# Meet Percy: The CS 221 Teaching Assistant Chatbot

SAHIL CHOPRA and RACHEL GIANFORTE and JOHN SHOLAR
Stanford University

Teaching assistants act as pillars of a college education, helping students grapple with new and challenging topics each week. Many students interact with their teaching assistants through online forums such as Piazza, where they may ask for assistance outside the purview of daily or weekly Office Hours. We propose and develop a T.A. chatbot to help answer Piazza questions for this class, CS 221. Specifically, we categorize questions into three types - "Policy", "Assignment", and "Conceptual". We scraped Piazza questions and answers, from the current Fall 2016 offering of CS 221, as well as previous offerings of CS 124, with permission from Professors Percy Liang and Dan Jurafsky, respectively.

To summarize, our algorithmic approach entails the use of three classifiers to determine a given question's category. If the question is classified as "Policy", we use regular expressions to match the policy question to a specific subcategory and return an appropriate pre-written answer from a representative set of solutions. If it is categorized as "Assignment", we return the closest Piazza answer according to cosine similarity of one of several feature vectors (including tf-idf and others). If the question is classified as "Conceptual" we perform an intelligent information retrieval from several academic sources (including Russel and Norvig's Artificial Intelligence textbook), and return the most appropriate paragraph, again according to cosine similarity of tf-idf vectors.

Our chatbot is able to differentiate "Policy" questions with low precision and high recall, "Assignment" questions with high precision and high recall, and "Conceptual" questions with low precision and moderate recall. We asked approximately 20 fellow students in CS 221 to evaluate the responses of our chatbot to a total of 15 randomly sampled "Policy", "Assignment", and "Conceptual" questions. Ultimately, our chatbot performs exceptionally well at answering "Policy" questions, moderately well at answering "Assignment" questions, and poorly at answering "Conceptual" questions.

Additional Key Words and Phrases: Chatbot, Education, Question Answering, Information Retrieval

## 1. INTRODUCTION

This project was inspired by news stories regarding a Georgia Tech computer science class that utilized a chatbot to respond to Piazza posts for an entire semester [1]. We were curious about the implementation but could not find any academic material on the subject, so we became interested in designing our own TA chatbot for CS 221.

Teaching assistants are crucial to guiding students through their educational experience. TAs have a rich understanding of the material that is being taught within their class, and as experts on the course, they can help provide students with both conceptual support and assignment-specific assistance.

Nowadays, most computer science classes at Stanford utilize Piazza, an online forum where students can ask questions at any time of day. With this ease of access, TAs are being inundated with more questions than ever before. With our project, we hope to alleviate some of this load by introducing our CS 221 TA chatbot, Percy. In an ideal world, our chatbot might not service a student directly but instead provide a potential answer that would automatically be posted to Piazza or that would go through a TA-approval process.

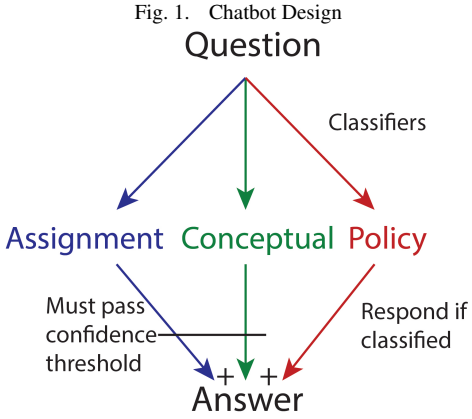
## 2. OVERVIEW OF APPROACH

A quick scan of Piazza posts makes it evident that similar questions are often asked multiple times by different people. Additionally, many Piazza questions pertain to predefined course policy. Utilizing these inefficiencies in questions, we have designed and implemented a TA chatbot to answer Piazza questions.

### 2.1 Designing the Chatbot

A student can ask literally anything on Piazza, so we aimed to broadly delineate the different types of questions. This way the chatbot could deferentially construct answers, according to the type of question being asked. Leveraging our prior experience with the platform, we devised three primary categories for the online questions - "Policy", "Assignment", and "Conceptual".

Fig. 1. Chatbot Design




Below we list a few examples of these three categories:

(1) **Policy Questions** - The chatbot should be able to answer questions regarding class policy, e.g. office hour timings, assignment due dates, etc. Example: "Where are office hours located?"

(2) **Assignment Questions** - The chatbot should be able to answer assignment specific questions. Example: "I am receiving the following output probabilities from my Bayesian network? ... What could my bug be?"

(3) **Conceptual Questions** - The chatbot should be able to answer conceptual questions pertaining to artificial intelligence. Example: "What is the difference between state-based and variable-based models?"

With this categorization of questions, our chatbot consists of three steps. First classifying the type of question, second generating an answer, and third providing an answer to the user if the prescribed solution meets a defined confidence threshold (Figure 1).

## 2.2 Data

We scraped Piazza question, answers, tags, followups, and notes from the Autumn 2016 offering of CS 221 as well as the 2013 - 2016 offerings of CS 124, with the permission of Professors Percy Liang and Dan Jurafsky, respectively. We then cleaned this data, by removing errant HTML and LaTeX symbols. Additionally, we procured a PDF copy of *Artificial Intelligence: A Modern Approach* by Stuart Russel and Peter Norvig. We converted this pdf into a text file and combed the manuscript, removing artifacts left from the conversion.

Ultimately, we trained our chatbot on both our dataset of approximately 1500 cleaned (question, answer) tuples from the 2016 offering of CS 221 and our the cleaned copy of *Artificial Intelligence: A Modern Approach*. The Piazza data was leveraged to classify questions and answer questions from the "Assignment" category. The textbook was utilized to answer questions from the "Conceptual" category.

Please note that our data set for CS 221 only extends through November 6, 2016 as we did not have sufficient time to scrape and clean additional data.

# 3. QUESTION CLASSIFICATION

## 3.1 Baseline

For our baseline, we attempted using a Linear SVM along with several simple features upon a 80-20 split of the data (Figures 2, 3, and 4). We leveraged simple features such as unigram, bigram, and trigram counts.

Additionally, we attempted training the SVM upon TF-IDF vector representations of the given questions. Given a set of input training questions, we computed a TF-IDF value for each of the tokens in the training set. TF-IDF is defined as follows:

$$tf\text{-}idf = tf * log\frac{N}{df_t}$$

$$tf = \text{Term frequency across corpus}$$

$$N = \text{Number of documents across corpus}$$

$$df_t = \text{Number of documnts containing the term, across corpus}$$

At first glance, it seems like the baselines are performing decently well across "Policy", "Assignment", and "Conceptual" classification with maximum F-Scores of 0.87, 0.73, and 0.93, respectively. Upon further analysis it becomes clear that these high F-Scores are artificial. The SVM is acting as a majority algorithm and classifying every inputted question as "Assignment". Since the data is so heavily skewed towards "Assignment" questions, this results in a decent F-Score – even though this not a classifier we would actually want for a chatbot.

Fig. 2. Question Classification Baseline - Policy Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| Unigram Count | 0.67 | 0.82 | 0.74 |
| Bigram Count | 0.53 | 0.73 | 0.61 |
| Trigram Count | 0.83 | 0.91 | 0.87 |
| Uni/Bi/Trigrams Count | 0.67 | 0.82 | 0.74 |
| **Unigrams TF-IDF** | **0.83** | **0.91** | **0.87** |
| Bigrams TF-IDF | 0.60 | 0.77 | 0.67 |
| Trigrams TF-IDF | 0.75 | 0.86 | 0.80 |

Fig. 3. Question Classification Baseline - Assignment Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| **Unigram Count** | **0.66** | **0.81** | **0.73** |
| Bigram Count | 0.51 | 0.72 | 0.60 |
| Trigram Count | 0.56 | 0.75 | 0.64 |
| Uni/Bi/Trigrams Count | 0.51 | 0.72 | 0.60 |
| Unigrams TF-IDF | 0.60 | 0.78 | 0.68 |
| Bigrams TF-IDF | 0.51 | 0.72 | 0.60 |
| Trigrams TF-IDF | 0.57 | 0.75 | 0.65 |

Fig. 4. Question Classification Baseline - Conceptual Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| **Unigram Count** | **0.91** | **0.95** | **0.93** |
| Bigram Count | 0.90 | 0.95 | 0.92 |
| Trigram Count | 0.88 | 0.94 | 0.91 |
| Uni/Bi/Trigrams Count | 0.91 | 0.95 | 0.93 |
| Unigrams TF-IDF | 0.90 | 0.95 | 0.92 |
| Bigrams TF-IDF | 0.87 | 0.93 | 0.90 |
| Trigrams TF-IDF | 0.82 | 0.91 | 0.86 |

## 3.2 Oracle

For comparison, our oracle was an SVM classifier that utilized Piazza metadata, which includes question tags such as "Other", "Hw1", etc. Most of the tags directly map to one of our three categories, so we believed that this additional information would help provide a much stronger signal as to the different question types.

Examining the results (Figure 5). the F-Scores are higher than our baseline at 0.92, 0.91, and 0.93 for "Policy", "Assignment", and "Conceptual" respectively. Digging deeper into these numbers, we found that the metadata tags was able to differentiate "Policy" from "NOT Policy", unlike our baseline classifier, though with low precision (0.20). The metadata classifier performed exceptionally well at discerning "Assignment" questions from those that were

Fig. 5. Question Classification Oracle - Piazza Metadata

| - | Precision | Recall | F-Score |
|---|---|---|---|
| Policy Questions | 0.90 | 0.95 | 0.92 |
| Assignment Questions | 0.79 | 0.80 | 0.77 |
| Conceptual Questions | 0.91 | 0.95 | 0.93 |

"NOT Assignment", though with low recall for "NOT Assignment" (0.37). Finally, the metadata tags did not help in differentiating the "Conceptual" questions from those that were "NOT Conceptual", with the SVM again acting as a majority algorithm. This is probably due to the presence of vary few "Conceptual" questions on Piazza and the fact that there is no single "Conceptual" category on Piazza. Rather, these questions are often tagged as "Other" - a category which also contains miscellaneous questions that outside the purview our "Conceptual" class.

### 3.3 Advanced Features

Distributional word vector representations have been gaining tremendous traction over the last few years - first with Google's word2Vec [2] and Stanford's GLoVe [3]. word2vec develops word vector representations to that are intended to highlight contextual similarities - grouping words that may not be associated by definition, e.g. "Barcelona" and "Spain". Meanwhile GLoVe vectors try to capture both local context and larger, global meaning. The goal of these word vector representations is to illuminate similarities to other words and phrases across a large corpora so that one can infer words' definitions and relationships.

For our question classification task, we attempted leveraging GLoVE vectors that had been pre-trained on the combination of the 2014 Wikipedia dump and the Gigaword-5 dataset. For each training and test sample, we iterated through the words of the question or answer, removing stop words and summing together the word vectors to devise a "one-hot-vector" representation of the entire text sample. Please note that, we only ran the Random Forest predictor on this feature function because the GLoVe representations are not guaranteed to be non-negative, and Multinomial Naive Bayes predictor assumes a multinomial gaussian distribution.

In addition to our experiments with GLoVe vectors, we attempted leveraging the regular expressions we wrote for determining the topic of a "Policy" question, e.g. office hours, as binary features to predict whether a question was of category "Policy".

### 3.4 Results

Improving on the baseline, we tried to correct for the tremendous skew towards questions of category "Assignment". Examining our CS 221 and CS 124 data, we found that roughly 70% of questions were "Assignment" focused, while 20% were "Policy" related, and 10% pertained to "Conceptual" ideas from the courses. To counter this intrinsic bias, we tried using a Multinomial Naive Bayes predictor (Figure 6, 7, 8) with the aforementioned priors and a Random Forest predictor (Figure 9, 10, 11).

We found that both Multinomial Naive Bayes and Random Forest predictors did approximately the same on the test set. Additionally, both improved significantly on "Assignment" and "Policy" question classification, in that they maintained relatively high F-scores without acting like majority algorithms. Though the F-Scores for "Conceptual" classification seem very high (0.96) the classifiers are just acting as majority algorithms, which once again is not helpful within the context of classifying questions for a chatbot.

Examining performance over the various features, it seems like unigrams was the best feature for delineating question types. This simplistic feature nearly universally outperformed the other feature functions across both predictors. Additionally, our attempt to leverage GLoVe vectors was somewhat unsuccessful, with the GLoVe features consistently scored lower than unigrams across all three classification tasks.

Fig. 6. Question Classification Multinomial Naive Bayes - Policy Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| Unigram Count | 0.92 | 0.86 | 0.88 |
| Bigram Count | 0.82 | 0.68 | 0.72 |
| Trigram Count | 0.81 | 0.41 | 0.36 |
| **Uni/Bi/Trigrams Count** | **0.95** | **0.91** | **0.92** |
| Unigrams TF-IDF | 0.82 | 0.45 | 0.43 |
| Bigrams TF-IDF | 0.88 | 0.18 | 0.12 |
| Trigrams TF-IDF | 0.05 | 0.23 | 0.08 |
| Trigrams TF-IDF | 0.05 | 0.23 | 0.08 |
| Regexes | .01 | .09 | .02 |
| Oracle | 0.83 | 0.68 | 0.68 |

Fig. 7. Question Classification Multinomial Naive Bayes - Assignment Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| **Unigram Count** | **0.81** | **0.80** | **0.81** |
| Bigram Count | 0.78 | 0.79 | 0.78 |
| Trigram Count | 0.69 | 0.72 | 0.70 |
| Uni/Bi/Trigrams Count | 0.85 | 0.84 | 0.84 |
| Unigrams TF-IDF | 0.82 | 0.82 | 0.79 |
| Bigrams TF-IDF | 0.72 | 0.73 | 0.68 |
| Trigrams TF-IDF | 0.67 | 0.71 | 0.80 |
| Oracle | 0.86 | 0.85 | 0.83 |

Fig. 8. Question Classification Multinomial Naive Bayes - Conceptual Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| **Unigram Count** | **0.92** | **0.92** | **0.92** |
| Bigram Count | 0.90 | 0.89 | 0.89 |
| Trigram Count | 0.85 | 0.87 | 0.86 |
| Uni/Bi/Trigrams Count | 0.92 | 0.88 | 0.90 |
| Unigrams TF-IDF | 0.87 | 0.93 | 0.90 |
| Bigrams TF-IDF | 0.87 | 0.93 | 0.90 |
| Trigrams TF-IDF | 0.86 | 0.93 | 0.89 |
| Oracle | 0.90 | 0.95 | 0.92 |

## 4. QUESTION ANSWERING

Once the chatbot has determined that the question belongs to one or more categories, it then generates a response specific to the type of question. Here we describe the three different methodologies for answering "Policy", "Assignment", and "Conceptual" questions.

Fig. 9. Question Classification Random Forest - Policy Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| Unigram Count | 0.86 | 0.82 | 0.80 |
| Bigram Count | 0.88 | 0.86 | 0.84 |
| **Trigram Count** | **0.97** | **0.95** | **0.96** |
| Uni/Bi/Trigrams Count | 0.75 | 0.86 | 0.80 |
| Unigrams TF-IDF | 0.91 | 0.91 | 0.91 |
| Bigrams TF-IDF | 0.68 | 0.73 | 0.67 |
| Trigrams TF-IDF | 0.75 | 0.86 | 0.80 |
| Trigrams TF-IDF | 0.05 | 0.23 | 0.08 |
| Regexes | 0.60 | .77 | .67 |
| GLoVe 50D | 0.67 | 0.82 | 0.74 |
| GLoVe 100D | 0.67 | 0.82 | 0.74 |
| GLoVe 300D | 0.66 | 0.77 | 0.71 |
| Oracle | 0.96 | 0.95 | 0.95 |

Fig. 10. Question Classification Random Forest - Assignment Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| **Unigram Count** | **0.79** | **0.80** | **0.77** |
| Bigram Count | 0.77 | 0.78 | 0.77 |
| Trigram Count | 0.69 | 0.72 | 0.70 |
| Uni/Bi/Trigrams Count | 0.78 | 0.79 | 0.76 |
| Unigrams TF-IDF | 0.78 | 0.78 | 0.75 |
| Bigrams TF-IDF | 0.70 | 0.73 | 0.70 |
| Trigrams TF-IDF | 0.63 | 0.68 | 0.64 |
| GLoVe 50D | 0.62 | 0.72 | 0.64 |
| GLoVe 100D | 0.67 | 0.71 | 0.64 |
| GLoVe 300D | 0.69 | 0.71 | 0.64 |
| Oracle | 0.84 | 0.83 | 0.80 |

Fig. 11. Question Classification Random Forest - Conceptual Question

| - | Precision | Recall | F-Score |
|---|---|---|---|
| **Unigram Count** | **0.95** | **0.96** | **0.96** |
| Bigram Count | 0.90 | 0.93 | 0.921 |
| Trigram Count | 0.89 | 0.91 | 0.90 |
| Uni/Bi/Trigrams Count | 0.92 | 0.94 | 0.92 |
| Unigrams TF-IDF | 0.89 | 0.92 | 0.88 |
| Bigrams TF-IDF | 0.90 | 0.92 | 0.89 |
| Trigrams TF-IDF | 0.89 | 0.92 | 0.90 |
| GLoVe 50D | 0.91 | 0.93 | 0.91 |
| GLoVe 100D | 0.96 | 0.99 | 0.97 |
| GLoVe 300D | 0.92 | 0.94 | 0.93 |

## 4.1 Policy Questions

To respond to a policy question, we extend the approach described in "Policy Question Classification". During the classification stage, a question is evaluated against a series of regular expressions and placed into one or more policy question subcategories. For any subcategory that a question is classified into, a pre-written answer is selected from a suite of responses. To facilitate fluid user interaction and eliminate a feeling that Percy simply gives "canned answers", the chatbot has multiple, similar responses for each of the categories of class specific questions. If a question is asked multiple times, the chatbot is able to return variations of the answer so that it does not seem too repetitive. For examples, if a question is placed under the "PRACTICE_EXAMS" subcategory, we might select the following response.

**"Past year exams and solutions are listed on the CS 221 website."**

Though this method is very straightforward, it works decently well for the fifteen topics the bot must address in regards to class specific questions. If a question matches multiple subcategories, we provide a relevant answer from each matching subcategory.

To allow the system to be more helpful over the course of a quarter, we implemented a method to allow real TA's to input new information into the knowledge base when things change. For example, a TA may notify students of a time change in office hours. If a student then asks a question related to the new information, both the appropriate pre-seeded answer and the new piece of information will be returned. This new information is also categorized and returned based on regular expression matching.

## 4.2 Assignment Questions

"Assignment" questions proved a significant challenge as these are often very detailed and specific. There is no way for a chatbot to generate an answer to questions about why a snippet of code is broken, so questions of this nature must be left to human teaching assistants. However, some questions pertain to the same common bug and are asked multiple times. Our bot's goal is to be able to respond to these repeated questions, leaving human TA's free to answer new questions.

We began by using cosine distance among tf-idf vectors computed across the training set and test question to return an answer from our existing knowledge base. The algorithm reads in all the questions from Piazza and calculates a tf-idf matrix across the document set. When a new question was posed, the bot creates a vector consisting of each word's tf-idf score that was computed during training. It then iterates through all the questions and finds the question that minimizes the cosine distance and returns the answer of that most similar question. We tested out algorithm both maximizing the dot product of the tf-idf vectors and minimizing the cosine distance between the vectors, and found that both performed similarly, often returning the same answer.

Ultimately, the chatbot could not reliably find the best answer as the top answer; however, it was often among the top few closest answers. To address this problem the chatbot in practice should return the top three questions produced by either algorithm. This greatly improves the overall helpfulness of the chatbot's answers, since the probability that the best answer is in the top three is much higher than the probability that it is the top answer.

We curb our chatbot's impulse to answer questions, even when it does not have a similar question in it's knowledge base and has no possible clue as to the true answer. To accomplish this we implemented a maximum threshold for the cosine distance such that if the distance between a new question and it's closest Piazza question is above the threshold, the chatbot returns a response akin to "I don't know". This method fits with our goal of answering repetitive questions rather than trying to generate answers.

## 4.3 Conceptual Questions

Conceptual questions are much more open-ended than class specific questions, and so we decided to use a much broader approach.

To respond to conceptual questions, we implemented information retrieval on unstructured documents as discussed in the paper by Yan, Zhao, et al[4]. Their method includes retrieving multiple answers and then ranking those answers according to tf-idf, surrounding context, chapter, topic, etc. Thus, we approached this an information retrieval task over a textbook - specifically Russell and Norvig's *Artificial Intelligence: A Modern Approach*.

Originally, our chatbot compared every line of the textbook with the question, but this ran slowly, taking up to 90 seconds, and sometimes returned unrelated material. Now we break the book up into sections, find the most relevant section, and then find the most relevant line within that section. In our chatbot's booting phase it reads the textbook in, splitting it by its section divisions. It then calculates a tf-idf matrix where each line represents a section of the book, and the idf is based on the entire textbook. It also calculates a separate tf-idf matrix for each section where each line of the matrix represent a line in that section and the idf is based on that section. Whenever a conceptual question is asked, the chatbot calculates the tf-idf of that question based on the idf of the textbook. It then calculates the cosine distance between the question and every section in the textbook. Once it has found the most similar section, it recalculates the question's tf-idf vector based on the idf of the most relevant section, and then finds the closest line in the section using cosine similarity. The chat bot returns that line and its surrounding paragraph, since conceptual concepts usually take more than a single sentence to explain.

Similarly to the assignment questions, we found that our answers were improved by returning the three closest paragraphs, rather than just the best.

### 4.4 Results

Evaluation of our results for the question classification stage was fairly straightforward, in that metrics for classification problems (e.g. precision, recall, f1-score) are well-established, and are recognized and understood by the broader research community. In contrast, evaluation of the question-answering is an inherently hard problem with no established evaluation metrics. Given a question, our methodology yields an answer. Our evaluation metric must then answer the question: "to what extent does this response constitute a 'good' answer to this question?". Setting aside the myriad interpretations of what might constitute a "good" answer (e.g. balance between succinctness and detail, etc...), this evaluation metric requires us to develop a meaningful and nuanced interpretation of both the question and answer and determine in what ways they might be related - an open problem in the artificial intelligence community. Concordantly, we were forced to pursue alternate strategies in the evaluation of our responses to questions.

Initially, we experimented with several automated evaluation metrics for string similarity, including most prominently BLEU score, and cosine similarity between TF-IDF vectors for question-answer pairs. However, as was expected, these methods only allow for the matching of syntactically similar question-answer pairings, not necessarily those with similar intentions.

After these and other initial attempts at evaluation for our question-answering stage, we turned to human evaluation - a technique which, while not scalable, provided an accurate, intelligent evaluation metric. For human evaluation, we curated a set of fifteen question-answer pairings (five conceptual, five policy-based, and five assignment-based) which we felt were representative of the overall performance of the chatbot. Twenty-two students were asked to rate the quality of each answer as a response to the given question, on a scale from one to five. We present in figure 12 the mean ratings for each question, as well as the standard deviation of the ratings for each question, to illustrate the presence or absence of contention over the quality of an answer (for brevity, we have excluded the actual question-answer pairings from the table; these question-answer pairing can be found in the appendix).

Fig. 12. Question Answering - Human Evaluation

| Question | Category | Mean Rating | Standard Dev. |
|---|---|---|---|
| 1 | Policy | 4.18 | 0.96 |
| 2 | Policy | 4.73 | 0.55 |
| 3 | Policy | 3.95 | 1.25 |
| 4 | Policy | 3.55 | 1.44 |
| 5 | Policy | 2.77 | 1.72 |
| 6 | Assignment | 4.05 | 1.00 |
| 7 | Assignment | 4.18 | 0.85 |
| 8 | Assignment | 2.50 | 1.22 |
| 9 | Assignment | 1.95 | 1.36 |
| 10 | Assignment | 3.73 | 1.24 |
| 11 | Conceptual | 2.00 | 1.27 |
| 12 | Conceptual | 1.32 | 0.89 |
| 13 | Conceptual | 2.59 | 1.10 |
| 14 | Conceptual | 2.14 | 1.25 |
| 15 | Conceptual | 2.27 | 1.28 |

## 5. ANALYSIS

### 5.1 Analysis: Question Classification

Our primary conclusion the results of the question classification sub-problem is that this is inherently a data problem. With our limited dataset of slightly more than one thousand past question-answer pairings, coupled with standard classification algorithms (e.g. SVM, Naive Bayes, Random Forest), we observed moderately successful classification rates, with significant room for improvement.

Interestingly, unigram features proved to be the best dilineator of question types, but this may be a result of the small data-set and high word-word variance across Piazza questions. Additionally, it was initially surprising the GLoVe features failed to perform at least as well as unigrams. However, upon further consideration it makes sense that our chosen GloVe representations may not be suitable for this classification task. The Wikipedia/Gigaword-5 dataecset that the GloVe vectors were trained upon may not be as applicable to our Piazza question classification dataset, since the questions deal with very assignment-specific terms. GLoVe features may have proved more valuable in differentiating "Conceptual" questions from the rest if they had been trained on a corpus of artificial intelligence texts, as these questions feature a wide array of terminology that is both diverse and exclusive to these types of questions.

Furthermore, it's likely that the results of our question classification phase could be improved significantly via the introduction of more data from previous iterations of the course (this data was

not included originally due to concerns over protecting student privacy under FERPA). Another potential improvement in this approach would be to incorporate data from other classes for question classification purposes only (and not for question answering purposes). It's possible that question categories in other classes are similar enough to question categories for CS 221, and could simply be folded into our existing dataset. We did obtain data for CS 124 (Introduction to Natural Language Processing) courtesy of Professor Dan Jurafsky, but ultimately were not able to incorporate it - an omission which paves the path for several obvious next steps.

## 5.2 Analysis: Question Answering

5.2.1 *Analysis: Policy Questions.* Policy-based questions were answered simply and effectively using a series of regular expression pattern matches to classify a question as belonging to one of several dozen different categories. Some might be quick to criticize this as an "unintelligent" approach, and indeed by any definition it certainly does not incorporate artificial intelligence. However, we, the authors cite this particular design choice as an example of the prudent application of artificial intelligence techniques. The number of different topics which fall under the umbrella of "policy questions" is small enough to be completely enumerated by a small team of people familiar with the course, and is unlikely to grow proportionally with the size of this course. Additionally, these questions have strict, predefined answers, which may change with time but do not change rapidly. Thus, it seems most appropriate to use a small, predefined set of answers to field these questions, and in practice this approach proved extremely effective. We cite as evidence of this point, our mean ratings for questions one through five in Figure 12. Question-Answer pairs 1 and 2 were consistently rated highly, while Question-Answer pairs 3, 4, and 5 were rated highly in general, with high variance in responses. With further refinement of techniques employed to answer policy questions, we postulate that these could be automatically answered in a completely satisfactory manner.

5.2.2 *Analysis: Assignment Questions.* Providing answers to questions focused on specific assignments, problem sets, and other course materials comprised the bulk of our research. Analysis of the ratings provided for questions six through ten in Figure 12 reveals that some assignment-based question-answer pairings were received quite well (notably questions six, seven, and ten), while others exhibited room for improvement, or high variance in responses, indicated disagreement among survey participants.

Our approach for question-answer pairing was inspired by that of Dror, Gideon, et al. [5], who pursued a similar approach when they attempted to use past question-answer pairings from Yahoo Answers to respond to new questions, with some success. However, Dror and Gideon had access to a dataset of approximately one million past question-answer pairings, while our team was working with a dataset of only approximately one thousand pairings. This proved to be one of our primary limitations in responding to assignment-based questions, but, luckily, will also be one of the easiest obstacles to conquer in future iterations of this research. This research was inherently limited in scope by the fact that it was conducted over the course of two months for class project, and given a larger project scope, it is likely that considerably more data could be acquired. With more data, the likelihood that any new question can be answered using a past question-answer pairing increases proportionally, and bolsters the strength of this approach.

One caveat to the strength of the current question-answering method for assignment-based questions is the fact that assignments for many classes will exhibit subtle changes year over year, shifting with instructor preferences and the unique and unexpected demands of any given academic quarter. This is particularly true of relatively cutting-edge classes such as CS 221, where advances in academia may be relevant to material taught in the class. The addition of a discretionary layer to our current question-answering model would allow human teaching assistants to evaluate any answer proposed by the chatbot for validity before releasing it to students. This would certainly prevent the chatbot from releasing any erroneous information, but the overhead of human error-checking could be almost as high as the overhead of human question-answering in the first place. In the ideal case, the chatbot will be able to release information without human approval, though such a release will require a high threshold of confidence, which we are unable to achieve for most questions as of this writing.

5.2.3 *Analysis: Conceptual Questions.* Conceptual questions proved much more difficult to answer, as is evidenced by our results for this category. We note that questions eleven through fifteen in Figure 12 received relatively low ratings, with low variance in responses. However, the goal of this research was to use artificial intelligence techniques to answer repetitive or unnecessary questions, to ensure that teaching assistants could spend more time helping students with difficult, conceptual topics. Currently, the chatbot's method for responding to conceptual questions consists of an intelligent information retrieval approach, designed to guide students to academic resources that may help them arrive at an answer on their own. However, often students struggling with conceptual topics seek the active dialogue and discussion of a topic that only a human teaching assistant can provide. Thus, the fact that we cannot yet answer difficult, conceptual questions is not so much a defeat as it is a recognition of the fact that some topics (for now) can and should be left to human teaching assistants.

# 6. CONCLUSIONS

Though certainly far from perfect, our chatbot produces viable results in some contexts, and provides a solid foundation for future research in this direction. Our original purpose was to handle repetitive questions on Piazza. Given that mission, the chatbot performs reasonably well. If a similar question has already been asked, the chatbot is generally able to retrieve it. If no similar question has been asked the bot is able to recognize that and respond, "I don't know." It also performs well on policy questions, which is the other type of question that consumes the time of teaching assistants, but does not necessarily require their skills. The limited number of categories makes it easy to determine exactly what a policy question is asking and return the relevant information.

The main issue we faced was responding to conceptual questions. When we began this project we recognized that this was an inherently hard problem, and it is one we believe still requires a human teaching assistant. The process of understanding a concept, understanding a student's question, and then synthesizing the appropriate information to directly respond to the student's question is one that even humans struggle with. We did not expect to be successful on this part of the project, and despite our attempts

our expectations of the difficulty of the problem were validated.

Ultimately, our chatbot strove to act question-answering vehicle, lacking a purely conversational mechanism. Given more time it would be fascinating to develop a knowledge base of facts that the chatbot could draw on to generate answers. But much like conceptual questions, this is a much harder problem than we could take on given the limited time frame.

## 7. FUTURE WORK

Given that our research was conducted over the course of only two months, several areas stand out as candidates for the continuation and extension of this study.

First, gathering more data from past courses would allow us to improve our classification of questions simply by having more data to train on. More data would also improve our chatbot's ability to retrieve answers to similar questions by increasing the number of question and answer pairs we can match on. Having past data would also allow us to begin handling issues of weighting past quarters' answers against current answers, and ensuring that the chatbot does not return information that is out of date.

Additionally, though we began using a GLoVe vectors as features for question classification, they performed worse than simple unigram features. As a next step we are considering training the GLoVe vectors on materials that are reflective of academic literature surrounding artificial intelligence, rather than just generic materials. This may better capture relationships between words in our specific context.

A possible improvement for assignment questions would be to utilize GLoVe vector cosine similarity for finding questions that have already been asked rather than tf-idf cosine similarity. This may better account for differences in wording and more effectively compare the the similarity of the key points of the questions.

To improve policy question answering, we would expand the chatbot's set of regular expressions. Though we believe that regular expressions are the best method for answering policy questions, the current set of regular expressions was designed by hand, but these may not entirely generalize. Thus, we need an unsupervised way of finding potentially relevant regular expressions. One candidate is the DIPRE method, first developed by Sergey Brin[3].

## REFERENCES

[1] Maderer, Jason. Artificial Intelligence Course Creates AI Teaching Assistant. 9 May 2016.

[2] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 3111-3119.

[3] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). doi:10.3115/v1/d14-1162.

[4] Yan, Zhao, et al. "DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents."

[5] Dror, Gideon, et al. "Learning from the Past: Answering New Questions with Past Answers". Proceedings of WWW. 2012.

[6] Brin, Sergey. "Extracting patterns and relations from the world wide web." International Workshop on The World Wide Web and Databases. Springer Berlin Heidelberg, 1998.

## 8. APPENDIX

### 8.1 User Survey Questions

The questions provided in the user survey are as follows. Users were asked to rate the quality of each answer from one to five as a response to its respective question.

(1) How do sign up for an alternate exam?
*Answer: Fill out the form listed on the website, if you want to need an accommodation to take an alternative exam, by Nov. 1*

(2) Is the code from lecture posted?
*Answer: Slides are posted on the website and lecture videos can be found at https://mvideox.stanford.edu/Course/787. Code can be found at http://web.stanford.edu/class/cs221/live/*

(3) Are there practice solutions?
*Answer: Practice exams can be found on the CS 221 website.*

(4) Do I need an exam monitor?
*Answer: SCPD students can the take exam either on campus or at a designated test center. If you choose to take the exam remotely, please schedule the exam for a 3-hour period between Monday, Nov. 28 at 9 a.m. and Tuesday, Nov. 29 at 9 p.m. (Pacific Time).*

(5) Are we allowed to import from the python library for our assignments?
*Answer: Try running your code on corn. If it works and passes grader.py you should be good*

(6) For 4a pacman. I'm having issues with pacman refusing to eat the last piece of food. He will wander around right next to it but never eat it.
*Answer 1: Maybe in your settings, the value when paceman is around the food is larger than the value after the pacman eat it. Maybe you put too much weight on the food, which is even larger than the end game bonus.It sounds like your evaluation function returns a larger value when it is sitting right next to the food capsule, than what it would get by the score increase by finishing the game.*

(7) For problem 0b It seems like parts 2 and 3 will have the same number of calls, because AC-3 must still be called for every assignment of x the way it is currently written, but the question also asks us to explain why AC-3 reduces the number of calls. What am I missing?
*Answer: AC-3 will (very likely) shrink the domain for the remaining unassigned variables. Thus when we consider the next variable, the number of times we call backtrack() is determined by how many values we can assign to it, which should be lower with AC-3.*

(8) For blackjack, I'm confused on the information given to use. How do we know the values of the cards that correlate the the counts in the deck we are passed?
*Answer: It's specific to this problem. Basically, you first learn some policy, then compare it to another policy. However, if you*

*call getAction() to compare the policy, and your exploration prob is not 0, with probability epsilon you will get a random policy.*

(9) I'm confused about how we are supposed to set up the feature keys. It says that each feature key is (state, action), but does this mean a single feature contains all the (state, action) pairs mapped to something or is each (state, action) pair a separate feature?
*Answer: No, the next state should be (6, None, None) with 0 reward.*

(10) What happens if the reward is not immediately apparent to Monte Carlo? For example, in chess or go, nothing really matter until the final move, how does reward account for that?
*Answer: I think the feedback backpropagates for a game like chess where the the only (and big) reward comes from moving a state to the terminal state (checkmate). The Qopt and Vopt recurrence would feed into each other such that you can infer rewards all the way back up to the start of the game.*

(11) What does Q-learning do in a terminal state?
*Answer: As in (a), but with the terminal state at (5,5) The actions are deterministic moves in the four directions In each case, compare the results using three-dimensional plots For each environment, propose additional features (besides x and y) that would improve the approximation and show the results 21*

(12) Why would people use policy iteration over value iteration?
*Answer: A 10 10 world with a single +1 terminal state at (10,10)*
*As in (a), but add a 1 terminal state at (10,1)*
*As in (b), but add obstacles in 10 randomly selected squares*
*As in (b), but place a wall stretching from (5,2) to (5,9)*
*As in (a), but with the terminal state at (5,5)*

(13) Beam Search w/ different K selection Are there any variations on pruning other than picking top k elements with highest weights? For example, using some kind of random distribution, or picking some elements in separate intervals. Would that generate better results?
*Answer: Just one It begins with k randomly generated states At each step, all the successors of all k states are generated If any one is a goal, the algorithm halts Otherwise, it selects the k best successors from the complete list and repeats*

(14) Motivation of SARSA I do not understand the motivation behind SARSA and other bootstrapping methods in the context of model-free learning. Why is it important to obtain feedback quickly if it is not used to modify the policy online?
*Answer: For this reason, Q-learning is called a model-free method As with utilities, we can write a constraint equation that must hold at equilibrium when the Q-values are correct. As in the ADP learning agent, we can use this equation directly as an update equation for an iteration process that calculates exact Q-values, given an estimated model This does, however, require that a model also be learned, because the equation uses P (s# — s, a)*

(15) Why do we care if hinge loss puts upper bound on 0-1 loss? Why is it an important feature of the hinge loss that it places an upper bound on the 0-1 loss? Couldn't we multiply the expression for the hinge loss by .1 arbitrarily (so it does not upper bound the 0-1 loss) and get the same results by using it?
*Answer: Is it possible to find an upper bound on the value of C before we have looked at all its children? (Recall that this is what alpha,beta needs in order to prune a node and its subtree). At first sight, it might seem impossible because the value of C is the average of its children's values, and in order to compute the average of a set of numbers, we must look at all the numbers But if we put bounds on the possible values of the utility function, then we can arrive at bounds for the average without looking at every number For example, say that all utility values are between 2 and +2; then the value of leaf nodes is bounded, and in turn we can place an upper bound on the value of a chance node without looking at all its children An alternative is to do Monte Carlo simulation to evaluate a position*

Received September 2008; accepted March 2009