ECS637U/ECS757P - Digital Media and Social Networks Group Project Deliverable 2

Domantas Miliauskas
d.miliauskas@se16.qmul.ac.uk
Report writing
25%

Gurpal Jabble
g.jabble@se16.qmul.ac.uk
Report writing
25%

Harvey Randall
h.randall@se16.qmul.ac.uk
Report writing
25%

Mehmet Keven
m.e.keven@se17.qmul.ac.uk
Report writing
25%

***Abstract*—Abstract?**

***Index Terms*—Modularity: a measure of the extent to which nodes of the same type are connected to nodes of the same type Edge betweenness: the number of shortest paths between pairs of nodes that run along an edge [?]**

I. Introduction

In this report, we gain further insight into three Glastonbury festival twitter datasets that consist of tweets collected through different stream collection techniques; Baseline (BL), Adaptive (AD), and Extra (EX). We aim to implement the Girvan-Newman community detection algorithm on each dataset to detect topic clusters, which will be analysed based on sentiment using a Naïve Bayes classifier to determine whether a given cluster contains 'Good' or 'Bad' tweets.

II. Background

A. Sentiment Analysis

There has been a growing interest in mining opinions and sentiment through Twitter, as nowadays everything is shared on this social platform with over 250 million tweets being made each day (Twitter, 2012). Many papers are available on the topic of performing sentiment analysis on tweets with the use of machine learning techniques such as the Naïve Bayes classifier, whether it be to see the overall opinion on a social event or to predict the financial market.

One paper describes a system for real-time analysis of public sentiment towards presidential candidates in the 2012 US election as expressed on Twitter. They have effectively gathered over 36 million tweets about the election. With this data, they attempt to explore whether Twitter provides insights into the unfolding of the campaigns and indications of shifts in public opinion.

To obtain a labelled train dataset, they used a crowdsourcing approach and used Amazon Mechanical Turk (AMT) to get a varied population of around 800 annotators. They designed an interface that allowed annotators to participate anonymously. They were shown a series of tweets and asked to annotate the tweets' sentiment. As a result, they had 17000 tweets for training.

With new tweets coming in, they pre-process them with Christopher Potts' basic Twitter tokenizer. A Naïve Bayes model on unigram features is used for the classification. The features are calculated from tokenization of the tweets that attempt to preserve punctuation that may signify sentiment as well as twitter specific phenomena. Based on the data collected, their classifier performs at 59% accuracy on the four-category classification of negative, positive, neutral, or unsure.

In conclusion, one finding is that their infrastructure and sentiment model evaluates public sentiment changes in response to emerging political events and news as they unfold. For instance, during the Republican Primary debate on Jan 19, 2012 in Charleston, NC, Newt Gingrich, a republican candidate, was asked about his ex-wife. His negative sentiment tripled in just two minutes.

B. Community Detection

Community detection is the process of finding groups of nodes within networks that are highly connected to one another and few connections to nodes outside of this group. When we look at small networks it can be quite easy to detect small communities but when networks become much larger it becomes difficult to identify communities. Identifying communities is worthwhile though as it enables us to find related nodes and can reveal the nature of social interaction within a network.

Community detection can be divided into two separate areas:

1) Hierarchical community detection
2) Overlapping community detection.

The former involves dividing the network in such a way that nodes can only exist within one community or another but never more than one, whereas the latter allows nodes to exist within multiple communities. We will focus on hierarchical community detection which can again be divided into separate areas:

1) Divisive detection methods
2) Agglomerative detection methods

Divisive methods can be considered top down approaches to community detection as they begin with the complete network

and iteratively remove edges, dividing nodes into separate, more densely connected groups. Agglomerative methods can be considered bottom up approaches to community detection as they initially consider each node as a single community and merge communities until every node belongs to the same community.

One method of edge removal is to remove bridges and local bridges between nodes to produce these new groups of nodes, this can be problematic though as sometimes there may be no bridges or local bridges in the network to remove. Another method is to remove edges using the notion of edge betweenness to find edges that are least central to communities [1]. This is calculated by finding the number of shortest paths between nodes that use this edge. The edge with the highest edge betweenness value is then removed to separate the nodes into separate groups. This approach still allows us to find edges for removal even if bridges do not exist in the network and therefore allows us to still find communities.
This approach runs with time complexity $O(m^2n)$ where m is the number of edges in the network and n is the number of nodes. This is not a problem for networks with small n however as networks become larger the time it takes to complete increases rapidly, particularly when trying to find communities in social networks which can have very large n and m values such as the dataset we are working with.

Agglomerative methods are performed in the opposite way to divisive methods, taking a bottom-up approach as opposed to a top-down approach. Each node begins as its own cluster and clusters are merged successively until all clusters have been merged into one cluster. One method of doing this is by measuring the Euclidean distance between clusters and merging the closest clusters until one cluster is formed. Euclidean distance is just one example of a distance metric that might be used, another commonly used metric is metric is modularity. Modularity is a measure used in networks to describe the strength of a community division, a high modularity value indicates a set of nodes is densely connected whereas a low modularity value indicates nodes are sparsely connected. Fast agglomerative algorithms can be implemented using greedy optimisation of modularity to efficiently find communities in large networks [2] [3] that run with better time complexity than the divisive algorithm described above. These algorithms work by merging clusters which provide the best increase in modularity at each iteration.
These algorithms can be graphically represented by a dendrogram. These visualisations are typically displayed bottom to top, ie at the bottom is each individual community (each node) and as you travel upwards branches show which communities have been merged.

*C. **EXTENSION:** Page Rank and its application to network analysis*

PageRank is an algorithm used to rank web pages in Google search engine results. It was developed by the founders of Google - Larry Page and Sergey Brin, and its functional prototype was released in 1998. The goal of PageRank is to get a measure of how popular webpages are based on other pages that are linked to them. Since not all links share the same value, the importance of a web page is calculated by the number, as well as the quality of links pointing to that page. PageRank is a recursively defined measure as a page becomes important if important pages link to it. With that said, the underlying assumption is that more important websites are likely to receive more links from other websites.

One way to think about PageRank is to imagine a random surfer on the web, following links from page to page. The page rank of any page is roughly the probability that the random surfer will land on a particular page. Since more links go to the important pages, the surfer is more likely to end up there. Google's random surfer is an example of a Markov process, in which a system moves from state to state, based on probability information that shows the likelihood of moving from each state to every other possible state.

When calculating PageRank, the distribution is evenly divided among all webpages at the beginning of the computational process. In a network with n nodes (nodes being webpages), we assign all nodes the same initial PageRank of 1/n. Then, we choose a number of steps k and perform a sequence of k updates to the PageRank values:

- Each page divides its current PageRank equally across its out-going links and passes these equal shares to the pages it points to. Also, if a page has no out-going links, it passes all its current PageRank to itself. Each page updates its new PageRank to be the sum of the shares it receives.

The PageRank computations require several iterations through the collection of webpages to adjust approximate PageRank values to more closely reflect the theoretical true value. This is when we get to the convergence case, where the PageRank values stop changing.

III. DATASET

We will be using a dataset of tweets collected during the 2013 Glastonbury Music Festival using a STRIM [4] framework to collect the tweets and detect sub-event occurrence in real time. This dataset is comprised of three streams:

- **Baseline**: the stream of tweets collected from the static user-specified keywords
- **Adaptive**: the stream of tweets collected using the adaptive crawler to dynamically update the keywords
- **Extra**: the stream of tweets collected only by the adaptive crawler ie the difference between the adaptive and baseline streams

In particular we will be analysing the hashtag co-occurrence network of each of these streams to identify hashtags which are used in combination with one another. Each node of our graphs will represent a hashtag and edges will represent

these hashtags being used in the same tweet, the weight of each edge will be determined by the frequency the hashtags are used together: the larger the edge weighting the more frequently they co-occur. Some preprocessing of the data was required to create a co-occurrence list that could be used to create a co-occurrence network. This was done by first loading the SQL files for each stream and iterating over their contents, the hashtags were extracted using a simple SQL query `SELECT 'hashtags' FROM <table>;` where `<table>` is the name of the stream. The returned string was split by space and sorted alphabetically, if the resulting list contained more than one element then the results were stored in a dictionary used to count co-occurrence before finally being written to CSV in a format that can be parsed by Gephi as well as networkx.

Initial analysis of the network shows that the graph consists mostly of low degree nodes with a rapid decrease in the frequency of nodes with specific frequency as node degree increases demonstrating that our networks display a power law distribution. The graph below shows the frequency of node by node degree shown on a log-log graph.

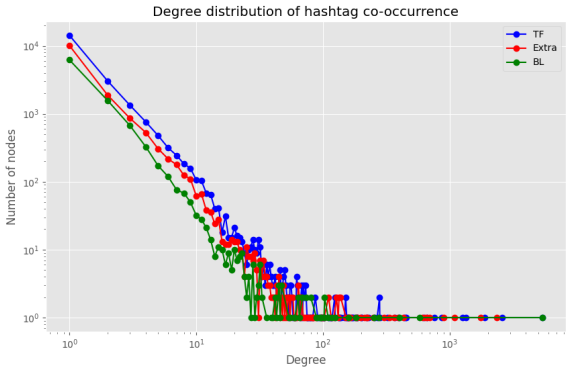

Fig. 1. Degree distribution of nodes for each dataset on a log-log graph.

A. Basic Network Statistics

	BL	AD	EX
Number of Nodes	10,777	24,651	16,787
Number of Edges	24,549	86,857	64,055
Network Diameter	8	9	9
Avg. Path Length	2.61	3.090	3.292
Average Degree	4.556	7.047	7.632
Avg. Clustering Coefficient	0.858	0.864	0.871
Modularity	0.374	0.607	0.565
Eigenvector Centrality	0.257	0.349	0.291

IV. Approach

A. Louvain Community Detection

We chose to use the Louvain agglomerative community detection algorithm [3] because it runs quickly whilst also maximising the modularity at every step. It is also the algorithm implemented by Gephi to calculate the modularity of a graph so we should see similar partitions to those produced by Gephi and should find a similar, if not exactly the same, number of communities.

The community detection algorithm is implemented using the Python igraph [5] implementation. First a weighted edge list is produced from the datasets where each vertex represents a hashtag, an edge represents that the hashtags co-occur, and the edge weight represents the frequency with which they have co-occurred. From this we create a graph using igraph and call its `community_multilevel` method to perform the community detection which returns a `VertexClustering` object which can be iterated over; at each iteration a community division of co-occurring hashtags is returned. We need to save the resulting community divisions to perform sentiment analysis later, so we use this to create a dictionary where the key is the community index and the value is the community division. Finally we save the dictionary as a JSON file which can easily be loaded later.

B. Sentiment Analysis: Naïve Bayes Classifier

The use of natural language processing is required in order to identify the overall 'sentiment' of an identified community of tweets from the given dataset. Sentiment refers to the overall view or opinion that is expressed in the tweets of each community. In our case, we have chosen to build a naïve Bayes classifier to do this which works by classifying the communities on overall positive, negative or neutral sentiment.

The classifier is implemented as follows. It loads in a train and test data set. The test dataset is the set of tweets for a given community and the train data set is a precompiled set of words/phrases that are labelled ('positive', 'negative' or 'neutral'). The training data set is used to train the classifier based on training features that are extracted from the train data set. Once the classifier has been trained, it can then be used to identify the overall sentiment of the unseen given test data set, which in our case is the set of tweets for a particular community. The classifier goes through the community tweets and applies a label to each based on what it thinks the sentiment of it is. When this has been completed, the labels are summed and the label with the highest count is considered the overall sentiment of the community.

V. Results

VI. Conclusions

References

Please number citations consecutively within brackets [1]. The

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even

if they have been Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English

REFERENCES

[1] M. Girvan, and M. E. J. Newman. "Community structure in social and biological networks". Proceedings of the National Academy of Sciences, Jun 2002, 99 (12), pp. 7821-7826

[2] A. Clauset, M. E. J. Newman, and C. Moore. "Finding community structure in very large networks". Physical Review E, Dec 2004, 70 (6)

[3] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. "Fast unfolding of communities in large networks". Journal of Statistical Mechanics: Theory and Experiment, Oct 2008, 2008 (10)

[4] L. Tokarchuk, X. Wang, S. Poslad. "Piecing together the puzzle: Improving event content coverage for real-time sub-event detection using adaptive microblog crawling". PLoS ONE 12 (11)

[5] G. Csardi, and T. Nepusz. "The igraph software package for complex network research". InterJournal, 2006, Complex Systems, pp. 1695

VII. STYLING FIGS AND STUFF

A. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 2", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 2. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.