Course Work – MSc
Event Detection

Total Marks – 100 marks & Weightage – 20%
Course work Deadline – 17 November 2017 4:30PM

The objective of this course work is to develop an event detection method and evaluate it on a collection of tweets. Students submit the **code and the report** on or before the specified deadline. Submission is through the Moodle page for the Web Science course.

This course work is based on the lecture on event detection. We will focus on Twitter data. You will be given a sub-sample of a week's tweets (3,988,534 tweets) and also the ground truth. To make the life easier, we will give output of various early stages, like pre-processing and clustering. For more details, you need to attend the lecture on Monday, 16th October 2017 (Thursday 19th for Singapore)

The course mark will be marked out of 100 and will be weighted at 20% of the final course marks. As the usual practice across the school, numerical marks will be appropriately converted into bands. Final written exam has 80% weightage, which will be in April/May 2018.

Specific task to do

You need to develop an event detection algorithm. One such method is given in Lecture 05 on event detection. You can modify it or develop a new one. Once an event detection method is developed you need to implement them in a programming language of your choice, though I would recommend, Python, or Java. Subsequently, you need to evaluate the method on the data set given, collecting statistics. A report needs to be written describing your method and the evaluation results.

References

1. Andrew James McMinn, Joemon M. Jose: Real-Time Entity-Based Event Detection for Twitter. CLEF 2015: 65-77
2. Andrew James McMinn, Yashar Moshfeghi, Joemon M. Jose: Building a large-scale corpus for evaluating event detection on Twitter. CIKM 2013: 409-418
3. Mahmud Hasan, Mehmet A. Orgun, and Rolf Schwitter., TwitterNews: Real time event detection from the Twitter data stream, PeerJ PrePrints, 4:e2297v1, 2016 (https://peerj.com/preprints/2297/)

Report structure & Mark Distribution

Report should be organised the following way

1. Section 1: Approach
    a. Description of the method and algorithms with pseudo-code.

2. Section 2: Code description & UML diagrams (along with explanations)
    a. Please do not include entire code here; just description along with examples if needed
    b. UML diagrams
3. Section 3: Evaluation
    a. Description of data and measures
    b. Evaluation results
    c. Critical examination of results
4. Section 5: Critical Discussion- Event Detection

Mark distribution

1. Clearly identify the problem and describe your solution, including high-level pseudo code – **This component is worth 20 marks!**
    a. In this part of the report, you should clearly identify and describe the steps you are using along with justification.
    b. For example, you normally perform clustering of tweets, as part of the event detection. This is computationally expensive and we have already conducted clustering and the output is given. Unless you are redoing this, you just need to mention that you worked on the output of the clusters.
    c. You may deviate from the techniques given in the lecture. This maybe to introduce new innovations, to reduce computation and/or to increase speed. Such aspects need to be described properly and is expected.
    d. Where appropriate use high-level pseudo-code. E.g., example of mathematical-style pseudo-code, for the Ford–Fulkerson algorithm is given at: https://en.wikipedia.org/wiki/Pseudocode
2. For the description of the software, including source code and high-level UML diagram – **worth 15 marks**
    a. It is not my objective to look into the super efficiency of your code. However, I would expect them to be in a state properly commented and readable. Organise them into modules.
    b. Appropriate UML diagrams will fetch *5 marks*
    c. Proper commenting and modularisation will get *5 marks*
    d. *5 marks* will be given, if the code is in a working condition and do what is expected to do.
    e. *5 marks* will be for the level of sophistication of the code; this include efficiency consideration, appropriate data structures etc.
    f. *Code will be submitted as it is (no need to add it to the report)*
    g. *You can use the software you like (e.g., python, Java, C/C++; Go)*
    h. *Report and code will be added to a zipfile, which should be submitted.*

3. Evaluation of the proposed method and description of the evaluation methodology a (data, measures etc.) and the results - **worth 25 marks**
    a. If you run the code against entire data set and collect measurements, this will fetch *10 marks*

b. Explaining the evaluation measures used and reporting the measures will fetch *8 marks*
c. Commenting critically on the results will fetch *7 marks*

4. Critical discussion – critically discussing the proposed method, results and alternative approaches - **worth 15 marks**

    a. Critical discussion includes commenting on the aspects of method, its scalability, usability, effectiveness, efficiency etc.
    b. You may also want to comment on some of the applications of such a method.
    c. Your arguments should be supported with data and/or justification. It shouldn't be just statements.
    d. This is an open problem and I will leave this to your imagination
    e. It will help you to read some of the papers on this topic.

Traffic Event detection – 20 marks

Traffic congestion is one of the big challenges for both travellers and infrastructure managers. Non-recurring congestion such as traffic accidents, work zones, adverse weather events and special events, which accounts for half of the total congestion.

The objective of this task is to detect traffic related events from social media data, Twitter stream. This is an open problem and I would invite students to come up with their own intuitive solutions.

Mark Distribution

5. Clearly identify the problem and describe your solution, including high-level pseudo code or traffic event detection– This component is **worth 10 marks**

6. Critical discussion – critically discussing the proposed method, results (if possible with numbers) and alternative approaches - **worth 10 marks**

Datasets & Tools

Dataset

The dataset contains two folders, one containing 1 days worth of tweets sampled from the Events2012 collection, and another containing a full weeks worth of tweets sampled from the Events2012 collection. These are in datasets/1day and datasets/7days respectfully.

The 1day dataset contains information about 592,391 tweets, while the 7day dataset contains information about 3,988,534 tweets. You are advised to use the 1day set during development, and the 7days set for evaluation.

Within each of the folders (1day, 7days), the following comma separated (csv) files are provided:

clusters.sortedby.clusterid.csv (1day: 37,634 tweets, 7days: 181,971 tweets)
This file is a good place to start with your analysis. This file provides clusters of tweets, one tweet per line, clustered using the entity-partitioning cluster method described in the paper "Real-Time Entity-Based Event Detection for Twitter" and covered in the lecture.

Note that the "detection" step has not been performed, only clustering. Tweets which do not contain a name entity or which have no "nearest neighbour" have been removed, meaning that less than 10% remain. Your task is to automatically decide which clusters discuss events, and which clusters are noise.

Each row represents a single tweet, with the following columns:

- cluster_id
- cluster_name_entity
- tweet_id
- timestamp_ms
- user_id
- tweet_tokens
- tweet_text

cluster_id is the unique ID for the cluster that the tweet belongs to. Tweets with the same cluster_id are part of the same cluster.

cluster_named_entity is the named entity which was used for the partitioning step before clustering. Tweets with the same cluster_id will have the same cluster_named_entity.

tweet_id is the unique id for the specific tweet. Note that the same tweet_id could appear in multiple clusters if the tweet contains multiple named entities.

timestamp_ms is the Unix Epoch timestamp for this tweet: the number of **milliseconds** since midnight on the 1st of January 1970 when the tweet was created.

user_id is the unique id for the user who created the tweet.

tweet_tokens provides a **space separated** list of tokens used in the tweet. These have been stemmed using the porter stemming algorithm.

tweet_text is the original tweet text as posted by the user, however to make parsing this file easier, newlines have been replaced with a space.

Rows are sorted by their cluster ID (lowest to highest).

clusters.sortedby.time.csv (1day: 37,634 tweets, 7days: 181,971 tweets)
The file is exactly the same as clusters.sortedby.clusterid.csv however has been sorted by time (oldest to most recent) rather than cluster ID.

tokens.sortedby.time.csv (1day: 168,916 tweets, 7days: 1,144,157 tweets)
If you wish to perform token level analysis on individual tweets, this file provides:

- tweet_id
- token1, token2, token3, …, tokenX

Tweets are sorted by their timestamp from oldest to most recent. This file **only contains rows for tweets which have at least one named entity as extracted by the Stanford NER.**

namedentities.sortedby.time.csv (1day: 168,916 tweets, 7days: 1,144,157 tweets)
If you wish to perform tweet level analysis of named entities, this file provides:

- tweet_id
- named_entity1, named_entity2, …, named_entityX

The first column will always be the tweet ID, followed by a variable number of comma separated columns depending on the number of named entities mentioned in the tweet. Tweets are sorted by their timestamp from oldest to most recent. These were extracted using the Stanford NER.

tweets.sortedby.time.csv (1day: 592,391 tweets, 7days: 3,988,534 tweets)
It is unlikely that you will need to use this file directly unless you wish to parse and tokenize the tweets yourself. This file contains "raw" information about the tweets, with the following columns:

- tweet_id
- timestamp_ms
- user_id
- user_name
- user_followers
- user_following
- tweet_text

Things to watch out for

- **Integer overflow**: Tweet IDs and Timestamps are 64 bit integers. A 32 bit int will not suffice. This can cause issues in some languages (namely Javascript) which do not support 64 bit integers.
- **Using the right order**: Some tasks are better suited to processing tweets in order of their cluster_id, whilst other tasks will be much easier if tweets are processed as a time-ordered stream.

Tools

eval.py

eval.py (eval/eval.py) is a Python script designed to evaluate the performance of event detection approaches by calculating precision and recall values. Unlike more traditional evaluation tools which work on a document by document basis, eval.py examines clusters of tweets to decide if the cluster as a whole is relevant to an event.

A special version of eval.py has been provided which can read csv formatted files similar to those provided with the coursework (such as clusters.sortedby.clusterid.csv).

To run eval.py on 1day/clusters.sortedby.clusterid.csv simply navigate to the eval folder using the command line and run:

```
python eval.py ../1day/clusters.sortedby.clusterid.csv
```

This will provide you with baseline results which you can compare to your modifications.

For example, if you have extracted all tweet clusters with 30 or more tweets into a file called clusters.min30tweets.csv, you can evaluate this by running:

python eval.py ../1day/clusters.min30tweets.csv

The tool will examine the file and produce output in 3 parts:

1. **A cluster by cluster summary**
   A list of clusters eval.py was able to match to an event in the groundtruth.

   Each row is a single cluster, where the cluster_id corresponds to the cluster_id in the file you asked it to evaluate.

   The second column describes how many tweets in the cluster are known to be relevant to the event, out of the total number of known relevant tweets for that event in the groundtruth. The percentage gives the percent of tweets in the cluster that are also in the ground truth - 100% means that every tweet in the cluster is also in the groundtruth. This column gives an idea of how well the clustering approach performs and is mostly a measure of tweet recall rather.

   The final column gives a description of the event written by one of the 5 crowdsourced evaluators for the event. You will notice that multiple clusters describe the same event. This is an example of "fragmentation", a very common problem in event detection.

2. **Event / Cluster Statistics**
   This gives aggregate information about how many of the 506 events in the relevance judgements were detected, and how many of the clusters were relevant to an event.

   It is worth remembering that the groundtruth covers 4 weeks worth of data, not just the 1 week that you have been given. This means that recall is always going to be quite low. Also note that you cannot improve recall above the baseline measurement since this is a (effectively) a measure of clustering performance. Instead, you should focus in improving precision with minimum impact to recall. The F-Measure provides a "balanced" measurement which factores precision and recall equally.

3. **Categories**
   This breaks the detected events down by category. This can give clues about how the detection algorithm is behaving and what types of event it is good (or bad) at picking up, however it more interesting than useful.

Example output from eval.py can be found below.

CLUSTER ID	TWEET COVERAGE	EVENT DESCRIPTION FROM CROWDSOURCED EVALUATION
1032	61/651 (84%)	It is about TV show by Keshya cole and her Husband
1035	7/651 (50%)	It is about TV show by Keshya cole and her Husband
1048	14/104 (82%)	Heriberto Lazcano Lazcano, the top leader of the c
1072	9/651 (47%)	It is about TV show by Keshya cole and her Husband
1079	76/444 (77%)	They all discuss about fat joe
1080	13/444 (100%)	They all discuss about fat joe
1082	14/444 (87%)	They all discuss about fat joe
1083	9/444 (69%)	They all discuss about fat joe
1187	38/294 (100%)	Malala Yousafzai, a 14 year old activist for women
1189	66/651 (82%)	It is about TV show by Keshya cole and her Husband
119	69/235 (41%)	is about an Award function. The nominees can be KE
120	6/235 (15%)	is about an Award function. The nominees can be KE
121	21/235 (7%)	is about an Award function. The nominees can be KE
122	29/235 (60%)	is about an Award function. The nominees can be KE
124	15/235 (62%)	is about an Award function. The nominees can be KE
125	23/235 (88%)	is about an Award function. The nominees can be KE
1283	12/307 (100%)	Penn State scandal involving imprisoned former foo
1295	11/307 (100%)	Penn State scandal involving imprisoned former foo
1300	7/104 (87%)	Heriberto Lazcano Lazcano, the top leader of the c
131	13/235 (19%)	is about an Award function. The nominees can be KE
134	14/235 (87%)	is about an Award function. The nominees can be KE
137	7/235 (77%)	is about an Award function. The nominees can be KE
1370	16/294 (84%)	Malala Yousafzai, a 14 year old activist for women
1377	6/104 (100%)	Heriberto Lazcano Lazcano, the top leader of the c
140	11/235 (100%)	is about an Award function. The nominees can be KE
141	6/235 (100%)	is about an Award function. The nominees can be KE
... lots and lots of rows removed ...		
926	22/269 (64%)	Some guy named omarion dancing on stage
938	17/269 (42%)	Some guy named omarion dancing on stage
939	7/269 (38%)	Some guy named omarion dancing on stage
955	13/269 (44%)	Some guy named omarion dancing on stage

------- EVENTS -------

EVENTS / CLUSTER STATISTICS:
- Of 506 events, 19 were detected.
- Of 8829 clusters, 120 could be matched back to an event.

Event Recall: 0.038
Cluster Precision: 0.014
Overall F-Measure: 0.020

----- CATEGORIES -----

CATEGORY NAME	RECALL	
Arts, Culture & Entertainment	5/53	(0.094)
Armed Conflicts & Attacks	3/98	(0.031)
Science & Technology	2/16	(0.125)
Miscellaneous	1/21	(0.048)
Sports	1/126	(0.008)
Business & Economy	1/23	(0.043)
Law, Politics & Scandals	6/140	(0.043)

Example output produced by eval.py evaluating the 1 days worth of clusters