CSC 594 – Topics in AI – Text Mining and Analytics – DePaul University
Professor Noriko Tomuro
Student: Ricardo Barros Lourenço

Final Project Report: Extension of Climate and Sentiment Analysis

Introduction

This report is result of the work extenden from midterm, also being a compliment on the project proposal presented earlier in the quarter. On this extension it was moved from a classifier of tweets, using regular expressions(regex) in a bag-of-words model, which the target was to evaluate these tweets in terms of sentiment, time and weather characteristics, to a regression analysis approach, using Ridge regression, and also TF-IDF hashing of tweets.

As a benchmark, we would compare the results to the midterm reference scores and also to Kaggle, as a baseline.

This report just cover the analysis of this development process. Algorithm and processing results are enclosed on a separate document.

Implementation

The extended implementation was targeted on predicting targets such as sentiment, time and weather characteristics, as expressed on several categories. For this implementation it was used several libraries, such as: pandas, nltk, sklearn and multiprocessing.

This approach involves some data treatment. For missing values, it was

chosen to drop all records containing missing values. The kaggle dataset represents a sufficient large corpus of text. The final dataset contains more than 50,000 records, which is somehow representative.

After that, another step was on data normalization and binning. The initial dataset contains, aside of tweets, several fields regarding on sentiment, time and weather categories. On that dataset, these are scores attributed by humans for each tweet, manually. On the kaggle website there is not much description on how this was measured, so it was decided to standardize those measurements, for benchmark comparison.

Considering that each field has different scaling, all fields were normalized using scikit learn MinMaxScaler, which applies a Min Max normalization, giving on all outputs floats that have range between zero and one. After that, those values were rounded to the closer integer, on a manner to get those variables on a binary form, which is useful, when needing to compare with regex results, that are generated on the presence or absence its regarding regular expression.

Regarding to the text preprocessing, each tweet was treated in terms of lowercasing, and on punctuation removal. I did not tokenize or stem the words, in terms of a initial benchmark. Also it was a problem when using scikit-learn TF-IDF, because it was generating a list that doesn't have *lower* attribute. Perhaps on a future revision, this would be done.

TF-IDF was done using *TfidfVectorizer* method, which generates a sparse attribute matrix with features reaching more than 4000 attributes. Considering that we peocessed almost 57000 tweets, it is not a small matrix.

After this, the data is submitted to regression analysis. The chosen routine is a scikit-learn *RidgeCV* module, which implements Ridge regression with n-fold cross-validation. On this program we also done random splitting for the targets and the TF-IDF matrix into training and testing partitions, being the test partition 10% of the data, an absolute value of 5000 tweet attributes and targets. For the cross-validation parameter it was chosen 10-fold cross validation with the goal of reaching model stability.

This was consecutively done for each target, being analyzed one per run. Each one doing random splitting, and performing regression analysis using 10-fold cross-validation, 24 times (the number of targets). After each model run, the predicted array is compared with the control labels, for training and testing, in terms of cosine similarity, and also on mean square errors.

Classification Results and Discussion

This new approach has reached better results, rather than using a bag-of-words approach. Not all targets were scored high, but the majority scored above the previous mean Midterm result of 0.31 (but having high values on misclassification matrix).

The testing scores are (values in green better than 0.31, yellow between 0.2 and 0.3 and red below 0.2) :

Atribute	Cosine Similarity	MSE
s1	0.07050809	0.00953531775287
s2	0.45649852	0.172600965991
s3	0.50172722	0.198293383659

s4	0.46448993	0.17751538096
s5	0.37629825	0.127938261081
w1	0.86051332	0.193434004848
w2	0.25605018	0.0710683285426
w3	0.30030455	0.0888833241971
w4	0.20712021	0.0461987779495
k1	0.12285882	0.0222598876411
k2	0.27806414	0.0810590196241
k3	0.02143839	0.00281965874134
k4	0.33818397	0.107129104
k5	0.19045296	0.0419778288215
k6	0.00817133	0.00209758942512
k7	0.52128612	0.20417983961
k8	0.03352883	0.00278145856612
k9	0.14905313	0.0241954254966
k10	0.30566032	0.0913343984262
k11	0.15360364	0.0350430510763
k12	0.3757311	0.128014170088
k13	0.37671485	0.130278352943
k14	0.11212666	0.0208136372975
k15	0.2118193	0.0532943580958

The classification results were very good, when comparing to the kaggle 2013 leaderboard has obtained scores around 0.14, using the Kaggle metric. The previous regex approach has already beat that score, but this new approach has enabled more stability on certain groups of scales such as sentiment perception (s*x*) and when (w*x*), which on its majorities has kept or

improved the standards when comparing to the previous benchmark. Even on the weather (k*x*) group some improvements were made.

Some criticism could be made on higher MSE values pointed in yellow, which are higher than the baseline, and have some relation with higher scores of improvement on cosine similarity.

As possible improvements, is trying to apply better preprocessing, trying to use tokenization and stemming, and if possible POS tagging. Initially I intended to work with this approach, but filtering the tags with NLTK is a tricky measure.

Another good room for improvement rely on the TF-IDF tuning parameters, which include dictionary building (or even providing one). Also on the resultant sparse matrix is possible to apply PCA for both dimensions, and rebuild it from both factor groups, in a Matrix Factorization algorithm, probably eliminating much of the noise on it.

Another is trying to use other regression methods, perhaps LARS regression, due to the high number of factors on the TF-IDF matrix.