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) [1]. 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 [44]. 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[44][27]. One of the requirements is that the questions should be of good quality [46, 48, 49]. 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 [47, 9].

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 examples here

discussing whether or not it should be acceptable to use greetings and sentiments in posts[8]. 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] [45] . 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-reviewed data. Because of the size of the data set, and the total amount of posted questions, going through all questions manually would be too time-consumings. 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/End%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[4]. SE has also been the focus of various researchers these past years [53]. 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

[4] Posnett et al. [35, p. 1] noted that SO "ranked 2^{nd} among reference sites, 4^{th} among computer science sites, and 97^{th} overall among all websites".

2 Related work

2.1 Stack Overflow and gamification

Gamification is defined as "the use of game design elements in non-game contexts" [10], and is a part of Serious Games (e.g. educational, pedagogical, awareness, etc). Furthermore, SO does indeed fulfil the needs for the four player types[1] Achievers can farm reputation and badges through posting new and interesting questions, or offer answers to new questions. Explorers can explore the site by searching for new and interesting questions (or answers) based on tags, votes, or other search parameters. Socializers are able to share their knowledge and engage in discussions by either offering answers or comments. The only type left are Killers, who basically just are out to ruin the fun for others (e.g. Player-vs-Player in online games) [28, p. 3]. The closest representation for this group would be trolls, or people who are simply ignorant of whether or not the information they share is correct. However, this is not a group that would be widely represented, due to the strict rules and desire for quality. Voting also has a penalty, such that if you down-vote a post, you lose some of your reputation.

One of the founders, Jeff Atwood said in an interview that he wanted users to not just give good answers, but also trick them into improving their communication skills [35].

[57] researched user activity on SO, where they classified users into two groups; sparrows (achievers) and owls (socializers).

"Linus' Law" - claiming that "with many eyeballs, all bugs are shallow"; suggesting that program bug-fixing efforts are facilitated by the participation of more users. Stack Exchange can be viewed as an instance of an online learning community, where individuals help each other gain expertise, both in terms of domain knowledge, and in answering questions well and to the point. It has long been argued that people learn better in co-operating groups than on their own. Important design goals of learning communities: help users find relevant content; help users find people with relevant knowledge and motivate people to learn. [35]

Small subset of users is responsible for the majority of the contributions, and ultimately, for the success of the Q/A systems. Standard expert identification methods often misclassify very active users for knowledgeable ones, and misjudge activeness for expertise. Contributes a novel metric for expert identification, providing a better characterisation of users' expertise by focusing on the

[1] The four player types are Achievers, Explorers, Socializers and Killers [28, p. 3].

quality of their contributions. Identifies two classes for the users, sparrows and owls, and describe several behavioural properties in the context of the StackOverflow Q/A system. Reasons for signing up/using qa systems: 1) to look for existing solutions to their issues; 2) to post a new question to the platform community; 3) to contribute by providing new answers 4) to comment or vote existing questions and answers Q/A platforms employ effective gamification mechanisms that motivate users by showing a public reputation score (calculated by summing the number of preferences obtained by all the posted questions and answers), and by assigning badges after achieving pre-defined goals (e.g. complete at least one review task, achieve a score of 100 or more for an answer). Used the StackOverflow dataset. Sparrows: An important part of the qa system. Numerous, active and highly "social". However, that does not mean that they are knowledgeable. Can be driven by gamification, wanting to score votes and reputation instead of giving informative and long-lasting answers. Can have low average score, and minimization of effort by targeting non-relevant and simple questions. Can guarantee responsive and constant feedback; helps in keeping the community alive. Owls: Active members that participate not for the achievements and gamification, but wants to share knowledge and information related to the platform. Experts in the related topic, providing useful answers and can show expertise by answering questions seen as difficult, complex or important to the users of the community. An expert can be defined as someone who is recognised to be skilful and/or knowledgeable in some specific field (based on judgment by the public or peers; e.g. academic peer-review). Expertise then refers to the characteristics, skills, and knowledge that distinguish experts from novices and less experienced people. In qa systems, social judgement is critical for expert identification. Question is answered by users, which are up or down-voted by other users. Answering questions can reflect how a user applies their knowledge to solve problems (actionable knowledge), and the votes for the answer can be seen as a cyber simulation of social judgement for the answerers' expertise level. Asking questions and posting comments can also say something about a users expertise level. Considered two dimensions: 1. The debatableness of a question, measured according to the number of answers it generated; 2. The utility of an answer, measured according to its relative rank in the list of answers. Intuitively, difficult questions generate a lot of discussions, and several answers; also, the higher in the rank an answer has been voted, the more potentially useful it is to solve the related question, and the more it provides evidences about the expertise of the answerer in the topic. The ratio between answered and submitted questions is significantly higher for sparrows. Owls, on the other hand, show a behaviour more similar to average users, thus further highlighting the distinctive "hunger" for answers of sparrows. Owls answer questions that are more difficult, and more popular. Question popularity, measured in terms of the number of times a question has been viewed in StackOverflow;

and time to solution, measured in terms of the number of hours needed for the question creator to accept an answer as satisfactory. Time to solution can also be an indicator of the difficulty of a question: intuitively, the longer the time to accept an answer, the more difficult is the question. Sparrows appear focused in building their reputation, which they increase by consistently answering to a lot of easy and non-interesting questions. Their behaviour is however providing important contribution to the community, as they can guarantee fast answers to many questions. On the other hand, owls intervene when their expertise is needed the most, i.e. in difficult question. Owls post questions that are more difficult, and more popular; Questions submitted by sparrows are less popular than those posted by the owls. Completion time for such questions is comparable. These results also suggest a difference in the expertise level of the two groups of users, as more popular questions might be a sign of the better understanding that owls possess on the subject. However, the higher (on average) difficulty and popularity of sparrows's answers w.r.t. the average of users, also suggests that sparrows are good contributors in terms of new problems to be addressed by the community. Gamification incentives can more effectively retain sparrows than owls. "Older" owls always contribute for the larger portions of the provided answers. However, owls consistently tend to decrease their activity in time, especially for more recently registered users. On the other hand, new sparrows significantly contribute to a share of answers produced by their group and, although in the long term a decrease in the overall activities of the older member can be seen, the effect is less important. These results suggest that the gamification incentives put in place by StackOverflow are really effective to retain the activity of sparrows. [57]

2.2 What is the definition of a question?

Question generation depends on the system to which it is embedded. Questions are intended to evaluate knowledge, understanding and skills. Socratic tutoring: Questions should help students understand something they did not understand before. Help system: System should learn what resulted in the request for help. Paper focuses on describing question taxonomy for tutoring systems (validated via human tutoring transcript analyses). Goal was to detail HCI design. Mapping from classification of learner interactions to tutor response. Revised the taxonomies considering redundancy, usefulness, and completeness. If two labels had the same meaning they were merged. When categories differed, the most generic was selected.-> Mostly primary taxonomy, p. 2, Figure 1 that is relevant for my thesis [34]

Hypothesis testing includes information selection (asking questions) and the use of information (answers to questions asked). Two experiments using different kinds of inferences (category membership of individuals and composition of sampled populations). Third experiment showed that certain information-

gathering tendencies and insensitivity to answer diagnosticity can contribute to a tendency toward preservation of the initial hypothesis. Critical elements in hypothesis testing is having a good question, and knowing what to do with the answer. Gives example of a fictional planet which contains only two creatures. These can only answer 'yes' or 'no', but they will answer truthfully. Based on a given feature list, what question would you ask to be able to classify the creature you encounter? This problem can be re-formulated into a Hypothesis, where H0 and H1 is either 'yes' or 'no' (or more defined, the race is...). Selection of hypothesis based on percentage/probability distribution, e.g. choosing those features that stands most out/is highly separated. Also a question on what type of answers (and how many) one can expect from each individual. Easy to assume that a good question will get a good answer (e.g. overestimation of yes/no when deciding which race creature belongs to). Sensitivity in regards to the yes/no answer given (e.g. treating all cases as identical, etc). [42]

2.3 Question-Answering (QA)

Question-answering (qa) system which includes question analysis, information retrieval and answer extraction. Question category is an important part of the question analysis, the same goes for follow-up process, since it affects the accuracy of the answer extraction. Challenging, since few language processing tools handle Chinese characters. Better to classify based on the domain of the question characteristics. Original method for question classification is usually rule-based approach, where the rules decide the category. Rule-based approach uses interrogative words and word combinations with other features of the rules extracted by experts. Difficult to extract these rules and impossible to make all of them, which will have an impact on classification results. Used RBF kernel in this experiment. Notes that the selection and organization of features is the main issue when working with classification. Features and feature space can have a significant impact on both accuracy and efficiency of the classifier. Compared with rule-based, question features for a specific domain can have greater benefits. To improve performance, the classification must improve the question characteristics. Their feature selection was based on semantic analysis and lexical analysis. They also needed to add a step for word segmentation and Part-Of-Speech (POS) tagging (since it was Chinese text processing). To improve classification efficiency, domain knowledge was added by using domain term concept hierarchy (basically if words had the same meaning or concept, then they were added to a feature vector). They used LIBSVM for coarse classification. The sub-classification was done by measuring the similarity between the users question and those in the sub-categories (calculated by utilizing word similarity based on term concept hierarchy). For the SVM, they used cross-validation, where the training set was divided into five parts of equal size. After a classifier had been trained on the first four, the last was used to get the cross-validation accuracy (to ensure

correct classification). [56]

QA is a method for finding the answer to a given question from an unknown amount of documents. The goal of most automatic qa systems is to find the exact answer to the question asked by the user. Existing QA technology involves two main steps: information retrieval (IR) and information extraction (IE). IR is used to retrieve the relevant documents after completed analysis or classification. IE then processes the retrieved documents to find the answers the user is looking for. High quality systems generally requires multiple external components. The purpose of question classification is to detect the answer type of the input question. Looking for answers in a small set is easier then searching all documents. By using a document retriever, only documents related to the question is retrieved, where the passage retriever divides the documents into paragraphs and ranks them based on the given evaluation metric(s). [58]

Fine-grained answer taxonomy improves qa performance, but difficult to create accurate answer extraction because ML methods require a lot of training data and training rules. They found that named entity/numerical expression recognition and word sense-based answer extraction contributed to system performance. The mainstream of the system is composed of four modules: Question Analysis, Document Retrieval, Answer-Extraction, and Answer Evaluation (AEv has several sub-modules). The question analysis module normalized question and determined answer types. Question normalization was used to simply the "answer-type determination rules". Useless expressions were removed from the question. Expressions with the same meaning were normalized into one. Abandoned use of TF-IDF based paragraph retrieval because the paragraphs were too short to cover all query terms. Same problem can occur when using passage retrieval modules. If it is too short terms will not be covered, if it is too long, passage scores will not reflect the density distribution.Answer extraction separates based on the candidate it belongs to (e.g. Location, organization, school, hospital, etc). Suffixes are then used to indicate which candidate group it belongs to. If it belongs to more than one, it is added in both. This is called suffix constraint rule. Use of noun-phrases in combination with suffixes.[19]

qa sites provides a place where people can ask either general or domain specific questions. Domain specific qa sites requires users to be familiar with both the site, rules and questions that can be asked. qa sites also works as an archive of knowledge which can be accessed later on. Mostly expert users that answers questions. [31]

qa is about getting an answer to a question, not a list of documents. qc maps a question to a category (pre-defined), which specifies the expected answer type. Focus is on fact based questions. Class determination can reduce feature space, in addition to specify the search strategy. A problem with qc learning systems is the high dimensional feature space (typically caused by the n-grams over all the vocabulary words). In qc, question is represented using vector space model,

the question is a vector which is described by the words inside it. Therefore, a question x is represented by vector x = (x1, ..., xd) in which xi is the frequency of term i in x and d is the total number of terms. This representation is also referred as bag-of-words or unigrams which is the simplest type of features that can be extracted from a question and is the most widely used feature space in document classification [25]

Even though expressive content and descriptive semantic content are distinct, sentiment information can be lost. It can learn that two words are close (similar/equal meaning), but cannot understand that two other words are the complete opposite (negative) of those words. 25,000 movie reviews from IMDB, including at most 30 reviews from any movie in the collection. built a fixed dictionary of the 5,000 most frequent tokens, but ignored the 50 most frequent terms from the original full vocabulary. Stemming was not applied because the model learned similar representations for words of the same stem when the data suggests it. Allowed non-word tokens (e.g. '!', ':-)'), since it could be sentiment relevant. Mapped ratings to [0,1]. Semantic component did not require labels. [29]

2.4 Stack Overflow (SO)

Focus is on developing indicators for problems and experts. Examine how complex problems are handled and shared between experts. Answering questions should be parallel to the experts knowledge, e.g. complex problems are handled by (domain) expert users, and basics by those that know the answer. Duration between the when a question was first posted and when an answer was accepted by the poster as a proxy for difficulty. Extracted events for the questions, to construct a language to describe a given questions activity. Used the following grammar: - Actor Type: Questioner (user posting the question), Accepted Answerer (user posting accepted answer), Unaccepted Answerer (users posting the other answers), and Someone (participants in the question (e.g. comments)). - Action: Posting, Commenting, Editing, Answering, Accepting, Voting Up, and Voting Down. - Object: Question or an Answer. [15]

Study of the user system on StackOverflow. Analysis of SO's reputation system. Although most of the questions are asked by low reputation users, the average shows that high reputation users asks more questions. Active users are rewarded with reputation (rep. score depends on activity). Before, asking questions gave more reputation, now giving answers give more reputation. Expert users can be identified not only by reputation, but also by the badges they have. Showed that users are awarded more reputation for good answers than good questions. Users with high reputation is considered as expert users (since answers give more rep). "System reputation can be considered as a measure of expertise". [31]

"The content pre-processing step takes in both normal text and code, performs tokenization, stop word removal, and stemming. Tokenization breaks a

paragraph into word tokens. Stop word removal removes commonly used words like: is, are, I, you, etc. Stemming reduces a word to its root form, e.g., reading to read, etc. For the code, we remove reserved keywords such as: if, while, etc., curly brackets, etc, and extract identifiers and comments. These are then subjected to tokenization, stemming, and stop word removal too." [54]

Many Q&A sites employ voting and reputation mechanisms as center pieces of their design to help users identify the trustworthiness and accuracy of the content. Since a lot of qa sites are focusing on having at least some domain experts, the questions and answers now have a lasting value. Since these are archived and quality measured by voting system, search engines can rank the answers. This way, users who have never been to the site can find questions similar to their problem, and in addition see suggestions to other solutions from the others who may have posted their answer. First principle: Generally expert users answers first. Latent "pyramid", where expert users are at the top. Question starts at the top, being looked at by expert users. Then it gets filtered down if unanswered. Second principle: Higher activity level signals not only question interest, but it is also beneficial to all answers given (evaluation and reputation). High activity can correspond to lasting value of a question. Subjective questions are frowned upon by SO community. Deep expertise and domain knowledge is thus often essential to providing a good answer. The longer a question goes unanswered, the more likely it is that no satisfactory answer will be given (i.e. no answer will be accepted). Expert users are more prone to be "answer-dominant", gaining reputation from answering questions. Users with over 100K reputation earn more from accepted and less from up-votes. Idiosyncrasy on SO, can only gain 200 rep points daily, after that you can only gain reputation by accepted answers or bounty. Only high-rep users hit the daily cap. Questions on Stack Overflow are supposed to be answerable factually and objectively (if not, they are marked as a "Community Wiki" question and actions on them do not count towards reputation score). This creates an incentive to answer quickly, since it is likely that the first correct answer may well be accepted. [2]

Growing participation in online qa forums. Classify online communities on StackExchange into two genres; technical and non-technical. Examines the effect of contribution by comparing differences between technical and non-technical communities (user participation). Started with qa for computer programming (SO, started in 2008), which later expanded by using the SO model, which is today known as StackExchange (SE). Forums can be categorized into technical (largely professional or problem based) and non-technical (largely hobbyists or interest based). A few highly active users are responsible for the majority of participation and overall users can be characterized as lurkers, help-seekers (askers) and givers (responders) In (Anderson2012), authors found that, the probability of a response being chosen as the best one, depends on temporal characteristics of its arrivals, like response speed which could be applicable to predict long-term

value of a post. Besides, the other work (Sinha2013) shows that, successfully of being answered about expert topic is not only because of the technical design, but also the regular involvement of design team of that community. Mentions reputation system (motivation factor). Compared the forum for code-review forum (CR) as technical instance and bicycle forum (BC) as non-technical one. Found strong correlation between number of answers given and reputation score achieved by a user which is quite similar for both the forums. [1]

Spolsky and Atwood built Stack Overflow to provide high-quality, relevant information in a freely accessible way. They envisioned that Stack Overflow would be a combination of collaboration technologies, including open editing (like Wikipedia), feedback driven user ranking (like Reddit or Digg), moderated content (like blogs), and forums to create a distinctive format. Users are rewarded for responding and voting with Badges and Karma. This system encourages users to return and make positive contributions to the site, greatly increasing the quality of responses posted. Stack Overflow immediately received positive reviews like "it's a Q&A site where the right answer isn't buried on page fifty, it's almost always at the top." [http://blog.stackoverflow.com/2010/05/announcing-our-series-a/] Building blocks: 1. Voting: To increase the quality of answers, visitors are encouraged to vote on answers they found useful. The answers are arranged by the number of votes received. This process ensures the best answers are displayed at the top, quickly showing the best solution in the presence of multiple possible answers. Users gain reputation points when others vote for their questions or answers. 2. Tags: To facilitate an easy retrieval of previous questions that have been answered, each question is associated with a variety of tags. Tags help direct questions to experts from a specific field, improving the quality and speed of responses. Users may also customize the Stack Overflow website by specifying tags of interest and filtering out uninteresting ones, making their search experience more efficient. 3. Badges: To encourage participation and improve the quality of discussion, StackOverflow introduced the concept of badges. These awards, shown on the user's profile, are given to users who consistently provide high quality answers or ask popular and relevant questions. 4. Karma and Bounties: Karma is a type of currency system that encourages and rewards participation. Users who earn enough karma enjoy special privileges. For example, only a user with sufficient karma can comment on answers and even modify questions. Users win karma by selecting the correct and most relevant answer to their questions. 5. Data Dump: Stack Overflow provides a "data dump", which stores all the site's user-generated content under a Creative Commons license.3.3 This license allows people to share and adapt content as long as it is correctly attributed to its author. Under the Creative Commons license, any derivative works must also be distributed under an "open" license. Monthly snapshots of the database are available for download via BitTorrent. 6. Pre-Search: Pre-Search provides a list of potentially related questions to a user before the

user is allowed to post a new question. This prevents having multiple copies of the same question, enabling efficient retrieval of relevant answers. 7. Search Engine Optimization: It is critical for Stack Overflow to score top results in searches to increase its user base and to build its reputation as the "one stop shop" for all programming related questions. Therefore, it uses various search engine optimization techniques to increase its visibility on search engines. For instance, the URL of a particular page on the site contains keywords from the corresponding topic. 8. Critical Mass: Atwood and Spolsky used their blogs to promote Stack Overflow and direct traffic to it, ensuring the presence of an initial community. Atwood answered questions so quality answers were available on the website from the outset. 9. Performance: Built on a Microsoft software stack, Stack Overflow achieves high performance at a very low cost. Spolsky boasts that Stack Overflow's performance is comparable to websites with similar traffic, but only requires one-tenth of the hardware. [27]

Analysis of data from SO to categorize questions that are asked. Exploration of answered/unanswered questions. Findings indicate qa websites are particularly effective at code reviews and conceptual questions. "Letovsky [8] identified five main question types: why, how, what, whether and discrepancy. Fritz and Murphy[4] provide a list of questions that focus on issues that occur within a project. Sillito et al. [12] provide a similar list focusingon questions during evolution tasks. LaToza and Myers [7] found that the most difficult questions from a developer's perspective dealt with intent and rationale. In their study on information needs in software development, Ko et al. [6] found that the most frequently sought information included awareness about artifacts and co-workers." 9 SO design decisions: 1. Voting 2. Tags 3. Editing 4. Badges (karma) 5. Pre-search 6. Google UI 7. Performance (search engine optimization) 8. Critical mass: several programmers were explicitly asked to contribute in the early stages of Stack Overflow. Research questions: 1. What kinds of questions are asked on Q&A websites for programmers? 2. Which questions are answered and which ones remain unanswered? 3. Who answers questions and why? 4. How are the best answers selected? 5. How does a Q&A website contribute to the body of software development knowledge? qualitative coding of questions and tags; tags for topics, question coding for question nature. Defines successful and unsuccessful questions as follows: A successful question has an accepted answer, and an unsuccessful question has no answer. [52]

Semantic web (sw); search and query has become difficult and challenging. Need user friendly UI for query and data exploration. qa ontology based on structured semantic information. New interest in search technologies, since we are close to reaching critical mass for large-scale distributed semantic web. Two most common uses of sw is interpretation of web queries/resources based on described background knowledge, and searching through large datasets/KB as alternative/complacent to current web. Difficult for end-users to understand the

complexity of the logic-based sw. Challenging to process and querying content, hard to scale models to cope with available semantic data. of qa systems is to allow users to ask questions the way they want to by using natural language (NL) to receive a related answer to their problem. We can classify a QA system, and any semantic approach for searching and querying SW content, according to four interlinked dimensions: (1) the input or type of questions it is able to accept (facts, dialogs, etc); (2) the sources from which it can derive the answers (structured vs. un-structured data); (3) the scope (domain specific vs. domain independent), (4) how it copes with the traditional intrinsic problems that the search environment imposes in any non-trivial search system (e.g., adaptability and ambiguity). Most qa systems focuses on factoids, where you have 'WH-' queries (who, what, how many, etc.), commands (name all, give me, etc.) requiring an element or list of elements as an answer, or affirmation / negation questions. As pointed out in (Hunter, 2000) more difficult kinds of factual questions include those which ask for opinion, like Why or How questions, which require understanding of causality or instrumental relations, What questions which provide little constraint in the answer type, and definition questions. Linguistic problems are common in most NL systems. Some use shallow keyword-based techniques to find interesting sentences (based on words that refers to entities that are the same as the answer type). Ranking is based on syntactic features such as word order or similarity to the query. Templates can be used to find answers that are just reformulations of the question. Most systems classify queries based on answer type (e.g. name, quantity, dates, etc). Question classes arranged hierarchically in taxonomies, requiring different strategies based on the question type. Utilize word knowledge through resources like WordNet or ontologies Suggested Upper Merged Ontology (SUMO) to get the qa type. Other options are also: named-entity (NE) recognition, relation extraction, co-reference resolution, syntactic alternations, word sense disambiguation (WSD), logical inferences and temporal-spatial reasoning. qa application with text has two steps: 1. identifying the semantic type of the entity sought by the question 2. determining additional constraints on the answer entity Ontology-based semantic QA systems (also called semantic qa systems in paper), takes NL queries and ontology as input, and returns answers from KB (where KB is based on passed ontology). Allows user to have no prior knowledge to vocabulary or ontology structure. Two different main aspects for ontology-based qa systems: (1) degree of domain customization required (correlates to retrieval performance) (2) subset of NL understanding (full grammar-based NL, controlled or guided NL, pattern based). Needed to reduce omplexity and the habitability problem. Main issue hindering successful use of NLI. p. 24+ in report notes listing more methods. [26]

Purpose of qa is to get a factual answer from a large collection of text, instead of a list of documents. Getting answer based on type (e.g. city) or getting answer based on what the question asks for (context, definition, reasoning). Reformula-

tions and syntactic structure can make it hard to create a manual classifier. The goal of the paper is to categorize questions into different semantic classes based on the possible semantic types of the answers. They developed a hierarchical classifier guided by a layered semantic hierarchy of answer types that makes use of a sequential model for multi-class classification and the SNoW learning architecture. qc can be seen as a text classification task, but some characteristics make it different. Questions are short, therefore contains less text. However, having short text improves accuracy and analysis. Developed a hierarchical learning classifier based on the sequential model of multi-class classification. The goal of the model is to reduce the set of candidate labels for a given question by concatenating a sequence of simple classifiers. The output of the classifier (class labels) is used as input for the next. Classifier output activation is normalized into a density over the class labels and is thresholded so that it can output more than one class label. qc is built by combining sequence of two simple classifiers; first=coarse class, second=fine class. [23]

For Discussion chapter: ———————— In the Bag of Words (bow) model, only single words or word stems are used as features for representing document content. The issue is that learning algorithms are restricted to detecting patterns in the used terminology only, while ignoring conceptual patterns. List of weaknesses with using bow (1-3 addressed issues on a lexical level, and 4 conceptual level): 1. Multi-Word Expressions with an own meaning like "European Union" are chunked into pieces with possibly very different meanings like "union". 2. Synonymous Words like "tungsten" and "wolfram" are mapped into different features. 3. Polysemous Words are treated as one single feature while they may actually have multiple distinct meanings. 4. Lack of Generalization: there is no way to generalize similar terms like "beef" and "pork" to their common hypernym "meat". WordNet database organizes simple words and multi-word expressions of different syntactic categories into so called synonym sets (synsets), each of which represents an underlying concept and links these through semantic relations. - Candidate Term Detection: Defined a Candidate Term Detection strategy built on the assumption that if you find the longest multi-word expressions appearing in the text, the lexicon will lead to a mapping to the most specific concept for that word (instead of querying single words, which may lead to wrong mapping). - Syntactical Patterns:Analysis by using POS-tagging. - Morphological Transformations: Entry form, base form reduction. Stemming if the first query for the inflected forms on the original lexicon turned out unsuccessful. - Word Sense Disambiguation (WSD): A lexical entry for an expression does not necessarily imply a one-to-one mapping to a concept in the ontology. - Disambiguate an expression versus multiple possible concepts. - Generalization: Going from specific concepts in the text to general concept representations. Mapping words based on generalization (up to a certain level). [4]

Question classification (qc): predict the entity type of the answer of a nat-

ural language question, mostly achieved by using machine learning. Used Latent Semantic Analysis (LSA) technique to reduce the large feature space of questions to a much smaller and efficient feature space. Two different classifiers: Back-Propagation Neural Networks (BPNN) and Support Vector Machines (SVM). Found that using LSA on question classification made it more time efficient and improved classification accuracy by removing redundant features. Discovered that when the original feature space is compact and efficient, its reduced space performs better than a large feature space with a rich set of features. They also found that in the reduced feature space, BPNN was better than SVM. Competitive with state-of-the-art, even though they used smaller feature space. [25]

Note to self: Map graph over feature impact (unprocessed, singular, all) Also add in estimated training time for exhaustive search (e.g. 120 minutes for SVC vs 100 for SGD over 16k questions (since 4k = test)).

Citation list (content to come):

- [6]: Libsvm
- [56]: Question classification, SVM, semantics
- [17]: SVC
- [24]: Sigmoid kernels SVM
- [58]: SVM, QA, Context ranking
- [34]: Question taxonomy
- [19]: Japanese Q/A System
- [31]: Reputation system SO
- [4]: Text classification, semantics
- [14]: fuzzy sigmoid svm
- [60]: sentiment analysis, linguistics
- [39]: Answer quality in qa community
- [50]: Answer type construction
- [52]: QA on the web (programmers)
- [42]: What is a good Q/A?
- [22]: predict closed questions on SO
- [**autocite {Stanley2013}**]: tag prediction on SO
- [18]: Question classification
- [59]: Question classification, svm
- [3]: random search, hyperparameters
- [51]: SVM, text classification
- [16]: svm
- [20]: svm, text processing
- [11]: Salton
- [33]: semi-supervised, question classification
- [36]: problem solving question, cs

- [30]: wordnet
- [32]: good code example, SO
- [15]: SO expertise & knowledge
- [54]: dev interaction SO
- [57]: expertise vs activity on SO
- [7]: question feature for answer attraction SO
- [40]: tag recommendation SO
- [35]: questions, answers, activity and reputation on SO
- [1]: stackexchange, technical vs non-technical communities activity
- [13]: domain ontology, qa
- [23] qa classification, semantics
- [2] so, qa, post quality
- [41] qa, design, anwer response
- [21, 43, 55] datasets

In the paper by Toba et al. [50], 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[2]). 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. [56] 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 [59] ("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.

2.5 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

[2] Low precision can occur if the feature sets are not fitted well enough during classifier training [50, p. 283].

documents are most relevant for the problem you are currently working on (e.g. searching for research papers that are relevant to your work).

Text categorization is a fundamental task in document processing, allowing the automated handling of enormous streams of documents in electronic form. Paper is about a N-gram-based system for text categorization that is tolerant of textual errors. The system worked well for language classification, in one test it got 99.8text categorization: an incoming document is assigned to some pre-existing category. characteristics: - categorization must work reliably in spite of textual errors. - categorization must be efficient, consuming as little storage and processing time as possible, because of the sheer volume of documents to be handled. - - categorization must be able to recognize when a given document does not match any category, or when it falls between two categories. This is because category boundaries are almost never clear cut. N-gram is an N-character slice of a longer string. the term can include the notion of any co-occurring set of characters in a string (e.g., an N-gram made up of the first and third character of a word), but in this paper they use the term for contiguous slices only. Their system use N-grams of several different lengths simultaneously. Appends space to beginning/end of string (_ = space): bi-grams: _T, TE, EX, XT, T_ tri-grams: _TE, TEX, EXT, XT_, T_ _ quad-grams: _TEX, TEXT, EXT_, XT_ _, T_ _ _ —> a string of length k, padded with blanks, will have k+1 bi-grams, k+1tri-grams, k+1 quad-grams, etc. Re-statement of Zipf's Law: The n'th most common word in a human language text occurs with a frequency inversely proportional to n. This means that there is always a set of words which dominates most of the other words of the language in terms of usage frequency. smooth continuum of dominance from most frequent to least. implies that classifying documents with N-gram frequency statistics will not be very sensitive to cutting off the distributions at a particular rank. if we are comparing documents from the same category they should have similar N-gram frequency distributions. determined the true classification for each test sample semi-automatically. - classification procedure works a little better for longer articles - classification procedure works better the longer the category profile it has to use for matching. there were some interesting anomalies (using more N-gram frequencies actually decreased classification performance). caused by some articles having multiple languages even though mixed types were removed also used the classification system to identify the appropriate newsgroup for newsgroup articles (collected article samples from five Usenet newsgroups). Chose these five because they were all subfields of computer science, and wanted to test how the system could confuse newsgroups that were somewhat closely related removed the usual header information, such as subject and keyword identification, leaving only the body of the article (to prevent influential matches). primary advantage of this approach is that it is ideally suited for text coming from noisy sources such as email or OCR systems. Possible to achieve similar results using whole word statistics (using frequency

statistics for whole words), but several issues exists; 1. The system would be much more sensitive to OCR problems (one wrong character, and the word is counted separately). 2. 2. Certain articles could be too short to get representative word statistics. there are simply more N-grams in a given passage than there are words, and there are consequently greater opportunities to collect enough N-grams to be significant for matching. 3. By using N-gram analysis, you get word stemming for free, since the plurals will all contain the base form. To get the same results with whole words, the system has to do word stemming, requiring knowledge about the language the documents were written in. Whereas N-gram frequency approach provides language independence for free. 4. Other advantages of this approach are the ability to work equally well with short and long documents, and the minimal storage and computational requirements. [5]

2.6 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)

2.7 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?

2.8 Dataset

short about the datasets, others who has used it, and datasets in general, e.g. [21, 43, 55]

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).

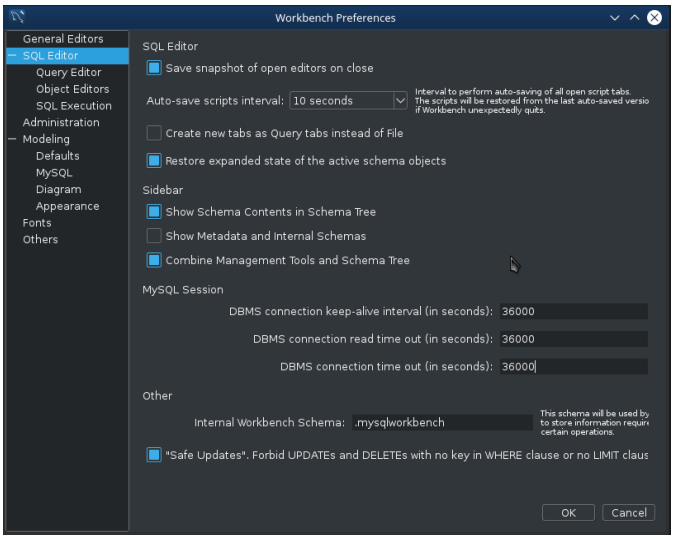

Figure 1: MySQL Workbench: Setting timeout values to avoid connection loss

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. 37. 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 [12]).

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 [38] [37, 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: Replacing tags in the question

```
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

Potentially move "all this failed and went wrong" here

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&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] William B Cavnar, John M Trenkle, et al. “N-gram-based text categorization”. In: *Ann Arbor MI* 48113.2 (1994), pp. 161–175.

[6] 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.

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

[8] 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).

[9] 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).

[10] Sebastian Deterding et al. “From game design elements to gamefulness: defining gamification”. In: *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*. ACM. 2011, pp. 9–15.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[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] B. Loni, S. H. Khoshnevis, and P. Wiggers. “Latent semantic analysis for question classification with neural networks”. In: *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. Dec. 2011, pp. 437–442. DOI: 10.1109/ASRU.2011.6163971.

[26] Vanessa Lopez et al. “Is Question Answering Fit for the Semantic Web?: A Survey”. In: *Semant. web* 2.2 (Apr. 2011), pp. 125–155. ISSN: 1570-0844. DOI: 10.3233/SW-2011-0041. URL: http://dx.doi.org/10.3233/SW-2011-0041.

[27] etl al M. Sewak. *Finding a Growth Business Model at Stack Overflow, Inc.* 2010. URL: https://web.stanford.edu/class/ee204/Publications/Finding%20a%20Growth%20Business%20Model%20at%20Stack%20Overflow.pdf (visited on 05/07/2016).

[28] Jitendra Maan. “Social business transformation through gamification”. In: *arXiv preprint arXiv:1309.7063* (2013).

[29] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT ’11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 142–150. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002491.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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).

[39] 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.

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

[41] 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.

[42] 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.

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

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

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

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

[47] 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).

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

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

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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

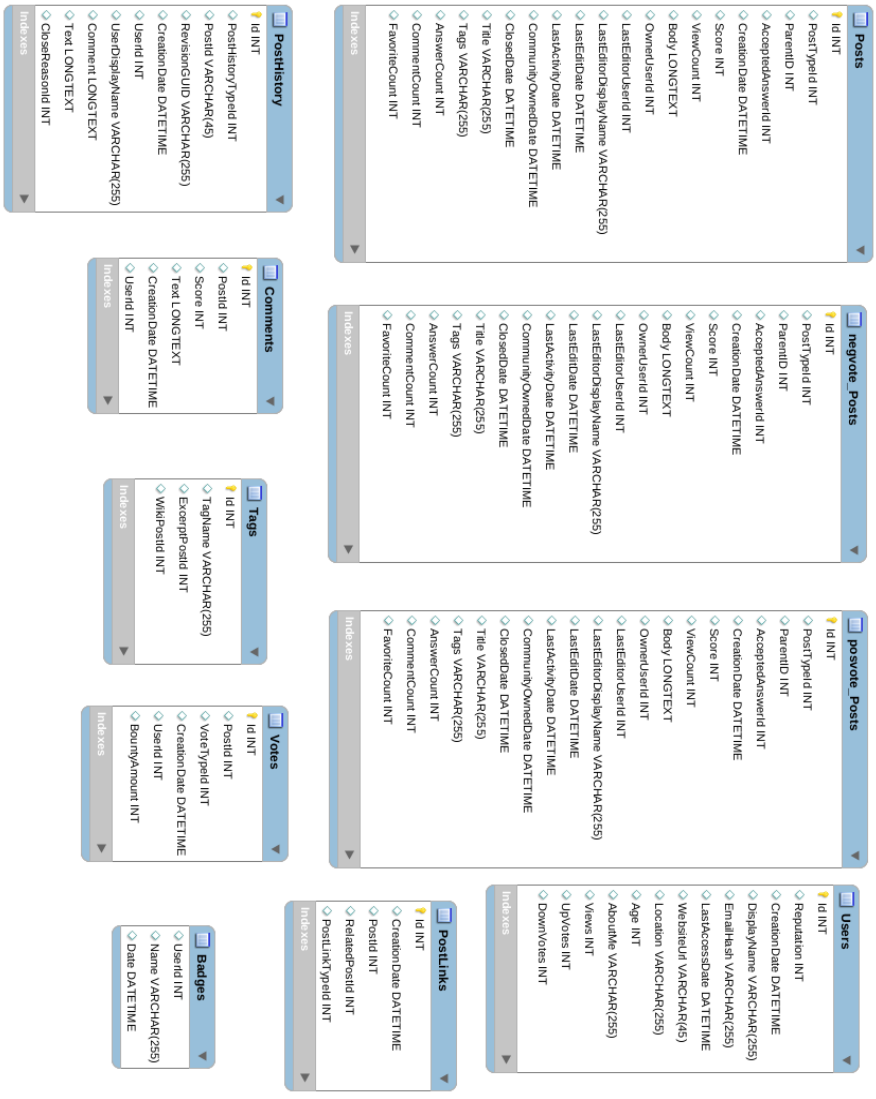

Figure 2: MySQL Database used for dataset