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

Acknowledgement

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 the Support Vector Machines (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

Stack Overflow (SO) is today for many developers a well known Question-Answering (QA) system. However, SO has a high requirement to the questions and answers posted, which is reflected through their voting and reputation system. This peer-review processes can be used as an indicator to a questions quality, where questions with high up-votes can be defined as good questions. In this thesis, a system has been developed using Machine Learning (ML) and Support Vector Machines (SVM) to see if it is possible to predict whether or not a new question will be considered as a good or bad question.

This was achieved by using the Stack Exchange (SE) data set, specifically using the one for SO. Questions were dived into two classes, where bad questions was question with a vote score below zero, and good questions were those above zero. Based on content in the various questions, a set of feature detectors was developed and tested against the raw data set. Surprisingly, the features actually lowered the accuracy score (the raw data set had an accuracy of XX%, and the ones using all the feature detectors had an accuracy of XX%).

Add numerical values, and also if time add comparison on those questions which only contained the actual feature

Contents

List of Figures

List of Tables

1 Introduction

Today, many uses the Internet as a resource to find answers to their questions and problems. In the past, one were often restricted to only use keywords and not being able to pose the problem as you would when asking another human being. Most search engines today can handle natural language queries, which makes it easier to find the answer you are looking for. The Internet offers a wide range of resources to acquire new knowledge, everything from encyclopaedias to blogs, forums and Question-Answering (QA) communities. One well known QA community is the Stack Exchange (SE) community, which is built upon the same model as Stack Overflow (SO). SE has grown large since its release in 2009, and now contains 154 different communities.

check year/ date

As a developer, one often find oneself in the situation that a part of the code does not work, you get weird error messages, or you are simply stuck. This is were SO comes in. SO is a part of the SE community, although SO was actually released before SE. Jeff Atwood and Joel Spolsky wanted to offer programmers a QA site where they could get the answer they wanted without having to read through a lot of text, see others posting "I also have the same issue" or having to subscribe and pay to see the solution [39]. Question (and answer) quality is maintained through the use of a peer-reviewed gamification system, where users are awarded with votes, reputation and badges for their participation[39]. One of the requirements is that the questions should be of good quality [41, 43, 44]. If a question is bad, users can vote to close or delete it (in which the question will be put on hold). A question can be put on hold or closed if they meet any of the following criterias: Exact duplicate (same question has been asked before), off-topic (not related to SO), unclear what is being asked, too broad (e.g. could write a book about question being asked) or primarily opinion-based [42, 8].

add cites here

1.1 Problem description

Most of the systems that have been developed so far focuses on finding the best answer to a question asked by the user. Few, if any, focuses on the quality of the question being asked. What defines a good question, and can we in anyway predict whether or not a new question posted on SO will be considered good or bad by the community? There are many users who has either a negative view or relationship in regards to SO. Many experience that their questions gets down-voted, closed or even deleted. For some, they simply do not know how to ask an acceptable question. Questions related to homework are one example of questions that are not accepted on SO. There is even a post on Meta.StackExchange

add ex-am-ples here

discussing whether or not it should be acceptable to use greetings and sentiments in posts[7]. Therefore, the question becomes: What is and is not a valid question on SO?

1.2 Research questions

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

1.3 Methodology to be used

The theoretical background in this thesis is mainly focused on Question Classification (QC) and similar research in relation to SO. What has been the focus of other researchers, and in what way did they proceed to solve their questions? The analysis of the questions are done by using the publicly available database dump, which is available via SE archive[1] [40] . There are several others who have used the same dataset . Taking into consideration that SO was released in 2008, it means that it now contains approximately 8 years of peer-revied data. Because of the size of the data set, and the total amount of posted questions, going through all questions manually would be too timeconsuming. Therefore only a select few were studied too see if it was possible to identify what separated the highly up and down-voted questions.

add cites here

The goal was to develop a Machine Learning (ML) learning system which was based on SVM, since many papers document that this has the best classification accuracy for text classification. The methodology therefore also includes a documentation on the development process, and how and why the given features used were selected.

For the sake of replicability, and also be able to undo potential errors, the system is available in a a GitHub repository[2]. In addition to the source code, the repository also contains both the samples that was used (stored in CSV files), and the models that was created.

1.4 Justification, Motivation and Benefits

Many systems focuses only on finding a good answer, and does not ask if it is a good question. As a famous Norwegian saying goes[3]: "A fool may ask more than ten wise men can answer". This means that new research possibilities could be opened up in relation to researching question quality by expanding the system.

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

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

[3] Although its origin comes from a Danish word collection from 1682: https://snl.no/En_d%C3%A5re_kan_sp%C3%B8rre_mer_enn_ti_vise_kan_svare.

Since all the communities within SE is based on the same model, few modifications would be needed to scale the program to be used within the other communities. As noted in several papers , question quality is measured based on the amount of votes given. Which can also be compared against the peer-review process in academia, and given that SO targets professionals and experts, using SO as a scientific reference is not that far-fetched (in CITE, XX% used SO as a reference). SE has also been the focus of various researchers these past years [49]. Improving ones own ability to ask better questions can also have a pedagogical effect, which means that this system could be implemented in education.

add cites here

1.5 Limitations

The greatest limitation is the time available. A large amount of time was spent on setting up the database, and retrieving the questions (the Posts table contains both questions and answers). Only a selection of the questions were selected (a total of 20,000 questions), and training the SVM over one sample set can easily take several hours. This also has an impact on classification accuracy, since in some cases there is only a small amount of the questions that contains a given feature (e.g. the hexadecimal feature, which only was present in 160/20,000 questions). A limitation is also that the focus is only on SO, which means that one would need to make additional adjustments and add more filtering to account for the differences that may occur in each community.

1.6 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. Instead of posting bad questions that can get downvoted or closed, the developed system could be able to give feedback to the questions quality. Furthermore, the research presented could open up for new research in relation to how we ask questions online, and in what ways these best can be analysed. It can also be used for educational purposes, e.g. having questions iteratively improve their question quality by asking the system questions.

1.7 Thesis structure

The following is the structure of this thesis:

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

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.

Citation list (content to come):

- [5]: Libsvm
- [52]: Question classification, SVM, semantics
- [15]: SVC
- [24]: Sigmoid kernels SVM
- [54]: SVM, QA, Context ranking
- [29]: Question taxonomy
- [17]: Japanese Q/A System
- [26]: Reputation system SO
- [4]: Text classification, semantics
- [12]: fuzzy sigmoid svm
- [56]: sentiment analysis, linguistics
- [34]: Answer quality in qa community
- [46]: Answer type construction
- [48]: QA on the web (programmers)
- [37]: What is a good Q/A?
- [20]: predict closed questions on SO
- [45]: tag prediction on SO
- [16]: Question classification
- [55]: Question classification, svm
- [3]: random search, hyperparameters
- [47]: SVM, text classification
- [14]: svm
- [18]: svm, text processing
- [9]: Salton
- [28]: semi-supervised, question classification
- [31]: problem solving question, cs
- [25]: wordnet
- [27]: good code example, SO
- [13]: SO expertise & knowledge

- [50]: dev interaction SO
- [53]: expertise vs activity on SO
- [6]: question feature for answer attraction SO
- [35]: tag recommendation SO
- [30]: questions, answers, activity and reputation on SO
- [1]: stackexchange, technical vs non-technical communities activity
- [11]: domain ontology, qa
- [21, 22, 23] qa classification, semantics
- [2] so, qa, post quality
- [36] qa, design, anwer response
- [19, 38, 51] datasets

In the paper by Toba et al. [46], they experiment with the use of statistical learning to find the expected answer pattern for factoid QA 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. [52] used SVM to create an online QA 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 [55] ("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).

[1] Low precision can occur if the feature sets are not fitted well enough during classifier training [46, p. 283].

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 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. [19, 38, 51]

3 Methodology

3.1 Dataset and MySQL Database

3.1.1 Dataset

The dataset contains all information that is currently available in the SE 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 SE, 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.

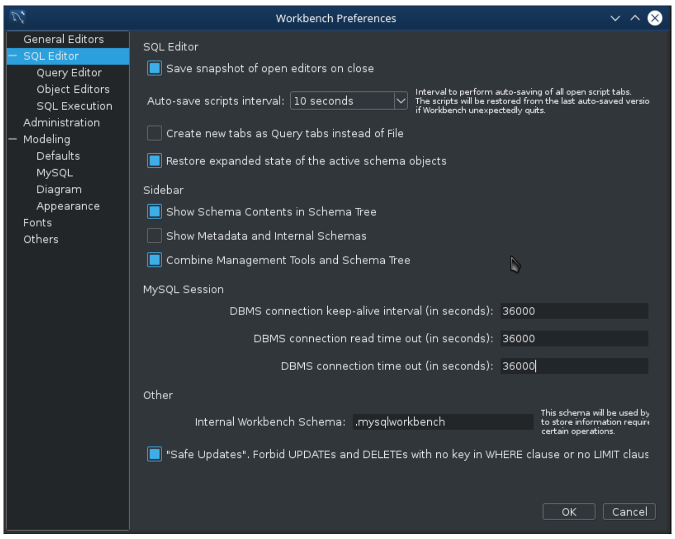
Listing 3.1: Content in stackoverflow.com-Tags.xml

```
<?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).



SQL Editor

[✓] Save snapshot of open editors on close

Auto-save scripts interval: 10 seconds — Interval to perform auto-saving of all open script tabs. The scripts will be restored from the last auto-saved versio if Workbench unexpectedly quits.

[ ] Create new tabs as Query tabs instead of File

[✓] Restore expanded state of the active schema objects

Sidebar

[✓] Show Schema Contents in Schema Tree

[ ] Show Metadata and Internal Schemas

[✓] Combine Management Tools and Schema Tree

MySQL Session

DBMS connection keep-alive interval (in seconds):	36000
DBMS connection read time out (in seconds):	36000
DBMS connection time out (in seconds):	36000

Other

Internal Workbench Schema: .mysqlworkbench — This schema will be used by to store information requir certain operations.

[✓] "Safe Updates". Forbid UPDATEs and DELETEs with no key in WHERE clause or no LIMIT claus

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 were 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 [10]).

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 [33] [32, 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) 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 setting it to 0.01, words that appeared in less 1% of all documents were ignored.

To be able to run the system without relying on an Integrated Development Environment (IDE), making it run from the Terminal using basic command setup seemed like a good idea. At first optparse was used, which ironically turned out to be deprecated and replaced by argparse. However, the problem was that you could only run one command at a time, whereas I wanted the program to be able to run until exited. The reason for this was because it needs to load a model before it can make a prediction, in addition the user might want to predict multiple questions. This was therefore replaced with a basic while loop that runs until the users enters the exit command. The setup used for argparse was kept, so users from *nix system might be more familiar with similar commands (shown in Listing 3.5). At the end, there were some commands that were not added to the menu, since these were mostly used for testing.

Listing 3.5: System menu

```
Menu:
d: Loads default model (if exists) from ./pickle_models
e: Exit the program
h: Displays this help menu
l: Load user created model. Arguments:
        path: Path to directory with model(s) (e.g. /home/user/my_models/)
        filename: The models filename
```

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/

suffix: File type – Optional (default: '.pkl')
p: Predict the quality of the entered question. Arguments:
question: Question to predict quality of
t: Train a new model based on an existing (or new) data set. Arguments:
path: Path to directory with training data (e.g. /home/user/my_data/)
filename: Filename for data set (model name will be the same as this)
db_load: Load from database (Enter 0: No, 1: Yes)
limit: Limit for database row retrieval (integer) – Optional unless 'db_load' is '1'
u: Create an unprocessed data set based on database content (from database set in dbconfig.py). Arguments:
filename: Filename for data set (model name will be the same as this)
limit: Limit for database row retrieval (integer)
feature_detectors: Create singular feature detectors based on data set? (Enter 0: No, 1: Yes)
create_model: Create classifier model(s) based on data set?
0: No, 1: Unprocessed model, 2: Feature detector model(s) 3: Both (1 and 2)

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. Therefore, the score was then set to less than -5. What was also found when using pandas.Categorical to get an overview (code snippet in Listing 3.7 and result in Table 1), one can see that for the 10,000 bad questions, the average vote score was -7. This could be an indicator that when a question has a vote score below -5, they are ignored.

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 3.6: 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'))
```

To be able to develop some theories on what the difference between good and bad questions was, a total of 200 questions were reviewed (by sorting questions based on votes[15]). It was easier to see certain patterns in down-voted questions rather than those that were up-voted. A repetitive pattern was that many had either no code example, or poorly written code. These questions could also show

[15]http://stackoverflow.com/questions?sort=votes

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.

indications of not having tried anything, or that they were based on either homework or school assignments. This in turn lead to a hypothesis that if a question contains indicator of word synonyms for homework[16], it would be considered a bad question. In addition, some code examples had syntax errors, which made the minimum working example (MWE) not executable. Some questions also contained links, either to external resources or indicators of potential duplicates. Therefore links was also considered a potentially useful feature. Tags was also considered as a feature, which was divided into two: Attached and External tag. Attached tags are tags which the user has linked to the question, whereas external tags are all the tags available on SO. Version numbering was also considered, but this was not included due to the complexity of writing a proper filtering method to account for all possible variations.

Features were added in the same manner as was done for code samples, numerical and hexadecimal values. However, there were some issues when attempting to replace the tags and the synonyms for homework. At first, WordNet was used for synonyms (using wordnet.synset()). The only problem was that for the word 'homework', wordnet.synset() only returns ['homework', 'prep', 'preparation']. Whereas Thesaurus[17] had a lot more suggestions, and was therefore used instead. Words were selected based on whether or not it was plausible that they could be used in programming related question setting. A new problem now arose, namely the issue that the word "assignment" did not necessarily need to occur in a homework setting, since it could also be used as a programming word (e.g. assignment operator[18]). Therefore features for homework were split into two types: 'has_homework' and 'has_assignment'.

Tags were without a doubt one of the most annoying features to detect and replace. Site tags (or external) are single text values in the database, whereas the question can have up to five tags attached. Those attached tags are then

[16] http://www.thesaurus.com/browse/homework
[17] http://www.thesaurus.com/browse/homework
[18] http://stackoverflow.com/questions/5368258/the-copy-constructor-and-assignment-operator

separated in the following format: "<c><multi-threading>", which had to be processed by removing the '<' and the '>'. After the removal, each tag value was added to a list, so that all attached tags was indexed based on the question they belonged to. Furthermore, a combination of string replacement and regular expression was needed. The regular expression was used for single character tags (e.g. 'C'), and word replacement for longer words. The reason for this was that when using string replacement, single character tags replaced occurrences even if they appeared in the middle of a word. If the tags contained characters that could be interpreted as a regular expression (e.g. C++), it would give error about multiple repetitions. In addition, the tags needed to be sorted based on their length, since for questions that contained tags which included both <C> and <C++>, if <C> came first, it replaced the <C++> with 'has_*_tag'++. The text also had to be converted to lower-case to ensure proper tag matching.

Listing 3.7: Getting Categorical data from pandas.DataFrame

```
for word in word_set:
    if len(word) == 1:
        # if its only one character (e.g. 'C'), ensure that it is a singular word by using regex
        text = re.sub(r"\b%s\b" % word, replacement_text, text, flags=re.IGNORECASE)
    else:
        text = text.replace(word, replacement_text)
```

4 Discussions

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)
Paths, Windows vs. Linux, parallellization, gpu_count, etc

4.1 Data and Testing

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

4.2 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?

4.3 Implementation Architecture

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

4.4 Limitations and other issues

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

5 Conclusion/Summary

5.1 Overview of main results

basically what the title says; a summary of the results

5.2 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&#38;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: 978-1-4503-3411-2. 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: 978-1-4503-1462-6. 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] CommunityWiki. *Should ’Hi’, ’thanks’, taglines, and salutations be removed from posts?* 2016. URL: http://meta.stackexchange.com/questions/2950/should-hi-thanks-taglines-and-salutations-be-removed-from-posts (visited on 05/07/2016).

[8] CommunityWiki. *What is a "closed" or "on hold" question?* 2016. URL: http://meta.stackexchange.com/questions/10582/what-is-a-closed-or-on-hold-question (visited on 04/05/2016).

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

[10] Eloff. *Strip HTML from strings in Python*. 2009. URL: http://stackoverflow.com/a/925630 (visited on 03/22/2016).

[11] 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.

[12] 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.

[13] 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: 978-1-4503-1051-2. DOI: 10.1145/2141512.2141550. URL: http://doi.acm.org/10.1145/2141512.2141550.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] Gary Klein. *Blinded By Data*. 2016. URL: https://www.edge.org/response-detail/26692 (visited on 04/24/2016).

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

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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: 978-1-4503-1868-6. DOI: 10.1145/2445196.2445253.

[32] Scikitlearn.org. *4.2. Feature extraction*. 2016. URL: http://scikit-learn.org/stable/modules/feature_extraction.html (visited on 05/07/2016).

[33] Scikitlearn.org. *sklearn.feature_extraction.text.CountVectorizer*. 2016. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html (visited on 05/07/2016).

[34] 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: 978-1-4503-0153-4. DOI: 10.1145/1835449.1835518.

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

[36] 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: 978-1-4673-2936-1. URL: http://dl.acm.org/citation.cfm?id=2487085.2487104.

[37] 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.

[38] SpaceMachine.net. *DATASETS OVER ALGORITHMS*. 2016. URL: http://www.spacemachine.net/views/2016/3/datasets-over-algorithms (visited on 04/24/2016).

[39] Joel Spolsky. *Stack Overflow Launches*. 2008. URL: http://www.joelonsoftware.com/items/2008/09/15.html (visited on 05/06/2016).

[40] Inc. StackExchange. *Stack Exchange Data Dump*. 2016. URL: https://archive.org/details/stackexchange (visited on 04/05/2016).

[41] Stackoverflow.com. *How to Ask*. 2016. URL: http://stackoverflow.com/questions/ask/advice? (visited on 04/05/2016).

[42] Stackoverflow.com. *What does it mean if a question is "closed" or "on hold"?* 2016. URL: http://stackoverflow.com/help/closed-questions (visited on 04/05/2016).

[43] Stackoverflow.com. *What topics can I ask about here?* 2016. URL: http://stackoverflow.com/help/on-topic (visited on 04/05/2016).

[44] Stackoverflow.com. *What types of questions should I avoid asking?* 2016. URL: http://stackoverflow.com/help/dont-ask (visited on 04/05/2016).

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

[46] 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.

[47] 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.

[48] 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.

[49] Bogdan Vasilescu. *Academic papers using Stack Exchange data*. 2012. URL: http://meta.stackexchange.com/questions/134495/academic-papers-using-stack-exchange-data.

[50] 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: 978-1-4503-1656-9. DOI: 10.1145/2480362.2480557. URL: http://doi.acm.org/10.1145/2480362.2480557.

[51] Alexander Wissner-Gross. *Datasets Over Algorithms*. 2016. URL: https://www.edge.org/response-detail/26587 (visited on 04/24/2016).

[52] 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.

[53] 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: 978-3-319-08786-3. DOI: 10.1007/978-3-319-08786-3_23. URL: http://dx.doi.org/10.1007/978-3-319-08786-3_23.

[54] 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.

[55] 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: 1-58113-646-3. DOI: 10.1145/860435.860443. URL: http://doi.acm.org/10.1145/860435.860443.

[56] 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
QA	Question-Answering. iii, 6
SO	Stack Overflow. iii
SVM	Support Vector Machines. ii, iii, 1–4, 6, 7, 14

A.2 Data sets/Statistical Overview

A.3 MySQL Database

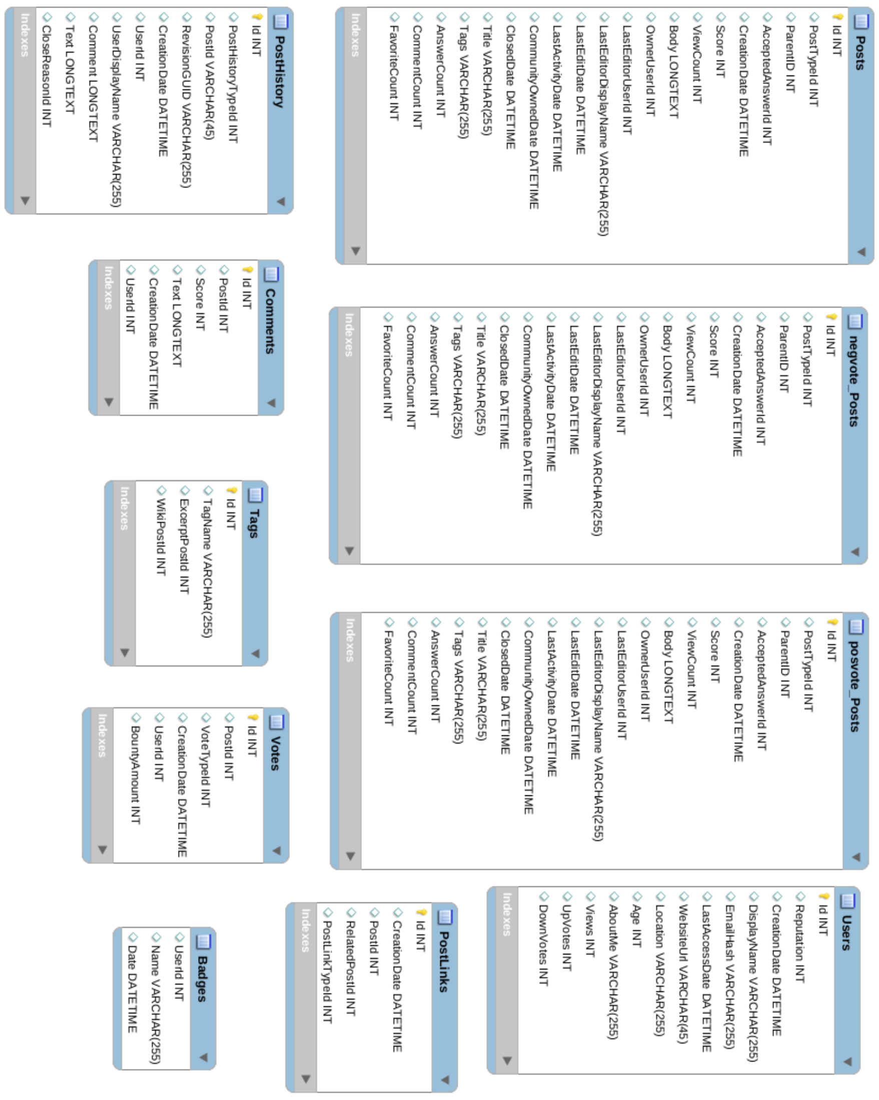


Figure 2: MySQL Database used for dataset