Message Classifier

System used: MonkeyLearn

MonkeyLearn is a text mining cloud platform, allowing companies to obtain relevant data from text using machine learning technologies. It can be accessed with a graphical user interface and can be integrated with any programming language via its API.

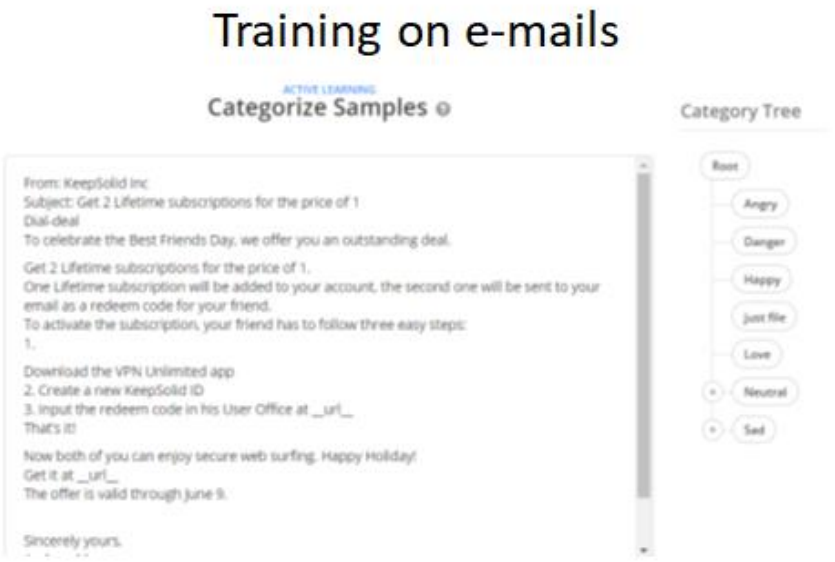
Problem:

To classify e-mails by emotions and topics

Solution:

1. Define the category tree.

2. Upload text samples for each category.



3. Train the machine learning classifier

How does MonkeyLearn work?

One of the key details you should be familiar with when building your own classifier is how the *classes* (or *categories*) are structured and how the process of *classification*and *training* is implemented.

Categories in MonkeyLearn are hierarchical, each category may have subcategories. There's a special category that is always present that doesn't have a parent category which is called the *root* category. Categories that don't have sub-categories are called *leaf categories*.

Category tree of a text classifier.

A classification result for any given text is a path from the root category to one of the leaf categories.

The training process

A key aspect to know is that, when training, the samples that represent a category are not only the ones directly associated with it, but also all the samples of the subtree.

The *training process* for each category has two subprocesses, the *evaluation process* and the actual *training process*.

The evaluation process

The *evaluation process* is the process that generates the stats for each category. In this process each category is trained with 90% of its samples. The other 10% are used as test samples, they are classified to measure the classifier performance.

The final training process

After a category is evaluated the final training can be performed, this time using all the available samples, that is, the final classifier that you will be able to consume via the GUI and the API, will be trained over all the data provided.

A machine learning classifier learns to assign the corresponding category to a text by using mathematical models and algorithms that learn to associate a particular input (text) to the corresponding output (label).

The text classification process consists of two main stages – training and classification. The training process involves loading the aforementioned hash table with the required information. In the hash table, for each word (i.e. key) in the entire training dataset, two values have been stored – a category 1 value and a category 2 value. The category 1 value is a count of the number of times that the word appears in a data instance labeled as category 1. Similarly, the category 2 value is a count of the number of times that word appears in a data instance labeled as category 2. The hash table thus acts as a training matrix consisting of data used to train the classifier. Once the hash table has been loaded, the training process is complete. Now, the classification process begins which makes use of the hash table, containing the trained data, to classify new data instances provided by the user. The classification task is similar to the task of loading the hash table. The classifier parses each data instance into its individual words and computes the category 1 and category 2 values, as defined earlier, for each word. It then computes the category 1 and category 2 values of the data instance itself by summing the category 1 and category 2 values of its constituent words. If the category 1 value of the data instance is higher than the category 2 value, it is classified as a category 1 data instance and vice-versa. The classifier then writes this newly classified data instance and its corresponding label to disk so that this information can be used in future iterations of the classifier for the same problem. This implements incremental data mining. The classifier then asks the user for the next input and the process continues until the user quits. The situation, in which the category 1 and category 2 values of a particular data instance are the same, indicates that the classifier has insufficient information to classify the data instance and cannot determine whether it belongs to category 1 or category 2.

Results:

Resuts for all

Confusion matrix

Actual \ Predicted	An	Da	Ha	Lo	Ne	Sa	ju
Angry	4	0	0	3	2	1	0
Danger	0	17	0	0	1	1	0
Happy	0	0	5	2	1	0	0
Love	0	0	0	9	2	0	0
Neutral	0	1	1	0	174	2	1
Sad	4	1	1	1	6	27	0
just file	0	1	0	0	1	0	19

For Neutral

Confusion matrix

Actual \ Predicted	Of	Sa	To	To	fr	gm
Official information	25	3	1	6	0	1
Sales	1	15	4	5	0	0
To Buy	0	3	10	3	0	0
To Study	4	2	2	52	0	1
from facebook	1	0	0	0	10	0
gmailbots	0	0	1	2	0	12

This classifier can be used in Java applications, here is the code to switch it on:

```java
import com.monkeylearn.MonkeyLearn;
import com.monkeylearn.MonkeyLearnResponse;
import com.monkeylearn.MonkeyLearnException;

public class App {
    public static void main( String[] args ) throws MonkeyLearnException {
        MonkeyLearn ml = new MonkeyLearn("7f7e035611bd48d5d493f7a62cbb5a0b8f79f2f5");
        String moduleId = "cl_W3QCChfS";
        String[] textList = {"This is a text to test your classifier", "This is some more text"};
        MonkeyLearnResponse res = ml.classifiers.classify(moduleId, textList, true);
        System.out.println( res.arrayResult );
    }
}

```