COSC 2671 Social Media and Network Analytics

Lab 7

Event Detection

Learning outcomes:

- Perform basic event detection
- Practice Python coding

Requirements:

- A PC with Internet connection and Python installed (preference is version 3.X, but 2.7 is also okay, but unfortunately, we don't have resources to provide support if there are version incompatibility issues)

Resources:

- This lab worksheet (available on Canvas)
- Associated code in lab07Code.zip (available on Canvas)

Python Packages Required:

- pandas
- numpy
- matplotlib
- nltk

Lab Introduction

In this lab, we study how to implement some trend detection algorithms and use it to detect bursty keywords.

Data & Packages

This week, we will a pre-processed dataset of tweets about the English football/soccer team during the 2014 WorldCup. No do not need to install any new Python packages.

The data is in the WorldCupData folder. When you unzip the code for this lab, this will create this folder and data within. Note the data is separated into hours, but the provided notebook will be able to handle this.

Data Loading

Cells 1 and 2 are the same from previous labs, to extract out tokens from the tweet text. Cell 3 is the one that does the grunt of the data loading. It essentially goes through each file (which separates the tweet stream of the English football team by hours. We load each hour into an entry in the global data lists (llDataTime, llTweetsText, ldFreq). Note this isn't the only option, we can easily load them all into a Pandas dataframe and segment the time, but for this lab we have segmented according to the file delimitations. Run this cell to load the

data, it will print out the name of the files as it processes them. It might take a few seconds to run this – chill for awhile.

Constructing Expected and Observed Frequencies

Recall in lectures that to detect events, we needed to detect bursts in frequencies. We studied two methods, 'Simple frequency ratios' and 'Burstiness score". We will explore both in this lab. But first we needed to construct the expected and observed frequencies for the words in the tweets.

To compute these ratios/scores, we need expected and observed frequencies. There are several approaches, one of which is to compute the long term frequency averages and use that for expected, then compare against recent frequencies (observed). Another approach, which we take, is to divide the time series data into a number of (sliding) windows, and for each adjacent pair of windows, let the older (in terms of time) be the window which we compute the expected frequencies, and the newer window as observed frequencies. Hence we are looking for substantial changes between the windows.

Examine cell 6. For this lab, we just use windows of 1 hour, which aligns with the file structures, but we encourage you to explore longer windows. The Simple frequency ratio is compute in the following line:

```
dRatio[word] = float(obsFreq[word]) / expFreq[word]
```

which is the observed frequency over the expected frequency for the keyword 'word'.

As we are only interested in the trending words, we use a manual threshold to determine what is trending. At the top of the cell, we have a variable called 'trendThresholdFreq'. It is currently set to 10. It means for a keyword to be trending, it most have a ratio of greater than 10 to be considered trending.

Run the cell, the output will only print out keywords that satisfy this threshold condition. Explore varying the threshold to search for the balance between false positives (words that aren't trending but because threshold was too long it got included) versus false negatives (words that are trending but threshold was too high to include it). Note that there is no absolute right answer for this, it is one of the challenges of using a manual threshold, have a go at varying it to get hands on experience about this.

Exercise 1:
Add a new cell. Now implement similar code but use the Burstiness score instead of the Simple frequency ratio.
Hint: the code for this is very similar to cell 6, the one for Simple frequency ratio, apart from computing the burstiness score. Look up lecture slides and ask your friendly demonstrator about the score if you need help.

Once you finished, print out the trending keywords, similar to cell 6, and compare the two approaches to detect bursts.

Plotting and Comparing with Frequencies

In cell 7, we plot the frequencies compared with the computed ratios. This gives a quick visual method to see how well the ratios are working. To avoid cluttering the figure, only ratios that are above the trendThresholdFreq are included and we only plot the top keywords in terms of their maximum bursts (there are many ways to select which top keywords to select, again want you to explore this as part of the process towards developing independence in data and approach exploration).

There is the parameter 'keywordNum' that specifies how many keywords plots to print. Currently it is 5, but you can change this as you wish. Run the code. Do the frequency bursts correspond with the times that the ratio is above the thresholds? Do the selected keywords interesting – if not, how might we alter this?

Exercise 2:
Repeat the plotting for the Burstiness score you computed in exercise 1. Are there difference with Simple Frequency scores?