ANM 2019 Fall Assignments and Project

Prerequisites

To succeed in ANM 2019 Fall assignments and project, you are expected to have the following skills (you can learn them gradually during the course):

- At least one programming language (Python is recommended).
- At least one data visualization tool, such as PowerBI, Matlab, GNUPlot, and Python matplotlib.
- Some background regarding AIOps (Artificial Intelligence for IT Operations).
- Some machine learning tools such as Scikit-learn, PyTorch, TensorFlow.
- **Using Google to search the things you do not know (important!)**

Overview

The ANM 2019 Fall tasks contains **two assignments(10% + 20%) and one project(60%)**. **Each student finishes each assignment alone** and **a team of 2-3 students finish the project as a group**.

For each assignment, you need to finish all the jobs (described by bullet points) and write them in one assignment report (one report for one assignment, in which you describe the results of all the jobs in detail). For example, some jobs require you to plot a figure, then you have to put the figure into the report; some give you questions, then answer them. We will give you the grade of each assignment based on the report **only**.

As for the project, you need to submit your algorithm result on the Kaggle-like competition website and get your rank.

- Leaderboard: 50%
- Presentation (scored by all students, TA, and instructor): 10%

Assignment #1 -- Data Preprocessing and Visualization

Deadline: Oct 8 23:59 Grade: 10%

Data preprocessing and visualization are important skills of managing Internet services such as search engines and online social networks. In this assignment, you can touch real data and learn some practical techniques to analyze the data, and you will have an unique opportunity to access **2 weeks of search logs** from a large search engine. Using the data, we give you the assignment to help learn both basic and advanced techniques of data analysis. The things you can learn include:

- **Basic statistics:** coding to count the data in different ways, e.g., how many queries are

served per minute. You can also use Power BI do it.
- **Visualization:** plot figures to show the data (e.g., line chart, histogram, and CDF). You can also use visualization tools (such as Power BI) to do it.

To finish the assignment, you will also learn to use several visualization tools and statistical methods, such as exploratory visualization tool Power BI.

Background and Data

Once you submit a query to a search engine, the search engine will log some related attributes regarding this query, such as when the query is submitted (e.g., timestamp) and the search response time (SRT).

In this assignment, we use 2 weeks of search logs from a global top search engine. Each day of search logs are written into one log file, so we have 14 (7 * 2 weeks) log files in total. Note that, because there are more than one billion queries submitted every day, we do not log them all but only a small random part of them. The log files are of the CSV (Comma Separated Values) format, and each column represents one attribute. The first line in the file contains the names of each attribute, and the following lines are the specific values for each query. A sample of the raw log file is as follows:

```
Timestamp,#Images,UA,Ad,ISP,Province,PageType,Tnet,Tserver,Tbrowser,Tother,SRT
1411315200,0,Chrome,noAD,CRTC,Heilongjiang,sync,1495.0,443.14,73.0,0.0,2011.14
1411315200,13,Chrome,noAD,CHINANET,Guangdong,async,200.0,80.0,47.0,359.0,686.0
1411315200,24,Chrome,noAD,UNICOM,Hunan,async,120.0,450.0,26.0,191.0,787.0
1411315200,4,Safari,noAD,OTHER,Guangdong,async,272.0,86.0,234.0,219.0,811.0
```

To be more clear, we show the above data in a table below (you can get the table view of the data by simply importing the CSV file into MS Excel, etc.).

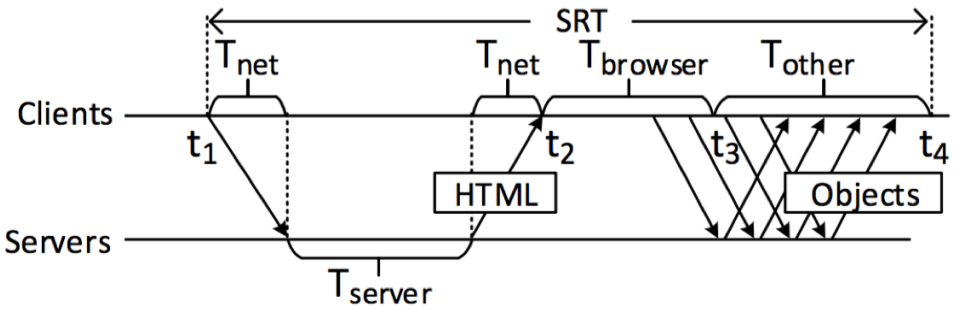
Timestamp	#Images	UA	Ad	ISP	Province	PageType	Tnet	Tserver	Tbrowser	Tother	SRT
1411315200	0	Chrome	noAD	CRTC	Heilongjiang	sync	1495.0	443.14	73.0	0.0	2011.14
1411315200	13	Chrome	noAD	CHINANET	Guangdong	async	200.0	80.0	47.0	359.0	686.0
1411315200	24	Chrome	noAD	UNICOM	Hunan	async	120.0	450.0	26.0	191.0	787.0
1411315200	4	Safari	noAD	OTHER	Guangdong	async	272.0	86.0	234.0	219.0	811.0

The description of those attributes in the above table is as follows.

- **Timestamp:** the unix timestamp when the query is submitted. For example, "1411315200" represents "2014/9/22 0:0:0".
- **#Images:** the number of images embedded in the result page.
- **UA:** user agent, the type of the user's browser where the query submitted from.
- **Ad:** whether the result page contains ads or not, "AD" for yes and "noAD" for not.
- **ISP:** the ISP (Internet Service Provider) that the user or the query comes from.
- **Province:** the location of the user (32 provinces for this attribute since the logs only contain queries from China mainland).
- **PageType:** whether the page is loaded synchronously or asynchronously.

- **Tnet:** the 1st component of SRT (ms), the page transmission time over the network (see the figure below).
- **Tserver:** the 2nd component of SRT (ms), the server-side processing time of the query.
- **Tbrowser:** the 3rd component of SRT (ms), the DOM parsing time of the browser.
- **Tother:** the last component of SRT (ms), the remaining time for acquire other embedded elements in the page, such as images.
- **SRT:** search response time (ms), which is the sum of the above four SRT components.

The figure below shows a simplified timeline of a search, where you can find the details of the four SRT components and their relationship with SRT.



A very basic skill when dealing with the data is to count the numbers for different purposes, and then visualize them. In this assignment, you need to do the following jobs using the data:

- Calculate the average SRT of every 10 minutes, and plot the SRT with a **line chart** (x axis for date time and y axis for the average SRT).
- Calculate the average of each SRT component of every 10 minute, and plot the four SRT components together with a **stacked area chart** (x axis for date time and y axis for time) and also a **100% stacked area chart** (y axis for the percentage).
- Plot the **CDF (Cumulative distribution function) chart** of SRT.
- Plot the **CDF chart** of #Images.
- Count the number of queries (also called page views or PVs) of each minute, and plot the minute-level PVs with a **line chart** (x axis for date time and y axis for the PVs).
- Count the PVs of each province, and plot it with a **histogram chart** (x axis for province and y axis for PVs).
- Count the PVs of each UA, and plot it with a **pie chart** (show the percentages in the chart).
- What are the differences among those charts (How to decide which one to use)
- Describe your experience or findings in doing those jobs. For example, experience of processing the data, observations from the charts, characteristics of the data, potential explanations, and any interesting things you would like to mention.

Reference

[1] FOCUS: Shedding Light on the High Search Response Time in the Wild. INFOCOM 2016.

Assignment # 2 — Log Analysis for Anomaly Detection

Deadline: Nov 19 23:59 Grade: 20%

Background and Data

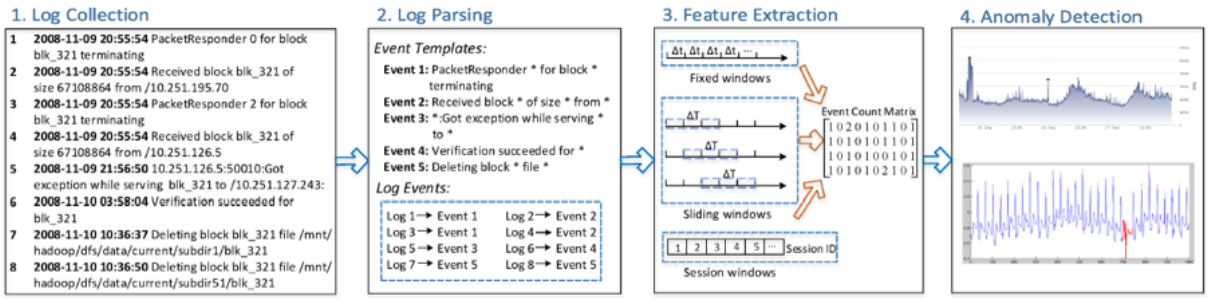
Anomaly detection plays an important role in management of modern large-scale distributed systems. Logs are widely used for anomaly detection, recording system runtime information and errors. Traditionally, operators have to go through the logs manually with keyword searching and rule matching. The increasing scale and complexity of modern systems, however, makes the volume of logs explode, which renders the infeasibility of manual inspection. To reduce manual effort, we need anomaly detection methods based on automated log analysis.

Raw log messages are usually unstructured texts. To enable automated mining of unstructured logs, the first step is to perform log parsing, whereby unstructured raw log messages can be transformed into a sequence of structured events. Then we are able to do anomaly detection based on these sequences.

Typically, one log message will record a specific system event with a set of fields: timestamp, verbosity level (e.g., INFO, WARN, ERROR), and free-format message content. A sample of the raw log file is shown below:

```
2008-11-09 20:35:32,146 INFO dfs.DataNode$DataXceiver: Receiving block
blk_-1608999687919862906 src: /10.251.31.5:42506 dest: /10.251.31.5:50010
```

The figure below shows a popular framework for log-based anomaly detection. The anomaly detection framework mainly involves four steps: log collection, log parsing, feature extraction, and anomaly detection.



This assignment is divided into two parts:

1) The first part is **Log Parsing** (second step in above figure).

The purpose of log parsing is to extract a group of event templates; whereby raw logs can be structured;

2) The second part involves **Feature Extraction and Anomaly Detection** (third and fourth steps in above figure).

In this part, we will identify whether or not a new incoming log sequence is an anomaly.

Part 1:

[1] introduces a number of data-driven approaches for automated log parsing. In this part, we offer some datasets (include five system logs, each file has 2000 labeled logs). In each dataset, “rawlog.log” is the raw log to parse. “template*XX*.txt” refer to logs belong to *XX* template.

You need to learn the paper [1] and do the following jobs using the given datasets:

- Grasp **four** log parsing algorithms: **LogSig**, **IPLom**, **SLCT** and **LKE**
- Use toolkit to run the four log parsing algorithms (toolkit: https://github.com/logpai/logparser).
- Plot the **runtime** with a bar chart when four algorithms parse these datasest.
- Try to adjust parameters for best **RandIndex**[2] (a metrics for evaluation clustering) and plot a bar chart when four algorithms parse these datasets.
- Try to adjust parameters for best **F-score** and plot a bar chart when four algorithms parse these datasets.
- Describe your own experience or findings in doing those jobs. For example, advantages and disadvantages of these algorithms.

Part 2:

[3] introduces a number of data-driven approaches for automated anomaly detection based on system log. In this part, we provide HDFS logs to do anomaly detection follow the framework mentioned above.

You need to learn the paper [1,3] and do the following jobs using the **HDFS logs**.

- Grasp the principles of **three** unsupervised anomaly detection models: **Invariants Mining, PCA and LogCluster**.
- Based on part1, choose one log parsing methods, and then use toolkit to run the three anomaly detection models (toolkit: https://github.com/logpai/loglizer). Notice that LogCluster suffers from very high computational complexity and it may take a very long time to obtaind its result (or reporting "memory error"). You can sample a part of data to run LogCluster algorithm.
- Try to adjust parameters for better **F-score, precision, recall,runtime** and plot them with bar charts when parsing logs.
- When running **Invariants Mining**, add some codes and display **three relationships** (please check the paper for more information), *e.g.*, n(A) = n(B), where n(∗) represents the number of logs which belong to corresponding template ∗. And explain why, *e.g.*, template A is “Interface *, changed state to down”, while template B is “Interface *, changed state to up”.
- Describe your own experience or findings in doing those jobs. For example, in which situation, these models do not detect anomalies correctly.

Reference

[1] He P, Zhu J, He S, et al. An Evaluation Study on Log Parsing and Its Use in Log Mining[C]// Ieee/ifip International Conference on Dependable Systems and Networks. IEEE Computer Society, 2016:654-661.

[2] https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

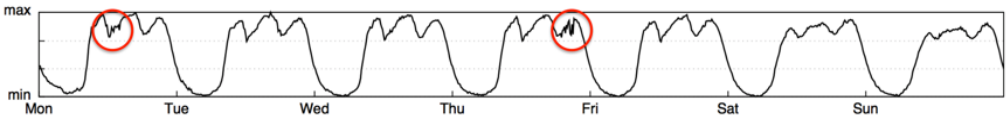
[3] He S, Zhu J, He P, et al. Experience Report: System Log Analysis for Anomaly Detection[C]// IEEE, ISSRE 2016.

Project — Anomaly Detection Competition

Leadboarder Deadline: Dec 19 23:59 Grade: 50%

Presentation Time: Dec 25 (Last class) Grade: 10%

To ensure undisrupted Internet-based services, such as search engines, online shopping, and social networking, large Internet companies need to closely monitor various Key Performance Indicators (KPI, e.g., Page Views, number of online users, and number of orders) to accurately detect anomalies and trigger timely troubleshooting/mitigation. The figure below shows 1-week example of PV and the red circles mark some obvious anomalies.



KPI: The KPIs are the time series data with the format of (timestamp, value).

Anomaly: KPI time series data can also present several unexpected patterns (e.g., jitters, slow ramp-ups, sudden spikes and dips) in different severity levels, such as a sudden drop by 20% or 50%.

KPI Anomaly detection can be formulated as follows: for anytime t, given historical observations $x_{t-T+1}, \cdots, x_t$, determine whether an anomaly occurs (denoted by $y_t = 1$). An anomaly detection algorithm typically computes a real-valued score indicating the certainty of having $y_t = 1$, *e.g.*, $p(y_t = 1 | x_{t-T+1}, \ldots, x_t)$, instead of directly computing y_t. Human operators can then affect whether to declare an anomaly by choosing a threshold, where a data point with a score exceeding this threshold indicates an anomaly.

In this project, we organize a KPI anomaly detection algorithm competition (http://iops.ai/competition_detail/?competition_id=11&flag=1), and we collect 26 labeled KPIs from 5 large Internet companies (Alibaba, Tencent, Baidu, Ebay, and Sogou). We divide each KPI into training and testing sets, whose ratios are 50%, 50% respectively. Training sets have labels used to train algorithm and testing sets have no labels. To be more clear, we show a example of training sets and testing sets in a table below. **KPI ID** denotes the name of 26 KPIs.

training set:

KPI ID	timestamp	value	label
A	1411315200	100.43	0
A	1411315260	90.5	1
A	1411315320	96.7	0
...	...	...	...

testing set:

KPI ID	timestamp	value
A	1511315200	80.67
A	1511315260	91.46
...	...	...

You should give a predict label for every observation in testing sets and submit you results. **Submitted file must be csv format and header must be "KPI ID, timestamp, predict"**.

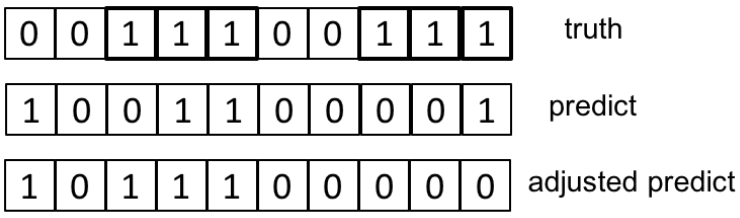
An example of submitted file:

KPI ID	timestamp	predict
A	1511315200	1
A	1511315260	0
...	...	...

Performance Metric:

The anomaly detection algorithm need give a predict label (0 or 1) for every observation in testing set. We use F-score on test set as the final performance metric in the ranking. In real applications, the human operators generally do not care about the point-wise metrics. It is acceptable for an algorithm to trigger an alert for any point in a contiguous anomaly segment if the delay is not too long. **For an anomaly segment with start point i, if any points between i to $T+i$ in the ground truth were detected, we say this segment is detected correctly, and all points in this segment are treated as true positives. Otherwise, all points in this segment are regarded as false negatives**. Meanwhile, the points outside the anomaly segments are treated as usual. For example (see the figure below), when $T=1$, the first anomaly segment was detected and the second segment was not, so the precision=0.75, recall=0.5. Baed on the above strategy, we calculate F-score. Evaluation script can bee seen in here.

$$F-score=\frac{2\times precision\times recall}{precision+recall}$$



In this project, you need to do the following jobs:

- Design a **generic** anomaly detection algorithm and submit your result on testing set in the website. The website will present a rank list of F-score.
- Submit a **report** including the details about your algorithm, such as data preprocessing (normalization? fill missing? *etc.*), algorithm implementation, parameter setting and so on.
- Submit runnable **codes**.
- Give a **presentation**.

Reference

[1] Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning. IMC 2015.

[2] Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. WWW 2018.

[3] Generic and Scalable Framework for Automated Time-series Anomaly Detection. KDD 2015.