# Ranking Course Reviews: A Classification Approach

Leif Jurvetson, Bhagirath Mehta and Griffin Tarpenning
*leijurv (CS221), bmehta18 (CS221), gritarp (CS221)*

**USER reviews are the cornerstone of countless recommendation systems, from Amazon to Netflix, and yet many reviews are unhelpful or minimally informative. We explored Stanford course reviews with the goal of removing general or non-specific reviews. Employing a novel approach to review ranking, we frame a review's utility as its specificity to a given department. Stanford's 5,000+ classes would prove impossible for rote classification; however, using a class' department as a proxy for topicality, we are able to achieve significant results. Our LSTM model reaches 58.6% accuracy for classifying reviews for the top 10 most reviewed departments and 32.3% on the top 100. However, perhaps more important than accuracy is the confidence of each prediction, which can be used to determine whether a given review is a general or specific claim. Our results show that department classification is a valid, if simplistic, way of purging non-course-specific, and therefore, less informative reviews.**

## I. Introduction

Every quarter, thousands of Stanford students pore over course descriptions and reviews to help them make an informed decision when they register for classes. A key factor in this decision making process is wading through the hundreds upon hundreds of pieces of written feedback and recommendations provided by fellow classmates. While analytics are provided in separate views, like grade distribution, intensity of course load, and professor rating, the review text is an honest unfiltered way of understanding what peers who have taken the course actually think. This can be much more informative than just the few numerical points, with review text containing everything from clarifications on course content, warnings about specific assignments, to other specific points that students think others should know before enrolling.

However, not all of these course reviews are alike in quality. Many reviews merely contain monosyllabic exclamation or very general feedback only tangentially related to the course. See Table 1 for some examples of reviews from CS221 (Spring '18), along with brief thoughts on their utility.

## II. Task Definition

How then, can we filter reviews to only include interesting and useful content? This task can, and has been, approached in myriad ways, with the most canonical solution likely a search-return ranking system like term frequency-inverse document frequency (TF-IDF). However, there is inherent bias in ranking reviews explicitly; our assumptions about what people value would indubitably skew our algorithm design and end product. So, we chose to employ an approach that doesn't discriminate between reviews deemed useful, which is accomplished by a classification algorithm.

When choosing a model, we required something that could take any review, and output a label of "useful" or "non-useful." With over 150,000 total reviews in the dataset we created, manually labeling these reviews was impossible. By making the assumption that useful reviews contained information specific to the course or field, we were able to reformulate the problem. An LSTM classifier was constructed to take a review and output the department of the class that the review discussed. Because useful reviews discuss course content, and non-useful reviews discuss general feelings, useful reviews are much more easily classified to the department it came from. Then, we use the accuracy of the multi-class classification (between department) as our labels for useful or not; reviews that can successfully be associated with their department are given the "useful" label, otherwise they are given the "non-useful" label. For example, "This course is great" is hard to assign to a department, while "I loved learning about sauropsids" can confidently be assigned to the biology department (and by our assumption, is more useful).

The output of our model are two pools of reviews, those with correct and incorrect labels (more on model design below). This system can be used to keep "useful" reviews for each quarter at the top and "unhelpful" reviews at the bottom, allowing students to navigate courses with hundreds of reviews more effectively by promoting the reviews we believe to be the most useful to the top. Because most courses have less than 50 reviews per course, and our initial inspection of reviews revealed that 50% of reviews were unhelpful, this would lead to many classes having around

**Table 1 Sample course reviews and analysis**

| Course Review | Useful | Explanation |
|---|---|---|
| Enjoy! (2018/2019, Spring) | No (0) | The positive sentiment in this review is likely already conveyed through the rating given to the class by this reviewer. Thus, this monosyllabic review adds no additional information and is not very helpful. |
| Although I did extremely well on all of the assignments, I still feel like I don't understand many of the topics covered in this course, which is rather unfortunate. (2018/2019, Spring) | Yes (1) | This is an informative review that gives a specific critique that may be helpful for students to consider. |
| A great introduction to AI. Start on the project early and be prepared to spend quite a bit of time on the Problem Sets. (2018/2019, Spring) | Yes (1) | This is an informative review that gives specific information about the course. |
| Expect no weekends free (2018/2019, Spring) | No (0) | Although useful information, this brief review is referring to the number of hours spent in the course and is already encoded in the other information shown in the Carta page for this course. |

10-20 reviews; students eager to enroll in classes lose attention fast, so this amount of reviews seemed perfect for our goal of improving the process.

## III. Literature Review

Understanding massive amounts of small snippets of text, whether it be tweets, other social media, or in our case reviews, is a red-hot area of natural language processing research. However, most research in this field focuses on a few main goals, the most prominent of which is sentiment. In fact, just in sentiment analysis alone, there are over 7,000 papers accessible in popular computer science journals [1]. Among common methods for identifying sentiment within reviews include Naive Bayes with N-grams, algorithms with word vector embeddings, neural networks, etc. Although these methods inform ways to extract meaningful information out of text, which we utilize in our own methods, sentiment is not the general topic that we focus on here. Instead, we must turn to research specifically on review text.

One such paper analyzes Yelp reviews, attempting to extract interesting restaurant features. Using a support vector machine with both bag of words and TF-IDF features, they were able to determine the most important words in review text that correlated with the overall review [2]. Although successful, their results focused on identify individual words that helped to identify multiple classes. Ultimately this proved less successful when trying to dissect course reviews, which we hypothesize was largely due to words too specific to an individual department for generalization.

In yet another useful direction, Haoyu Zhang and team highlight the effectiveness of BERT in digesting and semantic parsing of text reviews [3]. Moreover, they take it one step further by using this semantic understanding of text in summary generation. Here we see the massive important of pre-trained embeddings, which proved more significant that model differences and other features.

## IV. Infrastructure

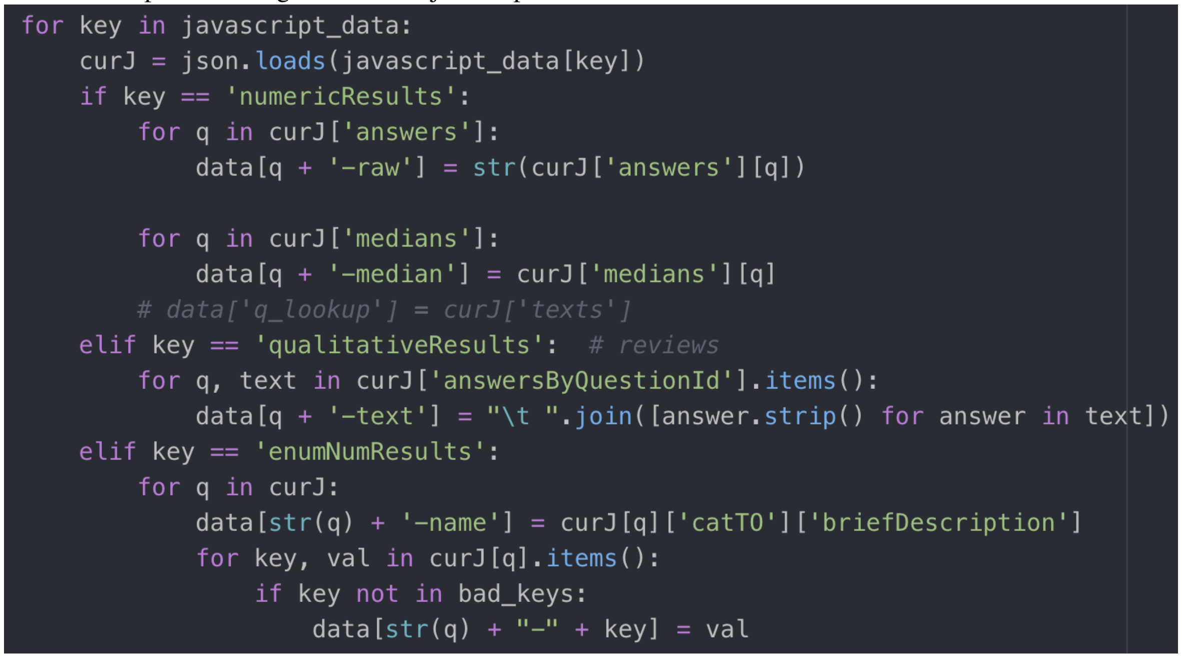
### A. Gathering Data

Due to sensitive review data and the internal workings of Stanford systems, we manually scraped the web for our review data. Stanford course reviews are stored by course/section leading to ~ 10,000 pages (8.4 GB) that required scraping. The following methods were employed to capture the review text and course metadata:

1) multithreaded bash/wget script for downloading the full contents of each page

```
jq -r '.data[] | ("showreport?c=" + (.id | tostring) + "&i=" + (.instructorId | tostring) +
"&t=" + (.termId | tostring) + "&r=3&embedded=true")' *.json | xargs -P 8 -n1 -I{} bash -c
"wget -nc -P reports --user-agent='Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0' --header='COOKIE REDACTED' 'https://www.applyweb.com/eval/new/{}'
|| exit 255"
```

2) BeautifulSoup for html digestion with a javascript renderer



```
for key in javascript_data:
    curJ = json.loads(javascript_data[key])
    if key == 'numericResults':
        for q in curJ['answers']:
            data[q + '-raw'] = str(curJ['answers'][q])

        for q in curJ['medians']:
            data[q + '-median'] = curJ['medians'][q]
        # data['q_lookup'] = curJ['texts']
    elif key == 'qualitativeResults':  # reviews
        for q, text in curJ['answersByQuestionId'].items():
            data[q + '-text'] = "\t ".join([answer.strip() for answer in text])
    elif key == 'enumNumResults':
        for q in curJ:
            data[str(q) + '-name'] = curJ[q]['catTO']['briefDescription']
            for key, val in curJ[q].items():
                if key not in bad_keys:
                    data[str(q) + "-" + key] = val
```

3) Querying ExploreCourses API for all Department Codes

### B. Processing

Text processing methods were crucial to model success; the order of manipulations was as follows:

1) special character removal
2) punctuation replacement
3) whitespace cleaning
4) tokenization
5) conversion to embedding with 20k unique word cap

### C. Key Characteristics Found in Reviews

1) 46985 reviews in top 10 most reviewed depts.
2) 119559 reviews in top 100 most reviewed depts.
3) avg. review length: 23.1 words
4) avg. word length: 5.75 characters

### D. Hardware/Software Used for Training

Training was done using the following hardware and software:

- Titan X Pascal GPU
- i7-4790k CPU
- NVIDIA driver 430.50
- Linux 4.15.0-72
- tensorflow-gpu 1.14.0
- CUDA 10.1
- CUDNN 7.6.5.32
- Python 3.6.9

### E. Zipped Data For Reference

1) data.zip : `https://drive.google.com/file/d/1cpMLegmPCnfezM5B15Zl2xSTcqqnYJUj/view?usp=sharing`

## V. Approach

### A. Oracles

Because of our unique approach, designing a useful oracle is non-trivial. For example, an obvious choice for an oracle might be the true department label for our multi-class classifier. While this oracle results in a 100% classification accuracy, this would correspond to concluding that 100% of course reviews were useful to students, which is the opposite of our goal. So, if true-labels for departments results in a bad oracle, where do we go from here? The answer is imperfect, but two separate kinds of human oracles were used to achieve much more significant results than the true-label oracle.

The first kind of human oracle was a human attempting department labeling. A human looks at a review, and decides which department it belongs to. With over 160 unique departments in our database, however, it proved very difficult for most humans to discern where a given review belonged. A further modification to this approach was to limit the reviews to belong to just the top 10 most reviewed departments: Computer Science, Biology, CEE, Education, History, Human Biology, Mechanical Engineering, MS&E, Political Science, and PWR. The human oracle was significantly better in this case, achieving nearly 50% accuracy. This exercise was also conducted for the top 100 departments.

The second kind of human oracle was a human attempting utility labeling. A human would look at a given review and determine whether it was useful to make a choice based on the following guideline: "Does this review contribute to more of a informed decision about this specific course relative to other courses a student might consider?" This guideline was chosen to encourage decision-making in a manner similar to that of real students, rather than a general question of usefulness.

### B. Naive Bayes n-Gram Baseline

We utilized n-grams as our non-human, non-perfect baseline. First, we split our data with a 90/10 training/test split. Next, we utilized bigrams, calculating the frequency of occurrence for each pair of words. Then, we made the Naive Bayes assumption with Laplace smoothing to calculate the probability that any given review would be found underneath a given department based on the product of the probabilities that each pair of words in the review would be found for a review in the department based on our training data. The Laplace smoothing prevented the probability of previously unseen bigrams giving a probability of 0 for a review being seen in a department. The equation for calculating the likelihood of $\theta$ being the department given data $b$ and making the Naive Bayes assumption is given below.

$$L(\theta) = \Pi_i(P(b_i \mid \theta)) \tag{1}$$

In order to simplify our calculations, we used the log of the likelihood of $\theta$ as illustrated in the equation below.

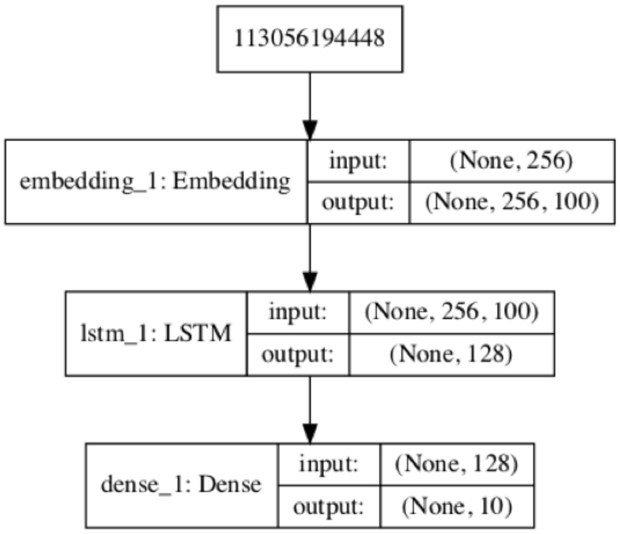
$$LL(\theta) = \Sigma_i(P(b_i \mid \theta)) \tag{2}$$

We picked the department that the review had the highest probability of belonging to. We can pick the department $\theta$, with the equation $\texttt{argmax}_\theta LL(\theta)$. We then followed a similar procedure for trigrams.

## C. Long Short-Term Networks (LSTMs)

Fig. 1 below describes an overview of our neural network.

**Fig. 1 Overview of LSTM network**



First, we have our embedding layer. This looks up each token in our input and outputs a unique vector for each one. Practically, this is a giant matrix multiplication whose input is a 1-hot for each possible token.

Second, our LSTM layer. This is a long short-term memory, with 128 neurons of internal state. An identical cell is applied to each token of the input going from left to right. State is maintained in the form of these 128 "memory" neurons.

The final output of the LSTM after the final embedded token of input is fed into our final layer, which is a dense layer. This means every neuron of the final has a connection to every neuron of the last LSTM cell.

An activation function is applied after this final layer, in our case, softmax. Softmax is what guarantees that all of our outputs sum up to 1, for this reason it is a common choice for classification.

## D. Zipped Code For Reference

1) code.zip : `https://www.dropbox.com/s/gnd9i0334epo97k/code.zip?dl=0`

## VI. Results

Our results for our human oracle, bigrams and trigrams are shown in Table 2. Our human oracle does best, as expected. Our results also improve when we narrow down the departments to the 10 department with the most reviews. Interestingly, our performance decreases when we from Naive Bayes and bigrams to trigrams, likely due to the fact that with limited training data, and thus, a limited set of triplets of words, this predictive algorithm has high variance, and cannot adapt well to new reviews.

**Table 2 Results for Baselines**

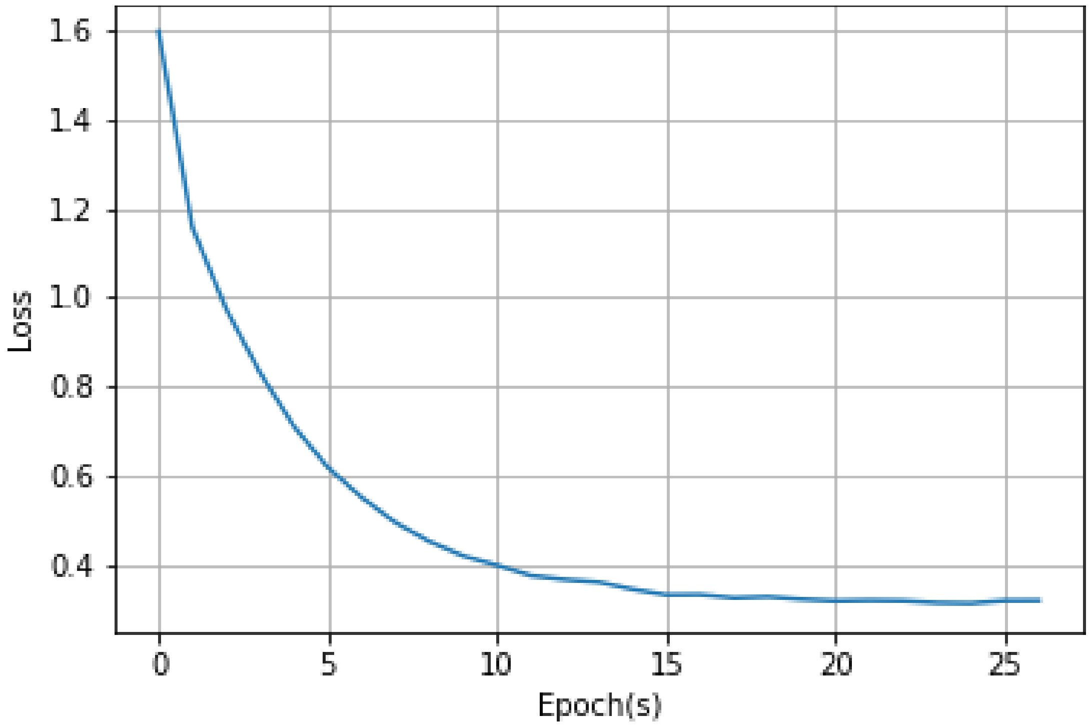
| Model | 100 | 10 |
|---|---|---|
| Human Oracle | 28% | 49% |
| Naive Bayes bigram | 18.24% | 43.7% |
| Naive Bayes trigram | 15.41% | 33.83% |

We first trained our model over departments and graphed the training loss as the number of epochs increased in Fig. 2. As the number of epochs continue, our training loss decreases until it stabilizes around 15 epochs and reaches its minimum around 26 epochs.

**Table 3 Experimental Results for LSTMs**

| Model | 100 | 10 |
|---|---|---|
| LSTM | 32.30% | 58.59% |

**Fig. 2 Training Loss Progression**



Our best accuracy as of the poster session was 55.78 percent. We tried various hyperparameter changes, of which only one was fruitful. For example, changing vocab size to 2000 decreased test accuracy on 10 departments to 46.09. However, changing maximum sequence length to 128 increased test accuracy on 10 departments to 58.59 percent after 4 epochs.

Because we had limited training data, we achieved our best results when we narrowed down the number of departments. For Fig. 3, we focused on only the 10 departments with the highest number of reviews and graphed training accuracy by each department. As the number of epochs increase, we need just 5 epochs before our training accuracy stabilizes around 85% on average for each department.

Again for the Fig. 4, we focused on only the 10 departments with the highest number of reviews and graphed test accuracy by each department. As the number of epochs increase, we need just 5 epochs before our test accuracy stabilizes around 60% on average for each department.

## VII. Error Analysis

In our latter two graphs, we see that we have a high level of accuracy for two of the departments (Computer Science and Management Science and Engineering) in both graphs when we have 0 epochs (i.e. before any training has occurred). Computer Science is around 40% accuracy in Fig. 3, and around 38% accuracy in Fig. 4. Management Science and Engineering is at around 30% accuracy in both figures. We realized that this is because a classification label is randomly assigned to each course review, and by chance Computer Science and Management Science and Engineering are assigned correctly to many of their course reviews. We should expect, given that there are 10 departments, and the

**Fig. 3 Train Accuracy Across True Departments by Epoch**

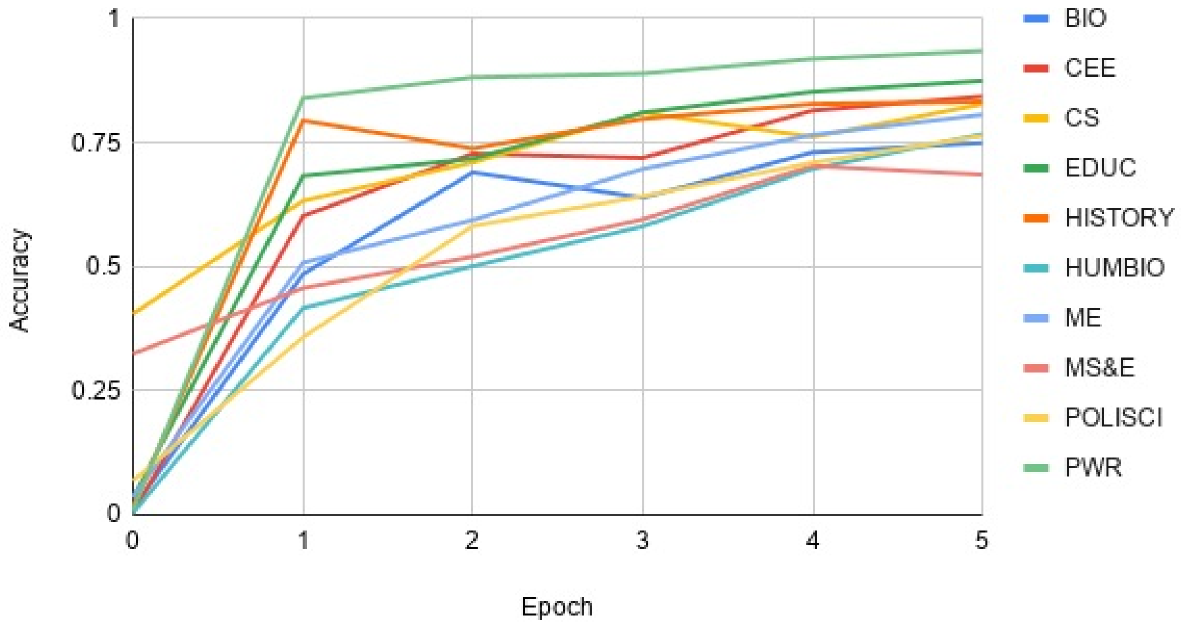
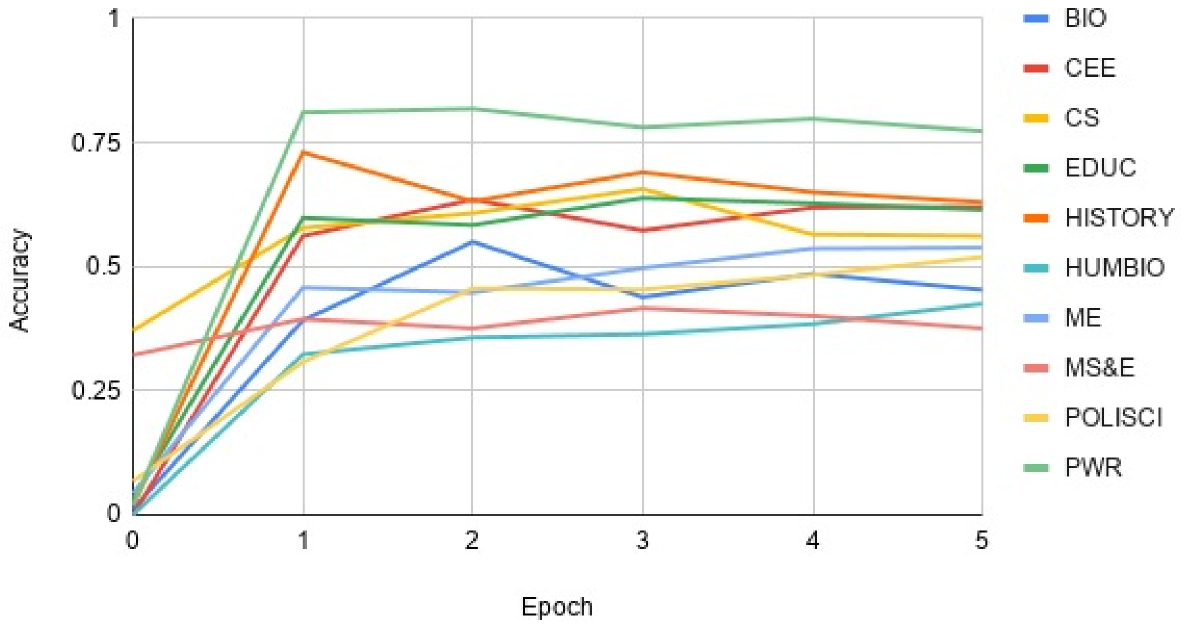


**Fig. 4 Test Accuracy Across True Departments by Epoch**



initial classifications are initialized randomly, that on average, approximately 10% of the reviews are correctly classified, and this is in fact the case, when we averaged the accuracy of our classifications for all 10 departments in each graph, at 0 epochs. Therefore, this high accuracy was in fact, due to random initialization, something we confirmed by rerunning the model and getting two different departments with accuracy between 25% and 50%. In essence, the untrained and randomly initialized model is behaving like a biased random number generator, operating completely independent of its input (when we reinitialize it with new random weights, the departments that it favors are randomly reselected).

We recognize that our method of utilizing rating a course review as "helpful" based on if we can identify the department it was written for has its flaws. It makes several potential assumptions such as the belief that reviews for different departments are written differently in a systematic way across the department. This will affect reviews in a Writing-In-Major class for Mechanical Engineering for example, as the content of the course review will likely appear similar to the Program in Writing and Rhetoric department. In addition, courses cross-listed across multiple departments can affect our results both when it comes to training and testing. For example, Natural Language Processing (NLP) classes are listed under both the Linguistics and Computer Science Departments and may cause NLP-related words to be associated with only one of the departments while training, and thus, will only identify one of the departments the class belongs to while testing.

Our algorithm is looking for keywords. For example, any review that contains the word "VR" will be identified as part of the Computer Science department, even if it belongs to the Emergency Medical Services or Electrical Engineering Department. This makes it susceptible to being ’tricked’. For example, a course review that has a keyword associated with another department will cause the review to be misclassified. In addition, this means that a review consisting solely of keywords like "VR. CS. AI." will be rated as "helpful" for any Computer Science class, even though the review is not compromised of meaningful sentences and may not even be relevant to the course it is written for. Similarly, someone may just paste the course name or description into the review and have a "helpful" review according to our algorithm, which would be able to correctly identify the review.

Looking at missed predictions is imperative to the utility of our model, as missed predictions occur when reviews lack department specificity. Table 4 contains random misses from the top 10 model.

**Table 4 Error Analysis**

| **Pred** | **Actual** | **Text** | **Explanation** |
|---|---|---|---|
| HIST | ME | Great course Great professor | Generic. Misclassification is to be expected (and desired) |
| CEE | POLISCI | Organized on top of responding to email | Though a useful review, misclassified because there is no content-related information, just information related to the professor’s methodology |
| CS | ME | Willingness to go beyond the scope of the course to explain concepts | Though a useful review, misclassified because there is no content-related information, just information related to the professor’s methodology |
| CS | ME | awesome | Generic and single word. Misclassification is to be expected (and desired) |
| CS | EDUC | I absolutely loved this class and highly recommend it | A positive review, but the effects of the sentiment should be captured in the average rating given to the course, and thus misclassification is acceptable. |
| HUMBIO | ME | Ability to care – she is so caring and can calm you down if you re a hot mess Wow such a great human | Though a useful review, misclassified because there is no content-related information, just information related to the professor’s methodology |

### A. Common Themes with Misclassification

We want to correctly classify reviews that are informative. However, some reviews that we misclassify that have useful general descriptions of the teaching style that professors use. We also misclassify reviews that focus solely on methodology, rather than content. This is of course, due to our heuristic that relies on the review describing the content of the course so that the department can be identified. Thus, to rate these reviews as "helpful", we would have to incorporate other heuristics that can also identify reviews that describe the professor's style and methodology well, perhaps by looking for keywords.

## VIII. Further Work

Given what our assumption overlooked (helpful reviews that described a teacher's method, if not the class content), we would use a feature selection algorithm; one that utilized the score from the LSTM, and others that looked for keywords (similar to sentiment analysis). By utilizing multiple criteria, we could account for the fact that descriptive reviews can contain information on both the content of the material and methodology of teaching used in the course. To expand on our use case, we can train on more data from previous years, and test on future years. We would also like to segment not just by department, but also by level (e.g. 100 vs 200 classes). These changes would likely increase the specificity of the algorithm, allowing for more specific review criterion, and thus decreasing the number of reviews shown. Although more successful from a technical standpoint, these changes might leave too little for the discretion of the reader; this is a topic rich for further analysis. Our proof of concept confirms the unencumbered usefulness of natural language processing systems. By careful problem construction, in conjunction with relatively boilerplate LSTM neural network structures, very interesting results can be found. This raises many possible applications for such an algorithm, in a much wider lens than just reviews. One in particular is the application of a similar method to the descriptions of different courses. What are the classes with the descriptions that best match reviews? Can we write a description for a class just based on the reviews of students? What would that look like? These questions are just a few of the directions we considered when developing the project.

## IX. Conclusion

In conclusion, we believe that our results obtained from our LSTM network are quite promising. We are able to identify course reviews that give useful insights into the content of the course. Because the current review structure in Carta is to show every review, in alphabetical order, no matter the content, our filtering mechanism clearly has potential. Used in conjunction with Carta, useful reviews would appear much more frequently and help students make more informed decisions when picking their classes and in a quicker fashion. Our unique approach yields high results proportional to the amount of computational power and generalizability of our model.

## Acknowledgments

We would like to thank Professors Percy Liang and Dorsa Sadigh, Head TA Reid Pryzant, our mentor Chuma Kabaghe, and the rest of the CS221 teaching staff for a great quarter! Have a great winter break!

## References

[1] Mäntylä, M. V., Graziotin, D., and Kuutila, M., "The evolution of sentiment analysis—A review of research topics, venues, and top cited papers," *Computer Science Review*, Vol. 27, 2018, pp. 16 – 32. https://doi.org/https://doi.org/10.1016/j.cosrev.2017.10.002, URL http://www.sciencedirect.com/science/article/pii/S1574013717300606.

[2] Yu, B., Zhou, J., Zhang, Y., and Cao, Y., "Identifying Restaurant Features via Sentiment Analysis on Yelp Reviews," 2015.

[3] Zhang, H., Xu, J., and Wang, J., "Pretraining-Based Natural Language Generation for Text Summarization," 2019.