2 and Case B.3 are both decided with the third annotation. In both cases the classifier is trained with a document twice but with different labels. The idea is to predict the determining label. For this approach to be useful a high classification accuracy is needed. In the DB project 372 annotations are affected by this method. The Naïve Bayes model yields an accuracy of 57.3%.

Finally, an evaluation of the proposed IAA vote could be realized as follows. Instead of three annotations each document is annotated by five different annotators. These documents are filtered for documents that are undecidable with the first three annotations but decidable with all five. Determine a label set based on the first three annotations with IAA vote. Determine a gold label set with on all five annotations and majority vote. Evaluate the IAA vote labels with the gold labels.

---

# Bibliography

- Agarwal, A., Biadys, F. & Mckeown, K. R. (2009), Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams, in 'Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics', EACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 24–32.
- Agarwal, A., Xie, B., Vovsha, I., Rambow, O. & Passonneau, R. (2011), Sentiment analysis of twitter data, in 'Proceedings of the Workshop on Languages in Social Media', LSM '11, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 30–38.
- Aggarwal, C. C., Kong, X., Gu, Q., Han, J. & Yu, P. S. (2014), Active learning: A survey, in C. C. Aggarwal, ed., 'Data Classification: Algorithms and Applications', CRC Press, pp. 571–606.
- Angluin, D. (1988), 'Queries and concept learning', *Mach. Learn.* **2**(4), 319–342.
- Artstein, R. & Poesio, M. (2008), 'Inter-coder agreement for computational linguistics', *Comput. Linguist.* **34**(4), 555–596.
- Asur, S. & Huberman, B. A. (2010), Predicting the future with social media, in 'Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01', WI-IAT '10, IEEE Computer Society, Washington, DC, USA, pp. 492–499.
- Atlas, L., Cohn, D., Ladner, R., El-Sharkawi, M. A., Marks, II, R. J., Aggoune, M. & Park, D. (1990), Training connectionist networks with queries and selective sampling, in D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems 2', Morgan Kaufmann Publishers Inc., pp. 566–573.
- Baldrige, J. & Palmer, A. (2009), How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation, in 'Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1', EMNLP '09, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 296–305.
- Banerjee, S., Ramanathan, K. & Gupta, A. (2007), Clustering short texts using wikipedia, in 'Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '07, ACM, New York, NY, USA, pp. 787–788.
- Baumeister, R., Bratslavsky, E., Finkenauer, C. & Vohs, K. (2001), 'Bad is stronger than good', *Review of General Psychology* **5**(4), 323–370.
- Berger, A. L., Pietra, V. J. D. & Pietra, S. A. D. (1996), 'A maximum entropy approach to natural language processing', *Comput. Linguist.* **22**(1), 39–71.
- Biemann, C. (2013), 'Creating a system for lexical substitutions from scratch using crowdsourcing', *Language Resources and Evaluation* **47**(1), 97–122.
- Biemann, C., Bontcheva, K., Eckart de Castilho, R., Gurevych, I. & Yimam, S. M. (2017), Collaborative web-based tools for multi-layer text annotation, in N. Ide & J. Pustejovsky, eds, 'The Handbook of Linguistic Annotation', Text, Speech, and Technology book series, Springer Netherlands.
- Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003), 'Latent dirichlet allocation', *J. Mach. Learn. Res.* **3**, 993–1022.

- 
- Blitzer, J., Dredze, M. & Pereira, F. (2007), Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification, in 'In ACL', pp. 187–205.
- Blitzer, J., McDonald, R. & Pereira, F. (2006), Domain adaptation with structural correspondence learning, in 'Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing', EMNLP '06, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 120–128.
- Bontcheva, K., Cunningham, H., Roberts, I., Roberts, A., Tablan, V., Aswani, N. & Gorrell, G. (2013), 'Gate teamware: a web-based, collaborative text annotation framework', *Language Resources and Evaluation* 47(4), 1007–1029.
- Chen, M., Jin, X. & Shen, D. (2011), Short text classification improved by learning multi-granularity topics, in 'Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three', IJCAI'11, AAAI Press, pp. 1776–1781.
- Cohen, J. (1960), 'A coefficient of agreement for nominal scales', *Educational and Psychological Measurement* 20(1), 37–46.
- Cohen, J. (1968), 'Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit', *Psychological Bulletin* 70, 213–220.
- Cohn, D., Atlas, L. & Ladner, R. (1994), 'Improving generalization with active learning', *Mach. Learn.* 15(2), 201–221.
- Cortes, C. & Vapnik, V. (1995), 'Support-vector networks', *Mach. Learn.* 20(3), 273–297.
- Cui, H., Mittal, V. & Datar, M. (2006), Comparative experiments on sentiment classification for online product reviews, in 'Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2', AAAI'06, AAAI Press, pp. 1265–1270.
- Daumé, III, H. & Marcu, D. (2006), 'Domain adaptation for statistical classifiers', *J. Artif. Int. Res.* 26(1), 101–126.
- Donmez, P. & Carbonell, J. G. (2010), *From Active to Proactive Learning Methods*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 97–120.
- Fellbaum, C., ed. (1998), *WordNet An Electronic Lexical Database*, The MIT Press, Cambridge, MA ; London.
- Fleiss, J. L. (1981), *Statistical Methods for Rates and Proportions*, Wiley series in probability and mathematical statistics, second edn, John Wiley & Sons, New York.
- Fleiss, J. et al. (1971), 'Measuring nominal scale agreement among many raters', *Psychological Bulletin* 76(5), 378–382.
- Go, A., Bhayani, R. & Huang, L. (2009), 'Twitter sentiment classification using distant supervision', *Processing* pp. 1–6.
- Jiang, J. & Zhai, C. (2007), Instance weighting for domain adaptation in nlp, in 'In ACL 2007', pp. 264–271.
- Landis, J. R. & Koch, G. G. (1977), 'The measurement of observer agreement for categorical data', *Biometrics* 33(1).
- Lang, K. & Baum, E. (1992), Query learning can work poorly when a human oracle is used, in 'In Proceedings of the IEEE International Joint Conference on Neural Networks', IEEE Press, pp. 335–340.

- 
- Lewis, D. D. & Catlett, J. (1994), Heterogeneous uncertainty sampling for supervised learning, in 'In Proceedings of the Eleventh International Conference on Machine Learning', Morgan Kaufmann, pp. 148–156.
- Lewis, D. D. & Gale, W. A. (1994), A sequential algorithm for training text classifiers, in 'Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '94, Springer-Verlag New York, Inc., New York, NY, USA, pp. 3–12.
- Manning, C. D., Raghavan, P. & Schütze, H. (2008), *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA.
- McCallum, A. & Nigam, K. (1998a), A comparison of event models for naive bayes text classification, in 'in AAI-98 Workshop on Learning for Text Categorization', AAAI Press, pp. 41–48.
- McCallum, A. & Nigam, K. (1998b), Employing em and pool-based active learning for text classification, in 'Proceedings of the Fifteenth International Conference on Machine Learning', ICML '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 350–358.
- Moschitti, A. (2006), Efficient convolution kernels for dependency and constituent syntactic trees, in 'ECML, Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Proceedings, Berlin, Germany, pp. 318–329.
- Nadeau, D. & Sekine, S. (2007), 'A survey of named entity recognition and classification', *Linguisticae Investigationes* 30(1), 3–26. Publisher: John Benjamins Publishing Company.
- Ng, H. T., Goh, W. B. & Low, K. L. (1997), Feature selection, perceptron learning, and a usability case study for text categorization, in 'Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '97, ACM, New York, NY, USA, pp. 67–73.
- Olsson, F. (2009), A literature survey of active machine learning in the context of natural language processing, Technical Report 06, Swedish Institute of Computer Science.
- Pan, S. J. & Yang, Q. (2010), 'A survey on transfer learning', *IEEE Trans. on Knowl. and Data Eng.* 22(10), 1345–1359.
- Pang, B. & Lee, L. (2008), 'Opinion mining and sentiment analysis', *Found. Trends Inf. Retr.* 2(1-2), 1–135.
- Phan, X.-H., Nguyen, L.-M. & Horiguchi, S. (2008), Learning to classify short and sparse text & web with hidden topics from large-scale data collections, in 'Proceedings of the 17th International Conference on World Wide Web', WWW '08, ACM, New York, NY, USA, pp. 91–100.
- Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Rozin, P. & Royzman, E. B. (2001), 'Negativity bias, negativity dominance, and contagion', *Personality and Social Psychology Review* 5(4), 296–320.
- Scheffer, T., Decomain, C. & Wrobel, S. (2001), Active hidden markov models for information extraction, in 'Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis', IDA '01, Springer-Verlag, pp. 309–318.
- Schwarz, G. (1978), 'Estimating the dimension of a model', *The Annals of Statistics* 6(2), 461–464.
- Scott, W. A. (1955), 'Reliability of content analysis: The case of nominal scale coding', *Public Opinion Quarterly* 19(3), 321–325.

- 
- Settles, B. (2009), Active learning literature survey, Computer Sciences Technical Report 1648, University of Wisconsin–Madison.
- Settles, B. (2012), *Active Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers.
- Shalev-Shwartz, S. (2012), ‘Online learning and online convex optimization’, *Found. Trends Mach. Learn.* **4**(2), 107–194.
- Smailović, J., Grčar, M., Lavrač, N. & Žnidaršič, M. (2014), ‘Stream-based active learning for sentiment analysis in the financial domain’, *Inf. Sci.* **285**(C), 181–203.
- Snow, R., O’Connor, B., Jurafsky, D. & Ng, A. Y. (2008), Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks, in ‘Proceedings of the Conference on Empirical Methods in Natural Language Processing’, EMNLP ’08, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 254–263.
- Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H. & Demirbas, M. (2010), Short text classification in twitter to improve information filtering, in ‘Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval’, SIGIR ’10, ACM, New York, NY, USA, pp. 841–842.
- Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S. & Tsujii, J. (2012), Brat: A web-based tool for nlp-assisted text annotation, in ‘Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics’, EACL ’12, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 102–107.
- Thisted, R. A. (1988), *Elements of Statistical Computing*, Chapman and Hall, New York.
- Tong, S. & Koller, D. (2002), ‘Support vector machine active learning with applications to text classification’, *J. Mach. Learn. Res.* **2**, 45–66.
- Toutanova, K., Klein, D., Manning, C. D. & Singer, Y. (2003), Feature-rich part-of-speech tagging with a cyclic dependency network, in ‘Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1’, NAACL ’03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 173–180.
- Whissell, C. & Charuk, K. (1985), ‘A dictionary of affect in language: Ii. word inclusion and additional validation’, *Perceptual and Motor Skills* **61**(1), 65–66.
- Yimam, S. M., Biemann, C., Eckart de Castilho, R. & Gurevych, I. (2014), Automatic annotation suggestions and custom annotation layers in webanno, in ‘Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations’, Association for Computational Linguistics, Baltimore, Maryland, pp. 91–96.
- Yimam, S. M., Gurevych, I., de Castilho, R. E. & Biemann, C. (2013), Webanno: A flexible, web-based and visually supported system for distributed annotations, in ‘Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics’, Association for Computational Linguistics, pp. 1–6.

---

# A *Annotate!* Documentation

---

## A.1 Configuration

---

---

### A.1.1 MySQL Configuration

---

Adjust MySQL with the following configuration. In the `mysql.conf` file under the `[mysqld]` section add the lines from code 5. In line 1-3 setup the cache to an appropriate size regarding the system at hand. Line 4 sets the character set to line `utf-8` and uses a maximum of four byte per character instead of just three with the regular `utf-8` setting. In line 5 the system variable `innodb_flush_log_at_trx_commit` is set to zero. This variable controls the balance between strict ACID compliance and higher performance. The value zero indicates a more performance orientated setting as log buffers are written to the log file approximately once per second. More information about this variable can be found in the MySQL Documentation. Line 6-7 configure MySQL to hold temporary files in memory instead of writing them to disk. This again is done to increase performance. This configuration is specific to Ubuntu and has to be adopted for any other operating system, yet it is also possible to leave out this configuration.

```
1 query_cache_size=1000000000
2 query_cache_type=1
3 query_cache_limit=80000000
4 character_set_server=utf8mb4
5 innodb_flush_log_at_trx_commit=0
6 tmpdir=/run/shm/mysqltemp
7 secure-file-priv=""
```

**Code 5: MySQL Configuration**

---

## A.2 Database Model

---

In this section the database model classes in the file `annotationApp/models.py` is described. The classes are used by Django's ORM to create their corresponding tables in the database. For the created tables in the database see Figure A.1.

---

### A.2.1 Annotation

---

This class holds all information regarding an annotation. For monitoring and evaluation purposes an annotation not only stores labels but also information about time, user, duration, etc.



---

document	ForeignKey	Realizes a ManyToOne relation from Annotation to Document. A document can have many annotations but an annotation can only belong to one document.
user	ForeignKey	Realizes a ManyToOne relation from Annotation to User. A user can create many annotations but an annotation can only belong to one user.
labels	ManyToManyField	In case of a multi-label annotation project an annotation needs to store more than one label. A label is more than a string see A.2.4 and can be referenced by many annotations.
proposals	ManyToManyField	Proposals are the suggested labels to the annotator. For them the same explanation is true as for the labels.
proposalFlag	CharField	This field stores which suggestion mode was used for the annotation. The three possible values are 'proposal', 'no proposal' and 'wrong proposal'
duration	CharField	The amount of milliseconds it took the annotator to submit the annotation.
dateTime	DateTimeField	Timestamp for when the annotation was added to the database.

---

### A.2.2 AnnotationQueue

---

Every annotation queue is assigned to a user. It stores the remaining annotation a user is supposed to do.

user	OneToOneField	It is mandatory to reference a user.
max_anno_num	IntegerField	The maximum number of annotations is stored here.

---

### A.2.3 Document

---

The Document class holds the document's id and its text. Additionally, information for the classifier are stored.

document	CharField	The document's text.
doc_id	CharField	The document's id, which has to be unique
preprocessed	CharField	On insertion documents are preprocessed and transformed into feature space. The result is stored here so it has not to be repeated every time a document is classified.
trainInstance	BooleanField	This flag indicated whether a document was added to be annotated, or solely to train the classifier.
active_prediction	ForeignKey	The predicted label from the last time this document was ranked is stored here.
margin	FloatField	The difference in probability for the most and second most probable label from the last ranking is stored here.
dateTime	DateTimeField	Date and time the document was added to the database.

---

## A.2.4 Label

---

A Label has its own class to store which kind of display option should be used.

label	CharField	The label's text as it is displayed in the application.
option	CharField	The display option which is either 'check' for check boxes or 'radio' for radio buttons.

---

## A.2.5 NBC\_class\_count

---

label	CharField
count	IntegerField
total_word_count	IntegerField

---

## A.2.6 NBC\_vocabulary

---

word	CharField	Holds the entire vocabulary
------	-----------	-----------------------------

---

## A.2.7 NBC\_word\_count\_given\_class

---

label	CharField
word	CharField
count	IntegerField

---

## A.2.8 QueueElement

---

A queue element encapsulates an annotation task. It realizes a many to many relationship between <AnnotationQueue> and <Document> and also stores additional information.

document	ForeignKey	A queue element references only one document but a document can be referenced by multiple queue elements.
queue	ForeignKey	The <AnnotationQueue> the element belongs to.
proposalFlag	CharField	This Field stores which suggestion mode will be used for the annotation.
rank	IntegerField	The position in the queue.

## A.2.9 Entity Relationship Model

The Entity Relationship Model as it was created in the database by Django’s ORM. Figure A.1 was created by MySQL Workbench.

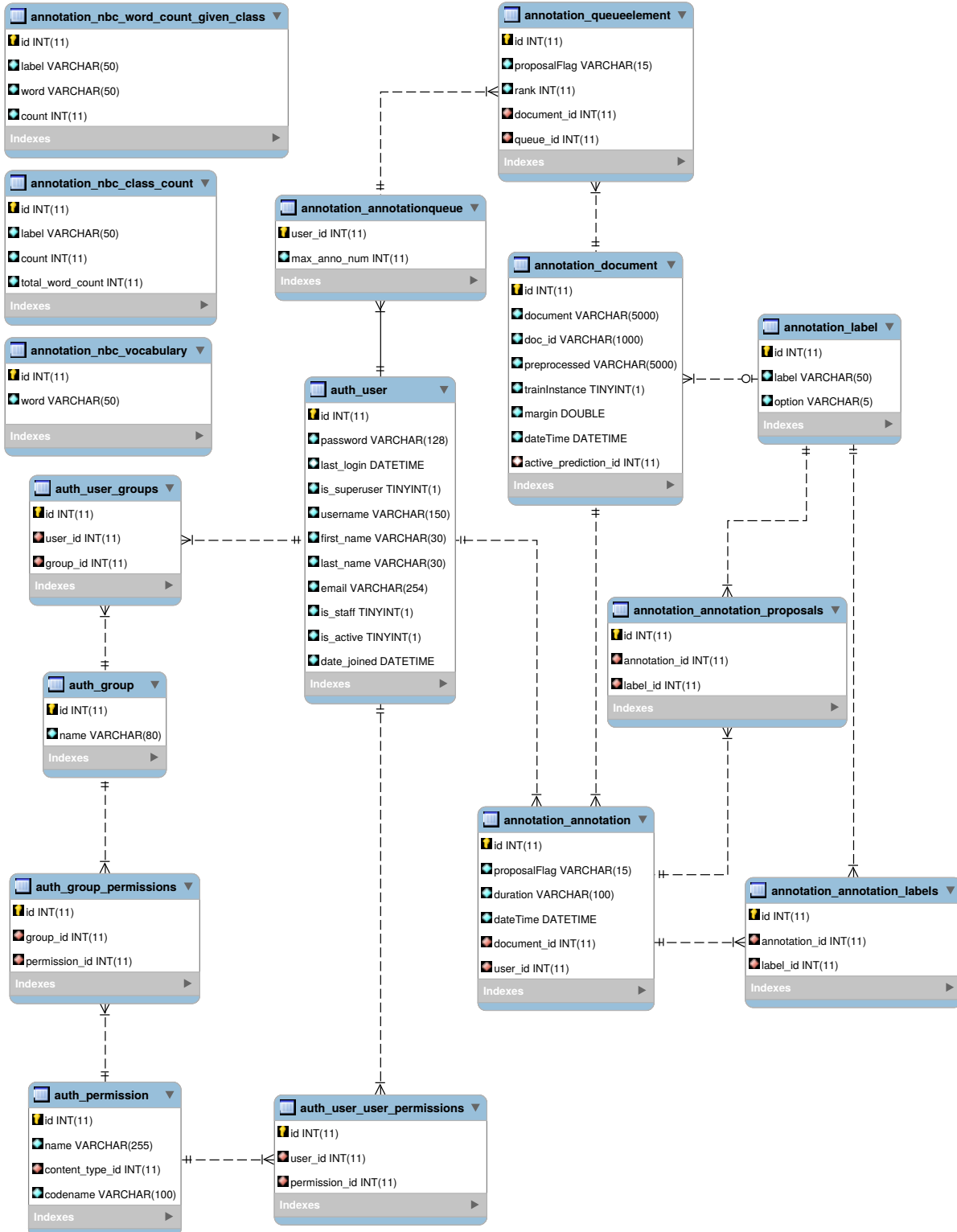


Figure A.1.: Entity Relationship Model Extended

---

## A.3 Commandline Tools

---

These scripts are a collection of tools for administrative tasks. All tools are called from within the projects root folder `annotationApp` with the command.

```
python manage.py <tool-name> [tool-args]
```

---

### A.3.1 addDocument

---

This command is used to read a file of documents and adds them to the database. A document file is expected to be a `.csv` file with document ids in the first column and document text in the second column. Newly added documents are preprocessed directly and transformed into feature space. Their feature space representation is saved in the database. See the database model A.2.3.

```
python manage.py addDocuments source delimiter maxTokenCount --train-instance
```

- `source` The path to either a `.csv` file or a directory of `.csv` files containing documents. Note that if `<source>` is a directory all files inside this directory are tried to process. The reason is to not be restricted to a file extension.
- `delimiter` Delimiter used in the `.csv` files.
- `maxTokenCount` The maximum number of tokens per document (white space tokenization). Additional tokens are ignored and not saved in the database. Type 'all' to use the entire document.
- `- -train-instance` This is an optional argument. If added all imported documents are flagged as training instances.

---

### A.3.2 addLabels

---

This command reads labels from a file and adds them to the database. The label file is expected to be organized as follows. Each line represents a label. The label's text is separated from the display option with an '@' character. Everything to the left of the '@' is considered label text. The display option is to the right. Use `check` for check boxes and `radio` for radio buttons.

Example row: `Positive@radio`

```
python manage.py addLabels source
```

- `source` Path to file containing labels.

---

### A.3.3 annotationQueueLoop

---

Ranking all unannotated documents according to the classifiers uncertainty takes to time to be done after every new annotation. The application will no longer be responsive. Therefore, ranking is done iteratively by placing this command in a cron table.

---

```
python manage.py annotationQueueLoop
```

---

### A.3.4 batchTrain

---

This command trains the classifier from a given file of examples. The file is expected to be a .csv file. Each row represents an instance, where the first row is the document id, the second row is the document text and the third row is the label.

```
python manage.py batchTrain source maxTokenCount
```

source	Path to file containing the training examples.
maxTokenCount	The maximum number of tokens per document (white space tokenization). Additional tokens are being cut off and not saved in the database. Type 'all' to use the entire document.

---

### A.3.5 createAnnotationQueue

---

This command ranks unannotated documents according to the classifiers uncertainty about them. The following cases are being distinguished. Documents that have no annotations are ranked according to the concept described in section 4.2. While documents that are partially annotated (meaning they have some annotations but not the number of annotations specified by maxAnno) are sorted ascending by their number of annotations. Afterwards, for every document  $k$  queue elements (see A.2.8) are created, where  $k$  is the number of missing annotations for this document. The queue elements are evenly distributed between the annotation queues (see A.2.2) of the annotators.

For evaluation purposes it is possible that not all annotations get a label suggested but also to get no suggestion or a wrong suggestion.

```
python manage.py createAnnotationQueue maxAnno noSuggestionNum wrongSuggestionNum  
noSuggestionFreq
```

maxAnno	The maximum number of annotations per document.
noSuggestionNum	Number of annotations per document that get no suggestion.
wrongSuggestionNum	Number of annotations per document that get a wrong suggestion.
noSuggestionFreq	This options expects 3 parameters which represent the frequency of documents that get no suggestion, a wrong suggestion and full suggestion. With the sequence 0. 2 0.3 0.5; documents have a 20% chance to get no suggestion where noSuggestionNum of the annotations have no suggestion, a 30% chance to get a wrong suggestion where wrongSuggestionNum of the annotations get a wrong suggestion and a 50% chance to get all suggestions. The three numbers have to sum up to one.

---

### A.3.6 createCurve

---

The learning progress of the classifier and the active learning strategy can be evaluated with this command. A training set is partitioned into learning steps. After the classifier is trained with the examples of a learning step its performance is evaluated on a test set. Then, the remaining examples are ranked with the active learning strategy and the next learning step is fed to the classifier and so on. For comparison the same process is performed without active learning. Every learning step can be repeated multiple times to account for randomness. The result of each learning step is saved to a file.

```
python manage.py createCurve trainFile testFile labelMap delimiter rep split
```

- trainFile Path to a .csv file containing the training examples. The first column is expected to hold the training instance, while the second column contains the respective label.
- testFile Path to .csv file containing the test examples. The same format as for the training file is expected.
- labelMap A file that maps the label representation from the train/test files to the label representation in the database. One mapping per line “training label representation tabular database representation”
- delimiter Delimiter used in the .csv file
- rep Number of repetitions of every learning step.
- split Number of training examples for each learning step.

---

### A.3.7 createGroup

---

This command creates the group Annotators in the database.

```
python manage.py createGroup
```

---

### A.3.8 createUser

---

Use this command to add annotators to the database. Users added via this command are automatically members of the Annotators group.

```
python manage.py createUser username password
```

- username Identifier of the user which has to be unique.
- password Initial password.

---

### A.3.9 evaluate

---

The current classifier’s performance is evaluated on a given test set.

```
python manage.py evaluate testfile delimiter
```

---

testfile Path to .csv file containing the test examples. The test file is expected to have the document's id in the first column and the document's text in the second column.  
delimiter Delimiter used in the .csv file

---

### A.3.10 exportAnnotation

---

For further processing of annotations use this command to export annotations from the database. The default export format is in .json see 4.

```
python manage.py exportAnnotation filename --incremental --tsv
```

filename Filename of the file to export in. (Without file extension)  
--incremental Optional argument. If this argument is passed the only those annotations of fully annotated documents are exported that have been added since the last call. This can be used in combination with `<createAnnotationQueue>` A.3.3.  
--tsv Optional argument. If added output format will be .tsv.

---

### A.3.11 predictLabel

---

Predicts the labels of a given test set with the current classifier.

```
python manage.py predictLabel testfile delimiter --save
```

testfile Filename of the test file. The test file is assumed to be a csv file with the document id in the first column and the document text in the second column.  
delimiter Delimiter used in the .csv file.  
--save Optional argument. Save labels to file.

---

### A.3.12 status

---

Creates an over the current annotation project. For an example output see Figure 5.6.

```
python manage.py status maxAnno
```

maxAnno The maximum number of annotations per document that was specified when calling `<createAnnotationQueue>`

---

### A.3.13 wipeDB

---

To clear an annotation project, the classifier's model or the entire content of the database this command can be used. Warning! If this command is called without any argument. All tables are being deleted.

---

```
python manage.py wipeDB exclude  
exclude Tables to exclude from deleting.
```