Feature Selection tool

Yaroslav Nechaev

May 31, 2014

1 Introduction

Feature selection has become very important step in Natural Language Processing due to large amount of features that usually can be found in the datasets of the NLP-related problems. This is happening because it's typical to represent a piece of text as a feature vector, where each feature measures the "importance" of the corresponding term in the text (using various term frequency metrics). With a large amount of features we want to reduce their quantity by discarding irrelevant or redundant features. By selecting only relevant features we provide three main benefits for future analysis: improved model interpretability, shorter training times and reduced overfitting [1].

In this essay I will briefly review some of the related work on Feature Selection and describe main features of my tool.

2 Related Work

One of the landmark systems in machine learning and data mining is The Waikato Environment for Knowledge Analysis (WEKA) [2]. The WEKA provide a comprehensive collection of machine learning algorithms and related tools including various feature selection techniques. The library is open sourced and is available under the CC BY-SA 2.5 license.

Another major one is the Feature Selection Toolbox 3 [3] by UTIA which is a standalone C++ library for feature selection. It contains wrappers, filters, hybrid methods, specialized methods for very high dimensional problems. Library is free for non-commercial use.

There is also a number of smaller tools can be found. One of them is Y.-W. Chen's et al. [4] tool implementing a simple feature selection technique F-score. Zheng Zhao et al. [5] maintain the repository containing 13 different algorithms and sample datasets for feature selection. Each algorithm is rated and performance report is available. Hanchuan Peng et al. [6] researched applications of mRMR method related to gene selection.

3 Technical Description

Feature selection algorithms generally fall into one of three categories: wrappers, filters and embedded methods [1]. My tool implements two common filter algorithms: pointwise mutual information (PMI) and Pearson's chi-squared test ($\chi 2$). The application takes a LIBSVM file as an input, then ranks features based on their score and outputs the desired amount of the top scored features. Note that the tool assumes that features are binary: it assigns 1 if feature is present in the sample and 0 if it's not.

The application is written in Objective-C 2.0 with Automatic Reference Counting (ARC). Grand Central Dispatch is used for multi-threading. It will automatically select the best number of processes to handle the task.

3.1 PMI

Pointwise mutual information [7] is widely used to determine association between two discrete random variables. It is defined like this:

$$PMI(x, y) = \log \frac{p(x, y)}{p(x)p(y)} \tag{1}$$

PMI is ranged as follows:

$$-\infty \leq PMI(x,y) \leq min\left[-\log p(x), \log p(y)\right] \tag{2}$$

where PMI equals to zero if variables are independent, lower bound for never occuring together and upper bound for complete co-occurence. For our feature selection task, when we select only the most relevant features, we want this metric to be as far as possible from zero, meaning that the feature and the class are associated in some way.

The tool use this metric to score features against every class c, normalizes the result so it would be in $[-1, 1]$, then summarize the score:

$$s_i = \sum_{c \in C} \left| \frac{PMI(f_i, c)}{\log p(f_i, c)} \right| \tag{3}$$

where s_i is the score for feature f_i.

3.2 χ^2 test

Pearson's chi-squared test [8] is a statistical test to evaluate whether paired observations on two variables are independent of each other. For example, we could test if some words appears in texts on certain topic more likely than on the other topics, or we could check if the certain diet increases the likelihood of the certain illness. The value of the test statistic is:

$$X^2 = \sum_{i,j} \frac{\left(O_{ij} - E_{ij}\right)^2}{E_{ij}} \tag{4}$$

Where values O_{ij} are observed values from the contingency table [9], E_{ij} are expected values. For feature selection task we have two variables — particular feature with two outcomes (feature is in sample or it is not) and class. This way we have $2 \times |C|$ contingency table (see Table 1). The tool constructs this table and calculates the metric.

	C_1	C_2	C_3	Total
f_i	4	5	6	15
$\neg f_i$	7	8	9	24
Total	11	13	15	39

Table 1: Example of contingency table for 3 classes

3.3 Code organization

Code consists of main file (*main.m*) and four classes (*ChiSquaredEvaluator*, *Evaluator*, *PMIEvaluator*, *DatasetCollector*). Main file contains all IO logic and also parses input in LIBSVM format.

DatasetCollector is a helper class used for iterative assembling of the sample info into the NSDictionary.

Evaluator is an abstract class with the main metric calculation routines, including the multi-threading support. Other two classes (*∗Evaluator*) extend *Evaluator* class and implement particular method. This method takes a feature info with some general precomputed statistics about the original data as an input and outputs the score of the particular feature.

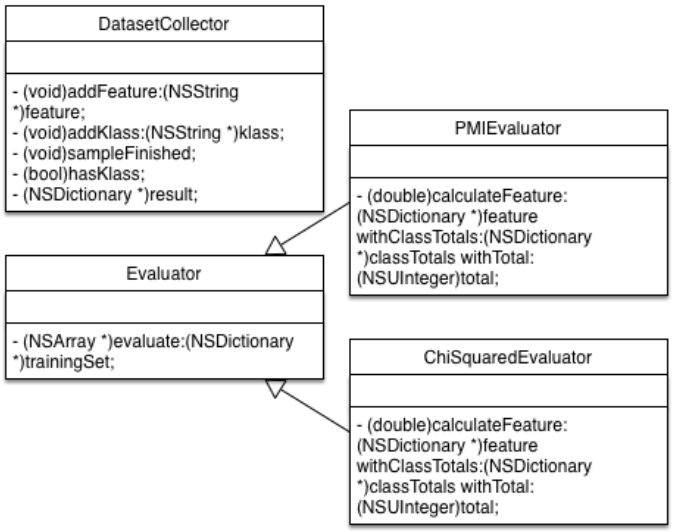

Figure 1: Code diagram

4 References

[1] Wikipedia. Feature selection, 2014.

[2] Geoffrey Holmes Bernhard Pfahringer Peter Reutemann Ian H. Witten Mark Hall, Eibe Frank. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[3] Feature selection toolbox 3, 2014.

[4] Chih-Jen Lin Yi-Wei Chen. Combining svms with various feature selection strategies.

[5] Shashvata Sharma Salem Alelyani Aneeth Anand Huan Liu Zheng Zhao, Fred Morstatter. Asu feature selection repository.

[6] Chris Ding Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 2005.

[7] Wikipedia. Pointwise mutual information, 2014.

[8] Wikipedia. Pearson's chi-squared test, 2014.

[9] Wikipedia. Contingency table, 2014.

[10] Hinrich Schütze Christopher D. Manning. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.