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

Preface i
Acknowledgement ii
Abstract iii
Contents iv
List of Figures vi
List of Tables vii
1 Introduction 1
1.1 Problem description 1
1.2 Research questions 2
1.3 Methodology to be used 2
1.4 Justification, Motivation and Benefits 2
1.5 Limitations 3
1.6 Thesis contribution 3
1.7 Thesis structure 3
2 Related work 4
2.1 Stack Overflow 4
2.1.1 Stack Overflows design 4
2.1.2 Stack Overflow and Gamification 5
2.1.3 Stack Overflow and reputation 6
2.2 Asking questions 8
2.2.1 What is the definition of a question? 8
2.2.2 Question classification 8
2.2.3 Question-Answering 10
2.3 Text categorization and classification 15
2.4 SVM 20
2.5 Dataset 20
2.6 undefined 20
3 Methodology 21
3.1 Dataset and MySQL Database 21
3.1.1 Dataset 21
3.1.2 MySQL Database 21
3.2 Development process 23
3.3 Feature sets, attributes and processing 26
4 Discussions 30
4.1 Data and Testing 32
4.2 Artificial Intelligence Methods 32

4.3 Implementation Architecture 32
4.4 Limitations and other issues 32
5 Conclusion/Summary 33
5.1 Overview of main results 33
5.2 Further work 33
Bibliography 34
A Appendix 40
A.1 Acronyms 40
A.2 Data sets/Statistical Overview 40
A.3 MySQL Database 41

List of Figures

1 List of questions, where one can see those with an accepted answers are marked with a green background. 5
2 Example of a question on Stack Overflow 5
3 MySQL Workbench: Setting timeout values to avoid connection loss 22
4 MySQL Database used for dataset 41

List of Tables

1 Results from pandas.DataFrame and pandas.Categorical. -1 is for bad questions (votes < -5), and 1 are for good questions (votes > 50). 27
2 Feature reduction steps before and after text was processed. . . . 28

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.

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 [40]. 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 [29, 25, 40, 52]. One of the requirements is that the questions should be of good quality [43, 48, 47]. 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 [8, 45].

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

add examples here

are not accepted on SO. There is even a post on Meta.StackExchange 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] [41]. There are several others who have used the same dataset [1, 2, 13, 29, 32, 37, 49, 57]. 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.

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 [29, 30, 32, 25, 52, 57], 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.

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 of 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. [32, 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 (SO)

2.1.1 Stack Overflows (SO) Design

The following list shows the design used in SO (based on M. Sewak [25, p. 6-7] and Treude et al. [52, p. 805]):

1. Votes: Questions and answers which are considered good (or bad) by the community can be given a score. This gives a filtering mechanisms, which allows users to ignore answers that are bad or wrong. Furthermore, answers are sorted by votes, and you can also sort questions on SO by vote score (see Figure 1).
2. Accepted answer: If the user asking a question gets an answer that they find satisfactory, they can select it as the "accepted answer". This answer will be the first displayed of the answers, and is also viewable when searching for questions (see Figure 1 and 2).
3. Tags: Each question is associated with a tag[1], which can be a topic, a programming language, a methodology, etc.
4. Badges: Similar to achievements in games, Badges are used to reward the user for their participation.
5. Reputation and Bounty: Currency system for user participation. E.g. voting for questions, getting your answer selected as the accepted one, etc. Bounty is a trade, where if your question goes unanswered for too long, you offer up parts of your reputation to receive an answer.
6. Data dump: Available data dump containing all content available within the SE community [41]. You can either download single files, or everything by using a Torrent client.
7. Pre-Search: Encouraging users to check that their question is not already posted by presenting a search bar when asking a new question.
8. URL keywords and Google: The questions title is included in the URL, allowing it to be processed by search engines. In addition, Google uses their crawlers every 10 second to have the latest updates from in their search engine [12].
9. Critical mass: Before Atwood and Spoolsky launched SO, they invited developers and programmers to participate to have some domain experts available.

[1] A full list can be seen here: http://stackoverflow.com/tags/

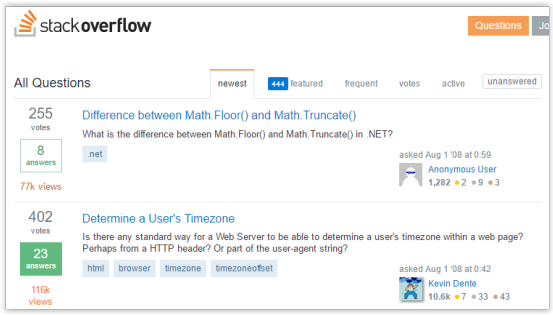

Figure 1: List of questions, where one can see those with an accepted answers are marked with a green background.

Figure 2: Example of a question on Stack Overflow[2]

2.1.2 Stack Overflow (SO) and Gamification

Deterding et al. [9] defines Gamification as "the use of game design elements in non-game contexts", and is the definition this section will be based on. Several papers make notes of the pedagogical and educational aspect of SO [30, 32, 57], and [30, 57] use the term gamification in their paper. 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 [32][3]. In

[2]Source: http://stackoverflow.com/questions/178325/checking-if-an-element-is-hidden

[3] From this interview:
http://www.wired.com/2012/07/stackoverflow-jeff-atwood/2012.

the course IMT4007 Serious Games Simon McCallum and Marius Nowostawski, presented their game GoRad, which was based on us students reading articles and posting questions which were voted on. The SO system awards users based their activity by using votes, reputation and badges [25, 29, 52, 42, 46]. In relation to gaming, there are four player types: Achievers, Explorers, Socializers and Killers [26, p. 3].

These player types can be used as a representation[4] for the various users of SO. Achievers are there for the reputation and badges, socializers are to interact, discuss and share knowledge. Explorers might find joy in looking at various topics, or searching for unanswered questions. The only exception would be the "Killer" type. Killers are those "... who always want to create trouble/problems for other participants" [26, p. 3]. In an online QA system (or Internet in general), these are what are commonly referred to as "Trolls" [11, 4]. However, due to the system used in SO, Trolls would not be able to survive, simply because the reputation controls what you have access to [44]. If you down-vote a post, you lose reputation. If your post gets down-voted, you also lose reputation. Users who are not willing to follow the guidelines can be locked out of SO [3]. However, today there is a lot of blogs complaining about the current structure of SO, who claims that a lot of the moderators are trolls[5].

2.1.3 Stack Overflow (SO) and reputation

Many QA sites includes domain experts to ensure some quality is upheld, and uses voting and reputation as a quality measurement [2]. Furthermore, questions topics, page views and votes can be used by search engines as a ranking mechanism, and it helps users to find the answers they are looking for. Anderson et al. [2] identifies two principles for the answer process. This process starts with the question being filtered down through the users, starting with domain experts. If the domain experts does not answer, it goes further down the chain, until it in the end either gets an answer, or is not answered at all. Both Anderson et al. [2] and Treude et al. [52] defines an unanswered question to be a question where no accepted answer is chosen[6]. The second principle is that a questions activity level does not just indicate the interest for the question, but could also be an indicator for quality (because a question can have multiple answers).

[4] Yang et al. [57] characterised users as "Sparrows" and "Owls", where sparrows answers question for reputation and owls answers the difficult ones (domain experts).
Ahmed et al. [1, p. 2] defined users as "lurkers, help-seekers (askers) and givers (responders)".

[5] https://www.reddit.com/r/programming/comments/3cafkp/is_stack_overflow_overrun_by_trolls/.
https://medium.com/@johnslegers/the-decline-of-stack-overflow-7cb69faa575d
Last accessed 23.05.2016.

[6] However, they do not take into considerations users who find a solution on their own, or simply forget or neglect to mark a an answer as accepted.

Since users can only gain 200 reputation points daily, the only way to earn more is by having your answer marked as accepted or through bounties [46]. Movshovitz-Attias et al. [29] found that users earn more reputation by providing good answers rather than good questions[7]. Most questions was asked by the users with a low reputation, but on average users with high reputation asked more questions. This indicates that reputation could be used as a measurement for expertise. Ahmed et al. [1] also found that there was a correlation between amount of answers given and the users reputation.

Yang et al. [57] found that the activity level of a user is not equal to knowledge, and divided users into two groups; "Sparrows" and "Owls". The sparrows are the basic users who earns reputation and badges by answering the easy questions, and has a greater interest in the gamification element. They found that the sparrows usually has a low average score and targets questions that are easy, or non-relevant. Nonetheless, they are still important since they are able to provide quick feedback. As for the owls, they are considered to be the domain experts. The owls earn reputation by asking more advanced questions, providing better answers (i.e. getting their answer accepted) and answering popular and difficult[8] questions.

Posnett et al. [32] views SE and SO as a learning community, since users help each gain new knowledge, and motivates learning. They wanted to see if the quality of the users answers improved over time. By constructing a posting history for each user, they found that the overall answer score decreased, and that the answer quality was static.

Nasehi et al. [30] did a qualitative analysis of code examples posted on SO. Their focus was on questions related to Java programming, with the requirements that the question should at least have a score of +4 and the answer +7. In addition, a code example should be included (by checking for <code> in the post). They found that the code explanation was just as important as the code examples (but you are still restricted to the quality of that example). For the code to be considered good, they listed the following attributes:

1. Concise code: Code samples should not be too long. They should be simple, and only focus on the parts that are relevant to the topic. Additional or non-relevant parts should instead be documented by using descriptive comments.

[7] However, as stated in Movshovitz-Attias et al. [29, p. 3], the reputation system was changed at one point. Originally, up-votes on questions and answers gave users a +10, but this was later changed into up-votes on questions only giving +5.

[8] Popularity was measured based on page views and the time between a question was posted until an answer was selected as accepted. The popularity can also therefore be seen as a measurement for difficulty. The longer it takes to answer, the more difficult the question is. [57, p. 273].

2. Question context: If the code is not working properly, suggestions for improvement should be added. One could also explain best practices and suggestions for improved readability. This will also have a pedagogical benefit, since the user asking the questions will learn to write better code.
3. Highlighting important elements: "Straight to the point", clearing up misunderstandings, pointing to relevant resources, etc.
4. Step-by-step solution: Splitting code into chunks, and explaining each chunk and its functionality. Comparison of languages; e.g. "How can I do X in C#, when I'm used to Java?"
5. Providing links to extra resources: Answers can be kept short by adding links to external resources, but a short summary should still be added.

2.2 Asking questions

2.2.1 What is the definition of a question?

The context of a question varies within the setting it is used. A question can be broad, where multiple answers can all be correct, or they can be factual, having only one right answer. When you are asking someone a question, you ask because you want to either find a solution to a problem, or learn something new. In the context of learning, questions are used for evaluating the students knowledge, or help them learn something new Nielsen et al. [31].

When doing research, you need research questions and hypotheses to decide what the goal of your research is. What questions are you trying to find an answer to, and what does that answer tell you? Slowiaczek et al. [38] defines asking a question as information selection and the answer(s) to a question as information usage. If you are working with statistical data, and you just post the numbers, this will not inform anyone. You need to explain what the numbers mean, and how you got them. The quality of an answer is also restricted to the quality of the question you ask. One can therefore assume that if you ask a good question, you will get a good answer [38].

2.2.2 Question Classification (QC)

QC is the process of categorizing a question into a class or category based on its structure, usually to decide what the expected answer type is [22, 23, 24]. To classify a question, it is important to select only those features that helps you identify the class it belongs to. Depending on the question type, and the information you want to extract, various methods exists.

WH-words

WH-words are mostly found in factoid questions [24]. Huang et al. [15] listed eight different WH-words: What, which, when, where, who, how, why, and rest (rest being the type does not belong to any of the previous type). Letovsky [20]

also listed "Whether" and "Discrepancy"[9]. However, not all are equally easy to use for classification, because even if the questions ask for the same answer, wording and syntactic structures can make it difficult to classify. Question containing words like "What", "Why", "How" and "Which", can be harder to classify due to the lack of limitation in regards to answer types[10] [15, 24].

Bag of Words (BOW) and N-grams

N-gram is a model that is used for splitting text into either characters (character model) or word frequencies (word model). The Bag of Words (BOW) model (or unigram) only looks at singular words, ignoring the order and relies only on the frequency for each word [28, 33]. Bi-grams takes dual values, tri-gram takes three, etc.

One problem with N-grams is that the dimension of the feature space is equal to the amount of words in the vocabulary [33, 23] When using categorization, there can be issues with mapping new words that does not exist in the vocabulary [58]. The impact of N-gram is also related to the size of the text being analysed. Zhang and Lee [59] found that there was not a big difference when using between bag-of-ngrams (all continuous word sequences in the question) and BOW as features.

Word mapping and processing: Case-sensitivity, Stemming, Stop words and Tokenization

To reduce the amount of words used, there are more steps that can be taken. By removing the case-sensitivity, all words will be equal (e.g. is the word 'Hello' equal to the word 'hello'?). [15] includes case-sensitivity under a definition called word shape, consisting of five elements: upper case, all lower case, mixed case, all digits, and other.

Semantics can be used for word filtering, e.g. removal of duplicate words or words with same meaning. WordNet has a built in function called synsets() which removes synonyms (words having the same meaning). You can also look for hypernyms (words belonging to a category with a parent-child relationship) or use stemming. Stemming reduces the word to its base-form, e.g. crying would be converted into the word cry. Word separation is also possible through tokenization, which splits the text into an array based on a set delimiter. There is also usage of stop words for removal of frequently used words in a given language.

Grammatical properties can be extracted by using Part of Speech (POS), e.g. by using Natural Language Toolkit (NLTK)[11], which can be helpful in reducing am-

[9] "Questions that reflect confusion over a perceived inconsistency." [20, p. 5]

[10] An answer type (or named entity) is the expected type of the answer to a given question (e.g. a Location, Organization, Person, Date, etc) [14, 24, 34, 58].

[11] NLTK includes in their POS tagger the following grammatical properties: Adjective, adposition, adverb, conjuction, determiner, article, noun, numeral, particle, pronoun, verb, punctuation

biguities [5]. Li and Roth [22] uses the word head chunks to identify what the question is asking for when multiple types are introduced (avoid ambiguity). The same concept is used in [15] and [23], but there it is referred to as headwords.

2.2.3 Question-Answering (QA)

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. [56]

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. Xu et al. [56]

QA system that includes question analysis, information retrieval and answer extraction. Categorizing questions is an important part of the question analysis (the same goes for follow-up process, since it affects the accuracy of the answer extraction). 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,

mark and others [50, See Section 2.3].

the classification must improve the question characteristics. 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 for most automatic QA systems is to find the exact answer to the question asked by the user. Existing QA technology involves two main steps: IR and Information Extraction (IE). IR retrieves the relevant documents after completed analysis or classification. IE processed the retrieved documents to find the answer to the users question. 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. Document retriever: for only retrieving documents that are related to the question. Passage retriever: divides the documents into paragraphs and ranks them based on the given evaluation metric(s) [58]

Developed a system composed of four modules: Question Analysis, Document Retrieval, Answer-Extraction, and Answer Evaluation. 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). Found that named entity/numerical expression recognition and word sense-based answer extraction contributed to system performance. 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). Used suffixes to indicate candidate group. 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. [16]

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. [29]

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. Paper focuses on factoid questions. 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. Representing a question as a vector (this is aka. BOW or unigram):
$x = (x_1, ..., x_d)$ in which x_i is the frequency of term i in x and d is the total number of terms.
BOW is the most widely used feature space in document classification. [23]

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 learn that two other words means the complete opposite (negative version). Ran sentiment analysis on 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. Did not use stemming, since the model learned similar representations for words of the same stem. Allowed non-word tokens (e.g. '!', ':-)'), since it could be sentiment relevant. Mapped ratings to [0,1], and the semantic component did not require labels. [27]

Search and queries have become difficult and challenging for the semantic web. 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. Semantic web is often used for interpreting web queries based on described background knowledge and for searches through large datasets/KBs. Difficult for end-users to understand the complexity of the logic-based semantic web. Challenging to process and querying content, hard to scale models to cope with available semantic data. The goal of QA is for users to ask questions as they would ask another human (natural language), rather then use keywords queries.
Classifies QA system and semantic approach for search and query based on four interlinked dimensions:

1. the input or type of questions it is able to accept (facts, dialogues, 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. ability to cope with problems that the search environment imposes in any non-trivial search system (e.g., adaptability and ambiguity).

Most QA systems focuses on factoid questions, which are usually based on 'WH-' (who, what, how many, etc.), commands (name all, give me, etc.), or affirmation/negation questions. Examples of more difficult questions are factoid questions asking for opinion (Why or How), What (lack of constraint) and definition questions. A common problem in natural language systems is linguistics. Some systems 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. Classification based on expected answer type (e.g. name, quantity, dates, etc). Classes of questions are arranged hierarchically in taxonomies and different types of questions re-quire different strategies. Utilization of word knowledge by using resources like WordNet or ontologies to find the QA type. Could also use 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 complexity and the habitability problem, the main issues hindering successful use of Natural Language Interfaces (NLI). [24]

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). Reformulations 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, hav-

ing 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. [22]

"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] :: An Empirical Study on Developer Interactions in StackOverflow

Proposes head word feature and present two approaches to augment semantic features of such head words using WordNet. Their linear SVM and Maximum Entropy (ME) models reached an accuracy of 89.2% and 89.0% over a standard benchmark dataset. Used Libsvm. Question classification is not trivial; simply using question wh-words cannot achieve satisfactory results. Difficult to classify "what" and "which" type questions. Even though multiple questions ask for the same answer, wording and syntactic structures can make it difficult to classify. Due to the large number of features in question classification, one may not need to map data to a higher dimensional space. It has been commonly accepted that the linear kernel of $K(x_i, x_j) = x_i^T * x_i$ is good enough for question classification. Used five binary feature sets;

1. Question wh-words (eight types): what, which, when, where, who, how, why, and rest (rest being the type does not belong to any of the previous type).
2. Head words: one single word specifying the object that the question seeks to avoid misclassification of entities. Requires syntactic parser.
3. WordNet semantic feature for head words: Use of WordNet for word similarity distance, computing the similarity between the head word of such question and each description word in a question categorization.
4. Word N-grams: Used to provide word sense disambiguation for questions
5. Word shapes: Five word shape features; all upper case, all lower case, mixed case, all digits, and other.

WordNet is a large English lexicon in which meaningfully related words are connected via cognitive synonyms (synsets). The WordNet is a useful tool for word semantics analysis and has been widely used in question classification. Using

WordNet for hypernyms; Y is a hypernym of X if every X is a (kind of) Y. For example, the hierarchies for a noun sense of domestic dog is described as: dog -> domestic animal -> animal, while another noun sense (a dull unattractive unpleasant girl or woman) is organized as: dog -> unpleasant woman -> unpleasant person. Unigram forms theBOWfeature, and bigram forms the pairs of words feature, and so forth. Considered unigram, bigram, and trigram. Two experiments to test classifier accuracy. The first was evaluation of individual contribution of different feature types to question classification accuracy, and in the second feature sets were incrementally fed to the SVM and ME (default parameter values for both).
Experimented with five machine learning algorithms:

- Nearest Neighbors (NN) (simplified version of kNN)
- Naive Bayes (NB)
- Decision Tree (DT)
- Sparse Network of Winnows (SNoW)
- SVM, using two kinds of features: BOW and bag of N-grams (all continuous word sequences in the question).

Using a search engine to retrieve list of documents vs getting the actual answer to the question. To answer a question, you need to understand what answer the question asks for. Classifying the question into semantic categories to understand what answer type is expected. Common words like 'what', 'is', etc. should be neglected for document classification, but these "stop-words" are actually very important for question classification. Writing heuristic rules for question classification can take time, and easily become complicated. Considers only the surface text feature for each question (extracting only two features: BOW and bag of N-grams). Every question is represented as binary feature vectors, because the term frequency (tf) of each word or N-gram in a question usually is 0 or 1. NB is regarded as one of the top performing methods for document classification. Sparse Network of Winnows (SNoW) algorithm is specifically tailored for learning in the presence of a very large number of features and can be used as a general purpose multi-class classifier. The learned classifier is a sparse network of linear functions. Parameters for all algorithms was default values. SVM: observed that the bag of N-grams features are not much better than the BOW features. The SVM based on linear kernel turned out to be as good as the SVM based on polynomial kernel, RBF kernel and sigmoid kernel. Syntactic structures; proposed using a special kernel function called tree kernel to enable the SVM to take advantage of the syntactic structures of questions. [59]

2.3 Text categorization and 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).

Text categorization is a fundamental task in document processing, allowing automated handling of enormous streams of documents in electronic form. 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 99.8% classification rate on Usenet newsgroup articles (written in different languages). It also worked well for computer-oriented articles (subject), where the highest classification rate was 80%. Text categorization is assignment of an incoming document to some pre-existing category. Has the following 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). In this paper, they use N-gram for contiguous slices only, with 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+1 tri-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. 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

Used the classification system to identify the appropriate newsgroup for newsgroup articles (collected article samples from five Usenet newsgroups). Chose those because they were all sub-fields of computer science, and wanted to test how the system could confuse similar newsgroups. Removed the usual header information, such as subject and keyword identification, leaving only the body of the article (to prevent influential matches).
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. Certain articles could be too short to get representative word statistics. 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. Ability to work equally well with short and long documents, and the minimal storage and computational requirements.

[6]

Using SVM to solve automatic text classification task. Text classification (aka. text categorization or topic spotting) is the activity of labelling natural language texts with thematic categories from a predefined set.
Examples of automatic document classification:

- spam filtering: separate spam email messages from legitimate emails
- e-mail routing: routing email to address or mailbox based on topic
- language identification: automatically determine the language of a text
- genre classification: automatically determine the genre of a text
- readability assessment: automatically determine the degree of readability of a text
- classical task of document indexing and filing: filing documents according to their content or other specific information.

The first issue in text classification is to find an adequate representation of the documents. The largely employed model in IR is the Vector Space Model (VSM) (aka. BOW model). Term extraction and dimensional reduction steps:

1. Tokenization: Separating text based on delimiters, e.g. space, EOF and tabs, where these text units are processed in the below steps.
2. Case standardization: Convert all characters to the same case

3. Tag filtering: if the document is an Internet page or a similar hyper document, the corresponding tags are eliminated
4. Stop word removal: "Stop words" are common natural language entities that do not carry strong semantics. They are therefore considered irrelevant for information retrieval and for text classification purposes.
5. Stemming (or lemmatization): text elements with small variations in their lexicon but with the same semantics must be associated in order to produce the same dimension in the VSM; some examples are singular and plural variations of the same word and person and time variations of a verb. Linguistic experts produce specific procedures that convert a word into its "stem" (radix), which is the element employed to represent the corresponding dimension
6. Other term associations: it is possible to associate terms of identical semantics using external dictionaries such as the WordNet, or by computing successive terms that always occur together in the text, e.g. "triple heart bypass", which is considered a triple compound term.

The "term-to-term" similarity is deeply analysed. Their work indicates that:

1. words can be analysed from the linguistic (semantic) point-of-view, to obtain synonyms, antonyms, hyponyms, hypernyms, meronyms, etc. these relations nowadays are included in large linguistic repositories such as the WordNet
2. statistical term co-occurrence computation in a corpus is a practical way to compute term similarity
3. a more abstract element than the terms, called a “concept”, can be used to represent document dimensions
4. a concept is characterized by a set of documents, or, more specifically, corresponds to the maximal subset of documents in the corpus that contains the concept;
5. two concepts are uncorrelated if the intersection between the corresponding sets of relevant terms of each document is empty; the greater the overlap between the sets of documents related to two concepts, the more related theses concepts are.

Perhaps the most important idea that can be extracted from this analysis is the relation between concepts and vectors:

- concepts are represented by vectors
- the determination of a vector basis in the term-space involves the identification of the "fundamental concepts" in the collection, whose corresponding representative vectors are orthogonal
- documents and queries are linear combinations of the fundamental concepts.

[18]

Text categorization has become one of the key techniques for handling and organizing text data. Text categorization techniques are used to classify news stories, to find interesting information on the WWW, and to guide a user's search through hypertext. Since building text classifiers by hand is difficult and time-consuming, it is advantageous to learn classifiers from example. Goal of text categorization is to classify documents into a number of pre-defined categories, where each document can be in one, multiple or none. Since categories may overlap, each category is treated as a separate binary classification problem. The first step in text categorization is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task. IR research suggests that word stems work well as representation units and that their ordering in a document is of minor importance for many tasks. This leads to an attribute-value representation of text. Each distinct word w_i corresponds to a feature, with the number of times word w_i occurs in the document as its value. To avoid unnecessarily large feature vectors, words are considered as features only if they occur in the training data at least 3 times and if they are not "stop-words" (like "and", "or", etc.). This representation scheme leads to very high-dimensional feature spaces containing 10000 dimensions and more. Need for feature selection to improve generalization accuracy and avoid "overfitting". From IR it is known that scaling the dimensions of the feature vector with their inverse document frequency (IDF) improves performance. SVM is based on the Structural Risk Minimization principle from computational learning theory. The idea of structural risk minimization is to find a hypothesis h for which we can guarantee the lowest true error. The true error of h is the probability that h will make an error on an unseen and randomly selected test example. An upper bound can be used to connect the true error of a hypothesis h with the error of h on the training set and the complexity of H (measured by VC-Dimension), the hypothesis space containing h. SVMs find the hypothesis h which (approximately) minimizes this bound on the true error by effectively and efficiently controlling the VC-Dimension of H. One remarkable property of SVMs is that their ability to learn can be independent of the dimensionality of the feature space. SVMs measure the complexity of hypotheses based on the margin with which they separate the data, not the number of features. This means that we can generalize even in the presence of very many features, if our data is separable with a wide margin using functions from the hypothesis space. The same margin argument also suggest a heuristic for selecting good parameter settings for the learner (like the kernel width in all RBF network). The best parameter setting is the one which produces the hypothesis with the lowest VC-Dimension. This allows fully automatic parameter tuning without expensive cross-validation.

Theoretical analysis concludes that SVMs acknowledge the particular properties

of text:

- high dimensional feature spaces,
- few irrelevant features (dense concept vector),
- sparse instance vectors.

Experimental results show that SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly. With their ability to generalize well in high dimensional feature spaces, SVMs eliminate the need for feature selection, making the application of text categorization considerably easier. Another advantage of SVMs over the conventional methods is their robustness. SVMs show good performance in all experiments, avoiding catastrophic failure, as observed with the conventional methods on some tasks. Furthermore, SVMs do not require any parameter tuning, since they can find good parameter settings automatically. All this makes SVMs a very promising and easy-to-use method for learning text classifiers from examples. [17]

2.4 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?

In QC, question is represented using vector space model, the question is a vector which is described by the words inside it. [23]

2.5 Dataset

short about the datasets, others who has used it, and datasets in general, e.g. [19, 39, 55]

2.6 undefined

Stanley and Byrne [49] - Predicting Tags for StackOverflow Posts

Short et al. [37] - Tag Recommendations in StackOverflow

Wang et al. [54] - An Empirical Study on Developer Interactions in StackOverflow

Lezina and Kuznetsov [21] - Predict Closed Questions on StackOverflow

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 3).

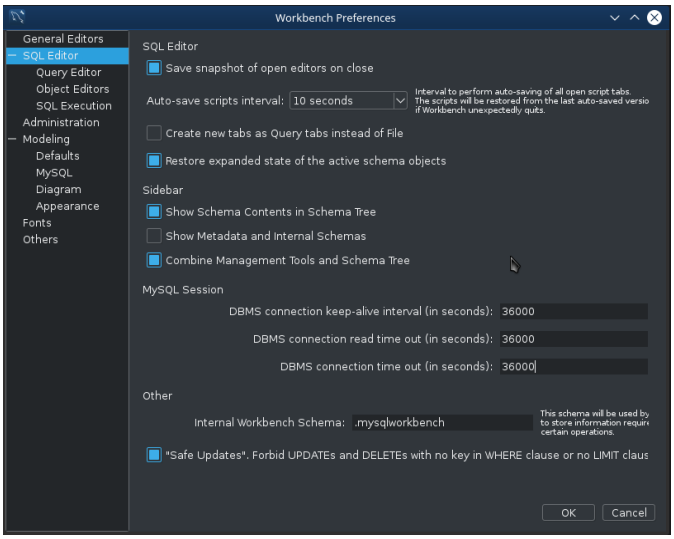

Figure 3: MySQL Workbench: Setting timeout values to avoid connection loss

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. 41. 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 [36] [35, see 4.2.3]. When looking at this vocabulary, a lot of of unimportant 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 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

For Discussion chapter: ——————

In the 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.
Conceptual Document Representation:

- Candidate Term Detection: Strategy built on the assumption that if you find the longest multi-word expressions 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).

[5]

QC: predict the entity type of the answer of a natural 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. [23]

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)).

In the paper by Toba et al. [51], 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). The retrieval of QA pairs is done by using a statistical relation framework: Bayesian Analogical Reasoning (BAR). Features sets are extracted from the training set by use of binary values checking if the question contains a given question word. The BAR framework then learns the related features and computes the estimation for them. Thereafter QA pairs are retrieved from the testing set and compared against the training set. Afterwards, the QA pairs that have identical question words are identified, and overlapping pairs are grouped according their named entity group. To retrieve named entities, they used two different recognizers. The first was Stanford (extracts the person, organization and location), and the second was dictionary based (extract number entities and fine-grained noun-based entities). Question words were extracted by building a question word list from the training set (achieved by using Stanford Part-of-Speech (POS) tagger). Then for each question, look for the appearance of the question word to create the feature set. Mapped named-entities. [51]

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)

[1] Low precision can occur if the feature sets are not fitted well enough during classifier training [51, p. 283].

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] Jeff Atwood. *A Day in the Penalty Box*. 2009. URL: http://blog.stackoverflow.com/2009/04/a-day-in-the-penalty-box/ (visited on 04/05/2016).

[4] Jeff Atwood. *What is Trolling?* 2015. URL: https://blog.codinghorror.com/what-is-trolling/ (visited on 05/23/2016).

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

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

[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] 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.

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

[11] Howard Fosdick. *Why People Troll and How to Stop Them*. 2012. URL: http://www.osnews.com/story/25540 (visited on 05/23/2016).

[12] Pascal-Emmanuel Gobry. *Google Is Indexing Stack Overflow At 10 Times Per Second*. 2011. URL: http://www.businessinsider.com/google-stackoverflow-2011-3?r=US&IR=T&IR=T (visited on 05/23/2016).

[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] Matthias H Heie, Edward WD Whittaker, and Sadaoki Furui. "Question answering using statistical language modelling". In: *Computer Speech & Language* 26.3 (2012), pp. 193–209.

[15] 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.

[16] 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.

[17] Thorsten Joachims. "Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings". In: ed. by Claire Nédellec and Céline Rouveirol. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. Chap. Text categorization with Support Vector Machines: Learning with many relevant features, pp. 137–142. ISBN: 978-3-540-69781-7. DOI: 10.1007/BFb0026683. URL: http://dx.doi.org/10.1007/BFb0026683.

[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] Stanley Letovsky. “Cognitive processes in program comprehension”. In: *Journal of Systems and Software* 7.4 (1987), pp. 325–339. ISSN: 0164-1212. DOI: http://dx.doi.org/10.1016/0164-1212(87)90032-X. URL: http://www.sciencedirect.com/science/article/pii/016412128790032X.

[21] 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.

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

[23] 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.

[24] 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.

[25] et 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).

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

[27] 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.

[28] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*. Vol. 1. Cambridge University Press, 2008. ISBN: 978-0521865715.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Pearson Education, 2013. ISBN: 978-1292024202.

[34] Yutaka Sasaki. “Question answering as question-biased term extraction: a new approach toward multilingual QA”. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 215–222.

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

[36] 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).

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

[38] 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.

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

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

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

[42] StackOverflow.com. *Badges*. 2016. URL: http://stackoverflow.com/help/badges (visited on 04/05/2016).

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

[44] StackOverflow.com. *Privileges*. 2016. URL: http://stackoverflow.com/help/privileges (visited on 04/05/2016).

[45] 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).

[46] StackOverflow.com. *What is reputation? How do I earn (and lose) it?* 2016. URL: http://stackoverflow.com/help/whats-reputation (visited on 04/05/2016).

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

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

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

[50] Ewan Klein Steven Bird and Edward Loper. *Categorizing and Tagging Words*. 2015. URL: http://www.nltk.org/book/ch05.html (visited on 05/06/2016).

[51] 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.

[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.

A Appendix

A.1 Acronyms

AI Artificial Intelligence. 3

BOW Bag of Words. 12, 14, 19, 20

IDE Integrated Development Environment. 28

IE Information Extraction. 11

IR Information Retrieval. 11, 16, 20, 21

ML Machine Learning. iii, 2, 3, 11, 16, 23

NLTK Natural Language Toolkit. 28

QA Question-Answering. iii, 1, 6, 8–16, 22

QC Question Classification. 2, 12, 14, 15

SE Stack Exchange. iii, 1–3, 5, 7, 24

SO Stack Overflow. iii, 1–3, 5–8, 10, 31

SVM Support Vector Machines. ii, iii, 2, 3, 16, 19, 21–23

A.2 Data sets/Statistical Overview

A.3 MySQL Database

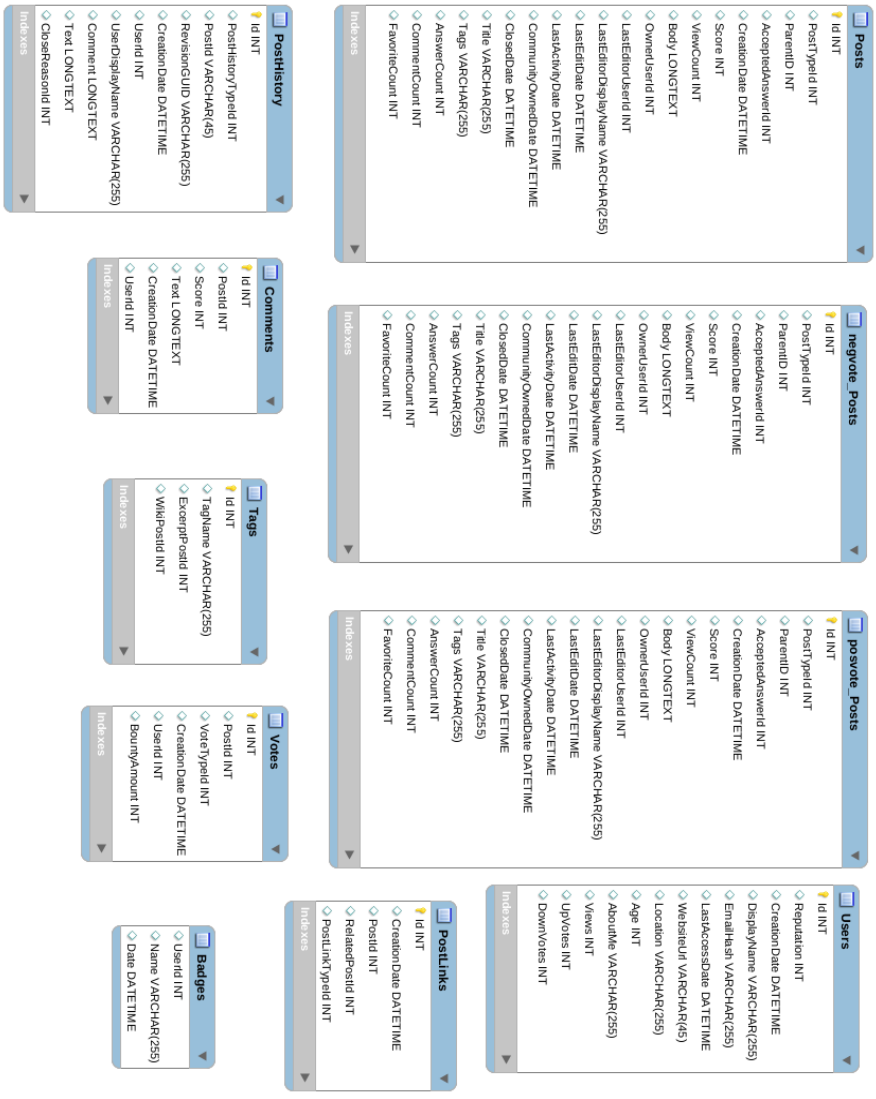

Figure 4: MySQL Database used for dataset