NTNU
Norwegian University of
Science and Technology

Question analysis of coding questions on StackOverflow

130533 - Knut Lucas Andersen

31-05-2016

Master's Thesis
Master of Science in Applied Computer Science
30 ECTS
Department of Computer Science and Media Technology
Norwegian University of Science and Technology,

Supervisor 1: Assoc. Prof. Simon McCallum
Supervisor 2:

Preface

This is my Master thesis concluding the two years spent at NTNU Gjøvik: Master Applied Computer Science - Web, Mobile, Games track. The thesis was carried out during the spring semester 2016, from January to the end of May.

The main concept for the thesis was based on discussions with supervisor. The original plan was to create a Chat Agent that could answers students questions and give feedback to their question quality, by using StackOverflow as a knowledge base. However, during the Master thesis project presentation, other professors noted that the scope of the project was to large for a Master thesis. The thesis were therefore narrowed down to focus on coding questions posted on StackOverflow, in an attempt to evaluate question quality and predict the future votes for a given question.

31-05-2016

Acknowledgment

I would like to thank the following persons for their help and support during these years. It would not have been possible without them.

My supervisor, Simon McCallum, for understanding my difficult situation and for his patience and helpful advices on how to proceed so that I could complete my Master thesis.

Mariusz Nowostawski for his advice on how to get started with text processing and ideas for building the SVM model.

Rune Hjelsvold for helpful advice in relation to text analysis.

My best friend, Njål Dolonen, for always being there for me, and helped me get through this.

My grandmother, Mimmi H. Underland, may she rest in peace. None of this would have been possible without your support, understanding, love and care. This is for you.

I would also thank my family and friends for believing in me and supporting me through this.

K.L.A

Abstract

When you have a question you need an answer to, the solution many uses is to go online and use Google or another type of search engine. However, the degree of difficulty in regards to finding an answer varies with the problem you need an answer to. Today, the Internet offers a wide range of resources to acquire new knowledge, everything from encyclopaedias to blogs, forums and Question-Answer communities. However, when you are writing code, or developing complex programs, it can be easy to get stuck and not figure out why something stopped working.

Depending on the complexity of the problem, the only solution might be to go to an online community for help. One such community is StackExchange (consisting of 154 different communities), which is home to StackOverflow. A requirement for all StackExchange sites is that questions should be of good quality. The quality of a question is measured by use of votes and reputation. The votes are used to grade the question (or answer), whereas the reputation belongs to the user.

By using Machine Learning (ML), I wanted to see if it was possible to predict whether a new question would be viewed as a good question on StackOverflow. This was done by using the dataset containing all posts on StackOverflow and the ML method Support Vector Machines (SVM) (by using scikit-learn for Python). A lot of time was used understanding how scikit-learn worked and how to properly implement SVM to analyse the questions.

One of the interesting finds was that the average down-vote score for 10,000 was 7. To develop the feature detectors, a total of 100 good and bad questions were looked at to see if there was anything that stood out. It was a lot harder to find anything significant for what was a good question, but it was easier to spot the bad questions (e.g. homework, no code-samples, no explanation to what had been done before, etc.).

Before any processing was done, the SVM had a vocabulary of 69,766 features. To reduce this amount, stop-words were added (reducing to 69,462 features) and removal of numbers and hexa-decimal values (27,624). The most significant change was when the document term frequency were set to 1%, which gave a total of 440 features.

Write something more about feature detectors, results, etc.

Note! Not actual abstract, more of a general introduction to later narrow down to abstract

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic

StackOverflow is a part of the StackExchange community, where each community is related to a specific field with domain experts. However, a requirement is that the questions are of good quality [39, 42, 43]. Through peer-review, the quality of questions are filtered by using votes. If a question is good, it gets up-voted, if it is bad, it gets down-voted. The votes is only for the current question, but they still have an impact on the user, by use of reputation [41]. If a question is bad, it can be closed or deleted (e.g. duplicate, off topic, unclear, etc) [40].

The goal of this thesis is to research and analyse coding questions posted on StackOverflow. Most systems focus on finding good answers for the question asked, and not if it is a good question. It would therefore be interesting to see if it was possible to develop a system that could predict the quality of a question before posting it on StackOverflow.

1.2 Problem description

Many systems focus on finding the answer matching the users question, and not on the quality of the question asked. Many users have a negative view on StackOverflow due to their questions being down-voted or closed, because they simply do not meet the standard. Being able to ask a question that is considered good would be helpful both to the community and the user, since it would help the user be better at asking questions. Furthermore, this could also be used in education to help computer science students improve their questions when seeking help.

1.3 Research questions

- What defines a good (coding) question?
- Can we predict a questions quality by using SVM?
- What type of features increases the accuracy of the SVM?

1.4 Methodology to be used

To be able to solve this problem, one needs to understand how text classification is done, and what methods others have used for this. A requirement is also to have a dataset that contains the questions that have been asked on StackOverflow. This is achieved by using the dataset that is available through Stack-

Exchanges archive[1] [38]. Stackoverflow was started in 2008, meaning it now contains approximately 8 years of data, which has been continuously reviewed by domain experts.

Considering the size of the dataset and the amount of questions that are posted on StackOverflow, going through all questions manually would be to big a job for such a limited timeframe. Therefore only a selected amount of questions was looked at to get an overview to see if it was possible to see any differences between good and bad questions.

To analyse more questions a program was developed in Python, based on SVM. Since SVM is complex, the library scikit-learn [28] was used. In the beginning the word quality was tested by using term frequency, and at the end feature detectors were developed to see if it was possible to further enhance accuracy.

To be able to have proper backup, and also be able to see the development process, the program was uploaded to a GitHub repository[2]. In addition to the source code, the repository also contains an SQL script for creating the database tables[3], the sampled dataset that was used (.csv file) and the created SVM models.

1.5 Justification, Motivation and Benefits

The quality of a question is based on the votes given by the users of the StackOverflow community. Therefore the vote score was used to label the question as either good (score > 0) or bad (score < 0). This is justified by the fact that you can draw a comparison between the peer-review process used in academia and the peer-review process used in StackOverflow. The process starts with someone posting a question, which is then read, scored and given feedback (through comments or answers). The feedback can then improve the question through edits by the poster, based on the feedback given (e.g. adding additional information or explaining how the given problem(s) occurred).

Scikit-learns SVM implementation is based on libsvm [5], but it is simpler to use. In addition, scikit-learn focuses on code quality[4], to ensure that everything works as it should [28, p. 3].

It is not a lot of research being done related to asking and defining good questions. Although the focus is only on programming questions, it can still be useful to improve question quality. Even if you cannot predict if it is a good question,

[1] StackExchange dataset: https://archive.org/details/stackexchange (Downloaded 30. March 2016).

[2] GitHub repository: https://github.com/klAndersen/IMT4904_MasterThesis_Code

[3] Note that this only creates the database and the table structures, it does not contain any of the StackOverflow data

[4] Coverage on their GitHub repository was 94% on 06. May 2015.

you can still avoid asking a bad questions. This in turn can help to improve the question quality (at least acceptability), and avoid asking questions that will be closed or ignored on StackOverflow.

1.6 Limitations

The greatest limitation is the time available. The total dataset has a size of 141,8 GB (but that is for all the StackOverflow data, not just the questions). Filling the database, retrieving the relevant questions and training the SVM takes time[5]. Furthermore, it also takes a lot of time to read through questions and look for differences between good and bad questions. Developed feature detectors may not have that much impact on accuracy, which means more time must be used on re-training and finding other features that can be of interest.

1.7 Thesis contribution

This thesis contribution can be summarized as to the following: Predicting (programming) question quality by using Artificial Intelligence (AI) and ML to improve the questions quality. Stackoverflow was started in 2008, which means that it now contains approximately 8 years of data, which has been continuously reviewed and judged by domain experts. Yet, many still fails at asking the good questions. Many are of course ignored or down-voted because they are interpreted as homework, but there are also a lot of users that just do not know how to ask a good question. Hopefully this research can help with that, so that future users (and students) can benefit to improve the quality of their questions.

1.8 Thesis structure

The following is the structure of this thesis:

- Chapter 2: State of the art and relevant research
- Chapter 3: Methodology
- Chapter 4: (unused placeholder)
- Chapter 5: Discussion on development, the thesis and limitations
- Chapter 6: Conclusion and suggestions for further work

[5]Between 2-3 hours after feature reduction

2 Related work

The goal of this target is to document the current state of the art technologies, and research that has been done by others. This is needed to both understand how to use text classification and classify questions. Furthermore, some of the papers have also done comparisons on SVM and other types of AI and ML algorithms, which concluded that for text classification, SVM had the highest accuracy.

Example for Simon:

I can either use `\cite{ChangLin2011}` and get: [5].
Or I can use `\citet{ChangLin2011}` and get: Chang and Lin [5].
`\cite` is based on `\renewcommand*{\cite}{\autocite}`,
and `\citet` is based on `\newcommand{\citet}{\textcite}`.
Unfortunately, you can't get the year, but at least you can now get in-text citation.

The listing below shows the whole command:

```
# HEADER/BEFORE \begin{document}

\usepackage{csquotes}% Recommended
%bibliograhpy
\usepackage[style=authoryear, % harvard style (author, year)
bibstyle=numeric, % lists bibliography as numeric [1] ..., [2] ..., etc.
citestyle=numeric, % numeric citing; where \cite gives numeric,
but \textcite gives authoryear
backend=biber, % backend
%% backref takes you back from bibliography to the page you clicked reference
%% example: (cit. on p. 7), where 7 would take you back to that page
% backref=true,
maxcitenames=2, % max number of names to list when using author names (et al.)
doi=true]{biblatex}

%% loads the bibliography for biblatex and for texniccenter
%% this is to enable auto-complete and being able to search through all entries
\makeatletter
\@ifpackageloaded{biblatex}{\addbibresource{bibfile.bib}}{\bibliography{bibfile}}
\makeatother

%% changes the cite command into autocite
\renewcommand*{\cite}{\autocite}
```

```
%% adds comma between author--year; (author, year)
\renewcommand*{\nameyeardelim}{\addcomma\space}
\newcommand{\citet}{\textcite}

# ADD AT THE END, WHEREVER YOU WANT BIBLIOGRAPHY PRINTED
% Start the references on a new page
\clearpage
\printbibliography
```

Citation list (content to come):

- [5]: Libsvm
- [50]: Question classification, SVM, semantics
- [13]: SVC
- [22]: Sigmoid kernels SVM
- [52]: SVM, QA, Context ranking
- [27]: Question taxonomy
- [15]: Japanese Q/A System
- [24]: Reputation system SO
- [4]: Text classification, semantics
- [10]: fuzzy sigmoid svm
- [54]: sentiment analysis, linguistics
- [33]: Answer quality in qa community
- [45]: Answer type construction
- [47]: QA on the web (programmers)
- [36]: What is a good Q/A?
- [18]: predict closed questions on SO
- [44]: tag prediction on SO
- [14]: Question classification
- [53]: Question classification, svm
- [3]: random search, hyperparameters
- [46]: SVM, text classification
- [12]: svm
- [16]: svm, text processing
- [7]: Salton
- [26]: semi-supervised, question classification
- [30]: problem solving question, cs
- [23]: wordnet
- [25]: good code example, SO
- [11]: SO expertise & knowledge
- [48]: dev interaction SO
- [51]: expertise vs activity on SO
- [6]: question feature for answer attraction SO
- [34]: tag recommendation SO

- [29]: questions, answers, activity and reputation on SO
- [1]: stackexchange, technical vs non-technical communities activity
- [9]: domain ontology, qa
- [19, 20, 21] qa classification, semantics
- [2] so, qa, post quality
- [35] qa, design, anwer response
- [17, 37, 49] datasets

In the paper by Toba et al. [45], they experiment with the use of statistical learning to find the expected answer pattern for factoid Question-Answering (Q/A) pairs. E.g. if you ask someone where a certain event took place, the answer pattern would be a location. They group question analysis into two approaches; pattern-based (high precision, low recall) and ML (high recall, low precision[1]). Pattern-based would match word sequences against a set of patterns (e.g. regular expressions), whereas ML would be based on the accuracy of the classifier (e.g. lexical or linguistic feature sets).

Xu et al. [50] used SVM to create an online Q/A tourism system in Chinese. The system included question analysis, Information Retrieval (IR) and answer extraction. The question classification accuracy was important to the overall performance of the system. The system was built using SVM and question semantic similarity, and the feature selection was based on lexical features and domain terms hierarchy. The original method for question classification is mainly rule-based, where the rules decide the category (difficult to extract all the rules for the question category). Another method is using statistics, as was done in [53] ("to classify questions in English. It uses tree kernel to extract features and classify questions with SVM classifier."). Questions were divided into 13 coarse categories and 150 sub-categories. The difference between these were that the sub-categories was built on sentences.

2.1 Text classification

Text classification can be done in many different ways, since the content and size rarely will be equal. The classification is also based on what you want to retrieve from the text. Do you want an answer to a question, or do you want to see which documents are most relevant for the problem you are currently working on (e.g. searching for research papers that are relevant to your work).

2.2 Question classification

Question classification and analysis can be seen as a sub-topic of text classification. In most papers, the basis is that the user has a question they need an answer to. What is the best way to find that answer, and in some cases what is

[1]Low precision can occur if the feature sets are not fitted well enough during classifier training [45, p. 283].

the quickest way to display said answer?

- question-answering techniques and analyses
- ontology and semantics
- StackExchange and other online communities for qa (potentially separate section)

2.3 Support Vector Machines (SVM)

SVM is the chosen ML algorithm that was chosen to use for the question analysis. This section is mainly intended as a light introduction to what SVM is, and what the most common uses are. Furthermore, in what way can this be utilized for text and question classification?

2.4 Dataset

short about the datasets, others who has used it, and datasets in general, e.g. [17, 37, 49]

3 Methodology

3.1 Dataset and MySQL Database

3.1.1 Dataset

The dataset contains all information that is currently available in the StackExchange community (at the time the dataset was created). The following is a list of the tables found in the dataset:

- Badges: Badges awarded to users.
- Comments: Comments given either to a question or an answer.
- Posts: Posts on StackExchange, this contains both questions and answers.
- Posthistory: The history of a given post (e.g. edits, reason for closing, etc.).
- Postlinks: Link to other Posts (e.g. duplicates).
- Users: Information about the given user registered at the given community.
- Votes: Type of vote given to a Post (e.g. up/down, vote to close, etc.).

In the beginning, the dataset that was used was downloaded in August 2015. However, since this turned out to be outdated, the latest dataset was downloaded from (https://archive.org/details/stackexchange) on 30. March 2016. The dataset comes in zip-files, where each zip-file contains all the rows found in the given table. These rows are presented in an XML file, as shown in Listing 3.1.

Listing 3.1: Content in stackoverflow.com-Tags.xml

```
<?xml version="1.0" encoding="utf-8"?>
<tags>
<row Id="1" TagName=".net" Count="227675"
ExcerptPostId="3624959" WikiPostId="3607476" />
<row Id="2" TagName="html" Count="511091"
ExcerptPostId="3673183" WikiPostId="3673182" />
...
</tags>
```

3.1.2 MySQL Database

In the beginning, the issue was getting access to the file and see how it looked like. Since most of these XML files had a large file size (ranging from 3,9 MB to 71,9 GB) none of the editors could open them. Attempting to open them through Python code also failed, since there was not enough memory to process everything. The only solution was therefore to create a MySQL database that could contain all the data.

Setting up the MySQL database was not a straight forward process. The operative system I was running was Arch Linux, where they had switched from using Oracle's MySQL to MariaDB[1]. One of the main problems was the available storage space[2] and the varying file sizes. Some of the issues were mainly connection timeout, no more disk space and connection loss (e.g. "Error Code: 2013. Lost connection to MySQL server during query"). To avoid losing the connection to the database, the timeout values had to be changed in MySQL Workbench (shown in Figure 1).

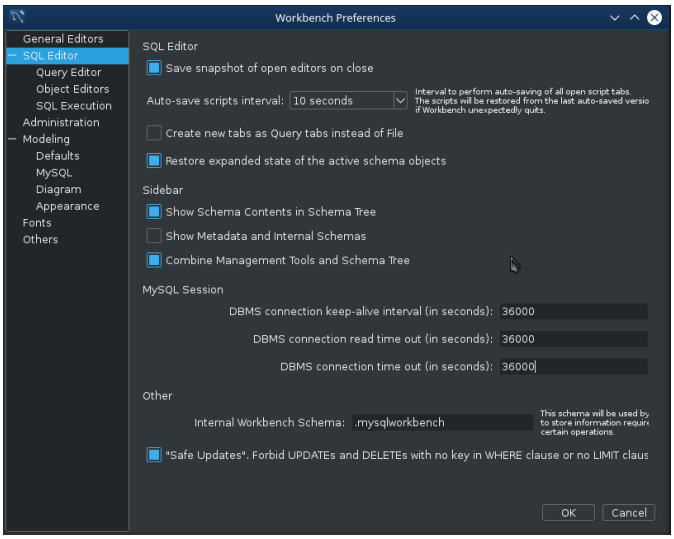

Figure 1: MySQL Workbench: Setting timeout values to avoid connection loss

The next problem was the lack of disk space. MySQL by default stores all databases and belonging tables in /var/lib/mysql/, and it also creates temporary backup files (where the file size is equal to the size of the current database). Since the default folder for temporary files was on /root, the disk space was used up in less than 30 minutes. Therefore, two things needed to be done. First, disable the storage of temporary files, and secondly change the storage location for the database. The problem when tinkering with the configuration file is that things easily break. Which is what happened, and a clean install was needed for both MariaDB and MySQL (the changed settings can be seen in Listing 3.2). The final step was to create symbolic links that linked the database to the location where the tables where stored (this has to be done before creating the tables, if not MySQL Workbench will store the tables in /var/lib/mysql/)[3].

[1]See https://wiki.archlinux.org/index.php/MySQL.

[2]The HDD with Arch Linux installed had a disk size of 500 GB, with four partitions; root, var, swap and home. 40 GB was used for /root and /var, 12 GB was used for swap and the remainder was used for /home.

[3]It should be noted that after an upgrade of MariaDB, MySQL and MariaDB could no longer

Listing 3.2: Changes made to config file: /etc/mysql/my.cnf

```
# disable storage of temporary files
#tmpdir = /tmp/
# disable storage of log files
#log-bin = mysql-bin

# set directory for storing database files
datadir = /home/mysql
```

Listing 3.3: Load XML file into a table in the MySQL database

```
LOAD XML LOCAL INFILE
path_to_xml_file
INTO TABLE db_table
ROWS IDENTIFIED BY '<row>';
```

Listing 3.3 shows how the files were loaded into the tables, and the complete database can be seen in Appendix A.3, p. 25. Since the Posts table is large (~29,5 million rows) and it contains both questions and answers, two new tables were created; "posvote_Posts"[4] and "negvote_Posts". posvote_Posts contains questions with a score higher then zero (score > 0) and negvote_Posts contains all questions with a score lower then zero (score < 0).

3.2 Development process

When starting the development, the focus was on retrieving the data from the database, and processing it for text analysis. To be able to store all the retrieved columns and the belonging rows without creating object classes, the pandas.DataFrame[5] was used.

The questions retrieved needed to be processed before any analysis could be done. The reason for this is because the questions was written as HTML (including HTML entities). An example is shown in Listing 3.4. Every question starts with the <p> tag, and if the question contains code samples, these are wrapped with a <code> tag. To convert the HTML text into readable text, a HTML parser class was created (based on answer by [8]).

find the tables, even if they still were in the /home/mysql/ folder. It is therefore advisable to dump the database after inserting all the tables, since it goes a lot faster to restore the database from dump rather than insertion from XML files.

[4]The Posts table has a file size of ~43,6 GB, whereas posvote_Posts file size is ~11,2 GB. negvote_Posts has a file size of ~1,33 GB.

[5]Pandas: http://pandas.pydata.org/.

Listing 3.4: Question before HTML is removed (Question ID: 941156)

```
<p>
Why do we need callbacks in ASP.NET or any server side technology?
</p>&#xA;&#xA;<p>One answer can be, to achieve asynchronous calls.
</p>&#xA;&#xA;<p>But I am not satisfied with this answer.</p>
&#xA;&#xA;<p>Please explain it in a different way.
</p>
&#xA;
```

To process the questions, CountVectorizer from scikit-learn was used. CountVectorizer uses the vocabulary found in the text and counts the frequency of each word [32] [31, p. 4.2.3]. When looking at this vocabulary, a lot of of un-important words was found (a lot which came from the code samples) provided in some of the questions. At first all code samples were removed from the text, but later on they were replaced with the value 'has_codeblock', indicating that this question contained one or more code samples. This was achieved by using a combination of lxml[6] and bs4[7] (BeatifulSoup). lxml was used to construct an XML tree containing all the tags (to be able to retrieve the content by searching for a given tag), and bs4 was used for beautifying the HTML (since in some cases an error was thrown complaining about "Missing end tag").

However, for some questions, part of the text was lost, and for others, some <code> tags was not removed. On inspection, it was found that the trailing text following the <code> samples was stored in a .tail attribute. Since the <code> was removed, the .tail attribute was also removed. This was fixed by storing the the content of the <code> .tail attribute into its <parent>[8] (where <parent> is the tag that contained the given <code></code>) .tail attribute. As for the non-complete removal of <code> tags, this error mostly occurred for code samples that contained XML or HTML code[9], because the lxml parser failed. The solution was to replace the lxml parser with bs4 and just change the content of the <code> tag to the value 'has_codeblock'.

Considering the size of the dataset, and that the source code was hosted on GitHub, I was hesitant to store the training data in a separate file. However, when loading 20,000 samples from the database with a 'WHERE' parameter, things tend to go more slow. At this point, it was decided to try to dump the loaded data from the database to a file. This was achieved by using pandas.DataFrame.to_csv[10].

[6] lxml: http://lxml.de/

[7] BeatifulSoup: https://www.crummy.com/software/BeautifulSoup/

[8] It was also necessary to check if the <parent> had a .tail, if not, the .tail attribute had to be set for the <parent> to avoid the error: "NoneType + str: TypeError".

[9] One example is this question:
http://stackoverflow.com/questions/19535331/print-page-specific-area-or-element.

[10] pandas.DataFrame.to_csv:

At a later point, the unprocessed dataset was also dumped to a CSV file for replicability[11].

Further examination showed that the vocabulary contained a lot of numerical and hexadecimal values, but also a lot of non-English words. The numerical and hexadecimal values were replaced using regular expressions to 'has_hexadecimal' and 'has_numeric'. The non-English words were a bit more troublesome to handle, since these were mainly used to prove a point or show an example of the issue they were having[12]. Attempts were made to filter them out by using corpus.words.words() and corpus.wordnet.synset() from Natural Language Toolkit (NLTK)[13], and PyEnchant[14]. However, WordNet does not have a complete database of all English words, and they all claimed some words were not English even though they were.

The solution turned out to be a lot simpler. Instead of creating filters, the CountVectorizer already had one built in. By adjusting the minimum document frequency (min_df) and set it to 0.01, it would ignore words that did not appear in less then 1% of all documents.

To write:
Tutorials that I went through
Using SGD (based on tutorials from scikit-learn)
Testing out different text classification algorithms (SVC, SGD and LinearSVC)
Attempting to make program runnable from command line; e.g. started with opt/argparse, ended with while loop

3.3 Feature sets, attributes and processing

When retrieving the questions from the database, the vote score was set to less than -10 for bad question and greater than 50 for good questions (retrieval limit set to 10,000; 20,000 total). However, the vote score was set too low for the bad questions, since only 683 rows was returned. The score was then set to less than -5. When using pandas.Categorical to get an statistical overview (code snippet in Listing 3.5 and result in Table 1), one can see that for 10,000 samples, the average vote score was -7. This could be an indicator that when a question has a vote score below -5, they are ignored.

http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html.

[11] The only change made to the unprocessed dataset was removing the HTML tags.

[12] http://stackoverflow.com/questions/856307/wordwrap-a-very-long-string.

[13] http://www.nltk.org/

[14] http://pythonhosted.org/pyenchant/

Class	Statistics	AnswerCount	Score	Question length
-1	mean	2.0483	-7.0275	319.226
	std	1.3129	2.676	382.115
	min	0.0	-147.0	13.0
	25%	1.0	-7.0	153.0
	50%	2.0	-6.0	239.0
	75%	3.0	-6.0	379.0
	max	20.0	-6.0	13673.0
1	mean	11.9379	182.5483	459.329
	std	13.707824	317.47217	531.187559
	min	0.0	51.0	13.0
	25%	6.0	67.0	189.0
	50%	9.0	96.0	328.0
	75%	14.0	173.0	558.0
	max	518.0	9432.0	18867.0

Table 1: Results from pandas.DataFrame and pandas.Categorical. -1 is for bad questions (votes < -5), and 1 are for good questions (votes > 50).

Listing 3.5: Getting Categorical data from pandas.DataFrame

```
from pandas import DataFrame, Categorical

# get statistics from pandas.DataFrame
temp_df = __so_dataframe.loc[:, ("Score", "Body", "Title",
        "AnswerCount", "length")]
temp_df.loc[:, CLASS_LABEL_KEY] = Categorical(__so_dataframe.loc[:,
                                                "label"])

# prints out the questions AnswerCount, Score and length
print(temp_df.groupby("label").describe())
# prints all selected columns
print(temp_df.groupby("label").describe(include='all'))
```

what features were selected and why?
how was data processed (e.g. retrieved from db, converted to scikit-learn format, and so forth)
attributes: length, symbols, question sentence only, code snippet, votes, closed, etc.

Step	Text processing	Vocabulary count	CountVectorizer
1	None	69766	analyzer="word"
2	Stop words	69462	analyzer="word", stop_words="english"
3	Removal of code, hexadecimal and numerical values	27624	analyzer="word", stop_words="english"
4	Minimum document frequency	440	analyzer="word", min_df=0.01, stop_words="english"

Table 2: Feature reduction steps before and after text was processed.

Hypotheses drafts (need to be re-written)

- Does the use of stemming increase the SVMs accuracy?
- Does the use of tags increase the SVMs accuracy?
- If a question has version numbering, does this increase SVMs accuracy?
- Does syntax errors have an impact on the question (may not be tested due to LINT check)?
- If question contains links to sources (e.g. repository, StackOverflow question, etc), does it have any effect?
- What about questions containing homework description (e.g. homework, assignment, textbook, etc)?

4 chapter4

empty placeholder section

5 Discussions

5.1 Data and Testing

discussion on the data set and how it was tested.
the results and what they showed.
potential improvements, etc.

5.2 Artificial Intelligence (AI) Methods

alternative methods and options (e.g. one could have used ann or k-nn, but as shown in. . .)
not sure if this section is relevant?

5.3 Implementation Architecture

discussion on the code that was written and its functionality
what worked, what should be updated/changed, etc.

5.4 Limitations and other issues

why didn't something work as intended?
why wasn't X completed/implemented?
etc.

6 Conclusion/Summary

6.1 Overview of main results

basically what the title says; a summary of the results

6.2 Further work

additions and updates to the system
new research possibilities based on results

Bibliography

[1] Saif Ahmed, Seungwon Yang, and Aditya Johri. "Does Online Q&A Activity Vary Based on Topic: A Comparison of Technical and Non-technical Stack Exchange Forums". In: *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*. L@S '15. Vancouver, BC, Canada: ACM, 2015, pp. 393–398. ISBN: 9781450334112. DOI: 10.1145/2724660.2728701. URL: http://doi.acm.org/10.1145/2724660.2728701.

[2] Ashton Anderson et al. "Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. Beijing, China: ACM, 2012, pp. 850–858. ISBN: 9781450314626. DOI: 10.1145/2339530.2339665. URL: http://doi.acm.org/10.1145/2339530.2339665.

[3] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 281–305.

[4] Stephan Bloehdorn and Andreas Hotho. "Boosting for text classification with semantic features". In: *WebKDD*. Springer. 2004, pp. 149–166.

[5] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[6] Derrick Cheng, Michael Schiff, and Wei Wu. "Eliciting Answers on StackOverflow". In: (2013).

[7] David Dubin. "The Most Influential Paper Gerard Salton Never Wrote". In: *LIBRARY TRENDS* 52.4 (2004), pp. 748–764.

[8] Eloff. *Strip HTML from strings in Python*. 2009. URL: http://stackoverflow.com/a/925630.

[9] J. Fu, K. Jia, and J. Xu. "Domain Ontology Learning for Question Answering System in Network Education". In: *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*. Nov. 2008, pp. 2647–2652. DOI: 10.1109/ICYCS.2008.337.

[10] Liu Han, Liu Ding, and Deng Ling-Feng. “Chaotic time series prediction using fuzzy sigmoid kernel-based support vector machines”. In: *Chinese Physics* 15.6 (2006), p. 1196. URL: http://stacks.iop.org/1009-1963/15/i=6/a=012.

[11] Benjamin V. Hanrahan, Gregorio Convertino, and Les Nelson. “Modeling Problem Difficulty and Expertise in Stackoverflow”. In: *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*. CSCW ’12. Seattle, Washington, USA: ACM, 2012, pp. 91–94. ISBN: 9781450310512. DOI: 10.1145/2141512.2141550. URL: http://doi.acm.org/10.1145/2141512.2141550.

[12] M. A. Hearst et al. “Support vector machines”. In: *IEEE Intelligent Systems and their Applications* 13.4 (July 1998), pp. 18–28. ISSN: 1094-7167. DOI: 10.1109/5254.708428.

[13] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. “A practical guide to support vector classification”. In: (2003).

[14] Zhiheng Huang, Marcus Thint, and Zengchang Qin. “Question Classification Using Head Words and Their Hypernyms”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP ’08. Honolulu, Hawaii: Association for Computational Linguistics, 2008, pp. 927–936. URL: http://dl.acm.org/citation.cfm?id=1613715.1613835.

[15] Hideki Isozaki. “An Analysis of a High-performance Japanese Question Answering System”. In: 4.3 (Sept. 2005), pp. 263–279. ISSN: 1530-0226. DOI: 10.1145/1111667.1111670. URL: http://doi.acm.org/10.1145/1111667.1111670.

[16] Celso Antonio Alves Kaestner. “Support Vector Machines and Kernel Functions for Text Processing”. In: *Revista de Informática Teórica e Aplicada* 20.3 (2013), pp. 130–154.

[17] Gary Klein. *Blinded By Data*. 2016. URL: https://www.edge.org/response-detail/26692.

[18] C Galina E. Lezina and Artem M. Kuznetsov. *Predict Closed Questions on StackOverflow*. Last Accessed: 25.09.2015. 2013. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.394.5678.

[19] Xin Li and Dan Roth. “Learning Question Classifiers”. In: *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*. COLING ’02. Taipei, Taiwan: Association for Computational Linguistics, 2002, pp. 1–7. DOI: 10.3115/1072228.1072378. URL: http://dx.doi.org/10.3115/1072228.1072378.

[20] Xin Li and Dan Roth. “Learning Question Classifiers: The Role of Semantic Information†”. In: *Natural Language Engineering* 1.1 (), pp. 000–000.

[21] Xin Li, Dan Roth, and Kevin Small. “The Role of Semantic Information in Learning Question Classifiers”. In: ().

[22] Hsuan-Tien Lin and Chih-Jen Lin. “A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods”. In: *submitted to Neural Computation* (2003), pp. 1–32.

[23] George A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748.

[24] Dana Movshovitz-Attias et al. “Analysis of the reputation system and user contributions on a question answering website: Stackoverflow”. In: *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*. IEEE. 2013, pp. 886–893.

[25] S. M. Nasehi et al. “What makes a good code example?: A study of programming Q amp;A in StackOverflow”. In: *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. Sept. 2012, pp. 25–34. DOI: 10.1109/ICSM.2012.6405249.

[26] Tri Thanh Nguyen, Le Minh Nguyen, and Akira Shimazu. “Using Semi-supervised Learning for Question Classification”. In: *Information and Media Technologies* 3.1 (2008), pp. 112–130. DOI: 10.11185/imt.3.112.

[27] Rodney D Nielsen et al. “A taxonomy of questions for question generation”. In: *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge*. 2008.

[28] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[29] D. Posnett et al. “Mining Stack Exchange: Expertise Is Evident from Initial Contributions”. In: *Social Informatics (SocialInformatics), 2012 International Conference on*. Dec. 2012, pp. 199–204. DOI: 10.1109/SocialInformatics.2012.67.

[30] Noa Ragonis and Gila Shilo. “What is It We Are Asking: Interpreting Problem-solving Questions in Computer Science and Linguistics”. In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE ’13. Denver, Colorado, USA: ACM, 2013, pp. 189–194. ISBN: 9781450318686. DOI: 10.1145/2445196.2445253.

[31] Scikitlearn.org. *4.2. Feature extraction*. 2016. URL: http://scikit-learn.org/stable/modules/feature_extraction.html.

[32] Scikitlearn.org. *sklearn.feature_extraction.text.CountVectorizer*. 2016. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[33] Chirag Shah and Jefferey Pomerantz. "Evaluating and Predicting Answer Quality in Community QA". In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '10. Geneva, Switzerland: ACM, 2010, pp. 411–418. ISBN: 9781450301534. DOI: 10.1145/1835449.1835518.

[34] Logan Short, Christopher Wong, and David Zeng. "Tag recommendations in stackoverflow". In: (2014).

[35] Vibha Singhal Sinha, Senthil Mani, and Monika Gupta. "Exploring Activeness of Users in QA Forums". In: *Proceedings of the 10th Working Conference on Mining Software Repositories*. MSR '13. San Francisco, CA, USA: IEEE Press, 2013, pp. 77–80. ISBN: 9781467329361. URL: http://dl.acm.org/citation.cfm?id=2487085.2487104.

[36] Louisa M. Slowiaczek et al. "Information selection and use in hypothesis testing: What is a good question, and what is a good answer?" In: *Memory & Cognition* 20.4 (1992), pp. 392–405. ISSN: 1532-5946. DOI: 10.3758/BF03210923. URL: http://dx.doi.org/10.3758/BF03210923.

[37] SpaceMachine.net. *DATASETS OVER ALGORITHMS*. 2016. URL: http://www.spacemachine.net/views/2016/3/datasets-over-algorithms.

[38] Inc. StackExchange. *Stack Exchange Data Dump*. 2016. URL: https://archive.org/details/stackexchange.

[39] Stackoverflow.com. *How to Ask*. 2016. URL: http://stackoverflow.com/questions/ask/advice?.

[40] Stackoverflow.com. *What does it mean if a question is "closed" or "on hold"?* 2016. URL: http://stackoverflow.com/help/closed-questions.

[41] Stackoverflow.com. *What is reputation? How do I earn (and lose) it?* 2016. URL: http://stackoverflow.com/help/whats-reputation.

[42] Stackoverflow.com. *What topics can I ask about here?* 2016. URL: http://stackoverflow.com/help/on-topic.

[43] Stackoverflow.com. *What types of questions should I avoid asking?* 2016. URL: http://stackoverflow.com/help/dont-ask.

[44] Clayton Stanley and Michael D Byrne. "Predicting tags for stackoverflow posts". In: *Proceedings of ICCM*. Vol. 2013. 2013.

[45] H. Toba, M. Adriani, and R. Manurung. "Expected answer type construction using analogical reasoning in a question answering task". In: *Advanced Computer Science and Information System (ICACSIS), 2011 International Conference on*. Dec. 2011, pp. 283–290.

[46] Simon Tong and Daphne Koller. "Support Vector Machine Active Learning with Applications to Text Classification". In: *J. Mach. Learn. Res.* 2 (Mar. 2002), pp. 45–66. ISSN: 1532-4435. DOI: 10.1162/153244302760185243. URL: http://dx.doi.org/10.1162/153244302760185243.

[47] C. Treude, O. Barzilay, and M. Storey. "How do programmers ask and answer questions on the web? (NIER track)". In: *Software Engineering (ICSE), 2011 33rd International Conference on*. May 2011, pp. 804–807. DOI: 10.1145/1985793.1985907.

[48] Shaowei Wang, David Lo, and Lingxiao Jiang. "An Empirical Study on Developer Interactions in StackOverflow". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC '13. Coimbra, Portugal: ACM, 2013, pp. 1019–1024. ISBN: 9781450316569. DOI: 10.1145/2480362.2480557. URL: http://doi.acm.org/10.1145/2480362.2480557.

[49] Alexander Wissner-Gross. *Datasets Over Algorithms*. 2016. URL: https://www.edge.org/response-detail/26587.

[50] Jinzhong Xu, Yanan Zhou, and Yuan Wang. "A Classification of Questions Using SVM and Semantic Similarity Analysis". In: *Internet Computing for Science and Engineering (ICICSE), 2012 Sixth International Conference on*. Apr. 2012, pp. 31–34. DOI: 10.1109/ICICSE.2012.49.

[51] Jie Yang et al. "User Modeling, Adaptation, and Personalization: 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings". In: ed. by Vania Dimitrova et al. Cham: Springer International Publishing, 2014. Chap. Sparrows and Owls: Characterisation of Expert Behaviour in StackOverflow, pp. 266–277. ISBN: 9783319087863. DOI: 10.1007/978-3-319-08786-3_23. URL: http://dx.doi.org/10.1007/978-3-319-08786-3_23.

[52] Show-Jane Yen et al. "A support vector machine-based context-ranking model for question answering". In: *Information Sciences* 224 (2013), pp. 77–87. ISSN: 0020-0255. DOI: http://dx.doi.org/10.1016/j.ins.2012.10.014. URL: http://www.sciencedirect.com/science/article/pii/S0020025512006792.

[53] Dell Zhang and Wee Sun Lee. "Question Classification Using Support Vector Machines". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. SIGIR '03. Toronto, Canada: ACM, 2003, pp. 26–32. ISBN: 1581136463. DOI: 10.1145/860435.860443. URL: http://doi.acm.org/10.1145/860435.860443.

[54] Zhihua Zhang, Guoshun Wu, and Man Lan. "ECNU: Multi-level Sentiment Analysis on Twitter Using Traditional Linguistic Features and Word Embedding Features". In: *SemEval-2015* (2015), p. 561.

A Appendix

A.1 Acronyms

AI Artificial Intelligence. 3, 4

IR Information Retrieval. 6

ML Machine Learning. iii, 3, 4, 6, 7

NLTK Natural Language Toolkit. 12

Q/A Question-Answering. 6

SVM Support Vector Machines. iii, 1–4, 6, 7, 14

A.2 Data sets/Statistical Overview

A.3 MySQL Database

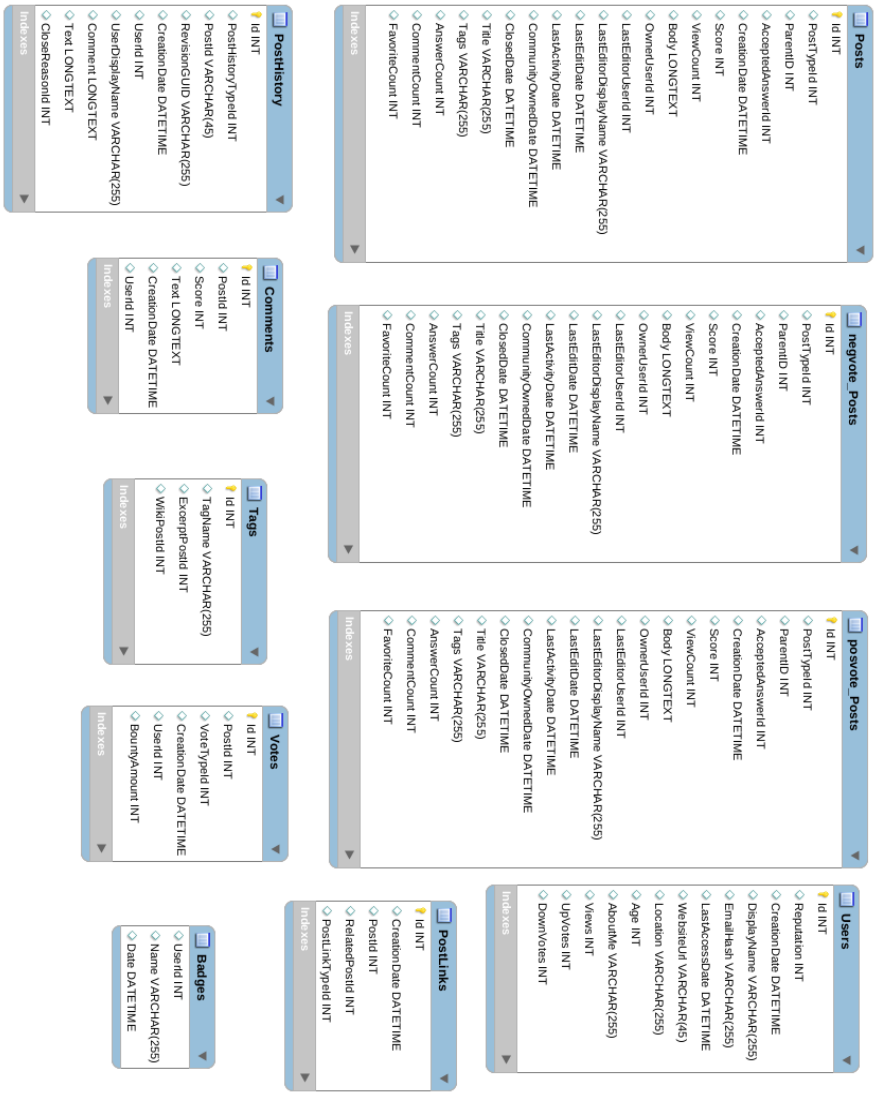

Figure 2: MySQL Database used for dataset