5-10 pages

Niclas

David

**Problem**

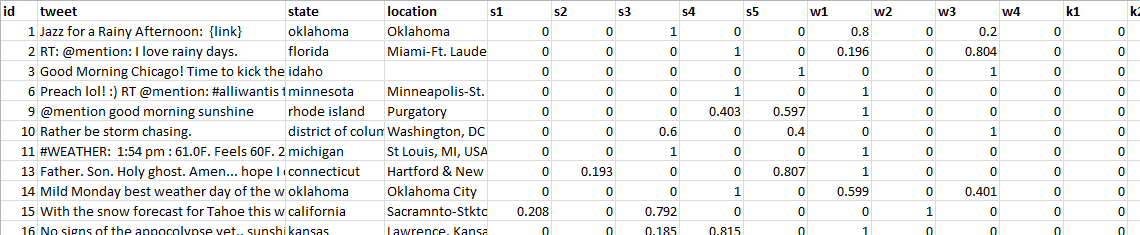
Our project is inspired by the Kaggle-competition “Partly Sunny with a Chance of Hashtags"[[1]](#footnote-1). The task is to classify tweets talking about the weather in time, sentiment and kind of weather (“cloudy”, “sunny” etc.). For simplicity, we will focus only on kind of weather. This choice is sensible as we are provided with a dataset containing only tweets concerning weather. For temporal or sentiment classification it would be more natural to use a general dataset.

Our motivation for the choice of this problem stems from our interest in Natural Language Processing. Furthermore, we decided to implement a Naïve Bayes Classifier ourselves instead of using a toolkit because this helps us to understand it more thoroughly.

**Data**

The data we will use are annotated tweets obtained from Kaggle, which only contain tweets from people talking about the weather.

The data has the following format (tweet ID, text, location):



**Predictions**

“kind of weather”-task requires to label the tweets with the following classes:

k1,"clouds"

k2,"cold"

k3,"dry"

k4,"hot"

k5,"humid"

k6,"hurricane"

k7,"I can't tell"

k8,"ice"

k9,"other"

k10,"rain"

k11,"snow"

k12,"storms"

k13,"sun"

k14,"tornado"

k15,"wind"

Multiple classes for one tweet are possible. Kaggle requires not to upload a classification but probabilities for each class. For this reason it comes in handy that we decided to implement Naïve Bayes ourselves as we can modify it to output probabilities instead of classes. The following sample shows the desired format for predictions, probabilities for each of the 15 possible classes:

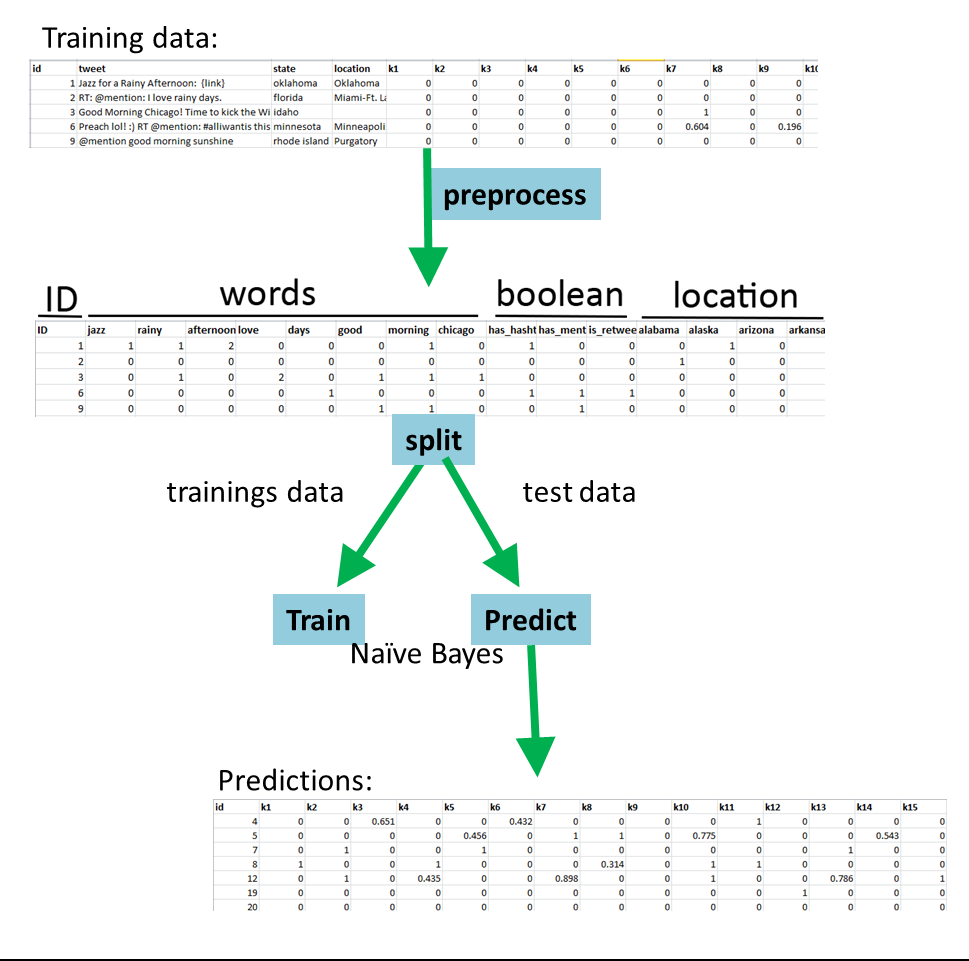
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TweetID** | **k1** | **k2** | **k3** | **k4** | **k5** | **k6** | **k7** | **k8** | **k9** | **k10** | **k11** | **k12** | **k13** | **k14** | **k15** |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0.604 | 0 | 0.196 | 0 | 0 | 0 | 0.201 | 0 | 0 |
| 11 | 0 | 0.203 | 0.176 | 0 | 0.376 | 0 | 0.421 | 0 | 0.176 | 0 | 0 | 0 | 0 | 0 | 0.579 |
| 13 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In this example many of the probabilities happen to add up to 1 in total but this is a coincidence; the the sum of the probabilities for tweet 11 is 1.931 for example.

**Approach**

**bag of words, nltk, sent**

**Solution**

****

1. **Preprocessing**
2. **Classification, Naïve Bayes**

As described above we created the option to either split the training data into training and test data or read the test data from a separate file as it is meant for the competition. We implemented the first case in the file *classify\_splitdata.py* and the second in *classify\_testdata.py*. We created a class for Naïve Bayes which has the methods *train()* and *predict()* and is instantiated in *classify\_splitdata.py* or *classify\_testdata.py.*

**Splitting Training Set**

Figure ?? shows the this scenario of splitting the training set: After preprocessing the data is split into training set and test set. There are 15 targets for which we want to run Naïve Bayes separately. Therefore, we loop through the 15 targets and each time train the classifier using the training set and predict using the test set.

**Using Separate Test File**

The scenario of using a separate file containing the test data is the scenario given by Kaggle. The training file given (*train.*csv) is 77,946 tweets long and the test file (*test.*csv) 42,157 tweets. When trying to preprocess all training data we got memory errors as the bag of words gets too big. About 10,000 tweets is the maximum we could test on. In order for a submission to be accepted by Kaggle at least all the test data needs to be processed, no matter with how much training data used. Therefore, we sliced the test data into separate file of 1000 tweets each and processed them separately. We used a training set of 5000 tweets for this test. However, it would be computationally very expensive to just loop through all the slices and apply the algorithms described above. This would mean that for each of the slices and each of the 15 targets we train the classifier using 5000 tweets which takes very long. Instead, we created a list of 15 *NaiveBayes* class instances. We train each of these first. Then we loop through the slices and predict for different targets using the respective *NaiveBayes* instance.

This gave us a training time of about an hour. However, the computation time for predictions was probably larger than 24 hours. We had to abort the process as we noticed that there were some errors in the files created. Because of these problems we decided to discard this approach and instead split the training set as described above.

**Results**

**Discussion: Evaluation of Approach and Solution**

**INTERMEDIATE REPORT:**

**Methods**

We will first preprocess the tweets: tokenize words, do simple spelling correction, use the nltk-toolkit for stemming. Then, we will use the bag-of-words approach to obtain the features. A Naïve Bayes Classifier will be fed with these features and output the likelihood for each class separately. The predictions are done separately because the prediction-labels are not dependent on each other, several labels can be predicted for one tweet.

(See: <http://www.kaggle.com/c/crowdflower-weather-twitter>).

**Possible Obstacles**

**Word Misspelling/Stemming:** As twitter language is often informal words are often written in many different (“wrong”) forms which impedes the performance of the classifier. For example *hot* might be written as *hottttt* to emphasize the intensity*.* A possibility to solve this problem with duplicate characters is to reduce all sequences of the same character longer than two characters to one character. Random misspellings could only be cured with a word lexicon which is outside the scope of this project. Moreover, one might want to stem words such that e.g. *cloudy, cloudier,* and *cloud* become *cloud.* This can be done using the nltk-toolkit.

**Kaggle-Evaluation:** As the Kaggle-competition is meant for three classifications (temporal, sentiment, kind of weather) it might be that the results for predictions will only give overall results. As we will only upload predictions for the kind of weather this would make it impossible to test. However, it might be that Kaggle also outputs results for each classifier separately. If not, we will have to separate the training set into test set and training set and use this data instead of uploading to Kaggle.

**Computation-Time:** The bag-of-words approach produces as many features as there are words in the dictionary. A preliminary analysis of our data (not using stemming or spelling corrections) reveals that approximately 10.000-20.000 words are used. Therefore, our classifier might take a lot of computation time.

1. https://www.kaggle.com/c/crowdflower-weather-twitter [↑](#footnote-ref-1)