Automatic Classification of Quiz Bowl Questions
LIN/COS 280 Final Project

Kenneth Jenkins

1 Introduction

The goal of automatic classification is to determine of which category out of a predefined set of categories a sample is most likely a member, given a training set of already-classified samples. For this project, I apply several classifiers to the problem of automatically categorizing quiz bowl questions. The American Competition Federation (ACF), in particular, makes past competition questions available in online archives,[1] but these archived questions are not categorized. It would be of considerable use for college quiz bowl teams to have searchable archives of categorized questions, and automating the categorization of these questions would be preferable. To this end, it seems the most successful system would be one that can classify a question and compute a confidence estimate of that classification; if the system classifies only the questions it is relatively 'sure' of. The others could then be manually classified by a human.

2 Data

The questions for the training corpus were taken from `http://www.carloangiuli.com/acfdb/`, a database of categorized quiz bowl questions from the ACF Question Archive. [cite] The data set contains 10697 classified questions, and was available for download as a text file with easily-parsable formatting. Table 1 shows the breakdown of questions by category and subcategory. Several duplicate questions were removed (including one apparently erroneous duplicate question which attributed *Tractatus Logico-Philosophicus* to Cyrano de Bergerac.)

I took several normalization steps with the data before using it to train the classifiers. I converted all the words to lower case and removed all punctuation. I also removed everything in between parentheses or brackets, because these seemed to be used only for pronunciation or other cues for the benefit of the reader of the question.

[1] `http://acf-quizbowl.com/archive.php`

Category		**Subcategory**	
History	22.4%	American	32.8%
		European	29.3%
		World	24.8%
		Ancient	13.1%
Literature	22.1%	European	52.5%
		American	28.3%
		World	18.3%
		Language Arts	0.9%
Science	21.1%	Biology	26.9%
		Physics	22.9%
		Chemistry	22.2%
		Mathematics	14.4%
		Astronomy	7.8%
		Earth Science	5.8%
Social Studies	10.5%	Religion/Mythology	39.5%
		Geography	21.3%
		Philosophy	19.7%
		Psychology	7.8%
		Economics	5.8%
		Anthropology	5.8%
Fine Arts	19.5%	Art	39.8%
		Music	47.5%
		Other	12.7%
Other	4.4%	Pop Culture	65.5%
		Other	34.5%

Table 1: Categories present in the data set.

3 Method

For classification problems with more than two categories there are two general approaches. Say we have k categories. The first approach is to train k binary classifiers to distinguish the ith category from all the rest. The classifier with the highest output determines the category. The second approach is to train $\frac{1}{2}k(k-1)$ binary classifiers to distinguish be-

tween each pair of categories. The category with the most positive results is selected. In practice, the first approach would likely be more efficient because it is only $O(n)$ in the number of classifiers required per number of categories, while the second approach is $O(n^2)$ in classifiers versus categories.

In this case, the Other category may present a bit of a challenge. There is not likely to be many positive distinguishing features for this category, so ideally we would use a classification scheme which sorted into the other five categories but was not forced to make a choice; if the output was beneath a certain threshold it would assign the Other category instead.

For a problem like the the quiz bowl questions, we have a hierarchy of categories (categories containing subcategories), which makes this problem different from the several examples I have found in the literature. [citez!] For instance, there is a simpler problem to attempt: one could train and test classifiers for just the six categories, which is presumably easier than doing so for the 25 total subcategories. For the larger problem of classifying category and subcategory, one could go about it two different ways: we could treat all the category-subcategory pairs as individual categories, or we could split the problem into two stages, first classifying the main category, and then running subcategory classifiers specific to that main category on the output from the first stage.

An easier task to consider would be to classify just the main category (Literature, History, Science, Social Studies, Fine Arts, and Other). Limiting to six categories reduces the complexity required to implement the second general approach considerably. This appears to be a relatively simple task, because there are many words which are unique to a given category. This suggests a very simple binary classifier as detailed in §3.2.

3.1 Feature selection and weighting

Half the battle for automatic classification is the selection of a suitable set of features to use for classification. For this task, the logical choice for features are the words that make up the questions, but it is not feasible to assign feature space vectors to each document based on every single word in the corpus. We want to make the classifiers as efficient as possible, so we want to choice words that are most distinguishing of a certain category. The nice thing about these quiz bowl questions is that they have a diverse lexicon given their length, and not surprisingly many words appear only in one category. For instance, the word "benzene" appears only in the Science category.

One pseudo-metric for feature weighting is the well-known TF·IDF (term frequency–inverse document frequency), which correlates terms with a given document according to how often they appear in that document times the reciprocal of the fraction of documents containing that term. While very common, this metric is rather time-consuming to compute for every word in every document in the testing corpus.

The traditional feature selection process is also complicated in this case by the fact that each question is very short, and may not contain any of the selected features. That would not be good, because there would be no possible way to distinguish zero-vectored questions in different categories.

Alternatively, it has been suggested that TF·IDF is not a very good feature weighting scheme and that the so-called *bi-normal separation* (BNS) is a better measure [For08]. According to Forman, the BNS score for a given word w with respect to category C_i is given by

$$\left|F^{-1}(P(w \mid C_i)) - F^{-1}(P(w \mid \sim C_i))\right|$$

(where F^{-1} is the inverse normal cumulative distribution function.) The basic idea is that for each word in the training corpus, we compute two probabilities: the maximum likelihood estimate that a random word in a particular category is that particular word, and the maximum likelihood estimate that a random word not in that category is the chosen word. The best features should occur much more often in one or the other, so we want to chose words that have the greatest difference between these two probabilities.

For my "modified Bayes classifier" (see §3.4) I employed a variant of the BNS score. As noted in [For08], the inverse normal cumulative distribution function is not available in common math libraries, so I opted to use a natural logarithm instead. I computed

$$\log\left(\frac{P(w \mid C_i)}{P(w \mid \sim C_i)}\right).$$

One nuance to this process is that it is actually much simpler if we add a smoothing term, which eliminates the difficulty of dividing by zero if a certain word does not appear at all outside a certain category. It also adjusts for those words which appear very few times: for example, both "benzene" and "acetylsalicylic" occur only in the Science category, but "benzene" occurs more often than "acetylsalicylic." With the smoothing term, "benzene" is ranked as a better classifier because it is apparently more closely associated with the category.

The best features will have the largest scores by this measure (the greatest magnitude, either negative or positive). Ordering the features by this mea-

sure, we can select the top thousand or so to use for each classifier.

3.2 Simple Classifier

The basic idea is to check whether a question has more words which are in category A and not category B, or more words which are in category B and not in category A. More formally, let C_i be the set of all words in category i. Then let $C_i \setminus C_j$ denote the set of all words in category i which are not also in category j. Let Q_k be the set of all words in question k. Then we examine $\left|Q_k \cap (C_i \setminus C_j)\right| - \left|Q_k \cap (C_j \setminus C_i)\right|$. If this quantity is positive, we classify question k as belonging to category i; if negative we classify the question as belonging to category j. If both sets have precisely the same cardinality, the classifier does not choose a category for the question.

We run this classifier for each pair of categories, and then each classifier's result is taking as a vote for or against a certain category. The category with the highest number of votes is chosen.

[This may be better than the {first approach} because there are likely more words separated each pair of categories than there are between one category and all the others.]

3.3 Naïve Bayes

The assumption behind the naïve Bayes model is that all the selected features are independent, i.e. each feature has the same probability regardless of the other features present.

I was not able to properly implement a naïve Bayes classifier in Python, and I am not entirely sure why not. Somehow the classifiers were not properly scaled, with the result that every question was being classified into the Other category. However, I did have some success with another classification scheme, based on my failed attempt at naïve Bayes.

After my failed attempt with Bayes I turned to the Java package Weka, which provides a number of utilities for data mining, automatic classification, and other wonderful things.[2] I used the package's `weka.classifiers.bayes.NaiveBayes`.

3.4 Modified Bayes

For my own version of this classifier I computed modified BNS scores for each word-category in the corpus. Then for each question to be classified, I averaged the BNS score for every word in the question, for each possible category. The category with the highest average modified BNS score was chosen.

I also attempted to compute a confidence level for each individual question's automatic classification, but this did not help very much. In order to raise the true positive rate to about 95% it was discarded about half the input.

3.5 Support Vector Machine

The Weka package also provided an SVM implementation. I used the package's `weka.classifiers.functions.SMO` with default values.

4 Testing

I intended to employ the testing strategy used in [DVDM01], in which the manually classified data is split into tenths, from which 90% forms the training corpus and the remaining 10% forms the testing corpus. If there was more time and computing resources, we could run ten trials (corresponding to the cyclic permutations of the tenths) so as to use all of the corpus for both training and testing, but not at the same time. The Weka package used the term "ten-fold cross-validation." However, my Python code for the modified Bayes classifier was so slow that I ended up running only one of the 90/10 tests.

5 Results

We can represent the resents from most of these classifiers as confusion matrices, where the rows represent the actual category and the columns represent the automatic classification. A perfect success would be a diagonal matrix.

5.1 Main category classification

5.1.1 Simple classifier

The results from the simple classifier on the first testing corpus are shown in Figure 1. As predicted, the Other category is a difficult one to classify. Only about 4% of the testing questions that were actually in the Other category were successfully identified.

5.1.2 Modified Bayes classifier

The results from the modified naïve Bayes classifier on the first testing corpus are shown in Figure 2. The overall accuracy is slightly better, but still not great.

5.2 Main- and sub-category classification

For kicks, I ran two of the default Weka classifiers: a naïve Bayes classifier and a SVM classifier. The comparison is shown in Table 2. Default feature selection was used, and due to memory constraints on my computer I was able to use only 1000 features. The poor performance of these classifiers is likely due to small feature space.

[2] `http://www.cs.waikato.ac.nz/ml/weka/`

	History	Literature	Science	Social Studies	Fine Arts	Other
History	192	16	0	11	0	0
Literature	8	231	1	9	1	0
Science	2	2	213	6	1	0
Social Studies	23	35	4	140	0	0
Fine Arts	5	25	0	3	84	0
Other	15	31	1	3	5	2

Figure 1: Simple classifier, first trial (testing corpus 0). Overall accuracy is 80.6% out of 1069 samples. Baseline accuracy would be 22.4% (if we simply assigned every question the most common category.)

	History	Literature	Science	Social Studies	Fine Arts	Other
History	199	3	0	3	1	13
Literature	1	224	1	5	3	16
Science	0	0	214	1	1	8
Social Studies	8	11	11	150	0	22
Fine Arts	0	4	0	0	107	6
Other	1	3	1	3	4	45

Figure 2: Modified naïve Bayes classifier, first trial (testing corpus 0). Overall accuracy is 87.8% out of 1069 samples.

	Naïve Bayes (ten-fold cross-validation)	SVM (single 90/10 split)
Accuracy	63.0%	64.2%

Table 2: Accuracy of Weka classifiers for category *and* subcategory. Baseline accuracy 11.6%.

6 Conclusion

Much work remains to be done to find the best method for the automatic classification of quiz bowl questions. If the Weka classifiers are any indication, an SVM is not much better than naïve Bayes for this problem. The other conclusion that can be made here is that my programming ability is somewhat lacking.

7 References

[DVDM01] Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. The form is the substance: Classification of genres in text. In *Proceedings of the ACL 2001 Workshop on Human Language Technology and Knowledge Management*, 2001.

[For08] George Forman. BNS feature scaling: An improved representation over TF·IDF for SVM text classification. In *ACM 17th Conference on Information and Knowledge Management*, 2008.

This paper represents my own work in accordance with University regulations.