Jarvis - A Smart Bot for Piazza

Pranav Ramarao, Isaac Bowen, Ke Yu
University of Michigan
Ann Arbor
{pranavr,irbowen,yuke}@umich.edu

ABSTRACT

In this work, we explore some of the challenges faced in Piazza specially for large sized classes. We present a smart bot - Jarvis that can tackle the problems by using state-of-the-art IR techniques. We show the Jarvis can greatly increase the efficiency of the instructors and also serve as a useful aid for students.

KEYWORDS

IR, piazza, search, similarity,FAQ generation

ACM Reference format:
Pranav Ramarao, Isaac Bowen, Ke Yu. 2017. Jarvis - A Smart Bot for Piazza. In *Proceedings of EECS 549/SI 650, Ann Arbor MI, USA, April 2017 (UMich'17),* 3 pages.
DOI: 10.475/123_4

1 PROBLEM

Piazza is a great tool for students and instructors to interact with each other. It allows students to post questions on a class wide forum, where other students and instructors can answer the questions. For small classes, where the question volume is very low, the system works well. Instructors can respond quickly, and scale isn't an issue. However, as the departments grow and class sizes increase, it becomes difficult to answer all the questions in a timely manner. For our project, we wanted to focus on a few specific issues related to Piazza at scale. The authors are instructors for undergraduate courses at UofM - these course have roughly 400 and 600 students and only a handful of staff members to manage the class. We are interested in solving these problems to assists in our courses.

1.1 Duplicate Questions

As one of the side effects of the number of students posting questions, it becomes difficult for other students in the class to know if their question has already been asked. Given the question volume, they may not have time to read every question that is asked nor use the search facilities that piazza provides. As a result, students repost a question that is very similar to one that has already been asked. For instructors, this means that they have to answer the same question over and over, or link students to the previous question. This is not an easy task as a similar question might have been answered by another instructor which requires instructors to search in the first place. For students, it could provide confusion, as many similar questions have very similar answers, but students are quite sure if its the same question (and therefore has the same answer). There is also the consistency issue where two instructors might not concur exactly on the duplicate questions which confuses students even more.

1.2 Top Questions

A related issue is that students aren't getting the most they could out of Piazza because of the number of questions. Students can't dedicate hours every day to read through all questions posted every day. Also, many of the question are very student-specific (they won't related to other students, so the whole class doesn't benifit from reading them). However, there are some questions that are very common, or exceptionally helpful, and it would be beneficial to aggregate these high quality questions. Ideally, students want to see a weekly report of 10 questions that they do not want to miss out on.

1.3 Better Search Features

Piazza's search features leave something to be desired. The largest problem seems to be that the results are very heavily skewed towards the most recent results, when that may not actually be what the student is looking for. We wanted to provide a method for students to more accurately search Piazza for their questions (or answers).

1.4 Information Reterival

How do we formulate the above problems as information reterival problems?

Finding duplicate questions is a natural IR problem. We have to find the most similiar posts in the collection, and then see if similarity score is above the threshold we have chosen. If so, we can flag this question as a duplicate.

Finding the top questions for the week for the FAQ is a question where both traditional IR techniques come into play, and we can use specific information made available from the Piazza platform.

Improve Piazza's search feature is directly an IR task. Piazza's search isn't very good at IR, and we want to make it better!

UMich'17, Ann Arbor MI, USA
© 2017 Copyright held by the owner/author(s). 123-4567-24-567/08/06...$Free
DOI: 10.475/123_4

2 BACKGROUNDS

Piazza is a free online gathering place where students can ask and answer questions 24/7 under the guidance of their instructors. Collaboration occurs in real time, which encourages participation and fuels discussion. Leading campuses across North America have a significant Piazza presence. Larger courses typically have over 500 students getting involved in the feed. Some of the painpoints however are :

(1) students tend to ask duplicate questions without searching for them first.
(2) students are overwhelmed by the huge number of questions on piazza specially in larger courses.
(3) the search facilities that piazza provides are average and rely on recency based features and fail to consider relevance based ones.

TODO: Cleanup above duplication

2.1 Piazza

Piazza is an online discussion forum where student can get their questions answered by peers and instructors. Students can post questions, answers, notes, show gratefulness, publicly or anonymously. Instructors can post notes, answer student questions, endorse good questions and answers, and collect student feedback about this course.Piazza provide students participation report which help instructors to see students engagement in the class. Instructors can view the class report to see when in the term the most questions are being asked but not the other way around

2.2 Chat Bots

A chatbot is known as a computer program which conduct conversational question answering with users in a certain domain or on a certain topic with natural language sentence.Such procedure is aiming at simulating a persons behavior as a conversation partner and designed to pass the Turing test. Chatbots are commonly used in various dialog system for information acquisition and other customer services. Some natural language processing techniques are used in building chatbots

3 METHOD

We designed and implemented Jarvis, a bot written in python that can accomplish these goals.

3.1 System Overview

Jarvis works by indexing all of the post on a given course on Piazza. On startup, it will bulk download all of the questions posted previously and constructed an index from there. After that, it sits in a loop, waiting for new questions to be posted.

3.2 Tools and packages used

We took advantage of several python packages and libaries to make the devolpment of Jarvis faster and the results more correct. We used python whoosh for indexing the documents. This allowed us to index the documents, but first we actually had to collect them in one place. To do this, we needed to download all olf the questions off of Piazza. Piazza does not have an offical supported API, but there is an unoffical API that has been constructed through some reverse engineering. While this did not work as perfectly as a fully supported first party API would have, the results were surprisingly good and required only a small amount of work on our part to work with.

3.3 Data Extraction

Piazza does not provide an elegant or easy way to problematically interact with its posts. To get the questions and answers on the site, we had to use an unofficial API. This API has been reversed engineered by doing some things todo fix me. WYue started from this API, and then built our own, higher level srapper that would read in posts and then store them as objects in memory that were easier to work with. To speed up processing, testing, and to prevent Piazza from potentially rate limiting us, we also developed a feature to bulk download and store course data as a .csv file, which we could then treat as our "Piazza" input data for indexing and pre-processing purposes. We extracted several past semesters of eecs 281 data, and were able to use that as our training set.

3.4 Indexing and Feature Extraction

3.5 Data Preprocessing

3.6 Duplicate Detection

We implemented a function called find_top_N_similar_docs(), which achieve following functionality: input a new post, and a collection of existing posts, and a integer n, we can return the top n posts with highest similarity scores against new input post.

In this function, we firstly did several preprocessing approach by removing all HTML paragraph tags from the post body, concatenating the title with the body together, and treating both of them as one document for convenience. Then, in order to improve reliability and efficiency in similarity measure, we did following preprocessing steps. For each document, we do the basic text cleaning: We first use regular expressions to get rid of symbols that are useless to the calculation. And then, transform each word to its lower case, after which we remove the stopword from the documents and applied Porter stemmer [Porter, Martin F. "Snowball: A language for stemming algorithms." (2001).] from nltk[Bird S. NLTK: the natural language toolkit[C]//Proceedings of the COLING/ACL on Interactive presentation sessions. Association for Computational Linguistics, 2006: 69-72.] package to further shrink the vocabulary size. After the documents are cleaned and stemmed, we split the text by space in order to built the vector space model.

By representing the text document using vector space model, we can calculate relevance between the documents. Before that, for the term selection, we calculated the TF* IDF(term frequencyinverse document frequency) score for each words in documents by using sklearn built-in function. And then calculate the similarity between new posted documents with all documents in the whole collection. The posts on piazza usually are more than 5 words long, and we consider the document not short yet not too long. In this case, Cosine similarity measure or Dice similarity measure would be more efficient since these measures tend not to over penalize non-overlapping terms. In our implementation, we used Cosine similarity measure for reliability. By comparing pair-wise similarity scores of new input post and all posts in the collections with input parameters N, we can return top N most similar documents with highest scores in the collections.

3.7 Top Questions

3.8 Improved Search Feature

4 RESULTS

Here we present the results of our system. To test our system, we set up a mock Piazza course where we could post fake questions to Jarvis. Jarvis has read only access to previous semesters of questions, and read/write access on the mock course. When new questions are posted, Jarvis can read those questions, and write answers to them, but it cannot write to the past semesters course.

4.1 Duplicate Detection

To test our system, we found all questions in previous semesters that had been marked with an '@'. This means that an instructor read the question, and decided that it was similar enough to a previous question, that, instead of answering it, they decided to simply link to a previous question. We used this set of questions to test our system. We found that our system performed very well on this data set, responding to more than 90 percent of known duplicate posts correctly.

Perhaps more exciting is the fact that our system was able to idenitfy some duplicate questions that instructors did not find themselves. When we fed our system questions that specifically had not been marked with an '@', we found that some of them were still marked by our bot. Many of these were, in fact, duplicates that had been answered twice, by two seperate instructors at two seperate occasions. However, since this data is not labeled, there is no easy way for us to determine our systems efficacy without looking at every duplicate it flags and checking to see if it seems logical. For the vast majority of the new duplicates flagged by our system, it seems to work well.

4.2 Top Questions

Jarvis's default behavior is to simply sit in a while loop, waiting for new questions to be posted. However, it can be instructed to create a special post on Piazza, containing the top questions for a certian week.

Jarvis will then look through all of the questions that have been posted in the given timeframe, and extract a variety of features from the questions. Based on these combinations of features, Jarvis will assemble as list of the top 10 (TODO FIX ME FIVE TEN WHAT?) questions posted.

Unfortunately, 'best' or 'top' questions is an exteremly subjective measure. We do not have a golden list of the best questions to compare our results to. So, we tried to do a few things. We looked through several weeks worth of questions, and tried to find a few, exteremly important posts that we knew *must* show up in the final output. These included instructor clarification posts, notes about when and where the exams were, and other important course logistic posts. For the questions we found, Jarvis always had these in the top ten. The other way we were able to evaluate is by manually looking at the questions that Jarvis has found, and attempt to determine if the questions reflect the best questions posted that week. Again, this was exteremly subjective, but for every question listed in the top questions, we were able to find something about it that made it worthy of being in the list.

5 DISCUSSION

5.1 Conclusion

5.2 Deliverables

We have open-sourced on implmentation of Jarvis, as well as the code to download and index Piazza. All of the Jarvis bot code is available on Github, here

5.3 Lessons Learned

A large portion of our work had to do with dealing with messy input data scraped from Piazza. Because the API is unoffical, there were many additional steps that we had to take before we had a clean dataset of questions and answers for each course. I'm sure we learned some other things. TODO TODO TODO TODO

5.4 Future Work

In the future, we can improve our algorithm by improving on term selection and similarity calculation. For term selection, we can improve by using n-gram, for example bigram or trigram, which may better capture the relationship in each document.Also, we could try implement grammar structure analysis of a sentence using a set of input transformation rules to represent grammar rules. For similarity calculation, we can try applying different similarity functions and use cross validation to find a better model. Besides, we could also try to applying applying new techniques, for example word2vec, to reduce the calculation.