Asymmetric Classification: Constructing Channels from Sources in Real-Time

Ron G. Katriel (rkatriel@acquiremedia.com)
Lawrence C. Rafsky (lrafsky@acquiremedia.com)

Acquire Media Corporation

Abstract

In this note we describe a system, developed under contract for major newspaper and newswire publishers (and currently deployed commercially), that constructs "topic channels" in real-time by simultaneously assigning one or more category codes to newspaper stories immediately upon publication and to newswire stories "on the run". The sources are diverse, but the category code taxonomy (developed by human domain experts) is unified. The system, named "Cogent" (**CO**din**G** **EN**gine **T**echnology), is asymmetric in the following two senses: (1) speed of classification is far more important than speed of training, and (2) precision is far more important than recall. The last two statements must be taken *in the extreme*: a failure to classify in under a few milliseconds, or the inclusion of an irrelevant story in a channel, are both considered system failures of the first magnitude, not mere "glitches". The approach is statistical, and the threshold-adjustment used to favor precision over recall has direct interpretation as a likelihood ratio. Novel aspects include a new feature selection algorithm that drastically reduces dimensionality, and the use of publisher-assigned metadata as features. Comparison with published results indicate that Cogent performs as well as the best available text categorizers for newswires but uses substantially fewer features and computational resources during classification.

1. Introduction

We wish to solve the following problem:

Immediately upon publication, classify articles from diverse newspapers and releases running on various newswires (hereafter we refer to both types of text as "stories") by assigning one or more categories from a pre-defined unified taxonomy (consisting in particular of *subject* and *industry* codes, but the exact nature of the taxonomy is immaterial for the research under discussion). Precision is far more important than recall, because instantly after classification the stories will be either:

- Used as input for automated stock-trading systems that depend on anticipating the investing public's reaction to news to achieve profitability,

or

- "Auto-posted" on a special purpose Web site that is dedicated to presenting a particular "channel" of information (corresponding to one of the assigned categories),

or potentially both.

In the case of automated stock trading, it is extremely damaging to present an incorrectly categorized story: to classify a story announcing a charity drive as one announcing a bankruptcy filing could trigger a

substantial loss of money. However, to miss a trading opportunity is of no great consequence – there will be another in a few minutes, and available capital limits the number of opportunities that can be pursued in any event. Precision trumps recall.

The case of auto-posting is somewhat more subtle. A special-purpose Web site is assured of getting all the stories devoted to its channel eventually, because it has editors that subscribe to all relevant sources, and all such sources will be read and reviewed in the normal course of editorial operations. Near-perfect recall is assured *over time.* But if auto-posting can produce a "scoop" – a story that breaks first for its dedicated channel readership, potentially before competing Web sites, then the auto-posting site increases its prestige, readership, profitability, etc. On no event, however, may an off-topic story be shown on the site. Auto-posters wish to both eat and have their cake – benefit from machine categorization only when it gives the "correct" answer. This can be achieved by "passing" on auto-posting except when one is "sure" the story is on point ... if there is doubt, let the auto-post candidate go by the boards and wait for editors to post it manually. In other words, for an auto-posting system to be of any use, precision must trump recall.

It is also clear that time-to-classify is of the essence ... the results are useful only if they are available substantially faster than those that could be produced by human editors. In the case of automated trading, this forces classification of a single newswire story to take no more than a few milliseconds, since we are competing with human reaction time and such stories are often short enough for a human to comprehend them in a single glance. In the case of newspapers and auto-posting there is no such stringent requirement, but realize that thousands of newspaper articles become available to the public electronically every day at just a few precise clock times (2 am Eastern, 3 am Eastern, 4 am Eastern, etc.), so that peak-load considerations again force lightening quick classification to insure that there are not long wait times for the "last" story to be categorized (without using expensive, massively parallel hardware).

The approach presented in this note assumes that a large collection of categorized stories is available for training. This is true for our application: commercial archival databases of newswire releases and newspaper articles expertly categorize the sources we are concerned with *after-the-fact* (i.e., not in real-time). Given such an oracle, our problem devolves to training a classifier to imitate in real-time other classifiers (human or human-assisted) that do not run in real-time, and to tune our classifier for extremely high precision.

The system we describe, Acquire Media's "Cogent" (**CO**din**G** **EN**gine **T**echnology), is in full production. It is the classifier of record for third-party syndication by ***Knight Ridder Digital*** (30 daily newspapers) and ***The New York Times Syndicate*** (11 daily newspapers or news services). It also classifies 6 real-time financial newswires for trading systems at a number of investment firms, including some of the world's largest.

The taxonomy we use consists of approximately 450 subject codes and 150 industry codes, arranged in a shallow (usually 2-level, but never more than 4-level deep) hierarchy. It is generally based on IPTC subjects and NAICS industries, with some modifications. Its exact content and structure are not of particular concern here. Codes are aggregated to from channels: approximately 20 subject channels (example: "real estate") and 30 industry (example: "software") channels. In general, any source may contribute to any channel. Of course a given story may be assigned to multiple channels.

There is an additional important consideration: the majority of the approximately 50 sources processed by Cogent arrive with some attempt at categorization undertaken by the original author or publisher. These attempts take the form of tags that we call "publisher topic metadata", and consist of categories from a publisher-specific (usually limited and not hierarchical) taxonomy, or a set of keywords. Publisher topic metadata, although different from publisher to publisher, can be mapped into nodes of our more comprehensive taxonomy in straight-forward ways by domain experts. We have in fact produced such a mapping, and it is trivial to apply it in real-time to all sources. But to simply "map and be done with it"

produces miserable results – certainly there is no way to tune for high precision. Classification errors by the publisher, or more usually an attempt to expand coverage by (in essence) weighting recall over precision, combined with imprecision in the mapping leads to cascading errors that are impossible to control. We take a different approach, which we believe is novel: we use the map to assign a "tentative category" in our taxonomy. This tentative category then serves as *just another feature* in our categorizer, along with words and phrases (see discussion below). The categorizer is free to pay attention to it or not – keeping it or removing it as the algorithm *and the tuning* sees fit. Furthermore, the potential inclusion of all (mapped) tentative categories as features – not just the one for the category under consideration – gives Cogent the chance to leverage information inherent in cross-correlations between non-independent categories without having to actually compute such correlations. Most classifiers do not consider cross-category inference, since doing so straight-forwardly essentially increases the computational burden by the square of the number of categories.

The publishers and institutions referred to above pay substantial fees to license Cogent, which we take as an indication that it in fact achieves its goal of high precision without reducing recall to the point of rendering the system useless. (It is of course possible to measure a-posteriori precision using an expert categorizer, but such measurements would tell us little since they would be derived from the same oracles used to estimate the classifier in the first place. These calculations would prove only that we did a good job in fitting our models of the oracles, which we know to be true by virtue of the estimation techniques presented below.) When we tune Cogent for optimality at the precision=recall point (as opposed to high precision), we can compare our results on public collections to published results; doing so leads us to believe that Cogent performs as well as the best available text categorizers (but uses substantially fewer features and computational resources.)

Cogent is based on the coordinated use of well-established statistical techniques which until recently were not practical due to their computational demands. In the rest of this note, we describe the algorithms and optimizations that allow us to take full advantage of the resources available on a moderately priced computer (e.g., a $5,000 Linux server).

2. The "Weight-Rate" Model

Given a predefined set of category codes that have been assigned by human editors to every article in a large corpus, what codes should an automated algorithm assign to a new, previously unseen article from the same source? Treating an article as a bag of words and assuming that the probability law governing the appearance of words is known – perhaps with parameters estimated from the coded corpus – the problem is relatively straightforward, assuming the various codes are independent of each other.

More concisely, for a given article with M words and for each code C, compute the ratio of the probability of observing these words assuming "assign C to the article" vs. assuming "do not assign C to the article". Then assign C if and only if this ratio exceeds a threshold chosen to minimize misclassification errors, weighting precision vs. recall as the application demands. (The amount by which the threshold is exceeded can be used to compute the "confidence" in this decision, although we do not explore this idea further in this note).

Bayes rule, likelihood ratio tests, linear discriminant analysis (Fisher or logistic regression) all come down to this same idea, under varying assumptions of what you know or are willing to assume about the probability laws in play. If word appearances are governed by a multinomial distribution, then the kernel of this ratio is just the weighted sum – a linear combination – of the observed word frequencies.

The battles over various approaches to machine learning (e.g., neural networks, classification trees), and document categorization (e.g., Salton's vector space model) seem to turn on how to estimate the weights. But in our view this is not the important problem. The real problems are elsewhere: what words do you use

(M gets much too big if you use all the words – you have to select your "features"), how do you define the counts (rates, truncated counts, etc.), and how do you manage the computational burden effectively (sparse matrix techniques will be needed, since most articles have very few words in common).

We therefore propose to simply assume that comparing linear combinations of feature occurrence rates to a threshold is the right answer, and we attack the estimation head-on without probability assumptions by using ordinary least-squares on the sample data, asking: "What weights produce minimal sample misclassification"? We call this technique – which dates back half-a-century to the seminal work of Mosteller [mosteller64] – "weight-rate". Back then computing power was so limited that the full potential of this approach could not be harnessed.

Word Rate Normalization

An important issue that needs to be addressed for the weight-rate model to work in practice is word rate normalization. Since the model counts the number of times a given word appears in a document, it is sensitive to variability in the length of documents. For example, a Dow Jones news feed article can range in length from 10 words (a headline-only story) to over 10,000 words (full-text of a press release).
One solution to this problem is to use binary features, that is, to note which words appear in a document but ignore the number of times each word appears in it. While this addresses the problem of variable-length documents, it throws away valuable information which can be used to improve results. A better solution is to truncate the articles to, say, the first 250 words (the first few paragraphs of a news story usually contain an accurate summary of the content of the document). We have found this to work well in practice and, in fact, the results are slightly better than using the entire article.

Limiting the length of articles is only a partial solution, however, since many articles start out shorter that the selected truncation point. We have found that additionally normalizing the rate of words with respect to the (trimmed) length of the article in which they appear yields noticeable improvements in performance. An alternative approach is to weight the contribution of each word by its position in the article, with words appearing near the beginning of the article getting very high weights and those further down receiving smaller and smaller weights. Our experiments show little gain from this approach, and we stick with truncation and normalization. Note that tentative category metadata is always included; metadata is not considered in the truncation step. Moreover, since by definition metadata appears only once, metadata rates are not normalized by trimmed story length; we up-weight their importance appropriately.

Phrase Recognition

Of course feature selection should not be limited to words. Phrases should also be used as features. The phrase "The Limited" is very specific, while the two words making up the phrase are not ("the" is in fact a "stop word" in some systems!). Other examples of important phrases in financial news are "first-quarter results", "labor agreement" , "economic recovery", etc.

Phrases can be extracted from documents using NLP techniques such as part-of-speech tagging and a variety of syntactic/semantic parsers. Unfortunately, these methods tend to be rather slow on the large collections used in our estimation step. The approach we use instead is to detect good candidate feature

phrases statistically using the same techniques that work for words, as described below. To keep the computational cost low, we limit phrases in length to no more than five words. Phrases are subject to trimming and weighting as in the previous section.

3. Feature Selection Techniques

Motivation

The statistical categorizer used by Cogent follows the weight-rate model described in Section 2. Each category is represented by a list of words, phrases, and/or tentative category metadata and their associated weights. Feature selection is the process of selecting a subset of all possible words, phrases, and tentative metadata for a given category. This is particularly important for computational efficiency and categorization accuracy:

1. **Computational Efficiency**: The time it takes to train Cogent on a particular category is proportional to the number of selected features. Since real-time systems need to be retrained occasionally (say, once a month) to keep up with changes in the news or subject vocabulary, there is a practical limit on the maximum number of features per category.
2. **Categorization Accuracy**: When the number of features exceeds a certain value based on the number of samples available for training, system performance on new text starts to deteriorate rapidly. This occurs because most words provide poor discriminatory power and only serve to introduce "noise" into the process. In the pattern recognition literature this is often referred to as the "curse of dimensionality".

Ideally, one would like to experimentally select the best subset of features out of all possible combinations. For example, suppose we somehow determine that 1,000 features (out of a total of, say, 100,000 possible features) is an optimal number to select. The combinatorics of selecting 1,000 out of 100,000 are enormous …enumerating all possibilities is not feasible.

A practical solution to this problem is to use a "greedy" algorithm which iteratively picks the next best feature from the pool of features not selected yet. This assumes an ordering of the features from best to worst. Obviously, there are many possible orderings based on different metrics. We have chosen Fisher's linear discriminant function as a first-order approximation to the full optimization we use on the selected feature set (described below). This is a statistically motivated solution that works well in practice and is extremely easy to compute … we use an approximation to cut the problem down to size, and then switch to exact optimization.

Fisher's Linear Discriminant

Fisher's linear discriminant could be used directly to categorize documents using features and rates. However, as will be seen in Section 5, there are better statistical techniques. Nonetheless, the method works well for feature selection and is computationally very efficient (its storage requirements and run time are proportional to the number of training samples).

We order features in decreasing value of their component weights in the full Fisher linear discriminant (assuming we actually computed it) and discard all words below a certain weight threshold. This is very easy computationally, involving essentially only a sort.

Optimal Number of Features

It stands to reason that the optimal number of features (in the general sense of the lowest application-specific error rate, or in our case the highest recall for a fixed (high) precision) should theoretically be some function of the number of samples available for training. More specifically, since the number of positive training samples (i.e., articles that belong to a particular category) is generally much smaller than the number of negative samples (i.e., articles that do not belong to the particular category), the former is the parameter of interest.

We did an experiment designed to quantify the relationship between the number of positive samples and the optimal number of features. This consisted of training our categorizer on 16 codes with sample sizes ranging from 12 to 22,424 and varying the number of features using powers of 2 starting at 8 and ending at 4,096. In each case, the features were selected using Fisher's linear discriminant as discussed above.

It turns out that the optimal number of features is proportional to the square root of the number of positive examples in the training sample (the constant of proportionality is determined experimentally). This rule-of-thumb has some statistical appeal: the square root of the sample size is proportional to the standard deviation of word occurrence rates under a variety of probabilistic word-generation models. Interestingly, our experience differs from the results reported in [lewis92], where the optimal number of features was found to be low (10 to 15) and independent of the sample size.

4. Approaches to Weight Computation

In this section we describe our approach to computing the word weights required for the weight-rate model. In theory this is straight-forward least-squares, but in practice care is required.

Least-Squares Minimization

Least-squares minimization, without probabilistic assumptions, is the problem of fitting N data points (stories previously classified in a corpus) to a model that has M adjustable parameters (feature weights) – see [press92], [duda00]. We need to solve the least-squares problem separately for each category. On paper, a closed-form solution can be found by forming the gradient and setting it equal zero. But in practice we are dealing with a matrix (the matrix elements are the frequency of feature J in document I, normalized as discussed above) that is very large (say, N = 100,000 and M = 2,000). To find a solution we need to combine subsampling, iteration, and sparse-matrix techniques. For a full discussion see [katriel02].

Observe that the result is the ***exact sample optimal linear feature weights*** for best separating the test corpus into "Category" and "Not Category". Fisher and other linear estimators based on distirbution assumptions do **not** produce this answer.

5. Cogent Results

Having estimated and stored the weight-rates for a given category code, it is now possible to compute the classification weight for that code in any given story with a trivial number of calculations, requiring only a few milliseconds, in real-time after the story arrives. Note that the parse required to enumerate the features found in a story (words, phrases, tentative categories) is re-used over and over for each code.

The resulting weight is then compared to a stored threshold. This threshold is a first-order approximation to the likelihood ratio test of "assign this category vs. do not assign this category", and can be set to achieve extremely high precision. If the overall categorization is well-fitted, the resulting recall will still be sufficient to allow the system to be useful.

So how good is Cogent? In this section, following [lewis95] and [lewis96], we describe an experiment using the F-measure, a 50-50 weighted combination of recall and precision, to facilitate benchmarking our results against those in the literature.

Practical Considerations

The experiment for a single category involves feature selection, computing the rate of occurrence of these features in each of the training stories, performing least-squares minimization on the resulting very large matrix of rates, running the categorizer with these estimated least-squares weights on a collection of "held-out" test stories (taken to be 25% of the corpus), and finally comparing the decisions the categorizer made against the codes in the corpus put on by human editors. As discussed in Section 1, such experiments will always produce optimistic recall and precision figures; the goal here is not to believe the actual values, but to compare them to similarly designed studies done by others.

Due to the large variability in the number of words selected by our feature selection algorithm (i.e., anywhere from 40 to 2,000), in some cases we used only a fraction of the stories available for training. We accomplished this by randomly sampling from the population of stories not coded with the category under consideration while using all the stories having the code. This was done since, in general, the number of stories with a particular code is a small fraction of the total number of stories (and one should not discard any of the latter.)

Dow Jones Sample

For the experiment we used 74,755 Dow Jones News Service (also known as the "Broad Tape" feed) stories covering a period of three months. The stories were partitioned into a training set (75%) and a test set (25%). Headline-only stories were discarded.

We selected 72 Dow Jones categories (referred to as "N Codes") with training frequencies of over 250 to test our system. This was out of a total of 179 subject codes present in our data set. The most frequent Dow Jones subject code is "Corporate Earnings Announcements" (N/ERN) with 12,341 training stories; the least frequent subject code is "Economic & Monetary Indicators, U.S." (N/EMI) with 254 training stories. The resulting collection of subject codes covers such diverse topics as government contracts, health, travel, taxes, labor and personnel issues, licensing agreements, and antitrust news.

We summarized the results of our categorization experiments using the F-measure on the test set. The macro averaged F-measure is 0.64 while the micro-averaged F-measure is 0.70. This compares favorably with Lewis' best results on the TREC-AP text categorization test collection (a somewhat similar corpus to ours) where the mean F-measure was 0.60 using 10,000 training documents and 0.72 using close to 150,000 training documents.

6. Bibliography

[duda00] New York, R. O. Duda, P. E. Hart and D. G. Stork, "Pattern Classification (2nd Edition)", John Wiley & Sons, A Wiley-Interscience Publication, 2000.

[katriel02] R. G. Katriel and L. C. Rafsky, "Cogent: A Scalable, Real-time Statistical Text Categorization", Technical Report, Acquire Media Corporation, Sep, 2002.

[lewis92] D. D. Lewis, "Feature Selection and Feature Extraction for Text", 212-217, Speech and Natural Language: Proceedings of a workshop held at Harriman, New York, Feb, 1992.

[lewis95] D. D. Lewis, "Evaluating and Optimizing Autonomous Text Classification", 246-254, The 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Jul, 1995.

[lewis96] D. D. Lewis and R. E. Schapire and J. P. Callan and R. Papka, "Training Algorithms for Linear Text Classifiers", 298-306, The 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Aug, 1996.

[mosteller64] Reading, Massachusetts, F. Mosteller and D. L. Wallace, "Inference and Disputed Authorship: The Federalist", Addison-Wesley Publishing Company, Inc., 1964.

[press92] Cambridge, Mass, W. H. Press and B. P. Flannery and S. A. Teukolsky and Vettering, "Numerical Recipes: The Art of Scientific Computing", Cambridge University Press, 1992.

About the authors: Ron Katriel is Head, A.I. Research, at Acquire Media. He holds a PhD from the University of Pennsylvania and an undergraduate degree from the Technion. He was formerly at IBM Research, Yorktown Heights. Lawrence Rafsky is CTO at Acquire Media. He holds a PhD in statistics from Yale University and an undergraduate degree from Princeton. He is a winner of the American Statistical Association Theory and Methods Award.

Discussion Points

71. What does the author mean by classification?

72. Does the research involve creation or implementation of a classification scheme?

73. How does the researcher use classification to improve the automated approach?

74. How do these methods compare to current human-generated approaches to classification?

75. How does the reported research expand our understanding of classification?

76. Does the research suggest an improvement over human-generated classification?

77. What do you think are the most important lessons learned in this research?

78. What do you think are the best practices reported in this research?

79. What would you recommend to the researcher as the next step in this approach?

80. Is there other related research that you would recommend the researcher become acquainted with?