Machine Learning 2013: Project 3 - Text Classification Report

anufer@student.ethz.ch
elmerl@student.ethz.ch
nivo@student.ethz.ch

December 23, 2013

Experimental Protocol

Usage:
Download the csv files to /data/3/....csv (... = training, testing, validation)
Run map.m
Results are in /data/3/....out (... = training, testing, validation)

1 Tools

We used .NET to do the pre-processing (Create word bag, assign entries to word list, fix input data, etc) and Mathlab to do the classification.

- Visual Studio (C#, LINQ) (code is in /code/3_map/preprocessing/external_tools/ML-3 directory)
- Matlab (code is in /code/ directory)
- Git / Github Repository [1]

2 Algorithm

Principal component analysis

Describe the algorithm you used for classification.

3 Features

To group similar words together, reprocessing was used. First, we used the Levinsthein distance[2], but then switched to a slightly modified version of the edit distance[3], which gave slightly better results. Also,

[1] https://github.com/lukaselmer/ethz-machine-learning
[2] https://en.wikipedia.org/wiki/Levenshtein_distance
[3] https://en.wikipedia.org/wiki/Edit_distance

instead of using an absolute value threshold for the distance, we used a ratio of about 75% to group similar words together. For this preprocessing, we used C#, because string handling in C# seemed much easier than in Matlab.

After the preprocessing, we then used Matlab to predict the city codes and country codes. For this, we used PCA.

In our implementation, the pre-processing step consists of the preparation of the input data, the selection of the words to put in the *bag of words* of the MAP and the mapping of the input data to a mapping to the selected words.

To prepare the data, we got a list of the top 400 words in the entire training set and we replaced every other word that is similar enough to one of this top word. The similarity is calculated by the edit distance of two words divided by the length of the longer of the two words. Two words are considered similar if the similarity is greater than 0.75.

To fill the bag of words we group the training data by classifier and count the occurrence of each word. We take the top 25 words for each classifier if they appear in at least 10% of the training entries of this classifier.

To be usable for the MAP algorithm, each data entry is then mapped to a vector v with $v_i \in \{0, 1\}$. $v_i = 1$ indicates that a certain word i from the bag occurred in the data entry and $v_i = 0$ it doesn't.

4 Parameters

To find the parameters, we used manual testing. The most important feature seemed to be the edit distance ratio, which is 75%. Another important parameter is the amount of top words which are picked at the start of the algorithm. These are the words, which are in a category of it's own before grouping them together.

5 Lessons Learned

Many other tools and algorithms have been tried. However, one we didn't try and might have worked well is SVM.