# IN4252 Web Science & Engineering
# Hands-on Assignment - 1

Ke Tao (k.tao@tudelft.nl)

November 19, 2013

## 1 Task 1: Retrieving via Twitter API

This task is one part of preparation for Task 3. The objective is to get familiar with Twitter API. It is an easy way for retrieving data from Social Web.

### 1.1 Start this task early

You are advised to **finish the task regarding Streaming API within 3 days (by November 23rd, Saturday morning 0800 CET)**. The reason is two-folded: (i) it is easy; (ii) you will need to collect your own data set early enough so that you can collect corresponding information later for Task 3. Why? When you start with Task 3, you need to evaluate a model with ground truth that is dependent on this. However, the ground truth will not be accurate enough until you wait a certain amount of time (like one week or so). Even the ground truth is ready, you need to crawl them again from Twitter, and this process **takes time** due to a limitation of Twitter API.

### 1.2 Twitter Streaming API

Twitter Streaming APIs provide developers with Twitter's global stream of Tweet data. If properly used, you can continuously receive Tweet data being pushed to you, possibly with some supported filters. You can refer to the official documentation[1] for details.

There are a couple of endpoints in the Streaming API, including *Public Stream*, *User Stream*, and *Site Stream*. In this assignment, you need to monitor *Public Stream*.

As in March 2013, Twitter enforced OAuth authentication for using Streaming API. That means you need to register an application on Twitter Developers in order to use it. You can access the application page via the following link (a Twitter account needed):

`https://dev.twitter.com/apps`

[1] `https://dev.twitter.com/docs/streaming-apis`

By clicking on **Create an application**, you would need to offer some basic information about the application. After that, you will be able to get following parameters:

- Consumer Key;
- Consumer secret;
- Access token;
- Access token secret.

You will need them to get your application authenticated with OAuth. Then you can crawl the public streams via the sample endpoint:
`https://dev.twitter.com/docs/api/1.1/get/statuses/sample` You can choose between calling the API with code or console.

### 1.2.1 Using code

The advantage of using code is that you can directly add some event processing code, e.g. for detecting the language[2], writing to a file, or reporting on crawling progress.

**Java** You can use *twitter4j* for monitor public stream. The code example can be found via following link:
`http://twitter4j.org/en/code-examples.html`

**Python** You can make use of *tweepy*[3] package for accessing Twitter Streaming API.

**NB:** the given example is using filter endpoint, you need to use sample endpoint by invoking *stream.sample()*.

### 1.2.2 Using console

In the application page, you can generate a cURL command after filling the Request URI field with the above url to the endpoint. If you have a Unix-like OS[4], you can execute it in a console and redirect it to a file. It may look like following (no line break in between):

```
curl --get 'https://stream.twitter.com/1.1/statuses/sample.json'
--header 'Authorization: OAuth
oauth_consumer_key="[your_consumer_key]",
oauth_nonce="[something_generated_automatically]",
```

[2]For Task 3, you can consider only English tweets if you want to exploit the semantic information.

[3]https://github.com/tweepy/tweepy/blob/master/examples/streaming.py

[4]If you are using a Windows machine, you can refer to this page for getting a cURL program: `http://curl.haxx.se/download.html`

```
oauth_signature="[something_generated_automatically]",
oauth_signature_method="HMAC-SHA1",
oauth_timestamp="[a_timestamp]", oauth_token="[your_access_token]",
oauth_version="1.0"' --verbose
> sampling.json
```

For this part, you need to monitor the Public Stream for 10 mins and answer following questions:

1. **What is the starting and ending time of the data that you have crawled?**
2. **What is the id of the first tweet you have got? And the last one?**
3. **How many tweets did you get?**
4. **How large is the result file (uncompressed file in JSON format)?**

Optionally, you are encouraged to try use the filter endpoint. While using this endpoint, you can specify the keywords that you want to monitor, the ids that refers to the users to be followed, or the geo-locations from which the tweets to be crawled are posted. More detailed information can be found at:
`https://dev.twitter.com/docs/api/1.1/post/statuses/filter`

## 1.3 Twitter REST API

Besides Streaming API, there are a set of REST APIs that provide most of Twitter functionality. For example, given an id of the Tweet, you can use Twitter REST API to retrieve the tweet including the metadata. Still, you need to use OAuth for authentication. However, there is usually a rate-limit for REST APIs. That means there is a certain limit on how frequent you can use the APIs. For example, the API call of retrieving a tweet with a given tweet id have a rate-limit of 180 times per 15 minutes[5].

You may try a couple of these APIs by utilizing the API wrapper, either twitter4j if you are using Java or tweepy for Python.

## 1.4 Language Detection

In the following analysis, we consider only English tweets. Therefore, we want to filter out the non-English tweets. There are already very accurate tools for this purpose. Python users may use langid[6]. Java users may use a similar library[7]. Both library will return the language code with a probability value. You can manually set a threshold, e.g. 0.9, which means if the probablity of being English is above the threshold then we think it is in English.

After finishing with this, please answer following question:

[5]https://dev.twitter.com/docs/rate-limiting/1.1/limits
[6]`https://github.com/saffsd/langid.py`
[7]`https://code.google.com/p/language-detection/`

1. **What is the threshold you have chosen?**
2. **How many English tweets have you found in the data that you have crawled with Streaming API?**

## 2 Task 2: Getting started with Weka for Mining Tweets

This task is another preparation for Task 3. The objective is to get to know how to use Weka for utilizing logistic regression as classification algorithm. The task is organized as a lightweight tutorial. Given a Twitter dataset as input for Weka, you will use Weka to analyze the following research question: what are the features of a tweet which allow for deciding whether the tweet is relevant to a given topic or not (cf. example in the slides of the lecture and corresponding research article [2]).

### 2.1 Download Weka

You can download Weka from the official website:
`http://www.cs.waikato.ac.nz/ml/weka/`

Weka is programmed in Java. Therefore, you have to install a Java Runtime Environment before you can get Weka running on your machine.

### 2.2 Dataset

Weka allows for loading datasets as ARFF files. During the lecture, we already introduced you to the format of ARFF files. For this task, the ARFF file is available via the following link:
`http://www.st.ewi.tudelft.nl/~ktao/wse2013/handson1-task2.arff`

This is the dataset that we used for investigating what makes a tweet relevant to a given topic [2]. For each *(tweet, topic)*-pair, the trained model should be able to make a decision whether the *tweet* is relevant to the *topic* or not. Each instance (= each line in the *data* part) in this dataset lists the 25 features (25 columns) and the relevance judgement (the last field) for a (tweet, topic)-pair. Please refer to Table 1 if you want to know the meaning of each feature/column. A more detailed description of the features can be found in [2].

### 2.3 Experiment Procedure

1. Load the dataset from the file or the URL given before (see Fig. 1).
2. We use the logistic regression to classify each (tweet, topic) pair into the relevant and the irrelevant instances. In the dataset, we only have 2817 (tweet, topic)-pairs (= 7% of all judged pairs) that are marked as *relevant*. Therefore, we use a cost-sensitive meta-classifier to fix the bias caused

Table 1: Description of Columns in the ARFF Data File

| **Column name** | **Description** |
|---|---|
| text_score | Retrieval Score (based on some IR models) |
| text_score_expansion | Retrieval Score using expansion on topic |
| hashtag | If the tweet contains hashtag(s) |
| hasURL | If the tweet contains URL(s) |
| isReply | If the tweet is a reply to another tweet |
| length | The length (in characters) of the tweet |
| sentiment | The polarity of the sentiment of this tweet |
| tweet_topic_time_diff | The time difference between the tweet and the query |
| semantic_overlap | Overlap of named entities between the topic the tweet |
| types_of_entities | #types of Named-Entities (NE) extracted from tweet |
| number_of_entities | #NEs extracted from tweet |
| organization_entities | #NEs with type of Orgranization extracted from tweet |
| person_entities | #NEs with type of Person extracted from tweet |
| work_entities | #NEs with type of Work extracted from tweet |
| event_entities | #NEs with type of Event extracted from tweet |
| species_entities | #NEs with type of Species extracted from tweet |
| places_entities | #NEs with type of Places extracted from tweet |
| nFollowers | #followers that the author has |
| nFriends | #followees that the author has |
| nFavorties | How many times has the tweets been marked as favorite by others? |
| nListed | How many lists has the author been listed in? |
| isVerified | Whether the tweet is posted by a verified account or not? |
| isGeoEnabled | Is there a geolocation attached to this tweet? |
| twitterAge | How many years has been since the author signed up on Twitter? |
| tweetsPostedByAuthor | How many tweets has the author posted on Twitter? |

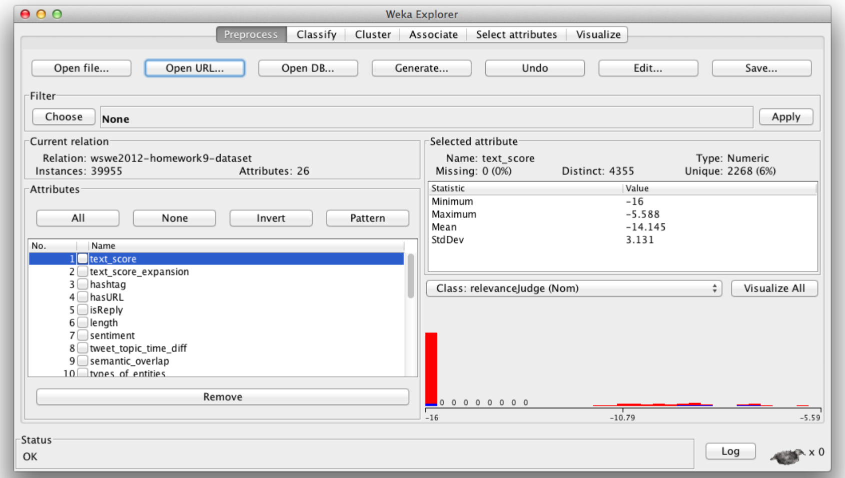
by the class-imbalanced dataset (Why? → `http://cling.csd.uwo.ca/papers/cost_sensitive.pdf`):

(a) Choose the tab of "Classify" (see Fig. 2).

(b) Choose "meta" → "CostSensitiveClassifier" as the Classifier.

(c) Click the area on the right side of the "Choose" button to modify the configurations of this classifier. The cost matrix should be like fol-

Figure 1: Load a dataset



Filter: None

Relation: wswe2012-homework9-dataset
Instances: 39955 Attributes: 26

| No. | Name |
|---|---|
| 1 | text_score |
| 2 | text_score_expansion |
| 3 | hashtag |
| 4 | hasURL |
| 5 | isReply |
| 6 | length |
| 7 | sentiment |
| 8 | tweet_topic_time_diff |
| 9 | semantic_overlap |
| 10 | types_of_entities |

Name: text_score Type: Numeric
Missing: 0 (0%) Distinct: 4355 Unique: 2268 (6%)

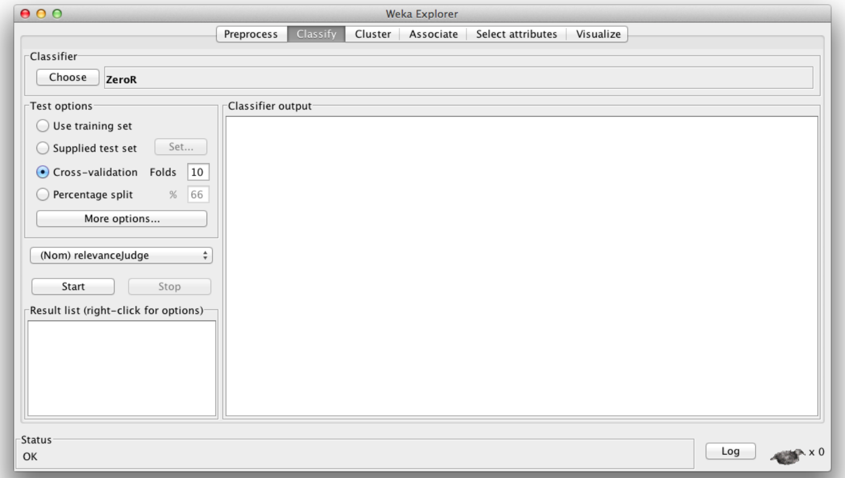
| Statistic | Value |
|---|---|
| Minimum | -16 |
| Maximum | -5.588 |
| Mean | -14.145 |
| StdDev | 3.131 |

Class: relevanceJudge (Nom)

-16 -10.79 -5.59

OK

Figure 2: Select Classify tab



Classifier: ZeroR

Test options:
- Use training set
- Supplied test set
- [✓ Cross-validation] Folds 10
- Percentage split % 66

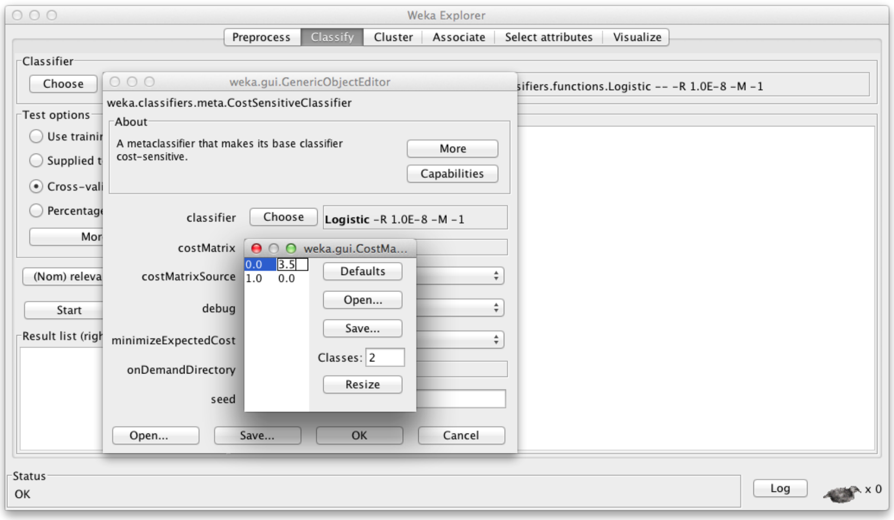
(Nom) relevanceJudge

OK

lowing (Click on “1x1 cost matrix” to modify the matrix, remember to resize the classes to 2, see Fig. 3):

$$\begin{pmatrix} 0 & 3.5 \\ 1 & 0 \end{pmatrix}$$

Figure 3: Change the Cost Matrix



Classifier: ...sifiers.functions.Logistic -- -R 1.0E-8 -M -1

weka.gui.GenericObjectEditor

weka.classifiers.meta.CostSensitiveClassifier

About: A metaclassifier that makes its base classifier cost-sensitive.

classifier: Logistic -R 1.0E-8 -M -1

costMatrix

weka.gui.CostMa...

|  |  |
|---|---|
| 0.0 | 3.5 |
| 1.0 | 0.0 |

Classes: 2

Test options: Cross-validation (selected)

(Nom) releva...

Status: OK

(d) Then in the same configuration dialogue, we choose "logistic regression" ("funtions" → "Logistic") as the classifier. Click "Okay" to finish the configuration.

3. The column of "relevanceJudge" is the last one. It is thus automatically selected as the class that should be predicted. By default, Weka uses 10 folds cross-validation for training and testing the classification model, i.e. just click on "Start" to obtain the results of the classification run.

## 2.4 Questions

Please answer each of the following questions in 2-3 sentences and include the answers into the report of this homework (see Section 4 for what you should deliver).

1. **What does "10-folded cross-validation" mean?**
2. **Why are we using a cost-sensitive classifier?**

## 3 Task 3: Retweet Prediction

The objective of this task is to design and implement your own strategies for representing tweets via features that allow for predicting whether a given tweet will be re-tweeted or not. You are recommended to start thinking about this task immediately after Task 2. In practice, this means that you are asked to generate an ARFF file by yourself and then use logistic regression as done in Task 2. You can have a look at existing publications to get some idea about the design of features that can be used for predicting whether a tweet will be re-tweeted. For example, Naveed et al. [1] propose a set of features that can be applied in order to characterize the interestingness of tweets and thus predict which tweets are likely to be re-tweeted. Example features are: does the tweet contains a URL, does the tweet includes a hashtag, does the tweet mention a negative word.

In this task, you are expected to implement features that allow for predicting whether a tweet will be re-tweeted. Using Weka, you should then analyze the impact of your features on the prediction.

## 3.1 Training Dataset

You will be given a sqldump, from which you can find nearly 50,000 tweets. They are crawled several weeks after being posted. All of them are identified as in English. About 14% of them had been retweeted while being crawled. Moreover, they were unlikely to be retweeted after that time because everyone is consuming and interacting with rather new information on Twitter. Therefore, it is reasonable to assume the rest 86% of tweets are not going to be retweeted ever. For this reason, you can use this dataset as your training data set.

The sqldump is available via following link:
`http://www.st.ewi.tudelft.nl/~ktao/wse2013/handson1-task3.sql.gz`

## 3.2 Preparation

Before you start with the design and implementation of your features as well as with the actual experiment with Weka in which you will analyze the features, you should do the following:

1. Have a look at the features that Naveed et al. used in their paper [1]. Which features are (in your opinion) most important?
2. Besides the features that are mentioned in the paper by Naveed et al., what features do you think might be useful and important for predicting whether a tweet will be retweeted or not?

## 3.3 Engineering Features and Generating ARFF File

You can select several features, either from the paper of Naveed et al. or based on your own ideas. Please come up with at least 5 features. **For EACH feature**, please answer following two questions:

1. **What is the meaning of the feature?**
2. **Why might the feature be useful for the prediction? What is your hypothesis for having this feature?**

You are encouraged to make use of the contextual information (Twitter profile of users who published tweets).

Generating the ARFF file: each of the tweets in the **training data set**, should be represented according the features that you selected so that you can generate the ARFF file. In the data part of this file, each line stands for a single tweet. The last column should be the class which the tweet should be classified into (i.e. 1 = the tweet was retweeted, 0 = the tweet was not re-tweeted. Since it might be too time-consuming to check for each tweet $t$ that has been published by one of your users whether $t$ has been re-tweeted, you can do the following:

- if "retweetedFromPostId > 0" (in the table of "tweets_sample") then the class should
- otherwise the class should be 0 (= tweet was not re-tweeted)

Basically, this means that you just check whether a given tweet $t$ is a re-tweet or not. Hence, you are not allowed to use "retweetedFromPostId", "retweet-Count", information such as "does the tweet content start with 'RT'?" or other information as a feature that directly marks the tweet as a re-tweet. Instead, you should represent the tweet $t$ so that it corresponds to the tweet $t_{original}$ that was originally re-tweeted.

## 3.4 Training the Model via Logistic Regression

You have already used Logistic Regression in Task 2. For this task, please use the ARFF file that you have just generated and follow the same procedure to train the model as done in Task 2. However, for test data set, please Please be careful when you configure the cost matrix: try to adjust the matrix and see how this impacts the performance/quality of the predictions.

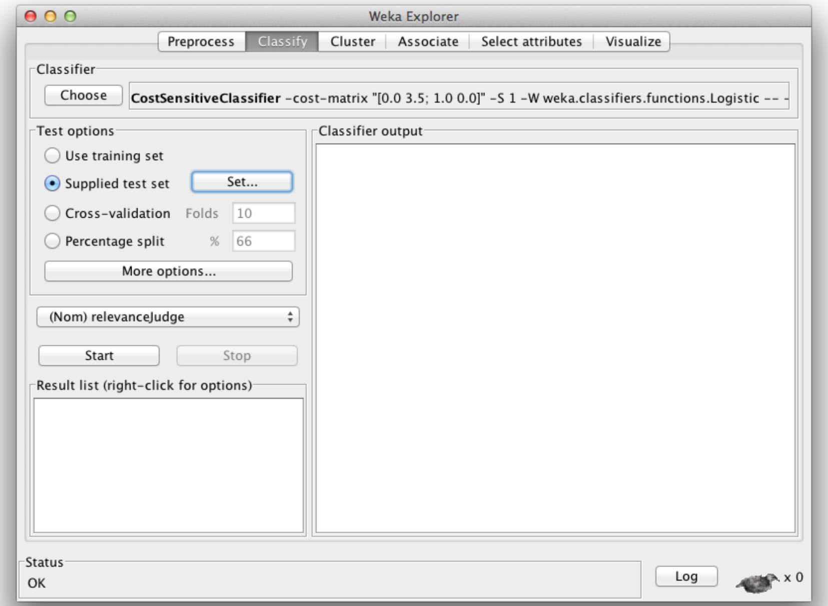
## 3.5 Test Dataset

In Task 2, due to limitation to relevance judgements, we applied cross-validations to evaluate our model with relatively small data set. However, for Task 3, we are not going to do the same thing. You will need to use your own test data set, based on what you have crawled for Task 1. However, when you crawled the data via Streaming API, the tweets were fresh. That also means, nobody knew whether they would be re-tweeted or not. Therefore, you need to use REST API (with limitation) to check those tweets again. Now (= **one week after you finish Task 1**), you can crawl the English tweets that you have found in Task 1. Be careful, because there is a rate limit on the REST API, you can only get 720 (=180 *times* 4) tweets per hour. So, this will take roughly 8 10 hours, depending how many English tweets you have got from Task 1. Then by checking the "retweet_count" field in JSON format, you should be able to see whether the tweets have been retweeted or not. After finishing with this, please answer following questions:

1. **How long did it take you to re-crawl the English tweets?**
2. **Are they all still available for crawling?**[8]
3. **If some of them are not available anymore, what are the reasons?**
4. **How many tweets do you get in the end?**

Then, you can follow the same procedure in Section 3.3 to generate another ARFF file. You can specify the test dataset as shown in the following:

Figure 4: Select a Test Dataset



```
CostSensitiveClassifier -cost-matrix "[0.0 3.5; 1.0 0.0]" -S 1 -W weka.classifiers.functions.Logistic -- -
```

( ) Use training set
(•) Supplied test set
( ) Cross-validation Folds 10
( ) Percentage split % 66

(Nom) relevanceJudge

Status: OK

## 3.6 Analyzing the Results

Given the model (coefficients of the logistic regression function) and the evaluation results from the test data set, please answer the following questions:

1. **Which features have (a) the highest positive impact, (b) the highest negative impact and (c) the lowest impact? Discuss possible reasons for these observations and explain whether your hypotheses on which you based the corresponding features was validated or not?**

[8]`https://dev.twitter.com/docs/error-codes-responses`

2. Optional: **What impact do the contextual features have on the classification results?**

For each of the sub-questions (1(a), 1(b), 1(c) and optionally 2), try to limit your answer to 3-5 sentences.

## 4 To be delivered

You are expected to compress following files in a zip file. Please name this file as "**[Lastname].[Firstname].zip**", e.g. Tao.Ke.zip, and upload it via the BlackBoard platform.

- Report (in PDF) which includes the answers to the questions (in bold font) of Task 1, Task 2, and Task 3.[9]
- The ARFF file (training dataset and test dataset) that you generated for Task 3.
- A text file that lists the results of the experiment as outputted by Weka.
- A text file that includes the problems you encountered with while doing the homework.

## 5 Deadline

Please submit your homework before **December 2nd, 2013, 0800 (CET)**. However, please try to finish **Task 1 before November 23rd 0800** (see reasons in the description of the task).

## 6 Q&A

If you have any question, make sure you have read this document and the FAQ web page via following link:

`http://www.st.ewi.tudelft.nl/~ktao/wse2013/faq-handson-1.html`

If you still don't get the answer after that, please send an email and good luck!

## References

[1] N. Naveed, T. Gottron, J. Kunegis, and A. C. Alhadi. Bad news travel fast: A content-based analysis of interestingness on twitter. In *WebSci '11: Proceedings of the 3rd International Conference on Web Science*, 2011.

[9] Pay attention to the questions in Sec. 3.3, for EACH feature, you have to answer two questions.

[2] K. Tao, F. Abel, C. Hauff, and G.-J. Houben. What makes a tweet relevant for a topic? In *Proceedings, 2nd Workshop on Making Sense of Microposts (#MSM2012): Big things come in small packages, Lyon, France, 16 April 2012*, pages 49–56, April 2012.