Automatic Feature Induction for Text Classification

Jason D. M. Rennie and Tommi Jaakkola

The Problem: All classifiers require a set of features that can be used to distinguish between different examples. In some cases, such as determining whether a chess position is a winning position, the features are clear (the positions of the chess pieces). In other cases, such as text, they are less clear. A document is simply a string of characters. Standard practice dictates that documents should be broken into words, common, uninformative words should be removed and a stemmer is to be applied for removal of common endings (such as "-s" and "-ing"). However, this processing is insufficient for many languages and many modern-day classification problems. For example, documents about the XWindow system include an unusually large number of words that begin with the letter "x," but traditional processing obscures this fact. Unsolicitied commercial e-mail ("spam") can often be identified by the use of foreign languages, a disproportionate number of exclamation points or capital letters. Again, such information will often be discarded.

Motivation: A fundamental evaluation of a system's understanding of a language is its ability to compress examples of that language. A system with no understanding of the English language (such as most computer compression programs) will compress based on statistical regularities in character occurrences and short words, such as common pairs of letters and short words such as "to" or "and." In doing so, much information about the English language is not compressed, such as sentence structure and capitalization rules. Similarly, we can evaluate a system's classification ability by the number of bits it needs to encode classification labels. Formally, let $D = \{d_1, d_2, \ldots, d_n\}$ be a vector of documents, let $L = \{l_1, l_2, \ldots, l_n\}$ be an associated vector of labels and let $s = g(D, L)$ be a string that is a function of D and L where $f(D, s) = L$. A compression algorithm is specified via f and g. The objective is to minimize the expected length of s. The string s is the information in addition to the set of documents needed to exactly predict the vector of labels. Its length is the amount of unexpected information, so it measures the degree to which the relation between the documents and labels is understood.

This compression framework has many advantages. In priciple, it is simple and well-understood. As a result, it may be applied to a wide range of problems. Though we use it to tackle the problem of classification, the principle of compression can be used in problems ranging from text summarization to clustering and even question answering.

Previous Work: Most relevant to our work is Carl de Marcken's work on language learning [1]. As do we, he poses it as a compression problem. A historical focus of the compression community has been speed and memory efficiency. De Marcken instead focuses on maximizing compression. As a result, his algorithm would not be a practical replacement for a computer compression utility, but he yields the best compression rates on English text corpuses and achieves a representation that segments documents in an intuitively pleasing way. For example, his algorithm separates out common prefixes and suffixes (such as "mis-" and "-ing") and joins words that are commonly found in phrases (such as "National Football League"). Our work extends his to the problem of text classification and builds the foundations necessary for extending it to other problems. Other work on inducing features for text classification includes Slonim, et. al.'s work on Variable Memory Markov Models [4], Lodhi et. al.'s work on string kernels [3] and Kudenko and Hirsh's work on feature generation [2].

Approach: We consider classification in the context of compression. Using the notation introduced earlier, we focus on learning f, specifically, the part of f that gleans information from the document set D. Any classification algorithm can be seen in this light, as a function of the document set, D that provides information about the label set, L. The string s simply bridges the gap between the information extracted from D and the correct set of labels. In this way, s judges the classification error, or the degree to which the algorithm cannot predict the labels. It also acts as a regularizer, since it must specify how label information is to be extracted from the documents.

We use as our representation a decision list, where each rule specifies a token and a label. If that token is found in a document, the rule makes a vote for a corresponding label. Part of the string s must be dedicated to the encoding of these rules. We assume a fixed-width encoding for tokens and labels, where each token uses tokSize bits and each label takes $\log m$ bits, where m is the number of

- Let $\text{tokSet} = \{""\}$.
- Let $\text{bestLen} = n \log m$.
- While $\text{minEncodingLength}(\text{tokSet}) < \text{bestLen}$
  - Expand tokSet by adding all strings that can be formed by appending a character to an element of tokSet.
  - Let bestLen be the shortest encoding achieved by adding a rule using one element of tokSet.
  - Delete any $\text{tok} \in \text{tokSet}$ where no token with prefix tok can achieve the encoding length bestLen.
- Choose $\text{tok} \in \text{tokSet}$ that yields the smallest encoding length.

Figure 1: An algorithm for feature induction.

different labels. With this, we can immediately put bounds on the impact of rules. Without any rules, s simply lists the labels, consuming $n \log m$ bits of information. Consider a decision list consisting of a single rule. Assume that rule applies to i documents, j of which it correctly labels. Then, we still encode $n - i$ labels directly. For each of the i documents to which the rule applies, we use one bit to specify whether or not the labeling is correct. We use a further $\log m$ bits to encode the label for each document that is misclassified by the rule. Hence, the single rule gives us an encoding length of $(n-i)\log m + \text{tokSize} + i + (i-j)\log m$. The rule compresses the encoding of the labels iff

$$\text{tokSize} + i < j \log m.$$

We propose to learn features by greedily finding rules that reduce the encoding length. For each rule, we find the token that yields the greatest encoding reduction. We search the entire space of tokens, but do so in an efficient manner by bounding the reduction that a token may achieve. Figure 1 provides an algorithm. The function minEncodingLength provides a bound on the encoding length by assuming $i = j$, that a string with the token as a substring can be found in each document correctly labeled but in no documents incorrectly labeled. This is clearly a lower bound on the true encoding length.

Impact: We use compression as an object for learning features in text. This provides additional flexibility over currently popular techniques because it does not require a set pre-processing scheme. The compression objective automatically determines those strings in the text that are useful for predicting labels. As a result, improved classification may be achieved by finding features that standard pre-processing ignores. It also provides promise for a system that is language independent since some languages, like Japanese, do not syntactically separate words.

Future Work: Our current implementation is limited in multiple ways: 1) it only matches exact strings, 2) it ignores frequency information and 3) it uses a greedy rule-based strategy to learn the relation between documents and labels. We plan to relax these restrictions by introducing regular expressions and feature counts and using alternate classification schemes. We are also interested in the extension of this work to other problems. Compression is a powerful idea that may be applied in many different ways. For example, if we replace document labels with document summaries, compression provides an objective for learning what aspects of a document should be included in a summarization.

Research Support: Jason and Tommi acknowledge support from Nippon Telegraph and Telephone Corporation and NSF-ITR Grant IIS-0085836.

References:

[1] Carl G. de Marcken. *Unsupervised Language Acquisition*. PhD thesis, Massachusetts Institute of Technology, 1996.

[2] Daniel Kudenko and Haym Hirsh. Feature generation for sequence categorization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[3] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. Text classification using string kernels. In *Advances in Neural Information Processing Systems 13*, 2001.

[4] Noam Slonim, Gill Bejerano, Shai Fine, and Naftali Tishby. Discriminative feature selection via multiclass variable memory markov model. In *Machine Learning: Proceedings of the Nineteenth International Conference*, 2002.