Classifier Stacking and Voting for Text Filtering

Rada MIHALCEA

University of North Texas
Denton, Texas, 76203-1366
rada@cs.unt.edu

Abstract

This paper summarizes the approach and the results of the TextCat system participating in the Filtering track in the Text Retrieval Conference 2002. The system relies primarily on statistical methods, and was designed with the main purpose of having a backbone system in which we can further integrate semantic components, and evaluate their relative performance as compared to traditional statistical approaches. The system is therefore simple, and is based on techniques for keywords extraction, and various classifier combinations including stacking and voting. TextCat participated in the *Batch* and *Routing* tasks. In the *Batch* task, it achieved a score of 39.02% normalized utility, and 26.37% F-measure respectively, averaged over all topics. The averaged uninterpolated precision for our best routing submission was 14.16%.

1. Introduction

The Filtering track has quite a long history in the Text Retrieval Conference (TREC) series. The goal of the track is to measure the ability of systems to classify new documents as relevant or irrelevant with respect to a given topic. While there are three different tasks organized within the Filtering track – adaptive, batch, and routing – our *Text Cat*egorization (TextCat) system participated only in the last two tasks. Few changes in the system would have probably allowed us to run TextCat on the adaptive filtering data as well; however, we decided to focus on the classification capabilities of the system, rather than on its adaptability to new incoming data. This is mainly because the purpose for building TextCat was to have a backbone text classification system, in which we can further integrate semantic modules and evaluate their relative performance as compared to the simple statistical approach. This follows up on our previous work in semantic-based Information Retrieval (Mihalcea, 2002), where various degrees of semantic knowledge where integrated into an existing Information Retrieval system (SMART (Salton and Lesk, 1971)). To extend this work to the text classification problem, we needed in the first place a basic text categorization system, which could then be expanded with more sophisticated modules. Since we were not able to find such a tool (reliable, free for download, with complete source code), we started building our own text categorization system, which ultimately resulted in the *UNT TextCat* system.

2. UNT TextCat

As stated in the title, TextCat relies on combinations of simple text classifiers, which includes stacking and voting. Starting with a basic *ngram-based classifier* and a *rule-based classifier*, we generate a range of new classifiers by making simple changes in the value of their input parameters. First, stacking is done by applying the rule-based classifier on the output produced by the ngram-based classifier. Second, classifier voting is performed using various degrees of inter-classifier agreement. In turn, different voting schemes generate new classifiers.

For the *Batch* task, we had a total of thirteen stacked classifiers, which were then fed to the voting scheme, such that ten more combined classifiers were generated. Out of this total number of 23 classifiers, one was chosen according to its performance during cross validation runs performed on the training data. This tuning on training data was done separately for the normalized utility measure and for the F-measure, resulting in two different submissions, *UNTextCatSU* (run optimized for the T11SU measure), and *UNTextCatF* (run optimized for the T11F measure).

For the *Routing* task, we used a single combination of the thirteen stacked classifiers (run *UNTextCatR*), and a combination of the thirteen ngram-

based classifiers, with no prior stacking (run *UNTextCatR1*)

2.1. Data

This year, the text collection for the Filtering track consisted in the Reuters documents published during August 1996 - August 1997. To speed-up the classification process, and avoid the overhead associated with the repetition of some initial transformation procedures, there is a pre-processing phase where all documents are transformed and saved in a format suitable for the text classifiers. During this phase, words are stemmed using Porter stemmer (Porter, 1980), and common words are eliminated based on a list of about five hundreds words that comes with the SMART system package. The output of this stage is a single large training file that includes all training documents, one document per line, each line being preceded by the document identifier. Similarly, there is one large test file that includes all test documents. Even though the pre-processing procedure was designed for the specific format of Reuters documents, we expect to be able to easily adapt it to new data formats, with a minimal number of changes.

2.2. Ngram-Based Classifier

The first classifier consists of a simple ngram-based classification scheme. Specifically, we have a module that generates candidate ngram keywords starting with the training files. So far, the candidate keywords consisted only of unigrams and bigrams. In future work, we plan to investigate the impact of longer ngrams on the quality and efficiency of the TextCat classifier.

Shortly, the ngram-based classifier proceeds as follows. First, we select an initial large set of ngrams from the training files, based on their frequency in the documents considered relevant for the given topic. Next, for each such ngram, we calculate several parameters, including frequency in each relevant document, ratio between relevant and irrelevant documents that can be extracted with the ngram, correlation coefficient, number of relevant documents that include the ngram. A fixed number of ngram-*keywords* is selected from this large list, based on their relative value with respect to some threshold values set for their parameters. Finally, if more than a certain number of keywords fulfill the minimum requirements, the highest ranked ngrams are selected. This list of ngram-keywords is then adjusted through several loops, where the number of keywords may be increased or decreased, based on the total number of test documents that are extracted. This is based on the intuition that a fixed number of keywords may not be satisfactory for all topics. An efficient classification for a certain topic may be performed with only 5 keywords, whereas another topic may need as many as 15 keywords or even more. We also set a maximum over the number of loops that may be executed, to avoid excessive running times for certain topics. Finally, once the list of ngram-keywords is selected, all documents from the test set that contain any of these keyword are classified as relevant, and all remaining documents are classified as irrelevant.

There are several parameters that may influence the number and the ranking of the ngram keywords. Consequently, the settings made for these parameters may also influence the number of documents classified as relevant or irrelevant to the given topic. In TREC 2002, TextCat included thirteen different settings, which therefore resulted in thirteen different classifiers. The list below details the various parameters that may be set for the ngram-based classifier.

TOP_NGRAMS Maximum number of ngrams preselected from the training files. Final set of keywords is selected from this preliminary list.
Default value: 200

RATIO_RELEVANT_IRRELEVANT The ratio between relevant and irrelevant documents in the set of documents used for ngrams pre-selection.
Default value: 512

TOP_NGRAM_KEYWORDS Initial number of ngram-keywords that is extracted from the preliminary list of ngrams. This number is subsequently adjusted through several loops.
Default value: 10

MIN_FREQUENCY The minimum value acceptable for the frequency of a ngram in each relevant training document, for the ngram to be selected as a keyword.
Default value: 3

MIN_DOCSREL The minimum number of relevant documents that should include the ngram, such that the ngram is selected as a keyword.
Default value: 2

MIN_CORRELATION The minimum value acceptable for the correlation coefficient associated with each ngram, such that the ngram is selected as a keyword. The correlation coefficient was defined in (Ng et al., 1997):

$$\frac{(N_{r+}N_{n-} - N_{r-}N_{n+})\sqrt{N}}{\sqrt{(N_{r+}+N_{r-})(N_{n+}+N_{n-})(N_{r+}+N_{n+})(N_{r-}+N_{n-})}} \quad (1)$$

where N_{r+} (N_{n+}) is the number of relevant (irrelevant) documents containing the given ngram, and N_{r-} (N_{n-}) is the number of relevant (irrelevant) documents that do not contain the given ngram.
Default value: 0.02

MIN_DOCS Minimum number of documents to be extracted from the test files. This is the lower bound in the loop that selects keywords. If the number of documents extracted is lower than this threshold, than the number of selected keywords is increased, so that additional documents are extracted from the test set.
Default value: 1000

MAX_DOCS Maximum number of documents to be extracted from the test files. This is the upper bound in the loop that selects keywords. If the number of documents extracted is larger than this threshold, than the number of selected keywords is decreased, so that fewer documents are extracted from the test set.
Default value: 2000

MAX_LOOPS Maximum number of loops that can be executed for keyword selection. Starting with the initial set of TOP_NGRAM_KEYWORDS, keywords are added or removed from this set, until the number of documents that are extracted falls in the range of MIN_DOCS - MAX_DOCS. This loop will stop after is executed for MAX_LOOPS times, regardless of the number of documents retrieved.
Default value: 10

2.3. Rule-Based Classifier

The rule-based classifier used by TextCat is Ripper, an off the shelf system available from AT&T (Cohen, 1995). The reason for choosing Ripper as our rule-based classifier was twofold. First, it was previously shown that Ripper is an efficient classification scheme for text categorization problems (Cohen and Singer, 1996). Second, Ripper handles set-valued features, and therefore we can feed the entire document as a single attribute, and let the machine learning algorithm decide upon the contribution of various keywords for the relevance classification.

2.4. Classifiers Stacking

The first method that we employ for classifier combination is stacking, where the rule-based classifier is applied on the output produced by the ngram-based classifier. The documents considered relevant by the ngram-based classifier are therefore the only documents seen and classified by the rule-based learner. This stacking procedure is also meant as a speed-up in the classification of large text collections, since the rule-based learner does not handle very well collections that exceed a certain size. Systems participating in the filtering task had to deal with collections of over 800,000 documents, and consequently the single use of the rule-based learner was not a feasible solution.

2.5. Classifiers Voting

Besides stacking, additional classifier combinations are performed through a simple voting scheme, where the number of inter-classifier agreements is collected, starting with a set of thirteen base classifiers. Basically, for each document that is considered relevant by any of these base classifiers, we count the number of votes the document receives from the remaining classifiers. Next, a minimum acceptable value is set for this vote, and only those documents that have a total vote exceeding the given threshold remain in the final classification. There are ten different threshold values considered in the TREC 2002 experiments, ranging from 0 (meaning that at least one base classifier should find a document to be relevant,

for the document to be considered relevant in the final classification) to 10 (meaning that a document is classified as relevant only if at least ten of the base classifiers give a relevance vote to that particular document). These ten different threshold values resulted in ten additional classifiers. Hence, the final number of classifiers used in this task was 23, that is thirteen base classifiers, plus ten combined classifiers.

3. UNT TextCat at TREC 2002

In TREC 2002, TextCat participated in the *Batch* and *Routing*. All filtering systems were evaluated based on: (1) normalized utility measure, which relates the relevant and irrelevant documents in the set of retrieved documents (N_{r+} and N_{n+}); the normalized utility is intended as a measure of the number of irrelevant documents that a user can tolerate in a given set of retrieved documents; (2) the F-measure, which is a standard measure in Information Retrieval (Van Rijsbergen, 1979), and combines the precision and recall figures obtained for a certain topic. Precision and recall were also measured individually for each topic.

In the *Batch* task, TextCat achieved a score of 39.02% normalized utility, and 26.37% F-measure, averaged over all topics. The normalized precision for our best routing submission was 14.16%.

3.1. The *Batch* task

In the *Batch* task, thirteen base classifiers are built by varying the relative ratio between relevant and irrelevant training documents. The first classifier uses all training documents provided in the *QRels* judgment file, with their corresponding relevance judgments. The second classifier adds to this initial set of documents an equal share of irrelevant documents. The third, and all subsequent classifiers, double each time the number or irrelevant documents, up to the thirteen classifier, which trains on all N documents listed in the *QRels file*, plus an additional number of 2048*N irrelevant documents. As a rule of thumb, classifier i includes all N documents listed in the *QRels* file, plus an additional set of $2^{i-1} * N$ irrelevant documents, all of them extracted from the training collection. If not enough irrelevant documents are found in the training set, the ratio 2^{i-1} reflects a maximum, rather than the actual rapport between relevant and irrelevant documents. In all these classifiers, all parameters except the RATIO_RELEVANT_IRRELEVANT were used with their default values. New classifiers can be easily generated by changing the values of these input parameters.

Next, the rule-based classifier is combined with each of these initial classifiers through stacking. As noted earlier, ten additional classifiers are built by combining the thirteen base classifiers using a voting scheme.

To select one classifier out of the total set of 23 different classifiers, cross validation runs were performed on the initial training set, with the ratio between training and test documents set to 90%-10%. There were two TextCat submissions in the *Batch* task, one optimized for the utility measure – *UNTextCatSU* – where the classifier leading to the highest normalized utility score during cross validation runs is the one that is selected, and one optimized for the F-measure – *UNTextCatF*.

Table 1 lists the thirteen base classifiers and the ten combined classifiers, with their corresponding average utility and F-measure, as well as the averaged precision and recall. Moreover, the last two columns list the number of times each classifier was selected in the cross validation phase, for each run (optimized for normalized utility or for F-measure).

As seen in the last two columns in Table 1, the classifier selection is quite uniformly distributed among the 23 different classifiers; however, base classifiers one and thirteen seem to be selected more often, as compared with the selection frequency for the other classifiers. In terms of performance, base classifier thirteen has an utility measure of 33.87%, which is 5% smaller than the score for the final classifier; in real world applications, this relatively small difference may not fully justify the additional running time brought by the cross validation phase, and therefore in some applications this could be the only classifier employed for the filtering task. The maximum F-measure that can be achieved with a single classifier (base or combined, with no selection) is 18.28%.

Figure 3.1. plots the scores obtained by all classifiers for the four different measures. For the base classifiers, a steady growing tendency is observed for the utility measure, which suggests that the larger the

Classifier	T11SU	T11F	Prec.	Recall	Times selected (opt.SU)	(opt.F)
Base classifiers						
1 (N)	2.01%	2.13%	2.28%	29.53%	17	12
2 (N + N irrel.)	16.00%	7.34%	10.14%	13.05%	2	1
3 (N + 2N irrel.)	17.96%	10.26%	14.31%	13.06%	3	3
4 (N + 4N irrel.)	17.38%	12.30%	16.06%	14.25%	4	6
5 (N + 8N irrel.)	20.29%	12.62%	16.65%	13.52%	4	5
6 (N + 16N irrel.)	21.03%	13.92%	19.37%	14.33%	2	4
7 (N + 32N irrel.)	21.79%	15.02%	19.97%	14.12%	3	4
8 (N + 64N irrel.)	24.13%	15.58%	20.77%	13.86%	4	5
9 (N + 128N irrel.)	24.94%	15.89%	20.17%	14.25%	5	3
10 (N + 256N irrel.)	25.80%	16.00%	22.98%	13.35%	6	9
11 (N + 512N irrel.)	30.50%	18.08%	25.41%	13.93%	4	6
12 (N + 1024N irrel.)	31.89%	16.02%	23.12%	10.98%	0	2
13 (N + 2048N irrel.)	33.87%	14.82%	22.95%	9.89%	19	12
Combined classifiers						
1 (min.1 vote)	4.24%	7.42%	6.76%	34.93%	0	0
2 (min.2 votes)	13.47%	13.88%	14.50%	26.15%	2	1
3 (min.3 votes)	19.13%	16.47%	19.17%	22.14%	2	0
4 (min.4 votes)	24.45%	18.28%	24.30%	18.04%	0	0
5 (min.5 votes)	25.78%	17.12%	26.19%	14.01%	3	3
6 (min.6 votes)	25.80%	16.56%	27.12%	11.78%	4	6
7 (min.7 votes)	23.00%	14.91%	27.31%	9.93%	3	4
8 (min.8 votes)	20.46%	12.66%	26.83%	8.03%	4	2
9 (min.9 votes)	16.19%	9.64%	23.57%	6.27%	4	2
10 (min.10 votes)	13.95%	8.68%	22.52%	4.34%	5	3

Table 1: Individual results obtained with each base/combined classifier

number of irrelevant documents, the better. The recall is relatively constant across different base classifiers, with the only exception being classifier one, which has a significantly higher recall. Precision and F-measure achieve a maximum for base classifier eleven, followed by a decrease in classifiers twelve and thirteen. In the case of combined classifiers, there is a peak in utility, F-measure, and precision, for combined classifiers four through six, followed again by a sharp decrease. As expected, the highest recall is obtained with combined classifier one. What this figure suggests is that new system settings may eventually lead to even higher scores for various measures (e.g. larger number of irrelevant documents for higher utility score, larger number of base classifiers in the voting scheme for higher recall, etc.).

3.2. The *Routing* task

In the *Routing* task, two different TextCat runs were submitted. The first submission, *UNTextCatR*, consists in the top 1000 documents returned by the combined stacked classifier with a minimum vote of one (i.e. combined classifier one). The second submission, *UNTextCatR1*, consists in the top 1000 documents returned by a combined classifier, again with a minimum vote of one, but this time with no prior stacking (that is, only the ngram-based classifier is employed during this run). The average normalized precision was 14.16% for *UNTextCatR*, and 8.15% for *UNTextCatR1*.

4. Conclusions

This paper has described the approach and the results obtained with TextCat – a simple text filtering system that relies on various classifier combination

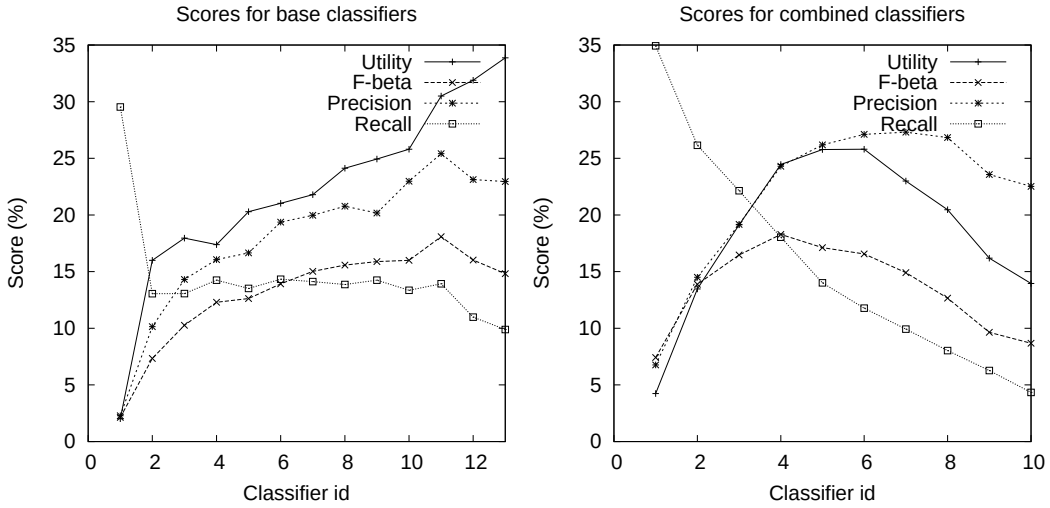

Figure 1: Utility, F-measure, Precision and Recall obtained with various classifiers

schemes. In the *Batch* task, the average utility score achieved with TextCat was 39.02%, and the average F-measure was 26.37%. In the *Routing* task, the two TextCat submissions achieved an average uninterpolated precision of 14.16% and 8.15% respectively. As a next step, we plan to integrate semantic modules into TextCat, and evaluate their relative performance as compared to the simple statistical-based approach.

5. References

W. Cohen and Y. Singer. 1996. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual International ACM SIGIR, Conference on Research and Development in Information Retrieval*, pages 307–315, Zurich, CH, July.

W. Cohen. 1995. Fast effective rule induction. In *International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July.

R. Mihalcea. 2002. Going beyond explicit knowledge for improved semantic based information retrieval. *International Journal on Tools with Artificial Intelligence*, 11(4), December.

H.T. Ng, W.B. Goh, and K.L. Low. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, PA, July.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

G. Salton and M.E. Lesk, 1971. *Computer evaluation of indexing and text processing*, pages 143–180. Prentice Hall, Ing. Englewood Cliffs, New Jersey.

C.J. Van Rijsbergen. 1979. *Information Retrieval*. London: Butterworths. available on-line at http://www.dcs.gla.ac.uk/ Keith/Preface.html.