DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2018

Sentiment Analysis on Stack Overflow with Respect to Document Type and Programming Language

LI LING

SIMON LARSÉN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Sentiment Analysis on Stack Overflow with Respect to Document Type and Programming Language

LI LING
SIMON LARSÉN

Degree Project in Computer Science, DD142X
Date: June 6, 2018
Supervisor: Richard Glassey
Examiner: Örjan Ekeberg
Swedish title: Attitydanalys på Stack Overflow med avseende på dokumenttyp och programmeringsspråk
School of Electrical Engineering and Computer Science

Abstract

The sentiment expressed in software engineering (SE) texts has been shown to affect both the productivity and the quality of collaborative work. This is one reason for why sentiment analysis on SE texts has gained attention in research in recent yerars. A large and open resource of SE texts is Stack Overflow (SO). SO is the largest question and answer (Q&A) web site in the Stack Exchange network, and has been the subject for several sentiment analysis studies. It has lately been established that sentiment analyzers trained on social media perform poorly on SE texts, which could challenge the credibility of some of these studies. The Senti4SD sentiment polarity classifier was developed and trained on SO documents to address some of these issues. In this study, random samples of SO documents are drawn and then classified with Senti4SD. The classification into positive, negative and neutral sentiment is used to model the sentiment probability distributions of different document types on SO as a whole, as well as for the eight most popular programming languages. The results indicate that the sentiment of a document is correlated to both the document type and the associated programming language. Among the three sentiment classes, neutral sentiment dominates throughout all SO documents. However, the reliability of the results are reduced by concerns regarding the accuracy of Senti4SD, vaguely specified pre-processing steps and possibly varying classifier bias in different subdomains. In conclusion, further research on sentiment classifiers for SE is needed before any detailed comparative studies of this kind can yield reliable results.

Sammanfattning

Attityden i tekniska texter har visats påverka både produktivitet och kvalitet i det relaterade arbetet. Detta är en av anledningarna till att attitydanalys på sådana texter har blivit uppmärksammad de senaste åren. Stack Overflow (SO) är en fråge- och svar-webbsida för programmering, och en stor resurs till tekniska texter. SO har undersökts i flertalet studier på attitiydanalys. Nyligen har det dock framkommit att attitydanalysverktyg som tränats på sociala medier presterar dåligt på tekniska texter, vilket kan utmana trovärdigheten hos flera av dessa studier. Senti4SD är ett attitydanalysverktyg för klassificering av attitydpolaritet som tränats specifikt på dokument från SO i syfte att bättre klassificera tekniska exter. I denna studie plockas obundna slumpmässiga urval av SO-dokument som klassificeras med Senti4SD. Klassificeringen av dokument i "negativ", "neutral" och "positiv" attityd används för att modellera sannolikhetsfördelningen för attitydpolaritet hos olika dokumenttyper på SO i sin helhet. Vidare genomförs likadana modelleringar av attityden för dokument relaterade till vardera av de åtta mest populära programmeringsspråken på SO. Resultaten antyder att attityden i ett dokument är korrelerad till både dokumenttyp och relaterat programmeringsspråk, samt att neutral attityd dominerar. Resultatens tillförlitlighet minskas dock av osäkerheter kring Senti4SDs korrekthet, vagt specificerade förbehandlingssteg samt potentiellt varierande systematiska klassificeringsfel bland olika underdomäner. Sammanfattningsvis bör mer forskning genomföras på attitydanalysverktyg för tekniska texter innan denna typ av detaljerad jämförelsestudie kan ge pålitliga resultat.

Contents

1 Introduction

Stack Overflow (SO) is the largest question and answer (Q&A) website in the Stack Exchange network, and deals solely with programming questions. A programmer who is unable to solve a problem can post a question, which can then be answered by other programmers. Questions and answers (collectively referred to as *posts*) can be commented, commonly to ask for clarification of a question or to point out errors in an answer [1], [2]. Answering a question is a collaborative effort of the community, as answers are posted, commented and subsequently edited to pin down the solution(s). It is common for authors of questions and answers to edit their posts after receiving feedback in comments, but expert users can circumvent this collaborative aspect and directly edit other users' posts [3]. As of March 25th 2018, SO has more than 8.6 million users, 16 million asked questions and 24 million answers [4]. The large amount of information contained in the posts on SO makes it a valuable resource for beginning and professional programmers alike. Questions are marked with 1-5 *tags* specifying which fields they relate to [5]. For example, a question asking about linked lists in Python may be tagged with `python` and `linked-list`, making it easy to find questions on a per-field basis.

1.1 Sentiment on SO

Sentiment in the workplace has been shown to be an important factor for both productivity and quality in collaborative work [6], [7], [8]. Considering SO as a collaborative place of work, the sentiment expressed in posts and comments could affect the efficiency of the site. Previous research has shown that user interaction on SO is not conducted in an entirely neutral fashion, but rather contains notable amounts of both positive and negative sentiment [9], [10], [11]. There is also evidence to suggest that SO is sometimes perceived as a hostile environment, as indicated by several notable Stack Exchange posts [12], [13], [14], the 2014-15 overhaul of the Stack Exchange *Be Nice* policy [15] and a recent post on the official Stack Overflow blog [16].

Previous research indicates that comments on SO are more likely to contain negative sentiment than questions and answers [11]. However, little effort seems to have been put toward investigating the general distribution of sentiment among the three primary SO document types (questions, answers, comments). This leads to the first research question of this study:

> RQ1: Does sentiment expressed in SO documents depend on document type?

Because of SO's size, there are many sub-communities within the site, and sentiment distribution could differ among these. A natural categorization into sub-communities is by programming language. A study on GitHub commit messages found that sentiment differed among programming languages [17]. Another study conducted on SO posts showed that politeness (which is related to sentiment) was also dependent on programming language [18]. This leads to the second and final research question of this study:

> RQ2: Does sentiment expressed in SO documents depend on the associated programming language?

In this study, we use the Senti4SD sentiment polarity classifier[1] to classify a total of almost 800,000 randomly sampled questions, answers and comments from 2017, gathered from the Stack Exchange Data Dump [19]. The results are then used to produce estimations

[1] Senti4SD was developed by Calefato et. al as described in [10], and is freely available on GitHub at https://github.com/collab-uniba/Senti4SD

of the sentiment probability distributions of different document types for SO as a whole, as well as for the 8 most popular programming languages.

1.2 Report Overview

The basic theory of natural language processing and sentiment analysis is presented in Section 2, along with a general overview of three sentiment analyzers specifically designed for analyzing software engineering (SE) texts. Related works relevant to this study are presented in Section 3. Because of the large amount of research that has been conducted on sentiment analysis in recent years, Section 2 and Section 3 have some slight overlap in terms of content. Section 4 presents details of how this study was conducted, including data collection and processing as well as statistical approach. Section 5 presents the results of the analysis. A critical analysis of possible flaws in this study is presented in Section 6, followed by a discussion of the results and thoughts about future works in Section 7. Finally, our conclusions are presented in Section 8.

2 Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is the extraction of meaning from human language by way of computation [20]. In the domain of spoken language, NLP comes in the form of *automatic speech recognition* (ASR) [21]. Commercial applications have already come a long way in this field [22]. However, it is still an active field of research, especially in terms of developing neural network-based solutions [23, pp. 1–9].

NLP as been a field of research for written language since the 1950's [22]. Possible applications are many, including text-to-speech, automatic translation and information retrieval. NLP has historically been approached in one of two ways: with our without statistics (including machine learning) [20]. Because of hardware limitations, early approaches primarily consisted of hand-crafted, rule-based solutions. However, the complexity of natural languages heavily challenged pure rule-based systems [22]. Statistical methods were later adopted to improve the performance of NLP for written language, and in this, the ASR community was the driving force [24]. One interesting subfield of NLP for written language, which is also the main focus of this report, is *sentiment analysis* of written text.

2.2 Sentiment Analysis

The increased popularity of machine learning methods and the large datasets provided by the World Wide Web in the 21st century have led to a boost in research related to sentiment analysis. Sentiment analysis, or opinion mining, is the study of opinion using computational methods [25]. More formally, it can be defined as the discovery of sentiment expressed by an opinion holder towards some aspect of an entity in an opinionated document [26], [27]. Document-level sentiment classification treats a multi-sentence document as a single entity, while sentence-level sentiment classification classifies sentiment on a per-sentence basis [25]. Sentiment can be classified based on its polarity (positive, negative or neutral), quantized according to some given scale or categorized into different emotions such as anger and joy [26], [28]. There are many possible application areas for sentiment analysis, including product review understanding, recommendation systems and question answering services [25].

Sentiment analysis methods can be broadly divided into two categories: the *machine learning approach* and the *lexical-based approach* [28]. The machine learning approach can be further divided into supervised and unsupervised methods, with the former requiring labeled training data guiding the classification, and the latter discovering groups of similar data without the provision of ground-truth [29]. The challenges of supervised learning methods include acquiring labeled data, feature engineering and selection of classification algorithms [30], [31]. Some of the widely considered features include presence of words, word frequencies, part of speech tagging, opinion words and phrases, negations and n-grams [25], [27], [31]. For classification algorithms, support vector machines and naïve Bayes classifiers have been shown to be effective in single domain document sentiment classification [31], [32]. Despite the ability of supervised methods to adapt to the domain of the training data, these methods suffer a few shortcomings. Firstly, apart from product reviews with user-provided rating, gathering labeled training data may be costly [33]. Secondly, multiple studies have shown that classifiers trained in one domain perform poorly when applied to other domains [25, p. 40], [32, p. 2], [27], [34], [35]. When the amount of labeled training data is insufficient, unsupervised methods can be used to cluster documents based on a selected feature set [36].

Lexical-based methods use a pre-defined list of words and n-grams tagged with associated sentiment and classify documents based on the frequency or presence of sentiment words [28]. Examples of lexical-based methods include Linguistic Inquiry and Word Count (LIWC) [37], Emolex [38] and VADER [39]. Manual labeling, dictionary-based approaches and corpus-based machine learning approaches have all been used for the development of opinion lexicons. Dictionary-based approaches build a set of opinion words by finding the transitive closure of synonyms and antonyms of some initial set of words. This is commonly performed using an online lexical database, such as WordNet. Corpus-based approaches also grow a set of words from some initial set, but use a domain specific corpora and a set of linguistic constraints instead of a lexical database. As a result, the latter is better at capturing domain specific information [27]. It is worth noting that lexical-based and machine learning methods can be combined, as exemplified by Melville et al. [40]. Several sentiment analyzers using machine learning methods also incorporate lexical dictionaries. Examples of such analyzers include SentiStrength [32], [41], Stanford recursive deep model [42] and SentiWordNet [43], [44].

2.3 Sentiment Analyzers

A comprehensive study on 24 published sentiment analyzers demonstrates that sentiment classifiers achieve different prediction performance when used to classify different datasets. Additionally, it shows that sentiment analyzers may not agree with one another, i.e. the choice of analyzers can lead to significantly different classification results [28]. In sentiment analysis for SE documents, SentiStrength is the most frequently used analyzer [45], [46]. SentiStrength is based on a lexicon of sentiment words, including words with non-standard spellings (e.g. "lol"), and was developed for sentiment classification of short, informal texts. To optimize the sentiment term weights, machine learning was utilized on a set of documents from MySpace, an online social network. SentiStrength analyzes sentiment by assigning both positive and negative scores to sentiment words in documents [32]. To improve the classification of negative sentiment, SentiStrength 2 was developed by extending the negative terms in the SentiStrength lexicon [41].

As mentioned in Section 2.2, analyzers and lexicons developed using dataset in one domain perform poorly in other domains, and the SE domain is no exception. Domain-specific lexicons in SE, such as "bugs", "kill" and "miss", can trigger false positives in negative sentiment classification [47], [48], [46]. Further, the question answering nature of SO implies that discussions intrinsically contain more negative vocabulary [48]. When used for sentiment classification on SE documents, existing analyzers trained on social media datasets achieve results that disagree both with each other, and with manual labeling. Recent studies show that this disagreement can seriously challenge previously published results [45], [46]. The above issues motivated the development of SE specific sentiment analyzers. As of 2017, at least three such sentiment analyzers have been developed, namely SentiStrength-SE [47], Senti4SD [10], and SentiCR [49]. A recent benchmark study suggests that the three mentioned SE analyzers outperform SentiStrength when used for sentiment analysis of SE texts [50].

2.3.1 SentiStrength-SE

Published in 2017, the SentiStrength-SE study was the first study to use a public benchmark dataset to identify challenges in sentiment analysis of SE texts. By manually analyzing the classification results of SentiStrength on 392 comments from the JIRA issue tracking system, the researchers identified 12 difficulties faced by SentiStrength, including misclassification of domain-specific words and sentimental words in code-snippets, stack traces and urls. SentiStrength-SE is an analyzer developed to address these difficulties. SentiStrength-SE incorporates a domain-specific dictionary on top of SentiStrengh, includes contextual information using part-of-speech tagging, adds neutralizer words as well as incorporates a preprocessing process. When used to classify more JIRA comments, Islam et al. claimed that SentiStrength-SE outperformed SentiStrength [47].

2.3.2 SentiCR

SentiCR is a sentiment analyzer for code review comments trained with supervised learning. It was trained on 2000 manually labelled comments that were randomly sampled from Gerrit. Similar to SentiStrength-SE, SentiCR incorporates a data preprocessing phase, where urls, stop-words and code snippets are removed from the input text, and other preprocessing such as emoticon handling, negation preprocessing and word stemming is also performed [49]. Term frequency and inverse document frequency (tf-idf) weighting [51] is utilized as the feature in SentiCR. By comparing the performance of different supervised learning methods, Ahmed et al. found that a gradient boosting tree achieved the highest accuracy [49].

2.3.3 Senti4SD

Senti4SD is a sentiment analyzer for SE texts trained with support vector machines. The training data consisted of approximately 4000 SO questions, answers and comments annotated using a model-driven approach [52]. For each input document, Senti4SD extracts a total of 76,369 features. These features fall into three categories: lexicon-based, keyword-based and semantic features. The SentiStrength sentiment lexicon is used to extract the 19 lexicon-based features. Examples of such features include the number of words in the input document with positive polarity, and the sentiment score of the last negative word in the input document. Keyword-based features are related to the existence and frequency of words in the input document, and account for the majority of the features. More specifically, keyword-based features consist of 10,496 unigram features, 65,844 bigram features and 6 other features, such as the existence of user mentions and uppercase words. Lastly, Senti4SD contains a distributed semantic model (DSM) trained with 20 million preprocessed SO documents. In the DSM, every *linguistic item* is represented as a vector in a high dimensional space. A linguistic item can be for example a word, a sentence, or an entire document of sentences, where the vector representation of a unit larger than a word is simply the vector sum of the words. The distance between two vectors is smaller for items that occur in similar contexts, and larger for items that occur in disparate contexts. Senti4SD computes the semantic features of a document as the distance between its vector representation and four prototype DSM vectors. The prototypes represent the three polarities and subjectivity, and were computed as the vector sums of words with the respective polarities according to the SentiStrength lexicon [10]. According to Calefato et al., Senti4SD outperformed SentiStrength, SentiStrength-SE and SentiCR in classifying SO documents [10], [50].

3 Related works

Sentiment analysis of written text has gained attention in the research community during the past ten years. Focus has mostly been put on analyzing short informal texts in social media [53], [32], [54] and evaluating opinions from product reviews [55], [56], [57], [25]. For example, Kucuktunc et al. investigated the relationships between sentiment of texts on a large online question answering site and multiple other factors, including gender, age, education level of the asker and answerer, the time of the day, etc. They also found a strong correlation between the sentiment of a question and the sentiment in answers to that question [58]. More recently, there has been an increase in research on sentiment analysis applied to the SE domain. To improve the emotional awareness within software development teams, Guzman et al. used topic modelling and sentiment analysis to analyze collaboration artifacts in software development projects, such as commit messages, emails and twitter messages [7]. Attempting to identify variables affecting the sentiment in commit comments in open source software projects, Guzman et al. used SentiStrength to measure the sentiment strength of commit comments in 29 open source projects hosted in Github. By modelling the sentiment against multiple variables, they found that negative sentiment is most prevalent in Java projects and commit comments posted on Mondays, while positive sentiment is most prevalent in more distributed teams [17].

Being a large resource for SE information, SO is also a popular subject for research, and has been since the early years of its operation. An early study investigated the common question types and question answering mechanism on SO. It was found that SO is particularly efficient at answering questions related to code reviews, explanation of conceptual issues as well as how-to questions posted by newcomers [59]. More recently, a few studies have tried to capture the factors affecting the successfulness of SO questions and answers. By randomly sampling and manually analyzing 400 unanswered questions, Asaduzzaman et al. concluded that the top five characteristics of unanswered SO questions include incorrect tagging, unclear question formulation, question duplicates, harsh sentiment in questions or comments and too specialized technology [60]. Building on Asaduzzaman et al. [60], Chua et al. further studied the answerability of SO questions. They found that the asker's popularity, participation and the time of posting influenced the answer rate. The specificity, clarity, level of detail and socio-emotional value of the question were also found to be important predictors for whether a question will be answered or not. Interestingly, impoliteness in questions was found to attract answers, rather than serve as a deterrent [1]. In contradiction to Treude et al. [59], Chua et al. concluded that short questions without code snippets are more likely to receive answers [1]. Calefato et al. investigated the success (upvotes) of answers to questions, using SentiStrength for sentiment analysis. Positive sentiment was found to significantly increase the success of the answer, while the opposite applies to negative sentiment. Reputation of the answerer, inclusion of code snippets and contextual information in the form of urls were also found to positively correlate to the success of an answer [9]. The same authors conducted a similar investigation on SO questions in an effort to develop question-asking guidelines. They found that successful SO questions are typically short, contain code snippets and are predominantly neutral in sentiment [11]. In another study, Calefato et al. used SentiStrength to classify SO questions, answers and comments and found that sentiment expressed in different document types may differ in their intent. They argued that more fine-grained affective state analysis than polarity classification is needed to fully understand the sentiment on SO. Notably, they found that negative sentiment does not imply impoliteness or rudeness toward others. Rather, in questions and answers, the negative sentiment is frequently used to show empathy or to express negative emotions toward a technical issue. In comments, however, sentiment is more prevalent and negative sentiment may be directed to the question asker [48]. It should be noted that these findings were gathered by manual analysis of the results, and not by an automated process. Indeed, finding the cause of an emotion is a difficult task in sentiment analysis, as the cause may precede the affective sentence, or could be implied by the context [61, p. 33]. Danescu-Niculescu-Mizil et al. studied politeness on SO and found that question authors are more polite than answerers, and that users with higher reputation are more impolite. Further, the researchers compared politeness of different programming language communities on SO and found Ruby to be most polite, followed by JavaScript [18]. Note that, although related, politeness and sentiment are not equivalent.

4 Method

The study was conducted in five distinct stages. For the sake of readability, the steps are not presented in chronological order. Table 1 lists the actual chronological order of the steps.

Table 1: Chronological order of this study.

Step	Described in
1. Data collection	Section 4.2.1
2. Sampling for baseline	Section 4.2.2
3. Analysis of baseline samples	Section 4.4
4. Sampling per-programming language	Section 4.2.2
5. Analysis of per-programming language samples	Section 4.4

4.1 Baseline and Programming Languages

The comparative analysis consisted of two primary parts: the baseline analysis and the per-programming language analysis. The baseline analysis was conducted on SO as a whole on a per-document type basis. In other words, samples were separately drawn from all questions, all answers and all comments. Establishing a baseline had a two-fold purpose. First, it provided a basis for comparison as the "norm" for SO as a whole. Second, the results were needed to answer RQ1, which was imperative to justifying whether or not to treat each document type separately in the per-programming language analysis. The sub-communities were selected as the eight most popular programming languages from the 2017 SO developer survey, listed in Table 2 [62]. Note that there is no direct correlation between the popularity in the survey, and the amount of posts on SO. For example, SQL was the second most popular technology, but ranked 7 in terms of post count, trailed only by C.

Table 2: Programming languages ranked by popularity in the 2017 SO developer survey.

Rank	Language
1	JavaScript
2	SQL
3	Java
4	C#
5	Python
6	PHP
7	C++
8	C

As mentioned in Section 1, a question is tagged with 1-5 tags to mark the fields to which it relates. Any answer to that question, or comments to either answers or question, implicitly share the tag. A document was only considered to belong to one of the programming languages if it was tagged with precisely one of the considered languages. In other words, the sub-communities examined in this report are all mutually exclusive.

4.2 Data

4.2.1 Data Collection

Stack Exchange provides a data dump of all user contributed content, which is updated every quarter. The data dump for each website is separated into multiple zipped XML files based on the content, with each XML file representing a database table [19]. In this study, only the XML files for posts and comments were used. Table 3 lists the requirements by which data was gathered from the dump.

Table 3: Data requirements.

Document type	Requirements
Question	• Posted between 2017-01-01 and 2017-12-03.
Answer	• Posted between 2017-01-01 and 2017-12-03. • Refers to an already collected question.
Comment	• Posted between 2017-01-01 and 2017-12-03. • Refers to an already collected post.

The data was migrated from the XML files into a relational database in three passes, in the same order as the document types appear in Table 3. The text preprocessing detailed in Section 4.3 was interleaved with the data migration, such that the relational database contained only sanitized text.

4.2.2 Sampling

Simple random sampling was used to draw samples from each population. Each document type was sampled individually within the baseline and the programming languages. That is to say, every `(category, document type)` pair was treated as a population, where category denotes either the baseline or a programming language, and document type denotes one of the types listed in Table 3.

For the baseline, sample sizes were calcualted with Cochran's sample size formula. The formula is defined as

$$n_0 = \frac{t^2 \times p \times (1-p)}{d^2} \tag{1}$$

where t is the t-value at a chosen confidence level, $p \times (1-p)$ is an estimate of variance and d is the acceptable margin of error. The confidence level was set to 95%. The value p was chosen to be 0.5, as this gives the maximum estimate of variance, representing a worst-case scenario [63]. The accepted margin of error d was chosen to be 1%. Plugging these values into Equation 1 yields a sample size of 9604. Due to time constraints, the sampling and subsequent analysis (Section 4.4) could only be repeated 15 times.

The per-programming language sampling was performed in the same fashion. The sample size was arbitrarily set to 1000, and sampling and analysis was repeated 15 times for each language and document type. Choosing a smaller sample size was again due to time constraints.

4.2.3 Data Description

The subset of data attributes from the Posts table that are considered in this study are presented in Table 4, while the relevant Comments table attributes are presented in Table 5.

Table 4: Relevant SO post attributes.

Attribute	Description
Id	Unique identifier for the post.
PostTypeId	1 denotes a question. 2 denotes an answer. 3-7 denote post types that are not considered in this study.
ParentId	Only answers contain this attribute. Indicates the Id of the associated question.
CreationDate	Creation date of the post.
Body	Content of the post.
Title	The title of a question. Only questions contain this attribute.
Tags	Denotes technologies/fields to which a question is related, e.g. "python", "java". Only questions contain this attribute.

Table 5: Relevant SO comment attributes.

Attribute	Description
Id	Unique identifier for the comment.
PostId	Id of the associated post.
CreationDate	Creation date of the comment.
Text	Content of the comment.

It is important to note that there are other post types in the data dump than the examined question and answer types. The comments on some data attributes saying they may be found only on some specific type of post are made from the perspective that question and answer are the only post types. These attributes may or may not be found on other post types.

4.3 Preprocessing

Before being classified by Senti4SD, documents need to be sanitized and formatted. Text that may interfere with the classification, such as code snippets and urls, should be removed, and each document must be contained on a single line. The preprocessing steps performed in this study differed slightly between comments and posts, and are detailed in Section 4.3.2 and Section 4.3.3, respectively.

4.3.1 General Preprocessing

The following general preprocessing algorithm was performed on all documents.

1. Remove the bodies of all `<pre>`, `<code>` and `<blockquote>` tags.
2. Extract all text (excluding any HTML tags).
3. Replace every sequence of consecutive whitespace (such as spaces, tabs and line feeds) with a single space character each.
4. Remove all `http` and `https` urls.
5. Strip leading and trailing whitespace.

4.3.2 Comment Preprocessing

Comments were sanitized according to the following steps:

1. Parse comments from Markdown to HTML.
2. Perform the general preprocessing steps detailed in Section 4.3.1.

Listing 1 shows what an SO comment could look like, and Listing 2 shows the same comment after preprocessing. Note how the `**` characters (Markdown syntax for **bold** text) and the code snippet have been removed.

Listing 1: Fictional example of an SO comment.

```
could you fix the **indentation**? Also, don't you mean `for i in range
    (10)`?
```

Listing 2: Fictional SO comment after preprocessing.

```
could you fix the indentation? Also, don't you mean ?
```

4.3.3 Post Preprocessing

Posts were sanitized according to the following steps:

1. Perform the general preprocessing steps detailed in Section 4.3.1.
2. Remove any code snippets enclosed in single or triple backticks.

Listing 3 shows an example of an actual SO answer,[2] and Listing 4 shows the same post after preprocessing. Note how all text inside of `<code>` and `<pre>` tags is removed along with the tags, while text confined in e.g. `<p>` tags is not.

Listing 3: SO answer.

```
<p>You can do this in just two lines.</p>

<pre><code>with open('path/to/file') as f:
    line_lists = [list(line.strip()) for line in f]
</code></pre>

<p><code>list</code> on a <code>str</code> object will return a list where each character is an element. <code>line</code> is stripped first, which removes leading and trailing whitespace. This is assuming that you actually want the characters as <code>char</code>. If you want them parsed to <code>int</code>, this will work:</p>

<pre><code>with open('path/to/file') as f:
    line_lists = [[int(x) for x in line.strip()] for line in f]
</code></pre>

<p>Mind you that there should be some error checking here, the above example will crash if any of the characters cannot be parsed to int.</p>
```

Listing 4: SO answer after preprocessing.

```
You can do this in just two lines. on a object will return a list where each character is an element. is stripped first, which removes leading and trailing whitespace. This is assuming that you actually want the characters as . If you want them parsed to , this will work: Mind you that there should be some error checking here, the above example will crash if any of the characters cannot be parsed to int.
```

[2]Post url: https://stackoverflow.com/questions/48946549/import-text-in-multi-dimensional-array-in-python/48947161#48947161
Author url: https://stackoverflow.com/users/8891266/slarse

4.4 Analysis

Among the three SE specific analyzers presented in Section 2.3, Senti4SD is the only one that was trained on SO data. As sentiment analyzers are domain specific, Senti4SD (described in Section 2.3.3) was deemed the most appropriate analyzer for this study. Senti4SD takes a file as input and produces a CSV file with classifications as output. Each line in the input file is treated as a separate document, and the output file contains a column with line numbers and a column with the corresponding classifications.

The sentiment probability distribution for a given population was modelled as a categorical variable with three classes, namely “positive”, “negative” and “neutral”. 15 samples were drawn from each of the populations described in Section 4.2.2, with each sample consisting of between 1000 and 9604 documents. For each sample, the *sample estimates* of the sentiment probabilities were calculated. For a given class c, the sample estimate is given by

$$p_i(c) = \frac{a_{ic}}{n_i} \tag{2}$$

where i denotes the sample number (between 1 and 15 in this study), a_{ic} is the amount of documents classified as class c by Senti4SD, and n_i is the sample size [63]. For each population, the mean of $p_i(c)$ of the samples, given by

$$p(\bar{c}) = \frac{\sum_{i=1}^{N} p_i(c)}{N} \tag{3}$$

is used as the sentiment probability estimate of the population, where N denotes the number of samples drawn from the population (15 in this study). For each probability estimation, the standard deviation is given by

$$\sigma(p(\bar{c})) = \sqrt{\frac{\sum_{i=1}^{N} (p_i(c) - p(\bar{c}))^2}{N-1}} \tag{4}$$

where N again refers to the number of samples [63].

4.5 Software

This study was made possible by various pieces of open source software, which were combined to form a toolchain for extracting data from the SO data dump, as well as sanitizing and analyzing it. The toolchain is publicly available at https://github.com/li-simon-collab/so-sentiment-toolchain, but its dependencies need to be installed separately.

The purpose of this section is both to provide attribution to the software that was used, as well as to detail some specific usage. While this kind of information would normally be confined in an appendix, we found some of the usage to be relevant to the discussion and consider proper attribution to software that was crucial to the study to be of high importance.

4.5.1 Senti4SD

As mentioned in Section 4.4, Senti4SD was the sentiment analyzer used for this study. Early on, three problems were discovered, two of which required non-trivial solutions for efficient usage.

1. Classifying files with several thousands of documents (lines) caused the machine to run out of physical memory, and the Java virtual machine instance terminated.
2. Senti4SD only utilizes a single core, and there was no obvious way to enable multicore processing. Simply running several instances of Senti4SD in parallel proved difficult as well, as the program stored intermediate results in a local file. This caused parallel processes to overwrite each others' intermediate results.
3. Running Senti4SD with `OpenJDK 8.u172-2` (Java 8) caused memory usage in the 10s of gigabytes within the first few seconds of operation.

The solution to the first problem was to split large files into smaller subfiles to be classified independently, and then concatenate the results. For the second problem, the only solution that was found to work reliably was to keep several copies of the Senti4SD software on the test machine, making sure to only run a single instance of each copy at a time. The third and final problem was solved by simply running Senti4SD with `OpenJDK 9.0.4.u11-1` (Java 9), which drastically reduced memory usage.

4.5.2 Additional Software and Runtime Environment

The toolchain was developed with `Python 3.6`, and depends on several open source packages. The packages are listed in Table 6.

Table 6: Python packages used in this study.

Library	Used for	Project url
SQLAlchemy	Database abstraction layer	https://www.sqlalchemy.org/
BeautifulSoup	Data preprocessing	https://www.crummy.com/software/BeautifulSoup
Python Markdown	Data preprocessing	https://github.com/Python-Markdown/markdown
Maya	Date string parsing	https://github.com/kennethreitz/maya
Pandas	Statistics	https://pandas.pydata.org/
SciPy	Statistics	https://www.scipy.org/
NumPy	Statistics	https://www.numpy.org/
matplotlib	Plotting	https://matplotlib.org/
seaborn	Plotting	https://seaborn.pydata.org/
Connector/Python[3]	Database connection	https://dev.mysql.com/downloads/connector/python/

[3]The database driver is configured via an environment variable, and the toolchain does not depend upon 'Connector/Python' specifically.

Table 7 lists the software that composed the runtime environment. The results of this study should be reproducible with later versions of the listed software, and possibly also with previous version and entirely different implementations.

Table 7: Runtime environment used in this study.

Software	Used for	Project url
Python 3.6.5	Python runtime	https://www.python.org/
R version 3.5.0	R runtime	https://www.r-project.org/
OpenJDK 9.0.4.u11-1	Java runtime	http://openjdk.java.net/
Arch Linux (kernel 4.16.9-1)	Operating system	https://www.archlinux.org/
MariaDB 10.1.33	Persistent storage	https://mariadb.org/

5 Results

The results are divided into four parts. The results from the baseline analysis are presented in Section 5.1. The following three sections detail the results of the per-programming language analysis of questions (Section 5.2), answers (Section 5.3) and comments (Section 5.4). Each part is structured in the same way. First, populations and sample sizes are presented. Then, the sentiment probability distribution estimation is presented in a table and a diagram. Due to round off errors, the sentiment probabilities do not always sum to 1. Note also that when two probabilities are compared, the largest of the two standard deviations is used.

5.1 Baseline

The baseline establishes the basis for comparison in the study. The populations and sample sizes for each of the three document types are presented in Table 8.

Table 8: Population and sample size for the baseline.

Document type	Population	Sample size
Questions	2331403	9604
Answers	2395077	9604
Comments	8196263	9604

The sentiment probability distribution estimates for SO questions, answers and comments are shown below in Table 9. The same results are presented in Figure 1. The full classification results from which the probabilities were calculated can be found in Appendix A.

Table 9: Sentiment probability distribution estimates for SO questions, answers and comments. Parenthesized values are the standard deviations of the samples.

Document type	p(neg)	p(neu)	p(pos)
Questions	0.3212 (0.0036)	0.4543 (0.0042)	0.2244 (0.0047)
Answers	0.0538 (0.0018)	0.7941 (0.0033)	0.1521 (0.0036)
Comments	0.0830 (0.0024)	0.6975 (0.0040)	0.2195 (0.0029)

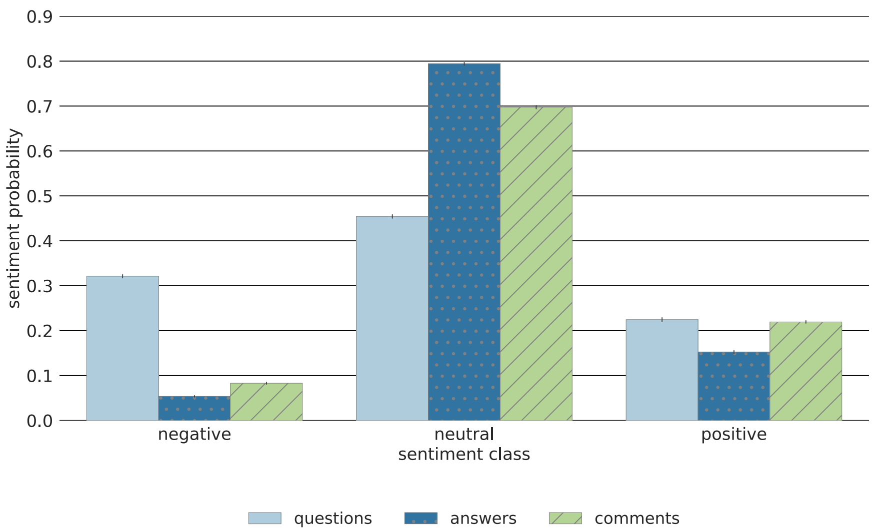

Figure 1: Sentiment probability distribution estimation for SO questions, answers and comments. Error bars represent ±1 standard deviation.

As can be seen in Table 9 and Figure 1, the sentiment probability distribution of SO questions differs significantly from that of answers and comments. Compared to answers and comments, questions appear to be considerably more negative and less neutral. Both $p_{question}(neg)$ and $p_{question}(neu)$ are separated from the corresponding probabilities for answers and comments by hundreds of standard deviations. The sentiment probability distributions for answers and comments are more alike one another. Both have $p(neu) > p(pos) > p(neg)$, and each probability is separated from the other two probabilities in the same document type by at least 27 standard deviations.

5.2 Sentiment of Questions per Programming Language

Table 10 lists populations and sample sizes for questions with respect to the baseline and each of the programming languages.

Table 10: Populations and sample sizes for questions.

Programming community	Population	Sample size
Baseline	2331403	9604
C	27815	1000
C#	132268	1000
C++	59073	1000
Java	182566	1000
JavaScript	242511	1000
PHP	131344	1000
Python	193557	1000
SQL	45678	1000

The sentiment probability distribution estimates for SO questions with respect to the baseline and each of the programming languages are shown below in Table 11. The same results are presented in Figure 2. The full classification results from which the probabilities were calculated can be found in Appendix B.

Table 11: Sentiment probability distribution estimates for SO questions. Parenthesized values are the standard deviations of the samples.

Programming language	p(neg)	p(neu)	p(pos)
Baseline	0.3212 (0.0036)	0.4543 (0.0042)	0.2244 (0.0047)
C	0.3717 (0.0136)	0.4456 (0.0170)	0.1827 (0.0122)
C#	0.3448 (0.0148)	0.4365 (0.0136)	0.2187 (0.0103)
C++	0.3931 (0.0173)	0.4368 (0.0185)	0.1701 (0.0172)
Java	0.3447 (0.0141)	0.4440 (0.0180)	0.2113 (0.0102)
JavaScript	0.3227 (0.0194)	0.4406 (0.0232)	0.2367 (0.0171)
PHP	0.3457 (0.0227)	0.4075 (0.0180)	0.2469 (0.0096)
Python	0.3228 (0.0167)	0.4572 (0.0197)	0.2200 (0.0101)
SQL	0.2496 (0.0160)	0.4692 (0.0147)	0.2812 (0.0151)

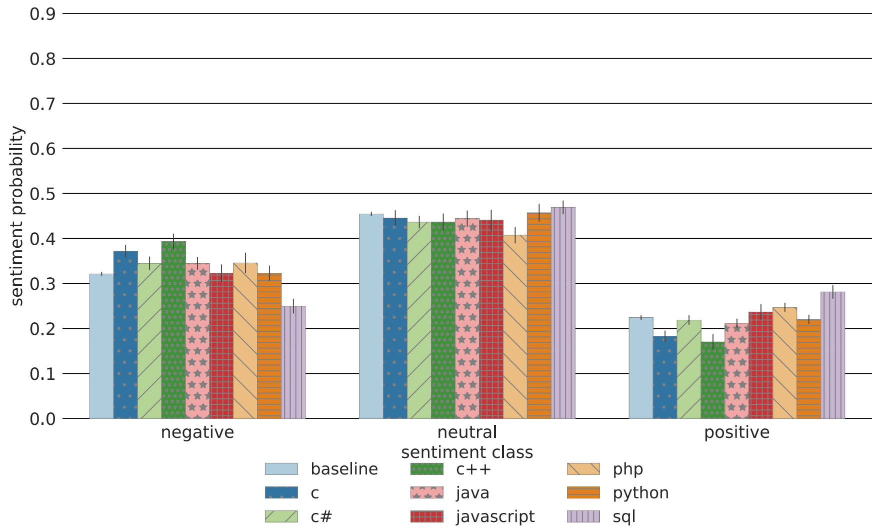

Figure 2: Sentiment probability distribution estimation for SO questions. Error bars represent ± 1 standard deviation.

The programming language results show standard deviations that are in general an order of magnitude larger than those of the baseline. Most of the probabilities within each sentiment class are separated by less than two standard deviations. A majority of the languages follow the same basic trend as the baseline does, in that $p(neu) > p(neg) > p(pos)$ by several standard deviations. There are a few exceptions, such as C++ and PHP having $p(neu)$ and $p(neg)$ separated by only 2-3 standard deviations. They do however still follow the baseline's general pattern. The only language to actually break the pattern is SQL, for which $p(neg) < p(pos)$, although only by two standard deviations.

5.3 Sentiment of Answers per Programming Language

Table 12 lists populations and sample sizes for answers with respect to the baseline and each of the programming languages.

Table 12: Population and sample size for answers.

Programming community	Population	Sample size
Baseline	2395077	9604
C	39089	1000
C#	140026	1000
C++	68397	1000
Java	199990	1000
JavaScript	294378	1000
PHP	143827	1000
Python	221830	1000
SQL	69080	1000

The sentiment probability distribution estimates for SO answers with respect to the baseline and each of the programming languages are shown below in Table 13. The same results are presented in Figure 3. The full classification results from which the probabilities were calculated can be found in Appendix C.

Table 13: Sentiment probability distribution estimates for SO answers. Parenthesized values are the standard deviations of the samples.

Programming language	p(neg)	p(neu)	p(pos)
Baseline	0.0538 (0.0018)	0.7941 (0.0033)	0.1521 (0.0036)
C	0.0845 (0.0083)	0.7829 (0.0152)	0.1326 (0.0123)
C#	0.0571 (0.0063)	0.7828 (0.0124)	0.1601 (0.0119)
C++	0.0885 (0.0094)	0.7902 (0.0106)	0.1213 (0.0090)
Java	0.0577 (0.0057)	0.7983 (0.0092)	0.1441 (0.0080)
JavaScript	0.0413 (0.0058)	0.8092 (0.0122)	0.1495 (0.0104)
PHP	0.0442 (0.0068)	0.7937 (0.0133)	0.1621 (0.0113)
Python	0.0475 (0.0065)	0.8251 (0.0124)	0.1274 (0.0107)
SQL	0.0374 (0.0081)	0.8311 (0.0126)	0.1315 (0.0110)

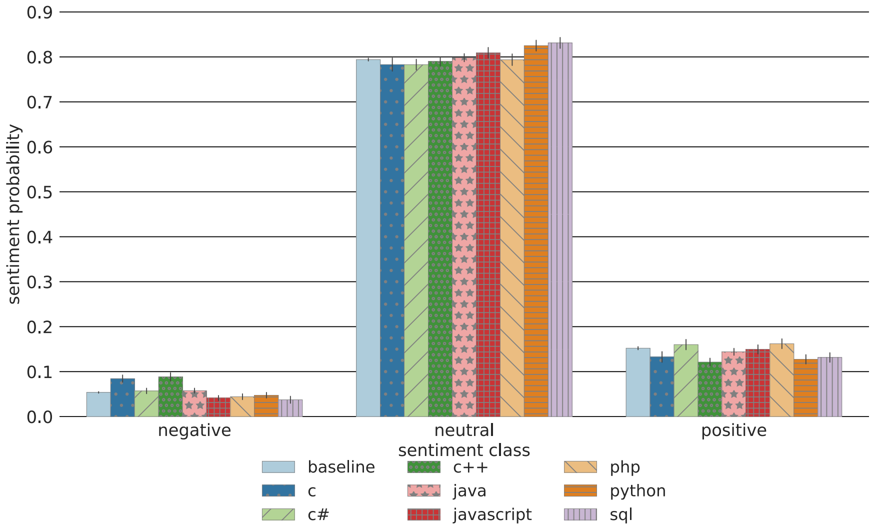

Figure 3: Sentiment probability distribution estimation for SO answers. Error bars represent ±1 standard deviation.

As in with the questions distributions in Section 5.2, the standard deviations of the programming language sentiment probabilities are larger than those of the baseline. However, in contrast to Section 5.2, all of the $p(neg)$ standard deviations are within the same order of magnitude as the baseline. For answers, all of the programming languages follow the same general pattern as the baseline does. For each language, $p(neu)$ is at least 50 standard deviations larger than either $p(neg)$ or $p(pos)$, while $p(pos)$ is at least a few standard deviations large than $p(neg)$. The closest call is for C++, where $p_{c++}(neg)$ is separated from $p_{c++}(pos)$ by 3.5 standard deviations.

5.4 Sentiment of Comments per Programming Language

Table 14 lists populations and sample sizes for comments with respect to the baseline and each of the programming languages.

Table 14: Population and sample size for comments.

Programming community	Population	Sample size
Baseline	8196263	9604
C	203682	1000
C#	575165	1000
C++	331590	1000
Java	719827	1000
JavaScript	964068	1000
PHP	579435	1000
Python	724879	1000
SQL	219776	1000

The sentiment probability distribution estimates for SO comments with respect to the baseline and each of the programming languages are shown below in Table 15. The same results are presented in Figure 4. The full classification results from which the probabilities were calculated can be found in Appendix D.

Table 15: Sentiment probability distribution estimates for SO comments. Parenthesized values are the standard deviations of the samples.

Programming language	p(neg)	p(neu)	p(pos)
Baseline	0.0830 (0.0024)	0.6975 (0.0040)	0.2195 (0.0029)
C	0.1081 (0.0116)	0.7103 (0.0100)	0.1816 (0.0083)
C#	0.0871 (0.0079)	0.7111 (0.0114)	0.2017 (0.0135)
C++	0.1045 (0.0114)	0.7122 (0.0145)	0.1833 (0.0074)
Java	0.0859 (0.0057)	0.7123 (0.0124)	0.2017 (0.0122)
JavaScript	0.0739 (0.0101)	0.6978 (0.0136)	0.2283 (0.0123)
PHP	0.0853 (0.0103)	0.7059 (0.0172)	0.2088 (0.0168)
Python	0.0781 (0.0084)	0.6893 (0.0131)	0.2327 (0.0137)
SQL	0.0748 (0.0061)	0.6891 (0.0167)	0.2361 (0.0169)

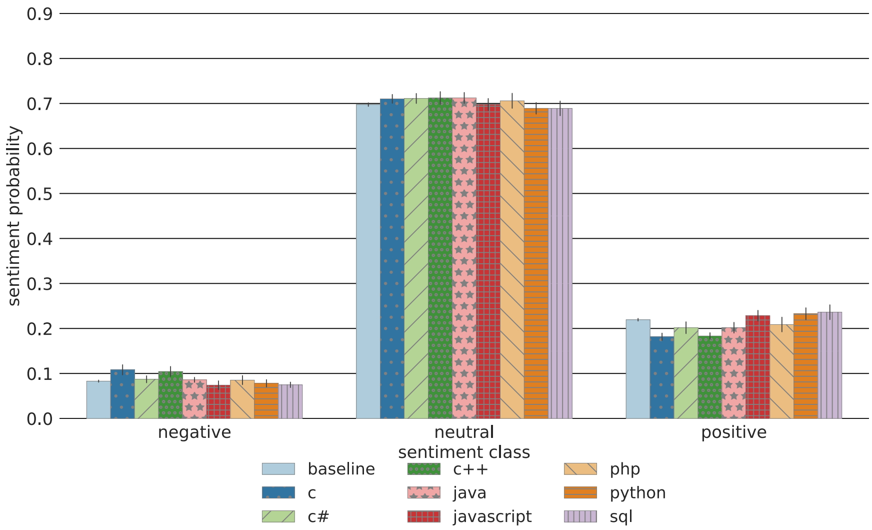

Figure 4: Sentiment probability distribution estimation for SO comments. Error bars represent ± 1 standard deviation.

Most of the standard deviations for the programming language sentiment probabilities are again an order of magnitude larger than those of the baseline, with a few staying within the same order of magnitude. The probability distributions look similar to the answers probabilities presented in Section 5.3. The difference is that, for all languages, $p(neg)$ and $p(pos)$ have increased at the expense of $p(neu)$, which has decreased. This is consistent with the behaviors displayed by the baseline distributions when comparing across document types presented in Section 5.1.

6 Threats to Validity

Sentiment analysis on SE texts is difficult, and several issues in this study threaten the validity of the results. The issues are mostly related to the tooling and the sampling techniques, which is discussed in Section 6.1 and Section 6.2 respectively.

6.1 Tooling

The largest issues with the tooling stem from the fact that Senti4SD is not entirely accurate, and is somewhat biased toward negative classification [10]. Questions especially seem to be over classified as negative. We find it unlikely that such a large amount of questions should be classified as negative. On a cursory manual inspection of questions classified as negative, we strongly disagreed with a considerable amount. The problem of over classifying documents as negative was one of the issues that Calefato et al. set out to counteract when developing Senti4SD, and it is possibly still an issue [10]. The fact that the prototype vectors used to extract semantic features (see Section 2.3.3) were computed from the sentiment scores of the SentiStrength lexicon could perhaps be part of the problem, as it is not an SE lexicon. There could also be other reasons for misclassification, such as failures in the preprocessing process. It is possible that the preprocessing process used in this study differs from the one used by Calefato et al., leading to sub-optimal application of their tool. Additionally, any code snippets and error messages not confined within html-tags or backticks would remain in the document, and could interfere with the classification process. It is also likely that questions could be misclassified as negative simply because the author describes the problem with negatively loaded words related to the issue. It is then possible that varying verbosity of error messages in different languages contribute to the differences.

Finally, there is also a possibility that the code written for this study contains errors affecting the results. Critical points include the non-trivial splitting and concatenation of documents as well as the parallelization, both of which are described in Section 4.5.1. We have attempted to unit test and integration test the behavior to the best of our abilites, but that is no guarantee that it works as intended.

6.2 Data Sampling

There are two principal issues with the data sampling that could reduce the validity of the results. The first one is that only data from 2017 was sampled from, as described in Section 4.2.1. The data is therefore not representative of the history of SO, but only the past year. However, the goal of this study was to investigate SO as it appears now, so not taking the history into account is not too large an issue. Furthermore, the data from 2017 constitutes a considerable part of the total amount of data. For example, the roughly 2.3 million questions sampled from in the baseline analysis amounts to more than 14% of the total 16 million questions. Overall, the impact of only considering 2017 data is likely to be small for the purposes of this study. The second issue with the sampling is that simple random sampling was performed on the documents without considering their dependencies on one another. For example, the strong influence that Kucuktunc et al. found between the sentiment in questions and answers on Yahoo! Answers could very well apply to SO as well [58]. It is also reasonable to assume that similar correlations may be found between posts and their related comments, or comments responding to other comments. Therefore, simply analyzing documents from the perspective that each document is independent of the others is clearly not entirely correct, and there may be a need for a more complicated model or sampling scheme. We are unsure how this affects the validity of the probabilities that were measured in this study, but it is a concern.

7 Discussion

7.1 Analysis of the Results

The results strongly indicate that there is a correlation between document type and the sentiment polarity of a document. Answers and comments appear to follow similar sentiment distributions, and comments tend to contain more positive and negative sentiment than answers. This is consistent with previous research finding a wide array of emotions in comments, which can potentially be because comments do not affect one's reputation on SO [9], [11]. Positive sentiment is however more prevalent than negative sentiment. This is interesting in light of the reportedly harsh environment on SO [14],

[16], which has been found most prevalent in comments [48], [11]. While the results do not precisely contradict the statements of the harshness on SO, it is interesting to note that sentiment in answers and comments is substantially more likely to be neutral and somewhat more likely to be positive than negative. What is somewhat surprising is that questions appear to be substantially more likely to contain both positive and negative sentiment than answers and comments. The results also indicate that questions is the only document type in which negative sentiment is more prevalent than positive sentiment. There could be many reasons for this. Close at hand is that the results are simply incorrect, as discussed in Section 6.1. There could however be other reasons. For one, the primary reason to post a question on SO is that the information seeker has been unable to solve some kind of problem. Questions may therefore contain negatively loaded content such as error descriptions and plain frustration, resulting in questions containing more negative sentiment than answers and comments [48].

Compared to the baseline, the standard deviations in the per-programming language analysis are large. This can most likely be attributed to the significantly smaller sample size of 1000, as opposed to 9604. Still, the results do indicate a correlation between the associated programming language and the sentiment expressed in a document. For example, in questions, $p_{SQL}(neg) < p_{C++}(neg)$ by 8 standard deviations, which is a fairly strong result. For both answers and comments, the separations between the sentiment probabilites of different programming languages tend to be small in relation to the standard deviations, so it is mostly difficult to conduct comparisons. There are also a few distinguishable trends throughout the document types. For example, $p_{C++}(neg)$ and $p_C(neg)$ are always larger than the corresponding probabilities for any of the other programming languages, regardless of document type. Overall, the distributions for C and C++ closely resemble one another. We also observe similar correlations between the distributions for Java and C#. One possible explanation for these correlations is that both pairs are fairly closely related. C++ is essentially a superset of C, and Java and C# are both object-oriented C-like languages with garbage collection.

An important fact to note is that the results in this study are not to be seen as indicative of the friendliness of any of the studied communities. While overt rudeness would probably be detected as negative sentiment, the presence of negative sentiment does not necessarily indicate the presence of rudeness or hostility. For example, Calefato et al. found that much of the negative sentiment expressed in questions is either directed toward the author herself, or toward the problem at hand [48], [11]. This also means that the results of this study do not contradict the previous findings of question askers being more polite than answerers [18].

There is also the issue of the accuracy of Senti4SD, as discussed in Section 6.1. Although Calefato et al. published data regarding the accuracy of Senti4SD on SO as a whole showing a bias toward negative classification [10], we suspect that different types of documents would result in different accuracy and bias. Bermingham et al. noticed that sentiment classification for longer blogs is more challenging than that of microblogs [34]. Since comments tend to be shorter and contain fewer code snippets than questions or answers, it might be reasonable to assume that Senti4SD would be more accurate for comments, and less so for questions and answers. Therefore, comparisons across document types may not be reliable. Assuming constant bias within document types, comparisons made between different programming languages for any given document type should still be viable [63, p. 380]. The assumption of constant bias within document types may not be entirely accurate either, however. Given the differing terminologies of different programming languages, in addition to varying formats of error messages and error descriptions, differing bias and accuracy between programming language communities can not be ruled out. It is therefore entirely possible that the trends observed in the per-programming language analysis are simply the results of differing classifier bias.

7.2 Our Contribution

In this report, we described a study on sentiment expressed by software developers on SO using Senti4SD, a sentiment analyzer trained with SO posts and comments. Our main theoretical and practical contributions are summarized below:

- We used a state-of-the-art sentiment analyzer to study sentiment in SE texts. This highlighted many difficulties with using this particular tool, as well as difficulties related to sentiment analysis of SE texts in general.
- We developed a toolchain for preprocessing and analyzing posts and comments from the SO data dump using Senti4SD. The toolchain is publicly available at https://github.com/li-simon-collab/so-sentiment-toolchain. Releasing the toolchain publicly makes reproducing our research less of an effort and more precise, and any mistakes we have committed could therefore be rectified with precision. We also see the toolchain as a valuable resource to junior researchers who may lack the programming skills to realize their research.
- We modelled sentiment distribution on SO in general, as well as for 8 programming languages. This enriches the field of sentiment analysis for SE.

7.3 Future Work

The differences between the sentiment distribution for questions and the ones for answers and comments is the strongest result of the study, and also the most surprising. Further research is needed to determine whether the actual sentiment polarity distribution in questions is so different, or if the results of this study have been affected by some systematic error. A first step would be to determine whether the classification bias of Senti4SD is dependent on document type. The SO data set manually labeled by Calefato et al. could for instance be utilized, which would effectively minimize the otherwise high cost of collecting labeled data [52]. As Jongeling et al. found that the use of different sentiment analyzers could lead to different conclusions [46], another way to validate the results of this study is to reproduce them with other sentiment analyzers.

The trends discovered in the per-programming language analysis should also be investigated further. First of all, larger sample sizes or more iterations are required to reduce the standard deviations and thereby increase the validity of the results. It is also important to find out how Senti4SD performs on different subdomains of SO. For example, the different programming languages examined in this study have different terminologies and typical error messages, leading to a different vocabulary. It would be worthwhile to investigate if this (or other dissimilarities) cause accuracy and bias to differ between subdomains. Seeing as the sizes of the communities considered in this study vary by a substantial amount, it is likely that some are more represented in the training set used to train Senti4SD than others. As sentiment analyzers generally perform poorly on domains they were not trained on, this could lead to different performance on different subdomains.

Another important area to research is how the preprocessing procedure affects classification. In this study, we tried to follow the preprocessing described by Calefato et al. [10], [52], but it is not clear how they arrived at this preprocessing procedure, nor the exact details of its execution. Thus, a thorough investigation into what parts to remove from an SE text and how to best format it for a sentiment analyzer is warranted.

Finally, there is also the matter of the limitations of sentiment polarity detection. Cambria et al. criticize the simplification of sentiment analysis into just polarity detection as being an oversimplification [61, p. v]. Future research could possibly use sentiment polarity detection to find subjective documents, and then utilize other techniques to, for example, find the causes of sentiment. To achieve the latter, context must also be preserved, and so treating single posts, answers and comments as separate contexts would be insufficient. Perhaps, a more appropriate approach would be to treat a question along with all answers and comments posted in relation to it as a single context. Sampling could then be performed only on questions, whereupon each document related to the question could be deterministically selected. Not only would context be preserved, but it would also alleviate the issue of the fallacy in assuming the samples in this study to be independent. Independence between questions is a more reasonable assumption than independence between, for example, comments that very well could have been posted to the same question. To use sentiment polarity detection as a guide for finding interesting documents, refining the tools that perform only sentiment polarity is of course a necessity, warranting further research solely on polarity detection as well.

8 Conclusions

The results show substantial differences between the sentiment distributions of different document types, implying a strong correlation between document type and sentiment. However, definitive conclusions about such correlations cannot be drawn due to uncertainties in tooling errors. What can be said with some degree of certainty is that all documents are predominantly neutral. As for the programming language analysis, the results indicate that there is a correlation between programming language and sentiment. Drawing conclusions from this would however be inadvisable because of the large standard deviations of the samples and the potential subdomain specific bias of the classifier. What can be said is that there appears to be a correlation, but further research is needed for any concrete conclusions to be drawn. Above all, this study has shown that sentiment analysis of technical texts is still in its early stages. Numerous issues remain, including analyzer bias, varying preprocessing methods and possible variations of accuracy and bias between subdomains. It is possible that sentiment analysis classifiers for SE are not yet mature enough for this kind of detailed comparative study.

References

[1] A. Y. Chua and S. Banerjee, "Answers or no answers: Studying question answerability in stack overflow," *Journal of Information Science*, vol. 41, no. 5, pp. 720–731, 2015.

[2] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos, "Analysis of the reputation system and user contributions on a question answering website: Stackoverflow," in *Proceedings of the 2013 ieee/acm international conference on advances in social networks analysis and mining*, 2013, pp. 886–893.

[3] "Privileges - stack overflow," 2018. [Online]. Available: https://stackoverflow.com/help/privileges. [Accessed: 12-Apr-2018].

[4] "Stack exchange site statistics." [Online]. Available: https://stackexchange.com/sites. [Accessed: 25-Mar-2018].

[5] "What are tags, and how should i use them? - stack overflow," 2018. [Online]. Available: https://stackoverflow.com/help/tagging. [Accessed: 28-May-2018].

[6] M. De Choudhury and S. Counts, "Understanding affect in the workplace via social media," in *Proceedings of the 2013 conference on computer supported cooperative work*, 2013, pp. 303–316.

[7] E. Guzman and B. Bruegge, "Towards emotional awareness in software development teams," in *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, 2013, pp. 671–674.

[8] J. M. George and A. P. Brief, "Feeling good-doing good: A conceptual analysis of the mood at work-organizational spontaneity relationship." *Psychological bulletin*, vol. 112, no. 2, p. 310, 1992.

[9] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli, "Mining successful answers in stack overflow," in *Mining software repositories (msr), 2015 ieee/acm 12th working conference on*, 2015, pp. 430–433.

[10] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Software Engineering*, pp. 1–31, 2017.

[11] F. Calefato, F. Lanubile, and N. Novielli, "How to ask for technical help? Evidence-based guidelines for writing questions on stack overflow," *Information and Software Technology*, vol. 94, pp. 186–207, 2018.

[12] R. Rodricks, "Why is stack overflow so negative of late?" 2014. [Online]. Available: https://meta.stackoverflow.com/questions/251758/why-is-stack-overflow-so-negative-of-late. [Accessed: 25-Mar-2018].

[13] K. Rudolph, "The rudeness on stack overflow is too damn high," 2014. [Online]. Available: https://meta.stackoverflow.com/questions/262791/the-rudeness-on-stack-overflow-is-too-damn-high. [Accessed: 25-Mar-2018].

[14] C. Upchurch, "Could we please be a bit nicer to new users," 2014. [Online]. Available: https://meta.stackexchange.com/questions/9953/could-we-please-be-a-bit-nicer-to-new-users. [Accessed: 25-Mar-2018].

[15] J. Hanlon, "The new new "be nice" policy ("code of conduct") - updated with your feedback," 2014. [Online]. Available: https://meta.stackexchange.com/questions/240839/the-new-new-be-nice-policy-code-of-conduct-updated-with-your-feedback. [Accessed: 12-Apr-2018].

[16] J. Hanlon, "Stack overflow isn't very welcoming. It's time for that to change." 2018. [Online]. Available: https://stackoverflow.blog/2018/04/26/stack-overflow-isnt-very-welcoming-its-time-for-that-to-change/. [Accessed: 11-May-2018].

[17] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: An empirical study," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 352–355.

[18] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, "A computational approach to politeness with application to social factors," *arXiv preprint arXiv:1306.6078*, 2013.

[19] "Stack exchange data dump." [Online]. Available: https://archive.org/details/stackexchange. [Accessed: 12-Feb-2018].

[20] C. A. Thompson, "A brief introduction to natural language processing for non-linguists," in *Learning language in logic*, Springer, 2000, pp. 36–48.

[21] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*, vol. 95. Prentice hall PTR Upper Saddle River, 2001.

[22] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.

[23] D. Yu and L. Deng, *Automatic speech recognition: A deep learning approach.* Springer, 2015.

[24] S. Armstrong, "Corpus-based methods for nlp and translation studies," *Interpreting*, vol. 2, no. 1, pp. 141–162, 1997.

[25] B. Pang, L. Lee, and others, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, nos. 1 – 2, pp. 1–135, 2008.

[26] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[27] B. Liu and L. Zhang, "A survey of opinion mining and sentiment analysis," in *Mining text data*, Springer, 2012, pp. 415–463.

[28] F. N. Ribeiro, M. Araújo, P. Gonçalves, M. A. Gonçalves, and F. Benevenuto, "SentiBench-a benchmark comparison of state-of-the-practice sentiment analysis methods," *EPJ Data Science*, vol. 5, no. 1, p. 23, 2016.

[29] M. B. Christopher, *PATTERN recognition and machine learning.* Springer-Verlag New York, 2016.

[30] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proceedings of the acl-02 conference on empirical methods in natural language processing-volume 10*, 2002, pp. 79–86.

[31] H. Tang, S. Tan, and X. Cheng, "A survey on sentiment detection of reviews," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10760–10773, 2009.

[32] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the Association for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[33] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha, “Comparing and combining sentiment analysis methods,” in *Proceedings of the first acm conference on online social networks*, 2013, pp. 27–38.

[34] A. Bermingham and A. F. Smeaton, “Classifying sentiment in microblogs: Is brevity an advantage?” in *Proceedings of the 19th acm international conference on information and knowledge management*, 2010, pp. 1833–1836.

[35] A. Aue and M. Gamon, “Customizing sentiment classifiers to new domains: A case study,” in *Proceedings of recent advances in natural language processing (ranlp)*, 2005, vol. 1, pp. 2–1.

[36] P. D. Turney, “Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews,” in *Proceedings of the 40th annual meeting on association for computational linguistics*, 2002, pp. 417–424.

[37] Y. R. Tausczik and J. W. Pennebaker, “The psychological meaning of words: LIWC and computerized text analysis methods,” *Journal of language and social psychology*, vol. 29, no. 1, pp. 24–54, 2010.

[38] S. M. Mohammad and P. D. Turney, “Crowdsourcing a word–emotion association lexicon,” *Computational Intelligence*, vol. 29, no. 3, pp. 436–465, 2013.

[39] C. H. E. Gilbert, “Vader: A parsimonious rule-based model for sentiment analysis of social media text,” in *Eighth international conference on weblogs and social media (icwsm-14). Available at (20/04/16) http://comp. Social. Gatech. Edu/papers/icwsm14. Vader. Hutto. Pdf*, 2014.

[40] P. Melville, W. Gryc, and R. D. Lawrence, “Sentiment analysis of blogs by combining lexical knowledge with text classification,” in *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining*, 2009, pp. 1275–1284.

[41] M. Thelwall, K. Buckley, and G. Paltoglou, “Sentiment strength detection for the social web,” *Journal of the Association for Information Science and Technology*, vol. 63, no. 1, pp. 163–173, 2012.

[42] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[43] A. Esuli and F. Sebastiani, “SentiWordNet: A high-coverage lexical resource for opinion mining,” *Evaluation*, pp. 1–26, 2007.

[44] S. Baccianella, A. Esuli, and F. Sebastiani, “Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining.” in *LREC*, 2010, vol. 10, pp. 2200–2204.

[45] R. Jongeling, S. Datta, and A. Serebrenik, “Choosing your weapons: On sentiment analysis tools for software engineering research,” in *Software maintenance and evolution (icsme), 2015 ieee international conference on*, 2015, pp. 531–535.

[46] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik, “On negative results when using sentiment analysis tools for software engineering research,” *Empirical Software Engineering*, vol. 22, no. 5, pp. 2543–2584, 2017.

[47] M. R. Islam and M. F. Zibran, “Leveraging automated sentiment analysis in software engineering,” in *Proceedings of the 14th international conference on mining software repositories*, 2017, pp. 203–214.

[48] N. Novielli, F. Calefato, and F. Lanubile, “The challenges of sentiment detection in the social programmer ecosystem,” in *Proceedings of the 7th international workshop on social software engineering*, 2015, pp. 33–40.

[49] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, “SentiCR: A customized sentiment analysis tool for code review interactions,” in *Proceedings of the 32nd ieee/acm international conference on automated software engineering*, 2017, pp. 106–111.

[50] N. Novielli, D. Girardi, and F. Lanubile, “A benchmark study on sentiment analysis for software engineering research,” *arXiv preprint arXiv:1803.06525*, 2018.

[51] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press, 2008.

[52] N. Novielli, F. Calefato, and F. Lanubile, “A gold standard for emotion annotation in stack overflow,” *arXiv preprint arXiv:1803.02300*, 2018.

[53] S. Bhattacharya, P. Srinivasan, and P. Polgreen, “Engagement with health agencies on twitter,” *PLoS One*, vol. 9, no. 11, p. e112235, 2014.

[54] A. Pak and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining.” in *LREc*, 2010, vol. 10.

[55] M. Malik and A. Hussain, “Helpfulness of product reviews as a function of discrete positive and negative emotions,” *Computers in Human Behavior*, vol. 73, pp. 290–302, 2017.

[56] N. Archak, A. Ghose, and P. G. Ipeirotis, “Deriving the pricing power of product features by mining consumer reviews,” *Management science*, vol. 57, no. 8, pp. 1485–1509, 2011.

[57] R. Ullah, N. Amblee, W. Kim, and H. Lee, “From valence to emotions: Exploring the distribution of emotions in online product reviews,” *Decision Support Systems*, vol. 81, pp. 41–53, 2016.

[58] O. Kucuktunc, B. B. Cambazoglu, I. Weber, and H. Ferhatosmanoglu, “A large-scale sentiment analysis for yahoo! Answers,” in *Proceedings of the fifth acm international conference on web search and data mining*, 2012, pp. 633–642.

[59] C. Treude, O. Barzilay, and M.-A. Storey, “How do programmers ask and answer questions on the web?: Nier track,” in *Software engineering (icse), 2011 33rd international conference on*, 2011, pp. 804–807.

[60] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, “Answering questions about unanswered questions of stack overflow,” in *Proceedings of the 10th working conference on mining software repositories*, 2013, pp. 97–100.

[61] E. Cambria, D. Das, S. Bandyopadhyay, and A. Feraco, *A practical guide to sentiment analysis*, vol. 5. Springer, 2017.

[62] “Stack overflow developer survey results 2017,” 2017. [Online]. Available: https://insights.stackoverflow.com/survey/2017. [Accessed: 28-Mar-2018].

[63] W. G. Cochran, *Sampling techniques.* John Wiley & Sons, 1977.

A Classification Results for Baseline

Table 16: Classification results for baseline samples.

sample id	negative	neutral	positive
questions 1	3089	4373	2142
questions 2	3118	4348	2138
questions 3	3078	4388	2138
questions 4	3082	4372	2150
questions 5	3069	4373	2162
questions 6	3051	4356	2197
questions 7	3069	4280	2255
questions 8	3171	4320	2113
questions 9	3071	4428	2105
questions 10	3079	4392	2133
questions 11	3130	4330	2144
questions 12	3110	4421	2073
questions 13	3029	4399	2176
questions 14	3058	4351	2195
questions 15	3074	4322	2208
answers 1	515	7632	1457
answers 2	509	7586	1509
answers 3	505	7678	1421
answers 4	557	7656	1391
answers 5	530	7622	1452
answers 6	529	7616	1459
answers 7	520	7604	1480
answers 8	508	7622	1474
answers 9	540	7618	1446
answers 10	492	7672	1440
answers 11	525	7607	1472
answers 12	504	7572	1528
answers 13	512	7659	1433
answers 14	510	7598	1496
answers 15	495	7649	1460
comments 1	804	6674	2126
comments 2	802	6744	2058
comments 3	796	6726	2082

Continued on next page

Table 16: Classification results for baseline samples.

sample id	negative	neutral	positive
comments 4	838	6682	2084
comments 5	821	6644	2139
comments 6	798	6716	2090
comments 7	765	6721	2118
comments 8	783	6721	2100
comments 9	797	6688	2119
comments 10	804	6636	2164
comments 11	753	6764	2087
comments 12	780	6745	2079
comments 13	830	6665	2109
comments 14	802	6664	2138
comments 15	783	6699	2122

B Classification Results for Questions per Programming Language

Table 17: Classification results for questions per programming language samples.

sample id	negative	neutral	positive
C questions 1	377	436	187
C questions 2	357	481	162
C questions 3	364	442	194
C questions 4	381	453	166
C questions 5	375	440	185
C questions 6	362	442	196
C questions 7	378	450	172
C questions 8	369	443	188
C questions 9	356	452	192
C questions 10	380	417	203
C questions 11	354	453	193
C questions 12	393	423	184
C questions 13	401	424	175
C questions 14	367	462	171

Continued on next page

Table 17: Classification results for questions per programming language samples.

sample id	negative	neutral	positive
C questions 15	362	466	172
C# questions 1	336	445	219
C# questions 2	329	450	221
C# questions 3	342	442	216
C# questions 4	359	422	219
C# questions 5	330	452	218
C# questions 6	354	428	218
C# questions 7	379	418	203
C# questions 8	344	422	234
C# questions 9	314	456	230
C# questions 10	348	435	217
C# questions 11	347	451	202
C# questions 12	344	439	217
C# questions 13	345	415	240
C# questions 14	347	445	208
C# questions 15	354	427	219
C++ questions 1	422	429	149
C++ questions 2	391	421	188
C++ questions 3	395	413	192
C++ questions 4	380	435	185
C++ questions 5	408	437	155
C++ questions 6	373	468	159
C++ questions 7	408	443	149
C++ questions 8	387	468	145
C++ questions 9	408	435	157
C++ questions 10	385	419	196
C++ questions 11	396	427	177
C++ questions 12	391	448	161
C++ questions 13	408	415	177
C++ questions 14	393	428	179
C++ questions 15	351	466	183
Java questions 1	326	458	216
Java questions 2	342	452	206
Java questions 3	334	441	225

Continued on next page

Table 17: Classification results for questions per programming language samples.

sample id	negative	neutral	positive
Java questions 4	327	467	206
Java questions 5	335	478	187
Java questions 6	345	436	219
Java questions 7	359	440	201
Java questions 8	350	441	209
Java questions 9	364	410	226
Java questions 10	336	452	212
Java questions 11	351	436	213
Java questions 12	335	442	223
Java questions 13	358	438	204
Java questions 14	335	457	208
Java questions 15	373	412	215
JavaScript questions 1	314	443	243
JavaScript questions 2	328	450	222
JavaScript questions 3	329	422	249
JavaScript questions 4	304	461	235
JavaScript questions 5	304	441	255
JavaScript questions 6	328	471	201
JavaScript questions 7	358	380	262
JavaScript questions 8	281	459	260
JavaScript questions 9	332	414	254
JavaScript questions 10	325	451	224
JavaScript questions 11	338	430	232
JavaScript questions 12	306	461	233
JavaScript questions 13	312	456	232
JavaScript questions 14	339	443	218
JavaScript questions 15	343	427	230
PHP questions 1	385	375	240
PHP questions 2	341	410	249
PHP questions 3	347	404	249
PHP questions 4	363	389	248
PHP questions 5	312	430	258
PHP questions 6	321	416	263
PHP questions 7	359	390	251

Continued on next page

Table 17: Classification results for questions per programming language samples.

sample id	negative	neutral	positive
PHP questions 8	351	413	236
PHP questions 9	335	424	241
PHP questions 10	349	405	246
PHP questions 11	370	384	246
PHP questions 12	330	412	258
PHP questions 13	312	435	253
PHP questions 14	331	429	240
PHP questions 15	379	396	225
Python questions 1	295	487	218
Python questions 2	325	445	230
Python questions 3	338	451	211
Python questions 4	340	441	219
Python questions 5	339	439	222
Python questions 6	309	462	229
Python questions 7	321	469	210
Python questions 8	335	435	230
Python questions 9	322	464	214
Python questions 10	333	428	239
Python questions 11	328	457	215
Python questions 12	334	442	224
Python questions 13	314	486	200
Python questions 14	327	461	212
Python questions 15	282	491	227
SQL questions 1	264	462	274
SQL questions 2	257	473	270
SQL questions 3	262	463	275
SQL questions 4	262	456	282
SQL questions 5	245	473	282
SQL questions 6	255	453	292
SQL questions 7	268	438	294
SQL questions 8	239	476	285
SQL questions 9	262	477	261
SQL questions 10	227	484	289
SQL questions 11	266	458	276

Continued on next page

Table 17: Classification results for questions per programming language samples.

sample id	negative	neutral	positive
SQL questions 12	221	473	306
SQL questions 13	223	468	309
SQL questions 14	243	492	265
SQL questions 15	250	492	258

C Classification Results for Answers per Programming Language

Table 18: Classification results for answers per programming language samples.

sample id	negative	neutral	positive
C answers 1	83	776	141
C answers 2	86	765	149
C answers 3	102	778	120
C answers 4	84	794	122
C answers 5	90	771	139
C answers 6	83	773	144
C answers 7	83	794	123
C answers 8	83	765	152
C answers 9	68	803	129
C answers 10	81	792	127
C answers 11	72	805	123
C answers 12	82	779	139
C answers 13	96	773	131
C answers 14	91	767	142
C answers 15	83	809	108
C# answers 1	50	811	139
C# answers 2	49	779	172
C# answers 3	50	795	155
C# answers 4	58	770	172
C# answers 5	60	774	166
C# answers 6	55	775	170
C# answers 7	60	794	146

Continued on next page

Table 18: Classification results for answers per programming language samples.

sample id	negative	neutral	positive
C# answers 8	59	791	150
C# answers 9	51	781	168
C# answers 10	65	761	174
C# answers 11	64	777	159
C# answers 12	47	782	171
C# answers 13	63	775	162
C# answers 14	66	794	140
C# answers 15	59	783	158
C++ answers 1	76	802	122
C++ answers 2	94	789	117
C++ answers 3	92	782	126
C++ answers 4	96	784	120
C++ answers 5	102	763	135
C++ answers 6	77	806	117
C++ answers 7	89	795	116
C++ answers 8	87	794	119
C++ answers 9	94	783	123
C++ answers 10	82	792	126
C++ answers 11	91	787	122
C++ answers 12	100	788	112
C++ answers 13	69	788	143
C++ answers 14	83	803	114
C++ answers 15	96	797	107
Java answers 1	61	801	138
Java answers 2	59	806	135
Java answers 3	59	792	149
Java answers 4	55	815	130
Java answers 5	58	801	141
Java answers 6	68	789	143
Java answers 7	58	794	148
Java answers 8	57	801	142
Java answers 9	58	807	135
Java answers 10	52	794	154
Java answers 11	53	791	156

Continued on next page

Table 18: Classification results for answers per programming language samples.

sample id	negative	neutral	positive
Java answers 12	51	812	137
Java answers 13	52	796	152
Java answers 14	71	780	149
Java answers 15	53	795	152
JavaScript answers 1	36	812	152
JavaScript answers 2	44	807	149
JavaScript answers 3	49	808	143
JavaScript answers 4	53	797	150
JavaScript answers 5	31	825	144
JavaScript answers 6	34	819	147
JavaScript answers 7	42	801	157
JavaScript answers 8	48	801	151
JavaScript answers 9	41	785	174
JavaScript answers 10	38	826	136
JavaScript answers 11	38	798	164
JavaScript answers 12	44	807	149
JavaScript answers 13	39	820	141
JavaScript answers 14	41	827	132
JavaScript answers 15	42	805	153
PHP answers 1	30	819	151
PHP answers 2	43	770	187
PHP answers 3	51	783	166
PHP answers 4	46	806	148
PHP answers 5	47	797	156
PHP answers 6	57	794	149
PHP answers 7	45	785	170
PHP answers 8	39	797	164
PHP answers 9	39	811	150
PHP answers 10	49	771	180
PHP answers 11	48	796	156
PHP answers 12	38	801	161
PHP answers 13	51	788	161
PHP answers 14	39	792	169
PHP answers 15	41	796	163

Continued on next page

Table 18: Classification results for answers per programming language samples.

sample id	negative	neutral	positive
Python answers 1	46	824	130
Python answers 2	35	848	117
Python answers 3	54	801	145
Python answers 4	37	831	132
Python answers 5	47	812	141
Python answers 6	50	824	126
Python answers 7	47	836	117
Python answers 8	59	823	118
Python answers 9	45	806	149
Python answers 10	43	838	119
Python answers 11	53	823	124
Python answers 12	42	830	128
Python answers 13	49	826	125
Python answers 14	51	836	113
Python answers 15	54	819	127
SQL answers 1	38	834	128
SQL answers 2	35	836	129
SQL answers 3	34	830	136
SQL answers 4	44	819	137
SQL answers 5	29	846	125
SQL answers 6	35	827	138
SQL answers 7	25	818	157
SQL answers 8	25	851	124
SQL answers 9	40	813	147
SQL answers 10	48	818	134
SQL answers 11	53	816	131
SQL answers 12	31	843	126
SQL answers 13	45	839	116
SQL answers 14	38	848	114
SQL answers 15	41	829	130

D Classification Results for Comments per Programming Language

Table 19: Classification results for comments per programming language samples.

sample id	negative	neutral	positive
C comments 1	127	697	176
C comments 2	100	709	191
C comments 3	126	705	169
C comments 4	89	734	177
C comments 5	107	697	196
C comments 6	99	718	183
C comments 7	107	724	169
C comments 8	128	702	170
C comments 9	112	708	180
C comments 10	106	707	187
C comments 11	107	708	185
C comments 12	113	704	183
C comments 13	103	708	189
C comments 14	93	718	189
C comments 15	105	715	180
C# comments 1	96	711	193
C# comments 2	82	718	200
C# comments 3	85	715	200
C# comments 4	83	706	211
C# comments 5	95	697	208
C# comments 6	81	709	210
C# comments 7	106	715	179
C# comments 8	88	711	201
C# comments 9	91	725	184
C# comments 10	87	696	217
C# comments 11	82	739	179
C# comments 12	74	701	225
C# comments 13	93	700	207
C# comments 14	82	718	200
C# comments 15	82	706	212
C++ comments 1	114	691	195

Continued on next page

Table 19: Classification results for comments per programming language samples.

sample id	negative	neutral	positive
C++ comments 2	87	747	166
C++ comments 3	106	720	174
C++ comments 4	115	708	177
C++ comments 5	106	710	184
C++ comments 6	97	720	183
C++ comments 7	107	711	182
C++ comments 8	100	713	187
C++ comments 9	118	696	186
C++ comments 10	90	728	182
C++ comments 11	107	707	186
C++ comments 12	103	710	187
C++ comments 13	129	689	182
C++ comments 14	91	713	196
C++ comments 15	97	720	183
Java comments 1	82	713	205
Java comments 2	88	692	220
Java comments 3	90	730	180
Java comments 4	91	710	199
Java comments 5	87	718	195
Java comments 6	81	736	183
Java comments 7	96	715	189
Java comments 8	74	713	213
Java comments 9	85	710	205
Java comments 10	89	699	212
Java comments 11	82	700	218
Java comments 12	78	728	194
Java comments 13	89	704	207
Java comments 14	87	717	196
Java comments 15	90	700	210
JavaScript comments 1	66	699	235
JavaScript comments 2	78	669	253
JavaScript comments 3	62	693	245
JavaScript comments 4	93	673	234
JavaScript comments 5	70	701	229

Continued on next page

Table 19: Classification results for comments per programming language samples.

sample id	negative	neutral	positive
JavaScript comments 6	65	708	227
JavaScript comments 7	64	696	240
JavaScript comments 8	72	718	210
JavaScript comments 9	73	708	219
JavaScript comments 10	65	702	233
JavaScript comments 11	80	702	218
JavaScript comments 12	89	701	210
JavaScript comments 13	66	709	225
JavaScript comments 14	76	706	218
JavaScript comments 15	89	682	229
PHP comments 1	102	670	228
PHP comments 2	90	706	204
PHP comments 3	92	700	208
PHP comments 4	82	703	215
PHP comments 5	79	707	214
PHP comments 6	81	710	209
PHP comments 7	105	683	212
PHP comments 8	71	712	217
PHP comments 9	89	734	177
PHP comments 10	82	726	192
PHP comments 11	72	688	240
PHP comments 12	89	716	195
PHP comments 13	72	696	232
PHP comments 14	80	728	192
PHP comments 15	93	710	197
Python comments 1	74	668	258
Python comments 2	71	711	218
Python comments 3	78	689	233
Python comments 4	81	692	227
Python comments 5	95	698	207
Python comments 6	71	691	238
Python comments 7	87	678	235
Python comments 8	68	691	241
Python comments 9	92	680	228

Continued on next page

Table 19: Classification results for comments per programming language samples.

sample id	negative	neutral	positive
Python comments 10	68	691	241
Python comments 11	80	663	257
Python comments 12	79	697	224
Python comments 13	71	710	219
Python comments 14	83	688	229
Python comments 15	73	692	235
SQL comments 1	76	695	229
SQL comments 2	73	692	235
SQL comments 3	80	679	241
SQL comments 4	67	684	249
SQL comments 5	75	686	239
SQL comments 6	72	715	213
SQL comments 7	80	682	238
SQL comments 8	77	695	228
SQL comments 9	71	719	210
SQL comments 10	67	658	275
SQL comments 11	79	669	252
SQL comments 12	74	691	235
SQL comments 13	77	670	253
SQL comments 14	65	707	228
SQL comments 15	89	694	217