Self-optimizing, n-gram based support vector machine text classification system. A simple tutorial for non-programmers.

Abeed Sarker
Division of Informatics
Department of Biostatistics, Epidemiology and Informatics
Perelman School of Medicine
University of Pennsylvania
abeed@pennmedicine.upenn.edu

Abstract

This document outlines the design and usage of a set of scripts for performing text classification. The scripts are intentionally kept simple and they only use n-gram features with minimal preprocessing. The scripts implement a support vector machine classifier, and, using only n-gram features, compute some key parameters that lead to good classification performance. The scripts were developed as part of a workshop for training biomedical informatics students with no background in machine learning, and no or limited background in programming.

Keywords: text classification, support vector machines.
Version: 3.1.
Date for current version: 04/03/2018
Date for first version: 03/03/2015

1 Introduction

Text classification is one of the most fundamental natural language processing tasks, and involves the categorization of texts based on their lexical contents. In its simplest form, text classification is binary in nature, such as the categorization of spam *vs.* non-spam email (Youn and McLeod, 2007). Researchers from distinct fields are exposed to a wide range of text classification problems. Even within a specific domain, such as the medical domain, there are a variety of text classification tasks and problems, such as assessing the qualities of published papers (Kilicoglu et al., 2009), outcome polarity classification (Sarker et al., 2011), biomarker classification (Davis et al., 2015), and adverse drug reaction mention detection (Sarker and Gonzalez, 2015) to name a few. Early automated text classification systems were rule-based in nature mostly because of the absence of annotated data (*e.g.*, Sarker and Molla-Aliod (2010)). However, such rule-based systems are generally limited in terms of performance and/or overfit to the target problem. With the rapid increase in text-based data in all domains (*e.g.*, the largest medical database, Medline,[1] now indexes over 27 million articles), most efficient text classification systems now use machine learning. Such systems utilize annotated data, and automatically extract features from large volumes of annotated text to perform classification. The rules that are used in classification are not hard-coded or predetermined, but are learned from the annotated data automatically. Text classification for various tasks has been thoroughly explored in the literature, and relatively recent

[1] https://www.nlm.nih.gov/bsd/pmresources.html. Accessed: 05/03/2018

progress has seen the use of techniques such as distant supervision (Mintz et al., 2009). However, recent progresses are very much outside the scope of this workshop paper, so we now move on to the technical design of our algorithm.

2 Methods

2.1 Support Vector Machines

Support Vector Machines (SVMs) (Vapnik, 2000) have been used for text classification tasks with significant success. They are particularly useful in text classification as they are capable of handling large feature vectors (Joachims, 1997). However, to optimize the performance of SVM classifiers, several parameters have to be tuned, as explained by Chang and Ling (2011). Because of their success in text classification applications, we also use SVMs for this text classification script. We use the python `sklearn` package to implement our classifier. We provide the technical details of the implementation in the following subsections.

2.2 Prerequisites

The implentation is in python (2.7.*). In particular, the following python libraries are required:

- `sklearn` – for performing the learning.
- `numpy` – for mathematical operations.
- `nltk` – for stemming and other natural language processing operations.
- `pandas` – for file loading and representations.

These libraries are available with the python anaconda distribution.[2] Note, if the stemmer gives errors, the specific `nltk` modules probably need to be downloaded. To download the modules, run python and use:

```
import nltk
nltk.download()
```

2.3 Technical Specifics

The current format assumes that the input is provided as a 3-column, tab separated file. The first column contains the `ID` for each instance, the second column contains the `CLASS`, and the third column contains the `TEXT`.

There are 3 python scripts for the classier. `main.py` calls runcrossvalidation.py to perform 10-fold cross validation using the supplied training set. Optimal values for C parameter and weights (which are particularly important when there is high imbalance among the classes) are learned from this script through iterations of the 10-fold cross validations. Once these parameter values are learned (the parameter values are stored in best `params.txt`), the `runclassifier.py` script classifies the test set and prints them out to a file (`classificationresults.txt`). The scripts can be run independently, as well as through main.py. Two run commands:

[2]Available at: https://www.continuum.io/downloads. Last accessed: 1/10/2018.

```
python main.py [trainingfilename] [testfilename]

OR

python main.py [trainingfilename]
```

Not specifying a test file simply performs the 10-fold cross validation on the training set and writes the results down.

3 Discussion

The results obtained by this classification system are not optimal. Current state-of-the-art in text classification involves the use of informative features and the use of feature engineering techniques for improving performance. However, based on our experimentation, this n-gram based approach is likely achieve close to optimal performance.

Files of different formats can also be easily used. Within the function `loadDataAsDataFrame()` (shown below), the `items[*]` need to be changed to represent the appropriate column numbers, starting from 0.

```
def loadDataAsDataFrame(f_path):
    '''
        Given a path, loads a data set and puts it into a dataframe
    '''
    datalist = []
    count = 0
    infile = codecs.open(f_path,'r')
    for line in infile:
        instance_dict = {}
        items = line.split('\t')
        instance_dict['id1'] = items[1]
        instance_dict['id2'] = items[2]
        instance_dict['class'] = strip(items[6])
        instance_dict['text'] = items[4].decode('iso-8859-1').encode('utf8')
        instance_dict['drug'] = items[3]
        instance_dict['date'] = items[5]
        datalist.append(instance_dict)
        count+=1

    return pd.DataFrame(datalist)
```

References

CC Chang and CJ Lin. 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ ˜cjlin/libsvm.

J Davis, M Maes, A Andreazza, JJ McGrath, SJ Tye, and M Berk. 2015. Towards a classification of biomarkers of neuropsychiatric disease: from encompass to compass. Molecular Psychiatry, 2015(2):152–153, Feb.

T Joachims. 1997. Text categorization with support vector machines. Technical report, University of Dortmund, ftp://ftp-ai.informatik.uni-dortmund.de/pub/Reports/report23.ps.Z.

H Kilicoglu, D Demner-Fushman, TC Rindflesch, NL Wilczynski, and R Brian Haynes. 2009. Towards Automatic Recognition of Scientifically Rigorous Clinical Research Evidence. Journal of the American Medical Informatics Association, 16(1):25–31, Jan-Feb.

M Mintz, S Bills, R Snow, and D Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL'09, pages 1003– 1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

A Sarker and G Gonzalez. 2015. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. Journal of Biomedical Informatics, 53:196–207, Nov.

A Sarker and D Molla. 2010. A rule-based approach for automatic identification of publication types of medical papers. In Proceedings of the 15th Australasian Document Computing Symposium.

A Sarker, D Molla, and C Paris. 2011. Outcome Polarity Identification of Medical Papers. In Proceedings of the Australasian Language Technology Association Workshop, Australasian Language Technology Association, pages 105–114.

V Vapnik. 2000. The Nature of Statistical Learning. Springer-Verlag New York.

S Youn and D McLeod, 2007. A Comparative Study for Email Classification, pages 387–391. Springer Netherlands, Dordrecht.