Mining for Confusion: Classifying Affect in MOOC Learners' Discussion Forum Posts

Akshay Agrawal
akshayka@cs.stanford.edu

Shane Leonard
shanel@stanford.edu

1. INTRODUCTION

Interest in educational data mining (EDM) has grown exponentially in recent years [5]. The Journal of Educational Data Mining was established to accommodate the surge in EDM research activity; in the following years, Learning at Scale and the International Conference on Learning Analytics cropped up as well. Broadly, EDM "is concerned with methods to explore unique types of data in educational settings and ... to better understand students and the settings in which they learn," [1]. Its applications cut across many domains within the educational setting, ranging from visualization of course activity to automating course construction.

The advent of Massive Open Online Course (MOOC) platforms edX and Coursera in 2012 unleashed vast troves of data for researchers to mine. Much effort has gone into developing "student models" – predicting students' next-question-accuracy and modeling their acquired skills. Some argue that such research is reaching the point of diminishing returns [1].

In contrast, relatively little research has been devoted to digging into MOOC discussion forums. Learners often resort to discussion forums in order to better understand the subject matter at hand. MOOC enrollments, however, are large: The typical Coursera class, for example, consists of more than 40,000 students [3]. These learners' posts get drowned out in a sea of other, non-academic posts, especially since some learners use forums as conversational chatrooms. As an unfortunate result, instructors might overlook questions posed by struggling learners, discouraging both forum participation and learner motivation [4].

Ideally, the MOOC platform would automatically feed instructors with posts relevant to them. Instructors could then simply answer posts out of this *intelligent queue*, relieving them of the burden of sifting through thousands of discussion posts by hand. Such a platform would empower learners, too, and would more generally boost the usefulness of discussion forums.

In this paper, we take an initial step towards an intelligent queue by presenting a pipeline that automatically surfaces posts that exhibit confusion. We say that a post exhibits confusion if it appears that its author either explicitly requests for clarification about a course topic, or if his language implicitly reveals a gap in comprehension. In contrast, we say that a post exhibits knowledge if it delivers factual information relevant to the topics studied in the course. If the post conveys neither confusion nor knowledge, we say that it is neutral. While EDM work on analyzing sentiment in MOOC forums exists [6], to the best of our knowledge, no previous work has explicitly attempted to identify confusion in posts. Other EDM efforts have attempted to predict instructor intervention patterns [2]; such an approach will only be successful in generating an intelligent queue if the instructor is already effective in answering relevant forum posts, which might not be the case. We instead take the approach of discovering content that we a priori believe will be relevant to instructors, regardless of their discussion forum history.

We frame the problem of retrieving confused posts as a multilabel classification problem, of the following form: Given the body of a discussion forum post P with a true unknown label L in { *knowledgeable, neutral, confused* }, apply some hypothesis h that correctly divines L. That the problem is a multilabel one is of significance: Solving the easier binary classification problem in which posts are labeled as confused or not-confused would force us to forfeit potentially useful information about where a post author lies on the knowledge-confusion spectrum.

The remainder of this paper is organized as follows. We describe our dataset in section two, detail our preprocessing, feature extraction and feature selection pipelines in section three, list our models in section four, present our results in section five, interpret them and discuss insights gleaned in section six, and propose future work in section seven.

2. DATASET

Two sets of nearly 10,000 distinct discussion forum posts each were collected from Stanford Online's OpenEdx offerings and scored by three distinct judges, hired from oDesk; one set consisted of six courses categorized under the humanities, while the other consisted of four courses categorized under medicine.

Our dataset has eight relevant columns that convey the following; the full set of columns is listed in table 2. Particularly interesting columns include the *confusion* column, which represents the extent to which the post communicated knowledge or confusion. Confusion was encoded as a number from 1 through 7, with 1 indicating plentiful knowledge and 7 encoding plentiful confusion. Other columns provided metadata about the post and its author.

The process [1] of gathering, cleaning, and preparing the dataset was labor intensive but necessary – no vetted tagged

[1]For more information about this process as well as an anal-

dataset of confused posts exists. The gold sets will soon be made publicly available through Stanford.

3. FEATURE GENERATION

We have a potentially $|D| + k + 6$-dimensional feature space, where $|D|$ is the number of unique words encountered in a training set (on the order of thousands), k is the number of additional features we engineer, and 6 corresponds to the last six rows in Table 2 above. We call the entire pipeline from pre-processing and feature extraction to feature selection *feature generation.* Our feature generation pipeline takes as input the forum post body text and the *confusion*, *up-vote*, *anonymous*, *anonymous-peer*, *type*, *reads*, *attempts*, and *grade* variables for each forum post. Each of these variables is processed in parallel and converted into a fractional feature vector. We take the union of these fractional vectures to produce a *full* feature vector, and finally apply a feature selection algorithm to winnow the full feature vector down to the feature vector fed to our classifiers. For each of our variables, there exists a simple mapping that takes them to an integer representation. Processing the text document, however, requires significant work. The text processing pipeline consists of three stages: (1) preprocessing, (2) feature engineering, and (3) feature extraction.

In the preprocessing stage, we tokenize the text with a handrolled regular expression[2] that matches single character words and punctuation, in addition to multiple character words. We clean text by mapping the families of LaTeX equations, numbers, and URLs to three unique tokens, and by converting all tokens to lowercase[3]. Finally, we discard words that are included in modified version of the Glasgow Information Retrieval Group's English stop words list – our version does not contain the tokens 'I', 'can', 'cant', 'cannot', 'could', and 'couldnt'.

In the feature engineering stage, we artificially manipulate our token space in order inflate the weights of particular tokens or sequences of tokens. In particular, we inflate duplicate each token that occurs in the first sentence, with the assumption that students might summarize their intent early on in their posts – unforutunately, our dataset did not capture the titles of each post. We also optionally leverage a chunk parser [**?**] and classifier to find noun phrases in the original text documents, and then duplicate each token that occurs in the noun phrase. We additionally include the option to construct k-character n-grams from the tokens.

In the feature extraction stage, we convert our tokens to integers. We begin by simply counting occurrences of each token; on a per-model basis, we convert these counts to binary indicators (see the next section for more information). Finally, we optionally replace raw counts with their TF-IDF equivalents [**?**]. Intuitively, doing so allows us to generate a dynamic 'sieve'-word list by giving less weight to words that occur in many documents, without necessarily rejecting them altogether.

After extracting features from our text documents and from our metadata variables, we take the union of the generated vectors to produce our full feature vector. We then optionally apply the chi-square univariate feature selection algorithm in order to reduce the dimensionality of our feature space. The chi-square metric is often used in bag of words feature selection – it performs relatively well and is computationally inexpensive [7]. In our tests, we experimented with no feature reduction, and reducing to the $200k$ most relevant features, k from 1 to 6.

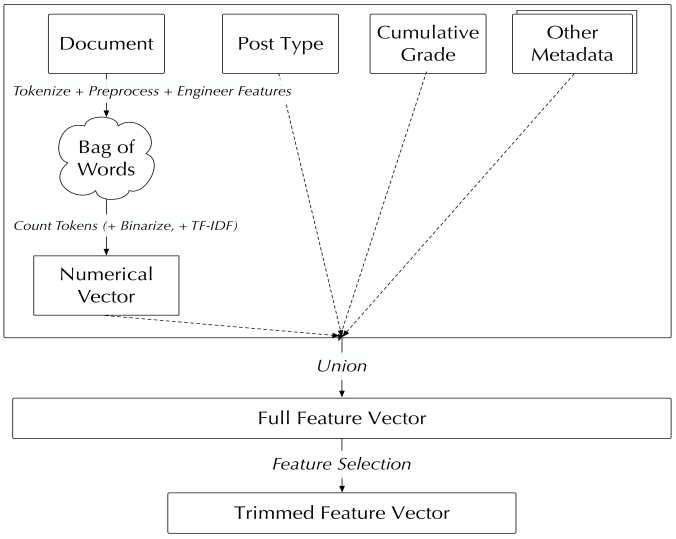

Figure 1: Generating the Feature Vector. Sketched here is the entire feature vector generation pipeline, from preprocessing and feature extration to feature selection. The input to our pipline consists of the columns listed in Table 2. Each variable is processed in parallel, but the most work (and the most interesting work) goes into processing the body of the forum post. The text document is first tokenized and cleaned. An array of feature engineering algorithms are then applied to the sequence of tokens, givin us an intermediate bag of words representation of the document. The counts are further processed, and are then unioned with representations of our metadata variables. Finally, we apply a univariate feature selection algorithm to winnow down our original feature vector into the one ingested by our classifier. The parenthetical actions in the figure are applied on a per-classifier basis.

4. MODELS

Multiple classification models were implemented and evaluated for the purpose of classifying forum post confusion. These models include multinomial Naive Bayes, linear SVM, and logistic regression. Each of these models was selected on the basis of several factors, including ease of implementation and their applicability to a unigram text model (i.e. bag-of-words approach). Current research suggests that, while bag-of-words is not ideal, "classification methods that use the bag-of-words feature space often achieve high perfor-

ysis of inter-rater reliability, see:
`http://bit.do/krippendorff`

[2]The Python regular expression:
`r"(?u)(\b\w+\b|[.,;!?])"`

[3]Case is sometimes helpful in detecting affect; for example, in some online forums, frustrated posters may write in all-caps. However, discussions in MOOC forums tend to be fairly civil, and such casing is rarely used.

Variable	Description
confusion	A number from 1 through 7, with 1 indicating maximum plentiful knowledge and 7 indicating plentiful confusion.
timestamp	When was this post submitted?
up-vote	How many up-votes did this post receive?
anonymous	Was the poster anonymous to all persons?
anonymous-peer	Was the poster anonymous to his peers?
type	Did this post initiate a thread, or was it a reply?
reads	How many reads did this post's thread garner?
attempts	How many submission attempts had this post's author made on the courseware before timestamp?
grade	What was this post author's grade at time *timestamp*?

Table 1: Dataset Columns. *Confusion* is a weighted average of the judges' scores; all other columns were computed from the data contained in Datastage (datastage.stanford.edu).

mance" [9]. In addition, the chosen classifiers have been shown to be near state-of-the-art compared to other bag-of-words approaches. This is particularly true for linear SVM [9, 10, 11].

We were able to implement each of these models using the scikit-learn toolkit in Python. Using the scikit API, we built a pipeline that handled preprocessing, feature selection, and classification. The pipeline provided a modular way to easily interchange and compare the classifiers, as well as the effects of feature selection algorithms like TF-IDF and k-best selection.

The multinomial event model for Naive Bayes was chosen over a multivariate Bernoulli model because it has been found to outperform this model when the vocabulary is large. To briefly differentiate between the two, the multinomial event model captures the number of occurences of each word in a document, while the multivariate model does not. Both models use the Naive Bayes assumption [8]. We chose to implement Naive Bayes classification as a first approach, and to provide a baseline for comparison with logistic regression and linear SVM. As expected, the Naive Bayes model worked reasonably well, but the other models proved to be more effective.

It is generally well-accepted that support vector machines (SVM) provide state-of-the-art performance for text classification, and in particular they perform well with simple feature spaces such as bag-of-words or bag-of-features [9, 10, 11]. In addition, SVMs tend to be more robust than Naive Bayes classifiers when applied to a skewed category distribution, as is the case with our dataset [13]. Therefore, implementing an SVM was a natural choice for our application. We ended up using a simple linear kernel, instead of the RBF (Gaussian), polynomial, or sigmoid kernels. This was partly because we lacked the computational and temporal resources to tune kernel parameters, and partly because the linear kernel was able to achieve high performance. We did not consider it necessary to spend time implementing nonlinear kernels because prior research has suggested that, for text classification, they offer little performance benefit over linear kernels [12, 13].

As a final model, we applied logistic regression. We chose logistic regression because several sources suggested that it can achieve similar performance as the SVM. For the text portion of the feature vector, we modified the multivariate event model. Instead of recording the counts of each word, we simply recorded whether the word was present or absent from the document. This modification had a positive impact on the classifier's performance. We found that the logistic regression did outperform Naive Bayes, and was similar to the SVM, as expected.

5. RESULTS

We evaluated the performance of the classifiers with several different options for the feature selection and feature space. For the feature space, we compared: 1) a text-only bag-of-words representation without any post metadata, and 2) a union of text features and the post metadata, as described in the Feature Generation section. For each feature space, we evaluated the logistic regression, SVM, and multinomial Naive Bayes classifiers using: 1) no feature selection, 2) TF-IDF, or 3) K-best feature selection with the chi-squared metric, also as described in the Feature Generation section. Figure **??** displays the results for both feature spaces using no feature selection, figure 2 displays the results using TF-IDF, and figure 3 displays the results using the k-best feature selection. These figures show the average precision, recall, and f1 scores from the cross-validation test for each model.

6. DISCUSSION

6.1 Interpretation of Results

We used precision, recall, and the f1 score as our primary metrics of success, as reported after applying our classifiers to our test sets. The best classification f1 score of 0.637794 was achieved by the logistic classifier. This score was achieved without TF-IDF or K-best feature selection, applied to the full text-plus-metadata feature space, using l2 regularization with a penalty parameter of $C = 0.4$. Every preprocessing step was used, namely custom tokenization, lemmatization of numbers, lemmatization of LaTeX equations, lemmatization of urls, and a custom stop words list.

Classifier	Precision	Recall	F1
Logistic: All Features	0.691571	0.597377	0.637794
Logistic: Text Only	0.66992	0.580685	0.618474
Mult. NB: All Features	0.298663	0.542611	0.374966
Mult. NB: Text Only	0.427971	0.575239	0.472454
Lin SVM: All Features	0.577934	0.552698	0.513804
Lin SVM: Text Only	0.641299	0.589635	0.611046

Table 2: Classifier performance on Humanities courses. Listed in this table are the precision, recall, and F1-score per classifier for the confusion class. We collected these numbers by using 10-Fold Stratified Cross Validation on the testing set. The 'all-features' classifiers used both textual data and the metadata listed in Table 2. The 'text-only' classifiers only used the text body of forum posts to generate features.

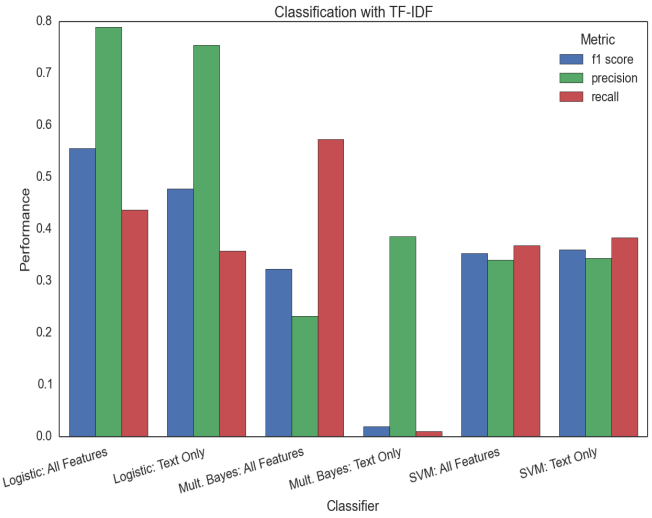

Figure 2: Classification results using tf-idf on humanities.

Overall, we found that the SVM and logistic regression classifiers performed reasonably well on the data. We also found that the multinomial Naive Bayes consistently performed significantly worse than these two models. This is consistent with our expectations based on prior research. Multinomial naive bayes performed particularly poorly when all variables were used because it treats its inputs as counts – e.g., a grade of 100 is equivalent to 100 occurrences of some token, so our classifier overfit to these metadata variables. What was surprising, however, was that the additional metadata only slightly improved the classification results of logistic regression compared to the text-only feature space. Indeed, performing ablative analysis on the logistic regression model demonstrated that the only variable that made an appreciable difference was *type*. This may be because the dataset was not sufficiently cleaned for edge cases (e.g., we have reason to believe that *grade* and *attempts* were unreliable, but it also may simply be because the features were noisy or generally less informative than the unigrams.

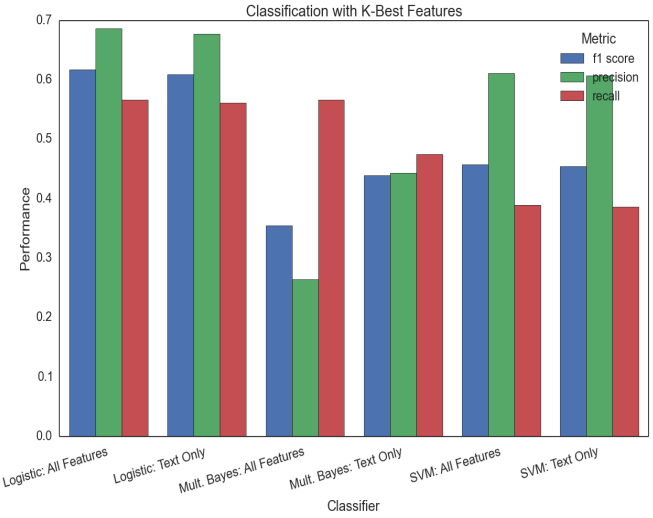

Figure 3: Classification results using chi-squared k-best feature selection (400 best features) on humanities.

As demonstrated in Figures 2 and 3, tf-idf and chi-squared feature selection did not necessarily help our results. Manually inspecting the most informative words suggested that the feature-space reduction caused our classifiers to select esoteric words that just-so-happened to be associated with confused posts in the training set.

6.2 Insights

Some interesting qualitative insights arose during the course of this research. When implementing the pipeline, we were able to use the scikit-learn API to surface the most relevant features for each model. With Naive Bayes, the most relevant text markers for confusion were very specific—usually names, elements from an equation, or other tokens that appeared in only a handful of forum posts. They did not reflect any implicit understanding of confusion in general; instead, the Naive Bayes model had overfit to some specific tokens that appeared only in positive examples. By contrast, the logistic regression and SVM identified text features that revealed a much better understanding of confusion: the features they deemed relevant included words like 'confusion', 'problem', 'help', '?', 'understand', and 'thanks'. Figure 4 illustrates the relevant features picked out by logistic regression during 10-fold cross-validation over the full dataset. The size of each word corresponds to the number of folds that identified the word as one of the top 50 text features.

Another interesting finding was that doing cross-validation with leave-one-course-out produced significantly better results than cross-validation across all the data. For example, in humanities, leaving one statistics course out and while still having another statistics course in our training set let us achieve an F1 of 0.80. This suggests that it may be easier to detect confusion in technical courses, and suggests that an online machine learning pipeline could work well, especially if we trained on previous iterations of courses.

7. FUTURE WORK

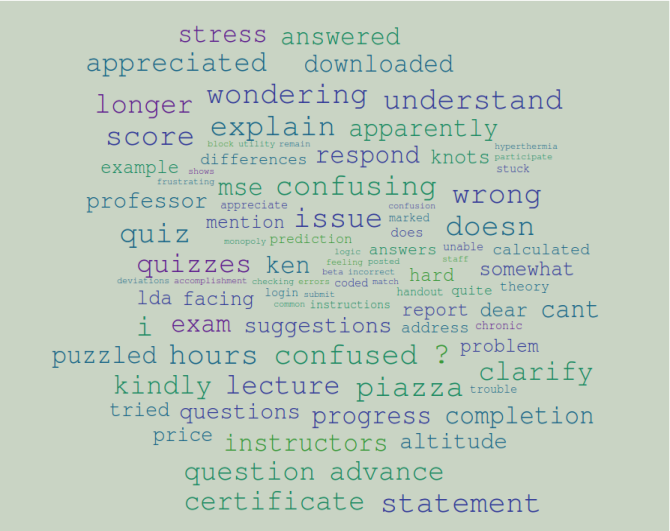

Figure 4: Unigrams identified as relevant by logistic regression during stratified 10-fold cross-validation.

We have several suggestions for the future direction of this research. First, there is much room to investigate more sophisticated feature spaces that augment or replace the straightforward bag-of-words approach. Since confusion detection is related to sentiment analysis and text categorization—topics of significant ongoing study—there exist approaches that will likely surpass the bag-of-words and bag-of-features implementations presented here. Building off of the results here also provides the opportunity to create a new kind of corpus labelled with confusion markers. As of yet, this corpus does not seem to exist.

8. CONCLUSION

We tackled a novel problem that had not been previously attempted by the educational data mining community – that of detecting confusion in forum posts. We had an involved feature generation pipeline and demonstrated that we can achieve fairly good results. In addition, we were able to tease out interesting insights, and found that our classifiers might work well in an online setting, at least for technical courses.

9. REFERENCES

10. REFERENCES

[1] Beck, Joseph E., and Xiaolu Xiong. "Limits to Accuracy: How Well Can We Do at Student Modeling?." Educational Data Mining (2013). APA

[2] Chaturvedi, Snigdha, Dan Goldwasser, and Hal DaumÃl' III. "Predicting InstructorâĂŹs Intervention in MOOC forums."

[3] Koller, Daphne, et al. "Retention and intention in massive open online courses: In depth." Educause Review 48.3 (2013).

[4] McGuire, Robert. "Building a Sense of Community in MOOCs – Campus Technology." Campus Technology. Public Sector, 3 Sept. 2013. Web. 12 Dec. 2014.

[5] Romero, CristÃşbal, and SebastiÃąn Ventura. "Educational data mining: a review of the state of the art." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 40.6 (2010): 601-618.

[6] Wen, Miaomiao, Diyi Yang, and Carolyn Penstein Rose. "Sentiment Analysis in MOOC Discussion Forums: What does it tell us?." Proceedings of Educational Data Mining (2014).

[7] Zheng, Zhaohui, Xiaoyun Wu, and Rohini Srihari. "Feature selection for text categorization on imbalanced data." ACM SIGKDD Explorations Newsletter 6.1 (2004): 80-89.

[8] McCallum, Andrew, and Kamal Nigam. "A Comparison of Event Models for Naive Bayes Text Classification." AAAI-98 workshop on learning for text categorization, 1998.

[9] Boulis, Constantinos, and Mari Ostendorf. "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams." Proc. of the International Workshop in Feature Selection in Data Mining. 2005.

[10] Pang, Bo, and Lillian Lee. "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts." Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2004.

[11] Forman, George. "An extensive empirical study of feature selection metrics for text classification." The Journal of machine learning research 3 (2003): 1289-1305.

[12] Colas, Fabrice, et al. "Does SVM really scale up to large bag of words feature spaces?." Advances in Intelligent Data Analysis VII. Springer Berlin Heidelberg, 2007. 296-307.

[13] Yang, Yiming, and Xin Liu. "A re-examination of text categorization methods." Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999.