School of Electronic Engineering and Computer Science

MSc Computer Science
Project Report 2017

Detecting Polar and Nonpolar Questions in both Dyadic and Multi-party Online Chat Discourse

Quinn Hisashi Koike

Queen Mary
University of London

August 2017

Disclaimer

Acknowledgements

The author of this research paper would like to pay special thanks to Android Chat and @andreasbackx for allowing this research to build a Slack Corpus from this channel. Furthermore, this research would like to acknowledge the University of Leuven and Hendrik De Smet. Department of Linguistics, for compiling and allowing this research to gain access to the Yahoo-based Contrastive Corpus of Questions and Answers (YCCQA).

ABSTRACT

This research paper focuses on training a machine learning classifier to detect questions in dyadic and multiparty online chat discourse. The classifier is trained on multiple datasets from forums, machine learning generated questions and online chat discourse from the Slack Corpus, created especially for this research. Classified questions will be used as the input for a polar detection algorithm which has been built as a non machine learning, rule based algorithm. This algorithm relies heavily on linguistic domain knowledge and outputs a list of polar and nonpolar questions. The final outcome is a question detection classifier and a polar / nonpolar identification algorithm. The machine learning section of this paper tests four different feature sets on both a Naive Bayes classifier and a Support Vector Machine with a linear kernel. The code for this research is included in support files and has a readme.txt file which explains the structure of the directories and details on the source code and the tools built to support the project.

Table of Contents

Introduction

The Oxford English Dictionary defines a chatbot as “A computer program designed to simulate conversation with human users, especially over the Internet” (27). Chatbots are a growing feature in society (24), from Facebook to Amazon and rely heavily on machine learning techniques as well as computational linguistics (25). Newman argues that “As more companies invest in strong AI systems, such as neural nets, deep learning, and real-world human trainers, chatbots represent a real digital disruption that could revolutionize major industries, individual departments, and consumer interactions” (24). Chatbots and programs which are expected to understand colloquial human interactions are becoming a disrupting force in the day to day lives of consumers. For this reason, this paper argues that the ability to map conversation structure is of great importance. Conversation structures are complex in both dyadic and multiparty online chat. Dyadic conversations can contain multiple conversations between two individuals at once and multiparty chat can increase this complexity even further as discussed later in this paper. Arguably, conversation is too complex to be treated as a holistic machine learning classification task. This paper proposes the idea of breaking conversation up into smaller interaction types and building labeling systems for each sub task. This paper will focus on the binary task of question and non question classification using supervised machine learning techniques and linguistic domain knowledge as well as polar and nonpolar identification via a rule based algorithm. Polar and nonpolar question identification and question classification are important tasks for various reasons. Question classification can help chatbots identify when a response is needed and polar and nonpolar question identification can help chatbots and those developing chatbots to identify what type of response is needed. This research is also key to the greater goal, of building labeling systems for individual linguistic acts, to aid in the end goal of conversation mapping.

This research focuses on finding the best feature set to train linear, supervised, machine classification algorithms on to detect questions and works on building a non machine learning algorithm, based on linguistic domain knowledge to identify polar and nonpolar questions. Generative models (Naive Bayes in the case of this research) and discriminative models (Support Vector Machines with a Linear Kernel) are tested. This paper will focus only on the English language and is therefore able to use the strict grammar rules provided by the language to build a polar / nonpolar question identification algorithm. An algorithmic, rules based approach to polar and nonpolar question identification can be much more efficient that training a machine learning classifier and could be used as a way to create features for machine learning classifiers where polar and nonpolar question detection is useful, such as automating meeting minutes (27) or improving chatbot dialogue when answering questions, as well as being used in non machine learning tasks where polar and nonpolar question identification is key.

Literature Review

Online chat discourse

Multi-party chat dialogue - in the context of this paper - describes online text-based conversations between three or more people. Online multi-party chat can be traced back as far as the 1970s (1) to MUDs (Multi User Dungeons / Dimensions / Domains) (1). Unlike dyadic conversation, multi-party chat creates more complexity, both structurally and semantically. Uthus and Aha found that *“multiparticipant chat may involve multiple, asynchronous, and simultaneous conversations, with messages interwoven among the various conversations”* (1). Stergos et.al analysed multi-party discourse and also found that *“multi-party chat involves multiple interlocutors that may address one or more interlocutors during their turn”* (2).

Different datasets were used to train this paper’s classifiers, including data from the Android Development Community Slack team which is a multi-party chat tool. The Slack data conforms to the complex structures outlined in the research by Uthus and Aha as well as the work by Stergos et.al. For example, inside the hangout channel - used to speak about topics other than Android development - complex, simultaneous conversations were extremely prevalent with embedded threads occurring between larger, more complex threads. Details on the dataset are outlined in a later section of this paper, however, to review current academic analysis of multi-party chat data and compare it with the finding of this paper, a brief overview of slack is needed.

Slack is an instant messaging service originally built for business teams to communicate in a more efficient and colloquial manner. “Teams” (or chat rooms) are created and dedicated to specific topics such as the Android Development Community. Inside each team's chat are channels (or sub threads) which are used for certain tasks. Commonly, you may find a “general”, “hangout”, “introductions” and “admin” channel although these are optional. These channels can be public or private. It is also possible to have one on one chats in direct message channels which are private and therefore not included in the dataset used for this paper.

Interestingly, although the user interface (UI) of Slack and other multi-party chat systems differ and are used for different purposes - SMS for more colloquial conversations, Slack for work and project based chats, MUDs and MMORPGs for gaming conversations (1) etc. - Werry (3) as cited by Uthus and Aha (1) states that major discrepancies occur in text-based conversations - dyadic and multi-party chat, regardless of the platform - in comparison to spoken language conversations. These include abbreviations, acronyms, clipped words, and deletion of subject pronouns (1). This is consistent and applicable to the data collected for this paper which encompasses not only online multi-party discourse data but also forum based corpora and machine learning generated data - which were used to train a classifier to detect question structure - indicating that the use of abbreviations, clipped words and deletion of subject pronouns is indicative of general online colloquial communication and not bound only by online instant messaging where multi-party discourse is prevalent. This poses multiple problems. The lack of formal syntax such as commas, renders a simple rule based, non machine learning technique futile. Furthermore, other insertions such as the use of emoticons (emojis) and GIFs were ubiquitous throughout the Slack corpus. As discussed by Kelly (3) *"...we use different emoticons/ emojis, depending on the context we have created with the person with whom we are communicating"* meaning that the interpretation of emoticons is vague enough to pose further problems in classifying questions where emoticons play a role. Arguably, commonly used emoticons such as a smiling face or a sad face are specific enough in nature to easily translate into language. This research suggests that text based chat, regardless of platform, intended use or number of participants, is a very different domain to spoken conversation and poses unique challenges. This research suggests that a rule based approach to classification could work, but the rules must account for English online chat conventions, not simply for the stricter rules of formal English.

Question Structure

The SLI Glossary of Linguistic Terms defines a question as "*...a sentence type that has a form (labeled interrogative) typically used to express an illocutionary act with the directive illocutionary point mentioned above. It may be actually so used (as a direct illocution), or used rhetorically."* (7) Arguably, questions can be placed into two broad categories - polar and nonpolar (4). Larsen-Freeman et.al argue that there are a minimum of 12 question types (6). However, for the purpose of this research, question types are seen hierarchically, with polar and nonpolar questions being categories of which other question types are children. These nested question types may fit into one or both of the parent categories. Debatably, most of these question types fit neatly into the polar and nonpolar categories. However, a small portion of question types can be classified into both polar and nonpolar types depending on the question's semantics. This is a simplified and arguably contradictory organisational structure to the research of Larsen-Freeman et.al, but it is used to help explain classification of question structures later in this paper.

Figure 1 uses the Larsen-Freeman et.al basic categorisations arranged in polar and nonpolar question structure to illustrate the above point, that this paper will treat question types hierarchically , categorising them as polar or nonpolar.

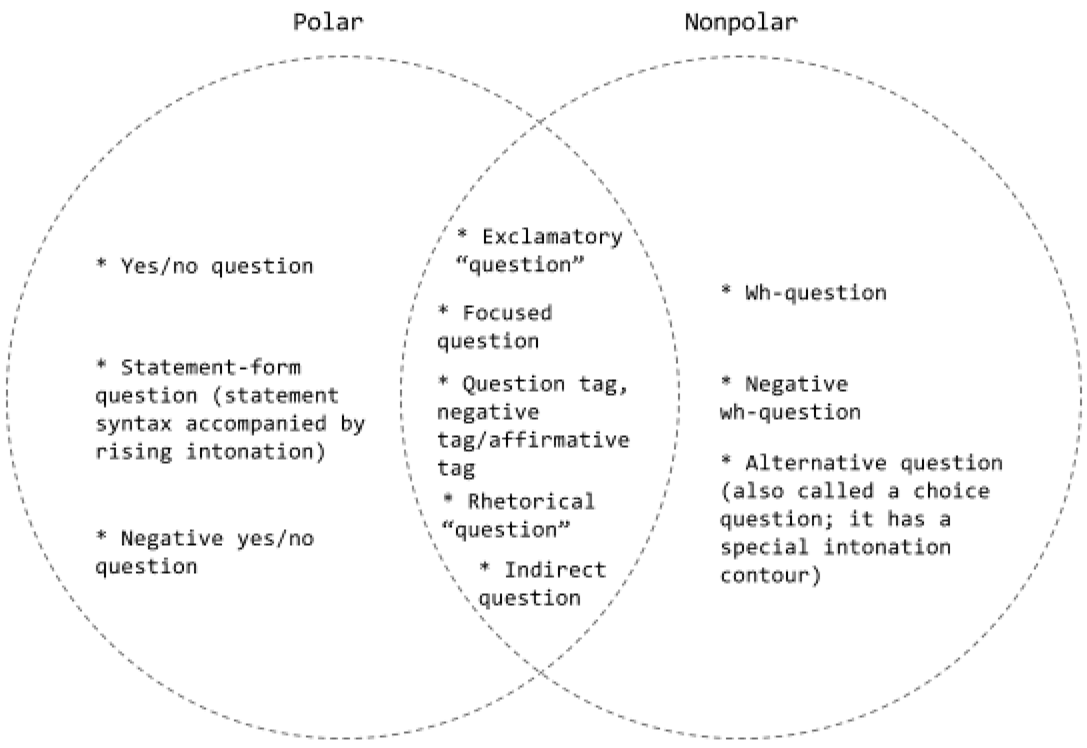

Polar
Nonpolar
* Yes/no question
* Statement-form question (statement syntax accompanied by rising intonation)
* Negative yes/no question
* Exclamatory "question"
* Focused question
* Question tag, negative tag/affirmative tag
* Rhetorical "question"
* Indirect question
* Wh-question
* Negative wh-question
* Alternative question (also called a choice question; it has a special intonation contour)

Figure 1 (6) - Organisational Structure of Question Types

Polar questions are also known as yes-no questions (4). These questions can be answered with a yes or no type of response such as “right”, “uh uh’, “yep”, “maybe” or in face to face conversations, via a nod of the head, a shrug or any other nonverbal indicator. As demonstrated by Dartmouth's School of Linguistics, generally, to form a polar question in English the subject and verb order are reversed (4). Figure 2 below has an example of this Subject-Verb reversal. Bold text is used to represent verbs, and brackets to represent the subject of the sentences.

1) "[He] **was** going to the shop" -> "**Was** [he] going to the shop?"
2) "[The job] **is** complete" -> "**Is** [the job] complete?"

Figure 2 - Subject-Verb reversal when forming a polar question

However, as stated by Larsen-Freeman et.al, polar questions can also be structured as statement-form questions where a statement is used to ask a question (6). In spoken conversation, the question aspect of the statement is indicated through intonation and via a question mark in the written form. This is indicated below in figure 3.

1) "You go to school?"
2) "He said that?"

Figure 3 - Statement Form Questions

Larsen-Freeman et.al also lists tagged questions as a type of question. These are formed of a statement, followed by an interrogative clause (8). These types of question can be found in either the polar or nonpolar categories, but are very commonly used as polar questions to ask for clarification or “...*serve as comprehension checks…”*(6). Figure 4 is an example of a conversation in which a question tag is used - taken from

Larsen-Freeman’s research. *B* asks for clarification using a question tag in this case. The interrogative clause is in bold.

A: I've got so much work that I don't believe it, so I'm just not thinking about that.
B: In school, **you mean?**

Figure 4 - Tagged Questions

Rhetorical questions, like tagged questions, can also be found in both polar and nonpolar categories. In a study of rhetorical questions by Ranganath et.al on social media an interesting method is provided to classify rhetorical questions. This study found that using syntactic analysis to build a feature set would not aid in the classification of rhetorical questions as “*they [rhetorical questions] are not syntactically different from other questions” (9).* Instead, Raganath et.al looked at the motivation of users based on previous tweet data. Social media allows researchers to assess political associations, sentiment and more, based on the user's previous posts. Although possible to access past messages sent from specific users and assess the motivation behind questions in online-chat discourse, this is beyond the scope of this paper. As rhetorical questions are by nature still questions, rhetorical questions will simply be classified as polar and nonpolar. Algorithmic decisions such as these are discussed in the Algorithm Analysis section below.

Machine Learning Classifiers

This research paper will focus on using Naive Bayes and Support Vector Machine classification respectively. This section of the paper will review the literature available on these two supervised machine learning classification methods only. The reasons for using these two classification methods are explained in the classifier analysis section below.

A lot of research around Naive Bayes classifiers exists. Briefly, a Naive Bayes classifier is a probabilistic classifier that gives a conditional probability of a feature *X* existing in a class *C.* This is done using Bayes rule which states $P(C|X) = \frac{P(X|C)P(C)}{P(X)}$ (10). In other words, the probability of a class given a feature is equal to the probability of a feature given a class, accounting for the probability that the class exists and accounting for the probability that a feature exists regardless of the class. One problem often highlighted by those researching machine learning techniques is that Bayes rule assumes "...that all attributes used to describe an instance are conditionally independent given the class of that instance" (10). This assumption means that Bayes rule cannot account for the probability of a word in a class given the fact it was preceded by another word. Essentially, the probability of "this is a sentence" being classified into class Ci is identical to the probability of "sentence is a this" being classified into the same class when treated as a bag-of-words (10).

However, as stated by Chen, "...[when] conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data" (11). A Naive Bayes classifier is very easy to implement and gives a baseline accuracy level to judge other machine learning classifiers on. Furthermore, as stated by Domingos and Pazzani, due to the simplicity of a Naive Bayes classifier and its basic linear nature, Naive Bayes classification is relatively robust to overfitting through a high dimensionality of features.

(12). Naive Bayes classifiers are also generative models, meaning that the classifier will predict the class for a feature set based on the maximum likelihood of a given feature set being generated by a class rather than fitting the feature set to a prebuilt linear model to generate a class prediction.

Support Vector Machines are discriminative models and build a hyperplane to separate features during the training of the model (16). Classification is done by placing the document in a high dimensional space in relation to the hyperplane. The features that help to distinguish the bounds of the hyperplane are known as support vectors (28). Therefore, unlike Naive Bayes which classifies the document by assessing which class is most likely to produce the document, SVMs predict the class of a given document based on its relation to the hyperplane, disregarding which class would generate the document. SVMs are commonly used in text classification problems. Joachims states that “...SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly” (17).

Current Research Regarding Question Detection and Polar Question Detection

Research into question detection has been conducted previously, but either at a macro level - classifying questions generally (2) - or at a micro level - detecting specific types of questions such as non-sentential utterances (26). Question detection has been studied before for specific types of online conversation. Arguably, the most notable paper on question detection comes from Afantenos et. al who aimed at detecting questions in forum data as part of a larger research project, trying to map questions to their corresponding answers (2). As stated by Afantenos et. al “The problem at first glance seems to be easy. Unfortunately, it turns out to be non-trivial on the basis of our analysis... ...Questions in forums are often stated in an informal way and questions are stated in various formats” (2). This is consistent with this research's dataset - made up not only of forum data but also online chat discourse - with statement form questions such as “You managed it?”, indirect questions and incorrectly formed questionss occurring in the dataset. Afantenos et al argue that a rule based classification system, simply looking at wh-words and a question mark were not adequate for the task and use a label sequential pattern to train their classifierseir (2). However, from the tests run in this paper, a high level of accuracy can be gained from rule based classification for question detection. It is possible however, that forum data, where a question is posed and then a larger explanation of the question and the reasons for asking is posted, may not respond well to rule based classification. In contrast to Afantenos et.al’s method where the appearance of a question word is marked as a single feature, this paper proposes the idea of a selective bag of words, where individual question words, made up of interrogative and auxiliary verbs are marked as individual feature, allowing the classifier to create a higher dimensional feature space and giving a greater insight into which words help predict which class. A full list of these words can be found in Appendix A.

Datasets and Corpora

The dataset used in this research is made up of data from multiple sources that exist in multiple domains. As this research focuses on dyadic and multi-party online chat discourse, a corpus of online chat was needed. As online chat conventions move relatively fast compared to the dialogue in spoken English, a modern corpus was needed. Before approaching groups of users to ask for online chat data, many online platforms were considered. For example, WhatsApp - the popular dyadic and multi-party chat application - was considered, but the functionality to export conversations does not exist. It was noted that Slack has an API that allows users to build and integrate chatbots. A chatbot was built to catch all conversations in real time and save them to a document database. At the start of this research, Slack groups were approached and asked if they would like to participate in the research. If so, they were instructed to install the bot application previously mentioned. However, whilst researching Slack in more detail and talking to Slack group administrators, it came to light that backdated conversations could be exported in JSON format. Therefore, the original idea of capturing data in real time was abandoned and a year and a half's worth of chat data was donated by the Android Chat community (18). The code for this chatbot has still been included separately in the support material accompanying this research paper in case others need to capture Slack data on the fly and perform preprocessing or any data manipulation in real time, as messages are being sent. The Android Chat channel has 5357 active members as of the date of this paper. Android Chat "...is an open community with a Slack team open to everyone. A place where every developer, idealist, entrepreneur or geek is welcome to share their ideas, questions and successes" (18). An overview of the dataset donated by Android Chat is shown in Figure 5 below. A more detailed insight into the dataset can be found in Appendix B.

Total number of Channels	Total number of JSON Files	Total number of utterances
25	7451	375473

Figure 5 - Summary of Android Chat Corpus

The hangouts channel was labeled by hand with questions labeled with a Q and non questions left without a label. Questions were also labeled with FULL, S or E. This is to denote if the utterance contains the whole question, or if the question was sent over a series of continuous messages, with S marking the start of a question, E for the end and FULL for a question encapsulated in a single message. A sample of this labelling can be found in Appendix C. These questions were appended to a question training data file and the non question data was appended to a non question training data file. All questions in the file were wrapped between *'__QuestionStart__'* and *'__QuestionEnd__'* markers and all non questions between *'__AnswerStart__'* (for ease of use, however, these are not answers, but non questions of any kind)

The Yahoo-based Contrastive Corpus of Questions and Answers (YCCQA) was also used as a dataset. This is a corpus of "about 90,000 questions and 575,000 answers" (19). These are colloquially written questions and are relatively representative of online writing convention. These were also appended to the respective non question and question training files.

The training data at this point contained approximately 753,894 non questions and 200,000 questions. Due to this large difference in data set sizes for each class, more questions were taken from the 30M Factoid Question-Answer Corpus (20). This increased the question dataset to approximately 25,923,705 questions. Although these question were generated by a machine, Serban et. al claim "...when [the generated questions were] presented to human evaluators, the generated questions appear comparable in quality to real human-generated questions".

Only a partial, randomly shuffled set of the data is used to reduce training time, meaning that the classifier is training on roughly 200,000 data points from each class. Increasing the size of the datasets did not yield better results past the 200,000 datapoint mark.

The corpus built for this paper contains a very large, broad dataset from forums, online chat and machine generated sources, with colloquial and less well structured questions and non questions to well structured, grammatically correct questions and non questions.

Methodology

Requirements and Intended Outcomes

The overall outcome for this research is to produce a machine learning classifier which can classify questions and subsequently produce an algorithm which reads these labeled questions and identifies polar and nonpolar questions. Although applicable to the general goal of discourse relationship mapping as mentioned below, detecting polar and nonpolar question types is also useful in other domains. Arguably, by classifying questions, chatbots understand which utterances need a response. Furthermore, polar questions are commonly used to clarify or refer to a comment that has already been made. Nonpolar questions can be longer and are open to interpretation. By building an understanding of this, artificial intelligence applications can cater responses and possibly optimise the speed and efficiency in information retrieval-like tasks.

The original purpose of this research was to aid in the overall task of mapping discourse relationships in multi-party chat conversation. Although this task was not in the scope of this project due to time restrictions and the large size of the task, this research can be built upon to produce the intended outcome and this outcome should therefore be discussed to aid in the understanding of the broader goal for this research.

Essentially, multi-party chat is made up of multiple threads, all interwoven, embedded and complexly structured (1), as discussed earlier. The broad task is to untangle these complex structures and build a labeled set of data with relationships between each utterance in an online conversation. One hypothesis that this research is based on is that a ‘one size fits’ all approach to conversation mapping is not the correct approach as conversation is too complex. Online conversation - as explained in the review of literature on this subject - comes with its own set of added problems often stemming from the colloquial manner with which users interact. Therefore, this paper proposes the idea of classifying utterance relationships for smaller sets of linguistic acts, with the goal of building up a library of parsers to eventually be able to label entire multi-party chat conversations. For this reason, the focus of this research is based on detecting polar and nonpolar questions to aid future research into mapping discourse relations. Resources and time did not permit the next step of research which would have been to map polar and nonpolar questions to corresponding answers using a combination of LDA and rule based queries such as distance between question and answers, timestamps, previous user interactions etc.

The ability to map question and answer relationships is applicable in many real world and academic pursuits. For example, Schegloff et.al seminal paper “The Preference For Self-Correction In The Organisation Of Repair In Conversations” (13), explains the idea of organisational repair, commonly known to linguists as the repair framework. Briefly, Schegloff et.al argue that common understanding is achieved through correction and clarification - speakers repairing their own statements, others repairing speakers’ statements and so on. This theory was proposed as a direct alternative to many theories which stated that humans came to achieve understanding in a conversation through a priori knowledge (13). Furthermore, Schegloff argued that repair could occur in the nth position, meaning that a correction or clarification statement may occur at any position in the conversation after the initial trouble source (13). Although possible for humans to understand this to a high accuracy, it appears from research that no adequate computational model exists to help link clarification to the initial trouble source. The aim of detecting polar and nonpolar questions in this paper, is to aid the development of those looking to build a way to detect repair and clarification statements (although other applications for this research exist and are discussed in the the Further Research and Recommendations section below). To illustrate, Facebook launched eCommerce within facebook messenger in late 2016 as covered by The Guardian - *“Facebook added e-commerce capability – allowing Messenger bots to accept payments without requiring users to leave the app. People with credit card information stored with Facebook or Messenger will be able to make instant purchases within the bots of their favorite stores and services” (14)*. Should a user order item X in the colour blue and later, before paying decide to change the colour to green, the chatbot must be able to account for this. Should the user have purchased multiple items and want to make a change or clarify information about one of the products before making a payment, this bot must understand where the trouble source is and what the clarification means in relation to the object. In order to detect repair in the nth position, conversations must be untangled and utterance relationships need to be mapped.

To complete the tasks of question classification and polar nonpolar question identification, the following languages, libraries and infrastructures have been used. All of the scripts used to traverse and combine the multiple datasets are written in Ruby as it is the prefered language for the researcher. Python has been used to build out the main application and classifiers, using the Natural Language Toolkit Library (NLTK) (15) which in turn uses Scikit-Learn, which contains pre-written classifiers used in this research. JSON libraries for Ruby and Python have been used to work with the structured datasets. The source code is held on GitLab, using Git as the version control system and due to the size of the dataset, a VM instance on Google Cloud was provisioned to speed up the classifier training and big data tasks. All development and testing has been on Unix systems (MacOS and Debian).

Problem Analysis

The following is an explanation of the problems found during this research. Proposed solutions to these problems are explained in the next section.

1) As stated in the literature review, abbreviations, acronyms, clipped words, and deletion of subject pronouns pose large difficulties for question classification and for polar question detection. For example, the polar question detection algorithm will use POS tagging on the word before and after the auxiliary verb. Should the word be misspelled or abbreviated, the word will not be correctly tagged. As an important section of this algorithm checks preceding and proceeding words around the auxiliary verb, should the tag be incorrect, the polar question will not be correctly identified. For example, *"Is Google in California"* is a polar question. Should the user type *"Is Gogle in California"* the misspelled word *'Gogle'* will not be tagged as a proper noun. Furthermore, semantics surrounding an utterance give the statement "am going tomorrow" meaning. English speakers can infer the word I proceeding the statement which should read "I am going tomorrow". However, as the polar detection algorithm will look for an auxiliary verb followed by a word in the POS tag list (figure 6), this could be incorrectly identified as a polar question.

2) Another problem is the use of URL’s and other symbols. The question classifier will use question marks as a feature. URL’s contain question marks to mark the beginning of a query string and emoticons made by users such as *‘:?’* can occur in questions and non questions. These may skew the classification of questions during training. In contrast to the overuse of question marks during non question statements, it should be noted that due to the colloquial manner of online chat, question marks and punctuation generally are often not apparent. This poses another problem for the classification problem when using a question mark as a feature.

3) Another problem lies in the use of interrogative words . Although interrogative words such as *‘where’* and *‘when’* are commonly used to form questions, they can also be used in non question sentences such as *‘I still remember when he proposed”’*. Again, much like question marks, the use of interrogative words in non question statements may skew the classification of questions during training of the classifier.

4) Although a lot of data has been collected to train the classifier, the question data was collected from online chats, forums and machine generated questions. Therefore, although large, the dataset does not reflect the exact nuances of multi party chat. Also, the question corpus created for this research is larger than the non question data by a factor of 10. This may skew the classification of

questions.

5) As mentioned in the Question Structure section of the literature review, polar questions that do not have a reversed subject-verb structure can be formed as a statement such as *'You going?' or even just "Going?"*. This is recognisable during spoken conversation because English speakers will rise at the end of the sentence changing the intonation. Statements questions can be identified in written communication from the use of a question mark or from the context of other utterances surrounding the statement. Should a statement question not contain a question mark, without looking at the semantic meaning surrounding a statement question it will not be feasible to classify a statement polar question as polar rather than non polar.

Solutions and Considerations

The following are possible solutions and approaches to the difficulties presented in the Problem Analysis section of this paper as well as general considerations that must be accounted for before building the machine learning classifier and the polar / non polar identification algorithm.

Abbreviated, clipped and misspelled words are very common in online chat data. However, unlike in classic bag of words tasks such as sentiment analysis, where frequency of words are counted and then used as features to train a classifier upon, due to the lack of domain knowledge around sentiment in the document, much academic literature exists around the linguistic rules of questions in the EEnglish language.

Therefore, using domain knowledge, a classifier can be trained on a feature set of predefined words. This in turn, means the majority of words can be discarded, reducing the reliance on correctly spelled, punctuated and well formed utterances.

The problem of question marks outside of the formal use of forming a question could be tackled during the preprocessing stage, or by writing a function to validate a given string. This function should use regular expressions to check for the presence of URL’s and discard utterances that contain them. Although it is possible for a question to occur in an utterance that contains a URL, this may work as a base line. Following this, stripping the URL’s but leaving the rest of the utterance for classification may be a possible next step.

The use of interrogative words in non questions is a challenging problem to solve. As stated by the Cambridge dictionary “We usually form *wh*-questions with *wh*- + an auxiliary verb (*be, do* or *have*) + subject + main verb or with *wh*- + a modal verb + subject + main verb” (21). For this reason, a binary feature could be added to the feature space, looking for an interrogative word + an auxiliary verb. This may help identify those utterances where an interrogative exists within an auxiliary verb, but it still does not account for uses of ‘*what, who, which* or *whose’* which when used as the subject of a sentence, can form a question without using an auxiliary verb (22). An example is shown in figure 7.

Who jumped off the bridge?

Figure 7 - Non polar question using an interrogative that isn’t followed by an auxiliary verb

Should a complete utterance start with an interrogative word, it has a high likelihood of being a question but it does not mean that a question must start with an interrogative, see figure 8.

And she's here until when?

Figure 8 - Question that doesn't start with an interrogative

The problem of interrogative words in non questions will affect both question classification and nonpolar question identification. However, some of the problems can be mitigated through the solutions stated earlier - creating a function to detect interrogatives + auxiliary verbs and a function to detect interrogative words at the start of a sentence. However, these help detect questions but this solution is still unable to identify non questions with interrogatives.

More generally, machine learning tasks have a high reliance on preprocessing. As stated by Shrivastava and Sridharan "The data preprocessing can often have a significant impact on generalization performance of a supervised ML algorithm" (23). Although utterances must be converted into feature vectors, this research will not be using stemming, lemmatization or removing stop words because individual words are not being used as features. However, to detect polar questions, POS tagging must be applied. Therefore, utterances will be tokenized and tagged. Furthermore, to detect an auxiliary verb following an interrogative word, utterances will also be converted into an array of bigrams.

Algorithm Design

The design of the full algorithm is explained below. This algorithm incorporates the solutions proposed in the Solution Analysis section of the Methodology.

The goal of this research has been split in the two parts as mentioned earlier. Firstly, questions and non questions need to be detected from structured data (due to the fact that most modern multi party chat platforms often export chat data as a JSON) . Secondly, the questions must be sorted into polar and nonpolar categories.

To approach the first task of classifying questions and non-questions, a classifier will be trained on 73 separate features. 70 of these features are *'question words'*, which for the purpose of this research are made up of interrogative determiners, interrogative pronouns, interrogative adverbs and auxiliary verbs as well as a question mark which although not treated as a word linguistically, will be treated as a feature and as such will be included in the *'question word'* list to allow the algorithm to identify it as part of the feature space. This acts as a selective bag of words approach to the classification problem. The list of question words can be found in Appendix A.

Another feature that will help to train the classifier are startwords. Arguably, if the utterance starts with a question word such as *'When'* or *'Where'*, it will be a question. Exceptions to this rule exist and are discussed in the Problem Analysis section.

Another feature that will be used to train the classifier is polar question identification. A function will be written to detect polar questions, treating the problem of polar/nonpolar as a rule based problem rather than a machine learning classification problem. The function returns a boolean, and should a polar question exist in the utterance a feature called *'Polar'* will be set to true. All features are binary for this research. The polar question function works by tokenizing and then using part of speech tags on each word. It then checks if the utterance contains more than one word. If it does, the algorithm loops eachthrough word. Should the word occur at the beginning of the sentence, the algorithm simply checks if the next word has a POS tag contained in a list of carefully selected POS tags, shown in figure 6.

```
["DT", "NNS", "NN", "NNP", "NNPS", "PRP", "PRP$", "VBG" "PDT", "RB", "RBR", "RBS", "IN", "TO"]
```

Figure 6 - POS tags used to detect polar words by looking at words in position: Auxiliary Verb + 1

Examples of the rule which focuses on the auxiliary verb at the beginning of the sentence followed by a POS tag in the POS tag list are displayed in figure 9 - Auxiliary verbs are marked in bold and POS tags and their corresponding words are in brackets.

Is [it DET] okay?
Are [you PRP] going?
Shall [we PRP] go now?
Didn't [John NNP] think about it?

Figure 9 - Polar questions formed of a starting auxiliary verb followed by a POS tag

These POS tags are chosen because they represent the type of words that can follow an auxiliary verb and form a polar question. For example, in the polar question "Is James there?" the auxiliary verb *'is'* is followed by *'James'* - which is a singular proper noun "NNP". As the auxiliary verb is in the first position of the utterance, the algorithm marks this as a polar question. However, should the auxiliary verb exist in a position greater or equal to the second position - i.e not at the beginning - then the algorithm checks if the POS tag of the word before the auxiliary verb exists in the POS tag list. This is because, although "Is James there" is a polar question, should the statement read “Why is james there?” then it becomes a nonpolar question. Therefore, the POS tag before the auxiliary verb is just as important as the POS tag for the word following the auxiliary verb. Finally, the algorithm checks for the word *‘how’*. If the word *‘how’* exists and is followed by an adjective (“JJ”) or an adverb (“RB”) then a polar question is turned into a nonpolar question. For example, “is it?” is a polar question. However, “How much is it?” is a nonpolar question. Therefore the algorithm must be able to detect *“how”* followed by a “JJ” or an “RB”.

The final feature - labeled as nonpolar - is a nonpolar detection function. Should a sentence contain an interrogative word followed by an auxiliary verb, it is likely to be a nonpolar question. Examples are shown below in figure 9 - Interrogatives are marked in bold and auxiliary verbs in brackets.

"**Where** [is] he"
"**What** [does] this mean?"
"I saw him there, **how** [did] the date go?"

Figure 9 - Interrogative word followed by an auxiliary verb

These are not catch all rules and exceptions do exist as discussed in the Problem Analysis section.

Four tests will be run on two classifier (Naive Bayes and Linear SVM) - eight iterations in total. Training of the SVM will be done using 10 fold cross validation. The first test uses all features explained above - selective bag of words, startwords, polar and nonpolar detection. The second test will be run using only startwords and the selective bag of words. The third test will only use selective bag of words and the fourth test will only use polar and nonpolar detection rules as features.

For the Naive Bayes classifier, 50% of the data is held as a training set and 50% as a testing set. The testing set is used to measure accuracy and to build a confusion matrix. From this, error and accuracy scores are generated for both classifiers to enable a comparison of the effectiveness of each.

Once the classifier is trained, an unseen channel will be chosen from the Android Slack Community which contains multiple files. Each file is read into the algorithm and each classifier - during each test - generates a txt file of questions from the corpus. These questions will be hand labelled to check for errors and to gain a false positive classifications rate on an unseen dataset. The txt file of classified questions generated by the test and classifier with the highest accuracy rate will be used to build two separate txt files. The polar identification function will be applied to the newly generated dataset and those questions that are detected as polar will be written to a polar txt file and those that are not polar will be classed as nonpolar and written to the respective nonpolar txt file.

Outcomes

Results

After running the full program the results look extremely promising. The following metrics were measured for all tests on both the Naive Bayes classifier and the SVM with a linear kernel: Accuracy, Recall, Precision. Accuracy measures the overall performance of the classifier by taking all the correctly classified occurrences divided by the total dataset. Let θ represent the whole dataset, α represent true positives, β

represent false positives, λ represent true negatives and ν represent false negatives. Therefore, $\alpha \subset \theta$, $\beta \subset \theta$, $\lambda \subset \theta$, and $\nu \subset \theta$ respectively. Accuracy is a measure of the overall correctness of the classifier (29). Recall is a measure of the true positive rate. In other words, the measure of how often the classifier correctly labels a question when given a question (29). Precision is a measure of how often a predicted question label is correct.

For Naive Bayes, question prevalence is also calculated and the raw confusion matrix is displayed. The question prevalence metric allows the calculation of the null accuracy rates. This is the percentage of times the classifier will be correct if it classified everything in the majority class (29). For example, using test 1 (figure 10.a) as an example, Questions are the majority class and therefore the null classification rate is 50.02% - calculated as $\frac{99871 + 169}{99871+169+7758+92202} * 100$. Essentially, if the classifier was to blindly classify everything in θ as a question, it would be correct 50.02% of the time. This gives a good baseline accuracy to assess the real accuracy level against.

Figure 10 shows the results from the question classification tests run through the Naive Bayes classifier.

	Question	Non Question
Question	99871	169
Non Question	7758	92202

Accuracy	95.934
Error Rate	3.9635
Recall	99.8310

Precision	92.7919
Question Prevalence	50.0195

Figure 10.a - Test 1 results: Naive Bayes

	Question	Non Question
Question	99792	151
Non Question	2034	98023

Accuracy	98.873
Error Rate	1.0925
Recall	99.8489
Precision	98.00247
Question Prevalence	49.9715

Figure 10.b - Test 2 results: Naive Bayes

	Question	Non Question
Question	99727	178
Non Question	4095	96000

Accuracy	97.9025
Error Rate	2.1365
Recall	99.8218
Precision	96.0557
Question Prevalence	49.9525

Figure 10.c - Test 3 results: Naive Bayes

	Question	Non Question
Question	43260	56702
Non Question	2066	97972

Accuracy	70.616
Error Rate	29.384
Recall	43.276445049118664
Precision	95.4419
Question Prevalence	49.981

Figure 10.d - Test 4 results: Naive Bayes

As well as the universal metrics used for both the Naive Bayes and SVM tests, the f-score was included for the SVM tests. F-score is the weighted average of recall and precision. These tests were run using k-fold cross validation. The dataset was large enough to use 10 folds. The advantage of using k-fold cross validation is to gain an average recall, precision and f-score, increasing the reliability of these results.

The full output of each test for both classifiers used, can be found in Appendix D. Figure 11 presents a summation of tests 1 through 4 using an SVM.

Accuracy	99.3385
Recall	99.3385
Precision	99.3408
F-Score	99.3384

Figure 11.a - Test 1 results: Linear SVM

Accuracy	99.2982
Recall	99.2984
Precision	99.3009
F-Score	99.2982

Figure 11.b - Test 2 results: Linear SVM

Accuracy	99.1802
Recall	99.1802
Precision	99.1823
F-Score	99.1802

Figure 11.c - Test 3 results: Linear SVM

Accuracy	70.7864
Recall	70.7863
Precision	79.5095
F-Score	68.4549

Figure 11.d - Test 4 results: Linear SVM

Each test (1-4) was identical for each classifier. Test 1 used the whole feature set which was made up of 73 features, explained in figure 12.

Feature Identifier	Explanation
Question Words (Actually made up of 70 separate features)	This was made up of 70 separate features, each of which was a question word. Question words is a grouping term for interrogative words and auxiliary verbs as well as "?". (A

	full list of these question words can be found in Appendix A)
Startword	Does the sentence start with a question word?
Polar	Does the sentence contain a polar question as determined by the polar identification algorithm?
Nonpolar	Does the sentence contain a non polar word (See Appendix D) followed by an auxiliary verb?

Figure 12 - Feature list

Test two used a feature set made up of start words and question words, disregarding polar and nonpolar identification features. Test three only used question words as a feature set and test four only used polar and nonpolar identification features.

Each Naive Bayes test produced a list of most likely features, which was a ratio that represented the strength of each feature in influencing the class prediction. For example, in test 1, the 6 most likely features to influence the class prediction are shown in figure 13.

```
? = 0                   NQ : Q     =    169.6 : 1.0
dont = 1                NQ : Q     =    134.2 : 1.0
should = 1              NQ : Q     =    116.8 : 1.0
cant = 1                NQ : Q     =    107.5 : 1.0
could = 1               NQ : Q     =     85.0 : 1.0
startword = 1           Q : NQ     =     64.2 : 1.0
```

Figure 13 - Top 6 most likely features in test 1 using a Naive Bayes classifier

These feature ratios will be discussed later. Briefly 0 represents the lack of the feature and 1 represents the existence of the feature. NQ is a non question and Q is a question. The lack of a question mark for example, indicates that this utterance is 169.6 times more likely to be a non question than a question.

After the training and testing of the classifier on the labeled dataset, unseen data from the slack corpus - all 271 JSON files from the troubleshooting channel (see appendix B) - were fed into the algorithm with the highest accuracy rate and subsequently, classified questions from this unseen dataset were written to a txt file. The content of this txt file was hand labelled, with non questions that appeared in the classified question list being marked as errors to check the false positive rate. The results of this hand labeled dataset are presented and explained in the Question Classification section below.

Question Classification

Question classification accuracy rates for all tests came out extremely high. At first, the running theory for this was around the training and testing dataset. It was thought that the training data producing these results contained some of the testing data. However, this did not appear to be the case. It was then thought that the dataset was too simple and question marks were more prevalent than might occur in real world scenarios. However, a program was build that allowed questions to be input in the terminal and the prediction was output back to the user. Complicated sentence structures that included questions without question marks such as *"I'm not sure, i think that she is going to go home around 6 so are you sure that she is still here"* were fed into the program and were correctly classified as a question. This test allowed the testing of the classifier on sentences that did not contain obvious features such as question marks or start words. It is possible that this machine learning task had such a high accuracy because it is a relatively simple problem that works well on generative models such as Naive Bayes. The reason this may work on Naive Bayes is because the assumption of feature independance that underpins this type of classifier, holds true for this problem. The feature “what” being present will be a strong indicator of a question and the existence of the word “when” also occurring in the same sentence should not increase the likelihood that the sentence belongs to the question class.

Interestingly, looking at the Naive Bayes results (figure 10) test 1, using the full feature set had a slightly lower classification rate than test 2 which only looked at start words and question words. Arguably, the main reason for this difference in accuracy, is due to the false positive rates. It would appear that the extra features of polar and nonpolar question identification increased the false positive rate from 2034 false positives in the test without the polar and nonpolar identification features to 7758 occurrences of false positives with these features.

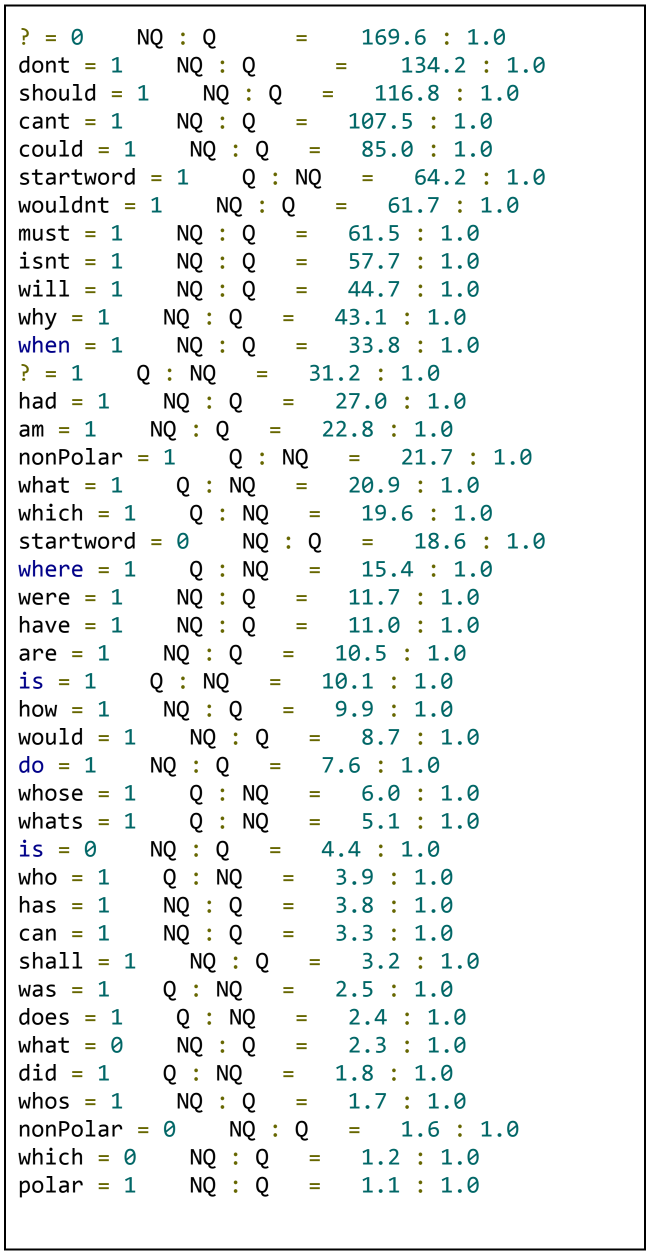
Although these two features reduced the accuracy of the classification task using a Naive Bayes classifier, these features on their own still produced positive results. Just using polar and nonpolar features produced a classifier with a 70.6% accuracy. This is much higher than the null accuracy rate of 50.019%. Therefore, these features are beneficial in classifying questions. However, they are not as important as the selective bag of words features (question words) or the start word feature.

The SVM with a linear kernel produced even better results with the highest f-score rate from the four tests being 99.3384%. This is in contrast to the Naive Bayes classifier which performed better without the polar and nonpolar identification features. However, the difference between test 1, test 2 and test 3 was approximately 0.2% and is therefore not statistically significant.

These extremely high values mean that this research is unable to come to definitive conclusions on the accuracy of the classifiers in detecting questions. Ideally, an unseen dataset would be hand labeled and then the same unseen dataset would be fed into the algorithm to produce a list of questions. These two lists can then be analysed to understand the true accuracy of this classification method by hand. Time and resources did not permit a large dataset to be hand labeled to test the classification. The hand labeled data that was produced was used to train the classifier. However, 21 of the 271 JSON files from the troubleshooting channel of the Slack corpus was labeled and the same 21 files were classified to produce a list of questions picked out by the classifier. These two lists were checked against each other and it was found that the classifier actually had a 79% accuracy. This is not a definitive test as the dataset was very small and fairly homogenous, with the same users appearing again and again. However, it would appear that the classification rate although not in the 99%, is very high and outperforms that of Afantenos et. al who also tested a rule based question classifier and concluded that a more complex classification method was needed.

Furthermore, the classified questions were checked for errors by hand. Although this approach cannot check the true negative rate - questions in corpus that were missed by the classifier - it was able to check the false positive rate - those questions which were classified as questions incorrectly. Using this hand checked data, it was found that in a set of 402 questions, 4 non questions were found. This means the classifier was predicting a question when given a non question 0.995% of the time. This is consistent with the test results and gives weight to the classifiers scores.

In regards to the use of features, the Naive Bayes tests demonstrate relatively common sense results. For example, figure 13 presents test 1 features - using a Naive Bayes classifier on the total feature set - and their impact on classification (All features below 1.1:1 ratios have been mitigated for brevity. Full lists of features can be found in the full test output in appendix D).



```
? = 0    NQ : Q      =    169.6 : 1.0
dont = 1    NQ : Q      =    134.2 : 1.0
should = 1    NQ : Q    =   116.8 : 1.0
cant = 1    NQ : Q    =   107.5 : 1.0
could = 1    NQ : Q    =   85.0 : 1.0
startword = 1    Q : NQ    =   64.2 : 1.0
wouldnt = 1    NQ : Q    =   61.7 : 1.0
must = 1    NQ : Q    =   61.5 : 1.0
isnt = 1    NQ : Q    =   57.7 : 1.0
will = 1    NQ : Q    =   44.7 : 1.0
why = 1    NQ : Q    =   43.1 : 1.0
when = 1    NQ : Q    =   33.8 : 1.0
? = 1    Q : NQ    =   31.2 : 1.0
had = 1    NQ : Q    =   27.0 : 1.0
am = 1    NQ : Q    =   22.8 : 1.0
nonPolar = 1    Q : NQ    =   21.7 : 1.0
what = 1    Q : NQ    =   20.9 : 1.0
which = 1    Q : NQ    =   19.6 : 1.0
startword = 0    NQ : Q    =   18.6 : 1.0
where = 1    Q : NQ    =   15.4 : 1.0
were = 1    NQ : Q    =   11.7 : 1.0
have = 1    NQ : Q    =   11.0 : 1.0
are = 1    NQ : Q    =   10.5 : 1.0
is = 1    Q : NQ    =   10.1 : 1.0
how = 1    NQ : Q    =   9.9 : 1.0
would = 1    NQ : Q    =   8.7 : 1.0
do = 1    NQ : Q    =   7.6 : 1.0
whose = 1    Q : NQ    =   6.0 : 1.0
whats = 1    Q : NQ    =   5.1 : 1.0
is = 0    NQ : Q    =   4.4 : 1.0
who = 1    Q : NQ    =   3.9 : 1.0
has = 1    NQ : Q    =   3.8 : 1.0
can = 1    NQ : Q    =   3.3 : 1.0
shall = 1    NQ : Q    =   3.2 : 1.0
was = 1    Q : NQ    =   2.5 : 1.0
does = 1    Q : NQ    =   2.4 : 1.0
what = 0    NQ : Q    =   2.3 : 1.0
did = 1    Q : NQ    =   1.8 : 1.0
whos = 1    NQ : Q    =   1.7 : 1.0
nonPolar = 0    NQ : Q    =   1.6 : 1.0
which = 0    NQ : Q    =   1.2 : 1.0
polar = 1    NQ : Q    =   1.1 : 1.0
```

Figure 13 - Most likely feature table generated from test 1 using a Naive Bayes classifier

As stated earlier, a lot of these feature impacts are intuitive. For example if a statement does not have a question mark, it is 169.6 times more likely to be a non question. This is not to say that questions without question marks will automatically be added to the non question list. Examples of classified questions that do not contain a question marks, taken from the list of classified question is shown in figure 14 below.

```
1. what doesn't work about it
2. <@U055GGAHS>: So i have this in my IabHelper.QueryInventoryFinishedListener which gets
called in oncreate. Is this the right idea
```

Figure 14 - Examples of classified questions that do not contain question marks

Furthermore, common words which are used in questions and non questions such as don’t, should, can’t and must are clearly used more in non questions than in questions as demonstrated by the high likelihood of non question classification given that these features exist in an utterance as presented in figure 13. Finally, the use of a start word also yields a high likelihood of a question. This should be a very obvious indicator of a question in formal English but due to the colloquial manner in online chat discourse, it is not as high of an indication.

Polar and Nonpolar Question Identification

Polar and nonpolar question identification using a rule based algorithm is more difficult to produce metrics for. On the troubleshooting dataset, after questions had been written to a txt file, each question was run through the polar identification algorithm. When the polar identification algorithm returned true, the question was written to a polar txt file and when false, to a non polar question list. There are no non polar questions in the polar question list, meaning the polar identification algorithm works at disregarding nonpolar questions. However, in the nonpolar question list, polar questions are found. An example of these incorrectly identified questions can be found in figure 15.

```
1. Someone got an M8/1+1 I can test it on?

2. yo guys. this is sick dude. anyone in the house have retrofit experience?

3. So you want to set the `Notification.FLAG_ONGOING_EVENT` on the notification correct?

4. Aren't emojis just utf8 characters?

5. that's not a real location, just a symlink to the camera folder. you get Uri via fetching
external storage?
```

Figure 15 - Examples of polar questions in nonpolar list

Figure 15, example 1 shows a common error in the identification of polar questions. As the algorithm in it's simplest form, looks for an auxiliary verb followed by a word that has a POS tag in a predefined list (figure 6), when a question does not contain an auxiliary verb in the correct position, the algorithm incorrectly identifies the word. For example, figure 15.1 - should read “Has someone got an M8/1+1 I can test it on?”. With the auxiliary verb ‘has’ preceding the word someone, the algorithm would correctly identify the question. One flaw that is evident after testing this algorithm is that any rule based linguistic algorithm will be rigid and partial based on formal rules of English. Therefore, for colloquial conversations, the accuracy of non machine learning techniques to detect linguistic tasks outside of formally structured English sentences, may be relatively low. Figure 15.5 is a very interesting example of problems with rule based linguistic algorithms. The question clause of the utterance is a statement question “you get Uri via fetching external storage?”. Using a machine learning technique to detect questions, this statement question was classified correctly. However, using a rule based algorithm to identify if this question is polar, meant that no question structure rules could be applied to identify the question, because without the question mark, this would be a non question.

Conclusion

To conclude, a selective bag of words approach to questions classification combined with a few other features generated through domain knowledge, works very well on the binary task of question / non question classification. Due to the fact that this solution assumes feature independance, very high classification rates can be achieved on a simple Naive Bayes classifier. Arguably, SVM’s help to increase the classification rate but not dramatically. Polar identification is possible through a non machine learning based algorithm but with a very high false negative rate. From this research, it would appear that a dramatic increase in accuracy of polar / nonpolar identification could occur if a) many more edge case rules are included such as finishing a statement with the clause ‘right?’ b) more preprocessing such as spell checking occurred on the list of questions before attempting to identify polar and nonpolar statements. This research paper has developed a novel approach to polar and nonpolar identification as well as tested and arguably refuted Afantenos et. al (2) statement that question classification cannot be achieved through a rules based approach to building feature vectors.

Further Research and Recommendations

As stated earlier in this paper, this research was originally undertaken to help aid in the development of a full, multi party chat discourse relation parser that maps utterances to corresponding utterances. The argument was that conversation is too complex and parsers and labelling systems for individual linguistic sub clauses and interactions such as questions and answers should be developed to further the development of this larger goal around discourse relation and conversation mapping. This research has focused on question detection and polar and nonpolar question identification. If more time was available, the nonpolar question would be used as documents in a LSA or LDA and topic modelling on utterances would be research. The idea was to use the topic probability distribution distances (using jensen shannon divergence) as a feature in a second classifier which would also use previous user interactions, distance in time and distance in number of separating utterance between questions and possible answers to produce a list of answers that mapped to corresponding questions. Others are urged to take this topic on and build on the research of this paper.

References

1. Uthus D, Aha D. Multiparticipant chat analysis: A survey. Artificial Intelligence [Internet]. 2013 [cited 24 July 2017];199-200:106-121. Available from: http://www.sciencedirect.com/science/article/pii/S0004370213000234#bl0010

2. Afantenos S, Kow E, Asher N, Perret J. Discourse parsing for multi-party chat dialogues [Internet]. Toulouse: Université Toulouse & CNRS, Univ; 2017 [cited 24 July 2017]. Available from: http://aclweb.org/anthology/D15-1109

3. Kelly C. Do you know what I mean > :(A linguistic study of the understanding of emoticons and emojis in text messages [Internet]. Stockholm: Halamstad University; 2017 [cited 25 July 2017]. Available from: http://www.diva-portal.org/smash/get/diva2:783789/FULLTEXT01.pdf

4. Questions [Internet]. Dartmouth.edu. 2017 [cited 26 July 2017]. Available from: http://www.dartmouth.edu/~deutsch/Grammatik/WordOrder/Interrogatives.html

5. Japanese Sentence Structure: A Beginner's Guide to Forming Japanese Sentences [Internet]. Fluentu.com. 2017 [cited 26 July 2017]. Available from: http://www.fluentu.com/blog/japanese/japanese-sentence-structure-patterns/

6. Larsen-Freeman D, Celce-Murcia M, Frodesen J. The grammar book. 3rd ed. Boston: Cengage Learning; 2016.

7. Question [Internet]. SIL Glossary of Linguistic Terms. 2017 [cited 26 July 2017]. Available from: http://www.glossary.sil.org/term/question

8. Carter R, McCarthy M, Mark G, O'Keeffe A. English Grammar Today + Workbook. 1st ed. Cambridge: Cambridge Univ Pr; 2016.

9. Ranganath S, Hu X, Tang J, Wang S, Liu H. Identifying Rhetorical Questions in Social Media [Internet]. Tempe: Arizona State University; 2017 [cited 28 July 2017]. Available from: http://www.public.asu.edu/~srangan8/resources/pub/ICWSM_2016.pdf

10. Langseth H, Nielsen T. Classification using Hierarchical Naïve Bayes models. Machine Learning [Internet]. 2006 [cited 29 July 2017];63(2):135-159. Available from: https://link.springer.com/article/10.1007/s10994-006-6136-2

11. Chen E. Choosing A Machine Learning Classifier [Internet]. echen. 2017 [cited 29 July 2017]. Available from: http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/

12. Domingos P, Pazzani M. On the Optimality of the Simple Bayesian Classifier under ZeroOne Loss [Internet]. Boston: Kluwer Academic Publishers; 2017 [cited 29 July 2017]. Available from: http://web.cs.ucdavis.edu/~vemuri/classes/ecs271/Bayesian.pdf

13. Schegloff E, Jefferson G, Sacks H. The Preference for Self-Correction in the Organization of Repair in Conversation. Language [Internet]. 1977 [cited 30 July 2017];53(2):361. Available from: http://icar.univ-lyon2.fr/ecole_thematique/TRANAL_I/documents/org_seq/scheSacksJeff77_repair.pdf

14. Walsh M. A tough sell: why Facebook's e-commerce dream failed to take flight. The Guardian [Internet]. 2016 [cited 30 July 2017];. Available from: https://www.theguardian.com/technology/2016/oct/01/facebook-businesses-online-shopping-chatbots

15. NLTK. NLTK Project; 2017.

16. Touretzky D. Advances in neural information processing systems: Volume 14. 1st ed. London: MIT Press; 2001.

17. Joachims T. Text categorization with Support Vector Machines: Learning with many relevant features [Internet]. Berlin: Springer; 1998 [cited 19 August 2017]. Available from: https://link.springer.com/chapter/10.1007/BFb0026683

18. AndroidChat [Internet]. Androidchat.co. 2017 [cited 19 August 2017]. Available from: http://androidchat.co/

19. De Smet H. CoRD | Yahoo-based Contrastive Corpus of Questions and Answers [Internet]. Helsinki.fi. 2009 [cited 19 August 2017]. Available from: http://www.helsinki.fi/varieng/CoRD/corpora/YCCQA/index.html

20. Serban I, García-Durán A, Gulcehre C, Ahn S, Chandar S, Aaron A et al. Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus [Internet]. 2nd ed. Cornell University; 2016 [cited 19 August 2017]. Available from: https://arxiv.org/abs/1603.06807v2

21. Questions: wh-questions [Internet]. Cambridge Dictionary. Cambridge: Cambridge Press; 2017 [cited 20 August 2017]. Available from: http://dictionary.cambridge.org/grammar/british-grammar/questions-and-negative-sentences/questions-wh-questions

22. Questions: wh-questions [Internet]. Cambridge Dictionary. Cambridge: Cambridge Press; 2017 [cited 20 August 2017]. Available from: http://dictionary.cambridge.org/grammar/british-grammar/questions-and-negative-sentences/questions-wh-questions

23. Shrivastava H, Sridharan S. CONCEPTION OF DATA PREPROCESSING AND PARTITIONING PROCEDURE FOR MACHINE LEARNING ALGORITHM [Internet]. Bangalore: M.Tech; 2017 [cited 20 August 2017]. Available from: http://www.academia.edu/9517738/CONCEPTION_OF_DATA_PREPROCESSING_AND_PARTITIONING_PROCEDURE_FOR_MACHINE_LEARNING_ALGORITHM

24. Newman D. Chatbots And The Future Of Conversation-Based Interfaces [Internet]. Forbes.com. 2016 [cited 20 August 2017]. Available from: https://www.forbes.com/sites/danielnewman/2016/05/24/chatbots-and-the-future-of-conversation-based-interfaces/#257691271fc7

25. Miller G. AI, Machine Learning, NLP, and Deep Learning? – Chatbots Journal [Internet]. Chatbots Journal. 2017 [cited 20 August 2017]. Available from: https://chatbotsjournal.com/ai-machine-learning-nlp-and-deep-learning-f55456394b82

26. Fernández R, Ginzburg J, Lappin S. Classifying Non-Sentential Utterances in Dialogue: A Machine Learning Approach. Computational Linguistics. 2007;33(3):397-427.

27. Chatbot [Internet]. Oxford Living Dictionary. Oxford: Oxford University Press; 2017 [cited 20 August 2017]. Available from: https://en.oxforddictionaries.com/definition/chatbot

28. Ng A. CS229 Lecture notes. Lecture presented at; 2015; Standford University.

29. Simple guide to confusion matrix terminology [Internet]. Data School. 2017 [cited 22 August 2017]. Available from: http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Appendix

A) is, are, were, was, am, has, had, does, did, wasn't, wasnt, weren't, werent, shall, shant, shan't, will, won't, wont, would, wouldn't, wouldnt, should, shouldn't, shouldnt, could,couldn't, couldnt, isn't, isnt, aren't, arent, do, didn't, didnt, don't, dont, can, can't, cant, must, musn't, have, haven't, havent, hasnt, hasn't, when, when's, whens, where, where's, wheres, why, why's, whys, whys, why's, what, what's, whats, who, who's, whos, whose, which, how, how's, hows, ?

B)

List of Channel Names:	Number of JSON files inside each channel
_announcements	608
android-chat-chat	84
apps	594
beginners	579
collab	280
conferences	375
design	562
development	628
gaming	61
hangout	631
hardware	192

ideas	145
jobs	461
kotlin	295
layout	1
libraries	586
music	63
ndk	138
new-features	134
policy	273
random	1
social	101
testing	294
troubleshooting	271
website	94

Total number of Channels	Total number of JSON Files	Total number of utterances
25	7451	375473

C)

```
{
   "user": "U04PLNDN9",
   "text": "What do you guys think?",
   "ts": "1431707037.000004",
   "id": "15ha125",
   "thread": [
     {
       "position": "S",
       "number": 13
     }
   ],
   "label": [
     {
       "type": "Q",
       "position": "FULL"
     }
   ]
},
```

D)

```
###################### NAIVE BAYES TEST START #######################

------------- TEST 1 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Training on 200,000 questions and 200,000 non-questions
Creating training set
Creating testing set
Training set contains 50.0% of the total data
Testing set contains 50.0% of the total data

             Question      |      Non Question
Question        99871                 169
Non Question    7758                 92202

Accuracy: 95.934
Error Rate: 3.9635
Recall: 99.83106757297081
Precision: 92.79190552732071
Question Prevalence: 50.0195

Training on start words, question words, polar and nonpolar


Test 1 - Most Likely Features:

Most Informative Features
                       ? = 0                  NQ : Q      =    169.6 : 1.0
                    dont = 1                  NQ : Q      =    134.2 : 1.0
                  should = 1                  NQ : Q      =    116.8 : 1.0
                    cant = 1                  NQ : Q      =    107.5 : 1.0
                   could = 1                  NQ : Q      =     85.0 : 1.0
               startword = 1                   Q : NQ     =     64.2 : 1.0
                 wouldnt = 1                  NQ : Q      =     61.7 : 1.0
                    must = 1                  NQ : Q      =     61.5 : 1.0
                    isnt = 1                  NQ : Q      =     57.7 : 1.0
                    will = 1                  NQ : Q      =     44.7 : 1.0
                     why = 1                  NQ : Q      =     43.1 : 1.0
                    when = 1                  NQ : Q      =     33.8 : 1.0
                       ? = 1                   Q : NQ     =     31.2 : 1.0
                     had = 1                  NQ : Q      =     27.0 : 1.0
                      am = 1                  NQ : Q      =     22.8 : 1.0
                nonPolar = 1                   Q : NQ     =     21.7 : 1.0
                    what = 1                   Q : NQ     =     20.9 : 1.0
                   which = 1                   Q : NQ     =     19.6 : 1.0
               startword = 0                  NQ : Q      =     18.6 : 1.0
                   where = 1                   Q : NQ     =     15.4 : 1.0
                    were = 1          NQ : Q      =     11.7 : 1.0
                    have = 1          NQ : Q      =     11.0 : 1.0
                     are = 1          NQ : Q      =     10.5 : 1.0
                      is = 1           Q : NQ     =     10.1 : 1.0
                     how = 1          NQ : Q      =      9.9 : 1.0
                   would = 1          NQ : Q      =      8.7 : 1.0
                      do = 1          NQ : Q      =      7.6 : 1.0
                   whose = 1           Q : NQ     =      6.0 : 1.0
                   whats = 1           Q : NQ     =      5.1 : 1.0
                      is = 0          NQ : Q      =      4.4 : 1.0
                     who = 1           Q : NQ     =      3.9 : 1.0
                     has = 1          NQ : Q      =      3.8 : 1.0
                     can = 1          NQ : Q      =      3.3 : 1.0
                   shall = 1          NQ : Q      =      3.2 : 1.0
                     was = 1           Q : NQ     =      2.5 : 1.0
                    does = 1           Q : NQ     =      2.4 : 1.0
                    what = 0          NQ : Q      =      2.3 : 1.0
                     did = 1           Q : NQ     =      1.8 : 1.0
                    whos = 1          NQ : Q      =      1.7 : 1.0
                nonPolar = 0          NQ : Q      =      1.6 : 1.0
                   which = 0          NQ : Q      =      1.2 : 1.0
                   polar = 1          NQ : Q      =      1.1 : 1.0
                    have = 0           Q : NQ     =      1.1 : 1.0
                     are = 0           Q : NQ     =      1.1 : 1.0
                      do = 0           Q : NQ     =      1.1 : 1.0
                    will = 0           Q : NQ     =      1.1 : 1.0
                   would = 0           Q : NQ     =      1.1 : 1.0
                     can = 0           Q : NQ     =      1.1 : 1.0
                   polar = 0           Q : NQ     =      1.1 : 1.0
                    when = 0           Q : NQ     =      1.1 : 1.0
                  should = 0           Q : NQ     =      1.1 : 1.0
                   could = 0           Q : NQ     =      1.1 : 1.0
                     how = 0           Q : NQ     =      1.1 : 1.0
                     has = 0           Q : NQ     =      1.1 : 1.0
                     had = 0           Q : NQ     =      1.1 : 1.0
                     why = 0           Q : NQ     =      1.1 : 1.0
                      am = 0           Q : NQ     =      1.1 : 1.0
                    were = 0           Q : NQ     =      1.1 : 1.0
                    dont = 0           Q : NQ     =      1.1 : 1.0
                    must = 0           Q : NQ     =      1.1 : 1.0
                    cant = 0           Q : NQ     =      1.1 : 1.0
                    wont = 0           Q : NQ     =      1.1 : 1.0
                   where = 0          NQ : Q      =      1.1 : 1.0
                   didnt = 0           Q : NQ     =      1.1 : 1.0
                 wouldnt = 0          Q : NQ     =      1.1 : 1.0
                    isnt = 0           Q : NQ     =      1.1 : 1.0
                shouldnt = 0           Q : NQ     =      1.1 : 1.0
                   wasnt = 0           Q : NQ     =      1.1 : 1.0
                   arent = 0           Q : NQ     =      1.1 : 1.0
                  havent = 0           Q : NQ     =      1.1 : 1.0
                 couldnt = 0           Q : NQ     =      1.1 : 1.0
                   shall = 0           Q : NQ     =      1.1 : 1.0
                   hasnt = 0           Q : NQ     =      1.1 : 1.0
                    hows = 0           Q : NQ     =      1.1 : 1.0
                    whos = 0           Q : NQ     =      1.1 : 1.0
                  werent = 0           Q : NQ     =      1.1 : 1.0
                  wheres = 0           Q : NQ     =      1.1 : 1.0
                  what's = 0           Q : NQ     =      1.1 : 1.0
                wouldn't = 0           Q : NQ     =      1.1 : 1.0
                  when's = 0           Q : NQ     =      1.1 : 1.0
```

```
                  musn't = 0                        Q : NQ     =      1.1 : 1.0
                   don't = 0                        Q : NQ     =      1.1 : 1.0
                   how's = 0                        Q : NQ     =      1.1 : 1.0
                   who's = 0                        Q : NQ     =      1.1 : 1.0
                   can't = 0                        Q : NQ     =      1.1 : 1.0
                  shan't = 0                        Q : NQ     =      1.1 : 1.0
                  aren't = 0                        Q : NQ     =      1.1 : 1.0
                couldn't = 0                        Q : NQ     =      1.1 : 1.0
                    whys = 0                        Q : NQ     =      1.1 : 1.0
                 where's = 0                        Q : NQ     =      1.1 : 1.0
                  hasn't = 0                        Q : NQ     =      1.1 : 1.0
                  wasn't = 0                        Q : NQ     =      1.1 : 1.0
                 haven't = 0                        Q : NQ     =      1.1 : 1.0
                  didn't = 0                        Q : NQ     =      1.1 : 1.0
                   won't = 0                        Q : NQ     =      1.1 : 1.0
                   whens = 0                        Q : NQ     =      1.1 : 1.0
                   why's = 0                        Q : NQ     =      1.1 : 1.0
               shouldn't = 0                        Q : NQ     =      1.1 : 1.0
                 weren't = 0                        Q : NQ     =      1.1 : 1.0
                   isn't = 0                        Q : NQ     =      1.1 : 1.0
31 errors were caught.

Writing Test 1 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedNBQuestions
-Test1.txt



------------ TEST 1 END ------------



------------- TEST 2 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Training on 200,000 questions and 200,000 non-questions
Creating training set
Creating testing set
Training set contains 50.0% of the total data
Testing set contains 50.0% of the total data


             Question     |      Answer
Question        99792                 151
Non Question    2034                 98023

Accuracy: 98.873
Error Rate: 1.0925
Recall: 99.84891388091212
Precision: 98.00247480996995
Question Prevalence: 49.9715

Training on start words and question words
```

```
Test 2 - Most Likely Features:

Most Informative Features
                    dont = 1                  NQ : Q      =    342.2 : 1.0
                       ? = 0                  NQ : Q      =    177.7 : 1.0
                    cant = 1                  NQ : Q      =    166.8 : 1.0
                  should = 1                  NQ : Q      =    103.7 : 1.0
                   didnt = 1                  NQ : Q      =     81.6 : 1.0
               startword = 1                   Q : NQ     =     61.4 : 1.0
                   could = 1                  NQ : Q      =     58.7 : 1.0
                    wont = 1                  NQ : Q      =     48.4 : 1.0
                    will = 1                  NQ : Q      =     44.1 : 1.0
                     why = 1                  NQ : Q      =     38.8 : 1.0
                    when = 1                  NQ : Q      =     34.0 : 1.0
                       ? = 1                   Q : NQ     =     30.9 : 1.0
                  havent = 1                  NQ : Q      =     28.3 : 1.0
                    must = 1                  NQ : Q      =     26.8 : 1.0
                   arent = 1                  NQ : Q      =     25.6 : 1.0
                     had = 1                  NQ : Q      =     21.9 : 1.0
                      am = 1                  NQ : Q      =     21.1 : 1.0
                    what = 1                   Q : NQ     =     20.9 : 1.0
                   which = 1                   Q : NQ     =     20.4 : 1.0
               startword = 0                  NQ : Q      =     18.7 : 1.0
                   where = 1                   Q : NQ     =     14.9 : 1.0
                 couldnt = 1                  NQ : Q      =     14.2 : 1.0
                    were = 1                  NQ : Q      =     11.5 : 1.0
                     are = 1                  NQ : Q      =     11.5 : 1.0
                      is = 1                   Q : NQ     =     10.1 : 1.0
                    have = 1                  NQ : Q      =      9.7 : 1.0
                     how = 1                  NQ : Q      =      9.6 : 1.0
                   would = 1                  NQ : Q      =      8.8 : 1.0
                   whose = 1                   Q : NQ     =      8.0 : 1.0
                   shall = 1                  NQ : Q      =      7.9 : 1.0
                      do = 1                  NQ : Q      =      7.4 : 1.0
                   whats = 1                   Q : NQ     =      5.1 : 1.0
                      is = 0                  NQ : Q      =      4.3 : 1.0
                     who = 1                   Q : NQ     =      3.9 : 1.0
                     has = 1                  NQ : Q      =      3.6 : 1.0
                     can = 1                  NQ : Q      =      3.4 : 1.0
                    whos = 1                  NQ : Q      =      2.7 : 1.0
                     was = 1                   Q : NQ     =      2.5 : 1.0
                    does = 1                   Q : NQ     =      2.5 : 1.0
                    what = 0                  NQ : Q      =      2.3 : 1.0
                     did = 1                   Q : NQ     =      1.6 : 1.0
                   which = 0                  NQ : Q      =      1.2 : 1.0
                    have = 0                   Q : NQ     =      1.1 : 1.0
                     are = 0                   Q : NQ     =      1.1 : 1.0
                      do = 0                   Q : NQ     =      1.1 : 1.0
                    will = 0                   Q : NQ     =      1.1 : 1.0
                   would = 0                   Q : NQ     =      1.1 : 1.0
                     can = 0                   Q : NQ     =      1.1 : 1.0
                    when = 0                   Q : NQ     =      1.1 : 1.0
                  should = 0                   Q : NQ     =      1.1 : 1.0
                     how = 0                   Q : NQ     =      1.1 : 1.0
                   could = 0                   Q : NQ     =      1.1 : 1.0
                     has = 0                   Q : NQ     =      1.1 : 1.0
                     had = 0                   Q : NQ     =      1.1 : 1.0
                     why = 0                   Q : NQ     =      1.1 : 1.0
                      am = 0                   Q : NQ     =      1.1 : 1.0
                    were = 0                   Q : NQ     =      1.1 : 1.0
                    dont = 0                   Q : NQ     =      1.1 : 1.0
                    must = 0                   Q : NQ     =      1.1 : 1.0
                    cant = 0                   Q : NQ     =      1.1 : 1.0
                    wont = 0                   Q : NQ     =      1.1 : 1.0
                   didnt = 0                   Q : NQ     =      1.1 : 1.0
                 wouldnt = 0                   Q : NQ     =      1.1 : 1.0
                    isnt = 0                   Q : NQ     =      1.1 : 1.0
                   shall = 0                   Q : NQ     =      1.1 : 1.0
                  havent = 0                   Q : NQ     =      1.1 : 1.0
                   wasnt = 0                   Q : NQ     =      1.1 : 1.0
                   arent = 0                   Q : NQ     =      1.1 : 1.0
                 couldnt = 0                   Q : NQ     =      1.1 : 1.0
                shouldnt = 0                   Q : NQ     =      1.1 : 1.0
                   hasnt = 0                   Q : NQ     =      1.1 : 1.0
                  werent = 0                   Q : NQ     =      1.1 : 1.0
                    whos = 0                   Q : NQ     =      1.1 : 1.0
                    hows = 0                   Q : NQ     =      1.1 : 1.0
                   whens = 0                   Q : NQ     =      1.1 : 1.0
                  what's = 0                   Q : NQ     =      1.1 : 1.0
                  wheres = 0                   Q : NQ     =      1.1 : 1.0
                wouldn't = 0                   Q : NQ     =      1.1 : 1.0
                  when's = 0                   Q : NQ     =      1.1 : 1.0
                  musn't = 0                   Q : NQ     =      1.1 : 1.0
                   don't = 0                   Q : NQ     =      1.1 : 1.0
                   how's = 0                   Q : NQ     =      1.1 : 1.0
                   who's = 0                   Q : NQ     =      1.1 : 1.0
                   can't = 0                   Q : NQ     =      1.1 : 1.0
                  shan't = 0                   Q : NQ     =      1.1 : 1.0
                  aren't = 0                   Q : NQ     =      1.1 : 1.0
                couldn't = 0                   Q : NQ     =      1.1 : 1.0
                    whys = 0                   Q : NQ     =      1.1 : 1.0
                 where's = 0                   Q : NQ     =      1.1 : 1.0
                  hasn't = 0                   Q : NQ     =      1.1 : 1.0
                  wasn't = 0                   Q : NQ     =      1.1 : 1.0
                 haven't = 0                   Q : NQ     =      1.1 : 1.0
                  didn't = 0                   Q : NQ     =      1.1 : 1.0
                   won't = 0                   Q : NQ     =      1.1 : 1.0
                   why's = 0                   Q : NQ     =      1.1 : 1.0
               shouldn't = 0                   Q : NQ     =      1.1 : 1.0
                 weren't = 0                   Q : NQ     =      1.1 : 1.0
                   isn't = 0                   Q : NQ     =      1.1 : 1.0
                   shant = 0                   Q : NQ     =      1.1 : 1.0
                   whose = 0                   Q : NQ     =      1.1 : 1.0
31 errors were caught.

Writing Test 2 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedNBQuestions
-Test2.txt



------------ TEST 2 END ------------



------------- TEST 3 --------------
```

```
Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Training on 200,000 questions and 200,000 non-questions
Creating training set
Creating testing set
Training set contains 50.0% of the total data
Testing set contains 50.0% of the total data


             Question     |      Answer
Question        99727                 178
Non Question    4095                 96000

Accuracy: 97.9025
Error Rate: 2.1365
Recall: 99.82183073920224
Precision: 96.05574926316194
Question Prevalence: 49.9525

Training question words

Test 3 - Most Likely Features:

Most Informative Features
                       ? = 0                   NQ : Q      =    190.6 : 1.0
                  should = 1                   NQ : Q      =     88.0 : 1.0
                    dont = 1                   NQ : Q      =     74.1 : 1.0
                   could = 1                   NQ : Q      =     71.0 : 1.0
                    cant = 1                   NQ : Q      =     63.3 : 1.0
                     why = 1                   NQ : Q      =     58.9 : 1.0
                    must = 1                   NQ : Q      =     44.5 : 1.0
                   didnt = 1                   NQ : Q      =     41.8 : 1.0
                    will = 1                   NQ : Q      =     37.4 : 1.0
                    when = 1                   NQ : Q      =     34.2 : 1.0
                       ? = 1                    Q : NQ     =     30.4 : 1.0
                   arent = 1                   NQ : Q      =     23.0 : 1.0
                    what = 1                    Q : NQ     =     21.2 : 1.0
                   which = 1                    Q : NQ     =     20.7 : 1.0
                     had = 1                   NQ : Q      =     19.9 : 1.0
                      am = 1                   NQ : Q      =     19.6 : 1.0
                  couldnt = 1                  NQ : Q      =     18.3 : 1.0
                   where = 1                    Q : NQ     =     14.8 : 1.0
                     are = 1                   NQ : Q      =     10.8 : 1.0
                    were = 1                   NQ : Q      =     10.5 : 1.0
                      is = 1                    Q : NQ     =     10.1 : 1.0
                     how = 1                   NQ : Q      =      9.5 : 1.0
                    have = 1                   NQ : Q      =      9.3 : 1.0
                   would = 1                   NQ : Q      =      8.7 : 1.0
                  havent = 1                   NQ : Q      =      8.4 : 1.0
                    whos = 1                   NQ : Q      =      8.3 : 1.0
                      do = 1                   NQ : Q      =      7.5 : 1.0
                   hasnt = 1                   NQ : Q      =      7.0 : 1.0
                   whose = 1                    Q : NQ     =      6.4 : 1.0
                   shall = 1                   NQ : Q      =      6.4 : 1.0
                   whats = 1                    Q : NQ     =      5.5 : 1.0
```

```
          is = 0              NQ : Q      =      4.3 : 1.0
         who = 1               Q : NQ     =      3.9 : 1.0
         has = 1              NQ : Q      =      3.7 : 1.0
         can = 1              NQ : Q      =      3.3 : 1.0
      wheres = 1              NQ : Q      =      3.0 : 1.0
         was = 1               Q : NQ     =      2.6 : 1.0
        does = 1               Q : NQ     =      2.4 : 1.0
        what = 0              NQ : Q      =      2.3 : 1.0
         did = 1               Q : NQ     =      1.7 : 1.0
       which = 0              NQ : Q      =      1.2 : 1.0
        have = 0               Q : NQ     =      1.1 : 1.0
         are = 0               Q : NQ     =      1.1 : 1.0
          do = 0               Q : NQ     =      1.1 : 1.0
        will = 0               Q : NQ     =      1.1 : 1.0
       would = 0               Q : NQ     =      1.1 : 1.0
         can = 0               Q : NQ     =      1.1 : 1.0
        when = 0               Q : NQ     =      1.1 : 1.0
      should = 0               Q : NQ     =      1.1 : 1.0
         how = 0               Q : NQ     =      1.1 : 1.0
         has = 0               Q : NQ     =      1.1 : 1.0
       could = 0               Q : NQ     =      1.1 : 1.0
         had = 0               Q : NQ     =      1.1 : 1.0
         why = 0               Q : NQ     =      1.1 : 1.0
          am = 0               Q : NQ     =      1.1 : 1.0
        dont = 0               Q : NQ     =      1.1 : 1.0
        were = 0               Q : NQ     =      1.1 : 1.0
        must = 0               Q : NQ     =      1.1 : 1.0
        cant = 0               Q : NQ     =      1.1 : 1.0
        wont = 0               Q : NQ     =      1.1 : 1.0
       didnt = 0               Q : NQ     =      1.1 : 1.0
        isnt = 0               Q : NQ     =      1.1 : 1.0
     wouldnt = 0               Q : NQ     =      1.1 : 1.0
       wasnt = 0               Q : NQ     =      1.1 : 1.0
       shall = 0               Q : NQ     =      1.1 : 1.0
       arent = 0               Q : NQ     =      1.1 : 1.0
    shouldnt = 0               Q : NQ     =      1.1 : 1.0
      havent = 0               Q : NQ     =      1.1 : 1.0
     couldnt = 0               Q : NQ     =      1.1 : 1.0
        whos = 0               Q : NQ     =      1.1 : 1.0
       hasnt = 0               Q : NQ     =      1.1 : 1.0
      werent = 0               Q : NQ     =      1.1 : 1.0
        hows = 0               Q : NQ     =      1.1 : 1.0
      wheres = 0               Q : NQ     =      1.1 : 1.0
      what's = 0               Q : NQ     =      1.1 : 1.0
    wouldn't = 0               Q : NQ     =      1.1 : 1.0
      when's = 0               Q : NQ     =      1.1 : 1.0
      musn't = 0               Q : NQ     =      1.1 : 1.0
       don't = 0               Q : NQ     =      1.1 : 1.0
       how's = 0               Q : NQ     =      1.1 : 1.0
       who's = 0               Q : NQ     =      1.1 : 1.0
       can't = 0               Q : NQ     =      1.1 : 1.0
      shan't = 0               Q : NQ     =      1.1 : 1.0
      aren't = 0               Q : NQ     =      1.1 : 1.0
    couldn't = 0               Q : NQ     =      1.1 : 1.0
        whys = 0               Q : NQ     =      1.1 : 1.0
     where's = 0               Q : NQ     =      1.1 : 1.0
      hasn't = 0               Q : NQ     =      1.1 : 1.0
      wasn't = 0               Q : NQ     =      1.1 : 1.0
     haven't = 0               Q : NQ     =      1.1 : 1.0
       won't = 0               Q : NQ     =      1.1 : 1.0
```

```
                  didn't = 0                  Q : NQ     =      1.1 : 1.0
                   whens = 0                  Q : NQ     =      1.1 : 1.0
                   why's = 0                  Q : NQ     =      1.1 : 1.0
                shouldn't = 0                 Q : NQ     =      1.1 : 1.0
                  weren't = 0                 Q : NQ     =      1.1 : 1.0
                   isn't = 0                  Q : NQ     =      1.1 : 1.0
                   shant = 0                  Q : NQ     =      1.1 : 1.0
                   whose = 0                  Q : NQ     =      1.1 : 1.0
                   where = 0                 NQ : Q      =      1.1 : 1.0
31 errors were caught.

Writing Test 3 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedNBQuestions
-Test3.txt



------------ TEST 3 END ------------



------------- TEST 4 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Training on 200,000 questions and 200,000 non-questions
Creating training set
Creating testing set
Training set contains 50.0% of the total data
Testing set contains 50.0% of the total data

             Question     |      Answer
Question        43260                56702
Non Question    2066                97972

Accuracy: 70.616
Error Rate: 29.384
Recall: 43.276445049118664
Precision: 95.4419097206901
Question Prevalence: 49.981

Training on polar and nonpolar


Test 4 - Most Likely Features:

Most Informative Features
                nonPolar = 1                  Q : NQ     =     21.0 : 1.0
                nonPolar = 0                 NQ : Q      =      1.6 : 1.0
                   polar = 1                 NQ : Q      =      1.1 : 1.0
                   polar = 0                  Q : NQ     =      1.1 : 1.0
31 errors were caught.

Writing Test 4 classified questions to
```

```
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedNBQuestions
-Test4.txt



------------ TEST 4 END ------------


###################### NAIVE BAYES TEST END #######################

###################### SVM TEST START #######################

------------- TEST 1 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Running cross vailidation on 200,000 questions and 200,000 non-questions
Precision: 99.34082099452975
Recall: 99.33851439486908
F Score: 99.33846560431209
Accuracy: 99.3385


Training on start words, question words, polar and nonpolar
0 errors were caught.

Writing Test 1 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedSVMQuestion
s-Test1.txt



------------ TEST 1 END ------------



------------- TEST 2 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Running cross vailidation on 803000 questions and 752294 non-questions
Precision: 99.30098801683425%
Recall: 99.29844111063477%
F Score: 99.29823114760374%
Accuracy: 99.29825000000001%


Training on start words and question words
```

```
0 errors were caught.

Writing Test 2 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedSVMQuestion
s-Test2.txt



------------ TEST 2 END ------------



------------- TEST 3 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data
Running cross vailidation on 200,000 questions and 200,000 non-questions
Precision: 99.18232258687813%
Recall: 99.18026094380996%
F Score: 99.18022199368917%
Accuracy: 99.18025%


Training question words
0 errors were caught.

Writing Test 3 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedSVMQuestion
s-Test3.txt



------------ TEST 3 END ------------



------------- TEST 4 --------------


Creating question hash from dataset...
Creating non-question hash from dataset...


Labelling Data
       Finished labelling non-question data
       Finished labelling question data

Running cross vailidation on 200,000 questions and 200,000 non-questions

Precision: 79.50950609153992%
Recall: 70.78638542703779%
F Score: 68.45490515935624%
Accuracy: 70.78649999999999%
```

```
Training on polar and nonpolar
0 errors were caught.

Writing Test 4 classified questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/classifiedSVMQuestion
s-Test4.txt



------------ TEST 4 END ------------



Writing Polar Questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/Polar.txt



Writing Nonpolar Questions to
/home/quinn/research/researchProject/Data/MachineLearningGeneratedData/NonPolar.txt
```