Replication of Event Detection and Summarization using Phrase Network

ALLIE LAHNALA

1 INTRODUCTION

In this paper, I present a replication study of "Event Detection and Summarization using Phrase Network" [Melvin et al. 2017]. The original work proposed a model they call `PhraseNet`, an unsupervised event detection model for Twitter using textual features.

Motivation. The `PhraseNet` paper was published in *ECML PKDD 2017* and had not gained much attention. However, I was curious about its implementation because I am interested in graph applications for language, and it seems interesting that the entire event detection in this method was mainly based on textual features.

Method summary. `PhraseNet` can be broken down into the following steps:

(1) Extract high frequency phrases from set of tweets
(2) Graphically represent relationships between these phrases in terms of tweet co-occurrence in a timestep.
(3) Apply a community detection algorithm over this graph to establish initial event candidates by phrase communities.
(4) For each event candidate, determine peaks in terms of relative frequency of its phrases across the timestep distribution.
(5) Determine an event candidate is an event by analyzing the number of peaks across timesteps, the peak height intensities, and the standard deviation of peak heights across timesteps.
(6) Evaluate the performance based on precision and recall on a set of manually determined target events.
(7) Consider the phrases of the event's phrase clusters to be the event summary, and compare these summaries to baseline Twevent [Li et al. 2012].

1.1 Notation

Throughout the paper the authors often refer to the same concept with different terms and notations, and different concepts with the same term and notation, which leads to confusion when trying to replicate their methods. Additionally, they introduced equations before or without providing a definition of the terms of the equation. Before describing my implementation, I would first like to provide clarity through a brief discussion of the notation and terms they used for important components.

A *phrase cluster* P is a set of phrases p at a particular timestep t determined by a community detection algorithm. In some points, the authors use *phrase cluster* and *event candidate* synonymously. However, they also use *event candidate* to refer to *trending topics* or *peaks*. I will describe *peaks* in section 2.1.5, but I will not use the term *trending topic* as the author did not use this terminology more than once or provide a clear definition of *topic*. In the meantime, consider an *event candidate* to be a *phrase cluster*. However, the set of *event candidates* that are being considered is being reduced throughout the approach, between when they initialize a set of *event candidates* using community detection over a graph, reducing them by merging similar communities from other graphs (see 2.1.4), and when they eliminate candidates in the peak detection and event determination stage (see 2.1.5).

A *phrase cluster weight* is the sum of the relative frequencies w of each phrase r in a phrase cluster P_i at timestep t, or $\sum_{p_r \in P_{i,t}} w_r$. This weight will be used for the peak detection computations described in section 2.1.5. The authors introduced this notation when discussing candidate merging (see section 2.1.4), however they abandoned it before providing the equations for the peak detection computations, using a notation that looks similar to the phrase count notation used for determining the Jaccard coefficients between two phrases (see section 2.1.3).

2 IMPLEMENTATION OF THE PHRASENET REPLICATION

I implemented the `PhraseNet` model and replicated the experiments with retrieving a set of event targets.

2.1 Implementation

2.1.1 Preprocessing. In the spirit of studying the reproducibility of this paper, I followed the text preprocessing steps that they listed which were: expand all contractions, remove all non-English characters, and remove all stopwords. I implemented the author_preprocess in data_prep/preprocessing.py to only use those steps as best as I could interpret them. They do not specify how they tokenize, so to start I split on spaces removed contractions with using an open-source library[1]. Then I removed non-english alphanumeric characters since they did not specify whether or not they kept punctuation, and finally stopwords from the NLTK stopword corpus [2]. There are many possible different implementation decisions under these specifications, so I discuss preprocessing concerns and modifications I made in 6.1.

2.1.2 ToPMine. ToPMine is a phrase mining approach that originally has two main components: phrase-mining with text segmentation and phrase-constrained topic modeling [Ahmed El-Kishky 2014]. It became clear after implementation that the the `PhraseNet` method only required the first phase, which is used to obtain frequent phrases for a timestep and their counts from tweets and to segment each tweet into those phrases. The authors did not provide any implementation details about this step. This method requires the specification of a minimum-support parameter (minimum number of occurrences of phrase), a maximum phrase size parameter, and an α parameter that defines a threshold for merging two words into a phrase. The authors specified that they used a minimum-support of 40 and max phrase size of 5 but did not specify or discuss α.

Author's address: Allie Lahnala, alcllahn@umich.edu.

[1] https://github.com/kootenpv/contractions
[2] https://www.nltk.org/

In my replication, I used an open-source implementation of the phrase-mining component of ToPMine.[3] This codebase also included its own version list of stopwords and preprocessing steps, which are discussed in section 6.1. My data is collected in four-hour chunks, so I treated each four-hour timestep as a corpus and each tweet a document as input to ToPMine. Since this method can be done over each timestep independently, I parallelized this process, however the authors did not specify how often or how they take in new data. I provide a discussion about the author's and my parameter choices in section 3.3.

2.1.3 FPGrowth. After using ToPMine to determine frequent phrases for a timestep, the authors then use the FP-growth algorithm [Han et al. 2000] to determine the most frequent co-occurring phrases. They used this method because they found that brute-force tallying of co-occurrences cause a bottleneck. The output of this step is used to construct the phrase network G_t by creating an edge between two co-occurring phrases with the Jaccard coefficient of the phrases as the weight, $w_e = \frac{F(p_a \wedge p_b)}{F(p_a)+F(p_b)}$ where $F(p_a)$ and $F(p_b)$ are the relative frequencies of phrase a and phrase b in timestep t's corpus respectively. This algorithm takes in a minimum support parameter, for which the author specified they used 8. Note that G_t can be empty depending on the results of ToPMine and FPGrowth.

The authors did not specify how they implemented this step. I started with the pyfpgrowth python library[4], but made modifications so that the output would work for the specified phrase network structure. The original implementation first finds frequent patterns and then outputs the patterns that meet a confidence threshold. In this case, the patterns could contain more than two phrases. Therefore, over each detected pattern, I process each combination of two phrases and compute their Jaccard coefficient.

It is not always the case that the pyfpgrowth program had an intermediate value stored that had the co-occurrence count of each of the two-phrase subsets of the full patterns. Therefore I used the partitioned output of ToPMine to compute the Jaccard Coefficient of these combinations. The authors did not specify how they handled pattern subsets or the Jaccard Coefficient computation, so I am unsure if this would be close to their implementation's efficiency or effectiveness.

2.1.4 Event candidates.

Community detection. In the next phase, we determine an initial set of event candidates using the phrase network. The authors use Louvain community detection [Blondel et al. 2011] to create phrase communities over the network, and each community of phrases, or phrase cluster, becomes a unique event candidate.

Alike the other phases, the authors did not provide implementation details. I used the python-louvain[5] library implementation of Louvain community detection. This method outputs a dictionary that maps nodes to their communities. Figure 1 shows an example of the community structure. I then constructed a community to nodes dictionary, so that each community key maps to the set of phrase nodes that belong to the community. In running this phase, I iterated through each timestep using its phrase network composed in the previous step, and I saved all phrase clusters for each timestep in one large dictionary of *initial candidates*, as this is useful for the next step of merging. The dictionary keys are $\{0, \ldots, T-1\}$ where T is the total number of timesteps t, and the values are dictionaries $\{0 : P_{0,t}, \ldots, P_{n,t}\}$ where n is the number of phrase clusters at timestep t.

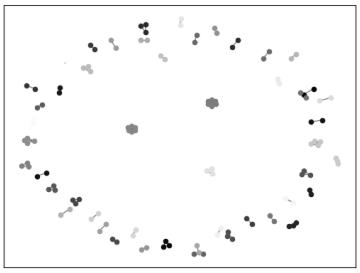

Fig. 1. Example of community structure

Candidate merging. For each phrase cluster in timestep t ($F(P_{i,t})$), the authors determine if any phrase clusters in timestep $t+1$ ($F(P_{i,t+1})$) are similar enough to merge them together. They define phrase cluster similarity by equation 1, and merge if the similarity is greater than a threshold, which they set to 0.5.

$$similarity = max\left(\frac{\sum_{p_s \in (P_{i,t} \cap P_{i,t+1})} w_s}{\sum_{p_r \in P_{i,t})} w_r}, \frac{\sum_{p_s \in (P_{i,t} \cap P_{i,t})} w_s}{\sum_{p_j \in P_{i,t+1}} w_j}\right) \quad (1)$$

In my implementation, I iterate over each timestep t where $G_t \neq \emptyset$, and if $G_{t+1} \neq \emptyset$ I compute the intersection of phrases for each cluster. Then, I compute the sum of the phrase weights (relative frequency in a timestep corpus) of the intersection cluster with respect to t's corpus, $t+1$'s corpus, and the s space (t and $t+1$ corpuses combined) in order to finally compute the similarity with equation 1.

How the merging of similar phrase clusters was done was not said in the paper, so I had to make an implementation decision. To merge, I treated the intersection $P_{i,t} \cap P_{i,t+1}$ as a new phrase cluster event candidate. I removed $P_{i,t}$ and $P_{i,t+1}$ from the *initial candidates* set, and then added $P_{i,t} \cap P_{i,t+1}$ to the set of event candidates at timestep t. For future computations for the peak detection phase, I saved a set of *secondary candidates* in a new dictionary. Similarly, keys $\{0, \ldots, T-1\}$ map to dictionary values $\{0 : P_{0,t}, \ldots, P_{n,t}\}$, but these dictionary phrase cluster values map further to its phrase cluster weight. A phrase cluster $P_{i,t}$'s phrase weight is sum of the relative frequencies of each phrase in the cluster in the timestep t tweet corpus, unless it was a merged candidate cluster ($P_{i,t} \cap P_{i,t+1}$) where I save ($\sum_{p_s \in (P_{i,t} \cap P_{i,t+1})} w_s$), the sum of the phrase weights in the s space of t and $t+1$. Since the authors did not specify how they did this, this was just a decision that could have gone other ways,

[3] https://github.com/anirudyd/topmine
[4] https://pypi.org/project/pyfpgrowth/
[5] https://github.com/taynaud/python-louvain

such as simply eliminating $P_{i,t+1}$ and keeping $P_{i,t}$ with its cluster weight with respect to t, instead of taking the intersection cluster and weight.

2.1.5 Peak detection. In the next phase, the authors determine event candidate (phrase cluster) *peaks* based on their phrase cluster weights. In figure 2, I highlight the components of their definition of a *sliding window mean*. The main idea is that for each phrase cluster, they compare its weight across timesteps within the sliding window size. They do so by computing the mean weight of the cluster across those timesteps, then for each individual timestep in that window, they compute the z-score to determine how many standard deviations away from the mean the weight is. Then they eliminate event candidates that do not meet a z-score threshold θ. The remaining candidates are then sorted by their total number of peaks α (least to greatest), the standard deviations of their peak heights β (greatest to least) where peak height = cluster weight at a timestep, and the maximum heights of the clusters χ (greatest to least). Then, they define a threshold for each of these variables and eliminate the candidates that do not meet all three of those thresholds, resulting in their final determination of events.

They specify that their sliding window is "midnight-to-midnight," but do not specify whether that window is made up of 24 timesteps (phrase clustering by hour) or some other timestep size. They used 3 for θ, 10 for α, .05 for β, and .5 for χ.

$$\mu_T = \frac{1}{\tau} \sum_{t=1}^{\tau} \omega_t \left(\sum_{m=1}^{k} \mathcal{F}(p_m^{(t)}) \right)$$

Fig. 2. Pink is the sliding window mean. Purple is 1 over the sliding window size. Yellow is a dampening coefficient parameter. Blue is the number of phrases in a phrase cluster, or event candidate. Green is the frequency of the phrase. In the true peak detection, they use the phrase weights, not the phrase count. Phrase weight it the relative phrase frequency for the timestep.

To implement this, I first took in the *secondary candidates* from the previous phase. Then for each phrase clusters across all timesteps, I sorted the phrases of the cluster alphabetically and joined them into one string by commas. This enables me to create a dictionary of unique phrase clusters, so that I can find the same phrase cluster across different timesteps in order to do the computation described above. I considered that it is possible to consider all phrase weights in the surrounding timesteps that match a phrase in cluster under consideration, but decided to implement it this way (only computing weights from exact cluster matches) since this is how the equations provided by the authors are interpreted. In discuss my parameters choices in the following sections.

3 REPLICATED EVENT DETECTION EXPERIMENT

In this experiment, I replicated the retrieval of target events with my `PhraseNet` implementation during the same window as the authors, January 1st, 2015 to March 31, 2015. As will be discussed further, the reproducibility of their experiment is lacking in some important aspects, therefore for to further analyze my `PhraseNet` implementation, I experiment with different sets of parameters to show their impact to the behavior of the model. This type of analysis was not performed in the original work, and they did not provide a discussion on the reasoning behind their parameter choices.

3.1 Datasets

I experimented with the same time period of tweets that the authors used in the paper, which was January 1st, 2015 through March 31st, 2015. The author's dataset had 2,747,808 for 90 days, so approximately 30531 tweets per day and approximately 1272 per hour. By closely following their preprocessing description I obtained 5,154,791 tweets for the 90 days (7275 tweets per day and 2386 tweets per hour). From now on I will refer to this dataset as the `AuthorSet`. I created an additional dataset after analyzing the behavior of the method over the first dataset and seeing that there are clear improvements to the preprocessing that could be made. In this dataset, I have 4,779,980 tweets for the 90 days (53111 tweets and 2213 tweets per hour). I will refer to this dataset as the `ModifiedSet`. Although both the `AuthorSet` and the `ModifiedSet` were larger than the dataset used by the author's in their experiments, I did not reduce my datasets to match their size because, as will be discussed in in following sections, we likely have very different looking data resulting in very different performance behaviors.

3.2 Evaluation metrics

Target Events. The authors evaluated `PhraseNet` by manually defining a set of target events from the January through March 2015 time frame, and computing precision, recall, and F1 score for the retrieved events. They did not list the events that they chose, so I emailed the authors to ask if they would share the target events but I did not get a response. Therefore, I did my best to follow how they came up with target events by using holidays and events listed for those months on the "Time and Date" website,[6] events listed in the "On This Day"[7] website, and listing all of the events that the authors including in their results. There were 77 events across these lists in total, which are listed in PaperEvents.txt in the code repository. After merging events that were the same across these lists there were in total 70 events.

Additional measures. The output of each stage of `PhraseNet` can contribute to the performance and efficiency. Therefore, for each experiment I recorded the intermediate outputs which could play a role in the final outcome. These are: the number of frequent phrases identified by the ToPMine phrase mining method, the total number of edges across all timestep phrase networks which were determined with the FPGrowth stage, the number of phrase cluster event candidates after Louvain community detection and merging similar events, and the final number of events. For each experiment, these values are listed in Table 3 under "Freq. phrases," "$|E| \in \bigcup_{t=1}^{T} G_t$," "2nd. Cand.," and "Final Events" respectively. Also experiments are timed and their duration is reported in seconds.

[6] https://www.timeanddate.com/holidays/us/2015
[7] https://www.onthisday.com/events/date/2015/

3.3 Experimental setup

3.3.1 Environment. All experiments reported in this paper were run on The Great Lakes Slurm cluster,[8] using 20 CPU threads for the parallelized stages.

3.3.2 Parameters. I experimented with different `PhraseNet` parameters on the `AuthorSet` and the `ModifiedSet`. The set of parameters I used are listed in 1 and named under the "Param. Set" header. Not all parameters necessary for PhraseNet were specified by the authors, for for the `AuthorParams` parameter sets I followed the parameters specified by the authors and used a reasonable guess for the parameters not specified. For the remainder of this section, I will define and describe each parameter. Discussion on these choices and their effects will follow in section 3.4.

Stopwords. There are two different sets of stopwords used in these sets, *nltk+* and *mods*. The *nltk+* set is based off the NLTK English stopword corpus. In the original list, there are cases where negations of stopwords are listed but not positives (e.g., wouldn and wouldn't are listed but would is not), so I augmented the set by including the positives of these stopwords. The *mods* set builds off *nltk+* even further. While performing initial analyses of the output of the intermediate steps and resulting events, I found that some words could be considered "stopwords" to Twitter-specific language because of their Twitter-specific relevance, for example, "follow," "followers," and "subscribe." For these terms and other words that were not in *nlt+* but seemed like necessary stopwords for this task, I included them in the *mods* set. These stopword sets are listed in their entirety in section B of the Appendix.

ToPMine α. This parameter (TM α) defines a significance threshold for word collocation to determine if the words should make up a phrase. A higher α would correspond to fewer but potentially higher quality phrases than a lower one. The authors did not specify the value they used for this parameter. On a open source repository hosting the author's original code (I did not use this because it would not run), they specified that they typically use 3 to 5[9]. Therefore for most experiments I used 5 for this value, and to demonstrate its effect I used 10 in `params-3`.

ToPMine minsupport. This parameter (TM minsupport) denotes the minimum number of word collocation occurrences before considering determining that the collocation is a phrase. The used 40 for this parameter. In my parameter sets, I lower this value to 10.

Topmine n-gram. TM n-gram is the maximum phrase size that ToPMine will consider. The authors specified that they used 5 for the maximum phrase size. I keep this parameter consistent at 5 throughout my experiments.

FPGrowth minsupport. FPg minsupport is the minimum number of co-occurrences between elements under consideration by the FP-Growth algorithm before determining it to be a pattern. In this case, the elements are the phrases identified by Topmine. The authors used 8 for this value, and I experiment with a generous value of 1.

Peak significance threshold θ. Denoted θ in Table 1, this value defines the minimum number of standard deviations above the sliding window mean a phrase cluster weight must be or else it is eliminated from the set of event candidates. The authors used 3 for this parameter which I use in my experiments as well.

Maximum peaks α. The α parameter denotes the maximum number of peaks of an event candidate. The original work used 10 for this value, which I use as well.

Minimum peak height standard deviation β. The β parameter specifies a minimum for an event candidate's standard deviation of peak heights. The original work used .05 for this value. I experiment with lowering this value.

Minimum peak height χ. The χ parameter specifies a minimum for an event candidate's peak intensity, or height of the peak. The original work used .5 for this value. I experiment with lowering this value.

3.4 Results and Analysis

The way that the authors presented the parameters affects the reproducibility of their experiment, but this will be discussed more in 6 definitions affect the reproducibility of their experiment. In this section, I will summarize the full results of this experiment and an analysis of the `PhraseNet` parameters. The full results of this experiment is included for reference in the Appendix in Table 3. For reference on the type of events and phrase cluster produced by the method, the top results from *params-4-mod* on *Modified set* are listed in in 4 in the Appendix.

In the parameter experiments reported, the ToPMine n-gram, θ, and α thresholds, remain constant. I had experimented with other values for the n-gram parameter, but I found that 5 served well for this value, likely due to the shortness of Tweet text, and larger values added complexity to the model. In the case of θ, there tended to be only one or two phrase cluster weights above zero in the sliding windows, and the standard deviations above the sliding window mean for these cases were for the most part just above 3. I maintained α at the maximum defined by the authors because in my experiments, this value ended up being quite generous, enabling me to observe more retrieved events. Most of the events detected in my experiments had very few event peaks, the top ones only having one.

With the author's parameter specifications provided in the paper, my implementation returned no final events on either dataset, except in one case on the the `ModifiedSet`. But in that case, it retrieved "love much, please follow," for March 31st, which is likely noise from fans of musical artists (there are a lot of tweets toward Harry Styles from people saying they love him so much, and asking him to please follow them). I believe this problem began with the parameter they specified for the very first stage, the minimum support value for ToPMine. With this value set to 40, I obtained few frequent phrases, and thus fewer edges between co-occurring phrases in my graph. In the `params-1` set experiments, I only lowered the ToPMine minimum support to observe if this is a factor of the number of phrases ToPMine was able to identify, however the results of this

[8] https://arc-ts.umich.edu/greatlakes/
[9] https://github.com/JSybrandt/ToPMine

Param. Set	Stopwords	TM α	TM minsupport	TM n-gram	FPg minsupport	θ	α	β	χ
AuthorParams	nltk+	*5*	40	5	8	3	10	.05	.5
AuthorParams	nltk+	*5*	40	5	8	3	10	.05	.5
params-1	nltk+	5	10	5	8	3	10	.05	.5
params-1-mods	mods	5	10	5	8	3	10	.05	.5
params-2	nltk+	5	10	5	8	3	10	.005	.05
params-2-mods	mods	5	10	5	8	3	10	.005	.05
params-3	nltk+	10	10	5	8	3	10	.005	.05
params-3-mods	mods	10	10	5	8	3	10	.005	.05
params-4	nltk+	10	10	5	1	3	10	.0005	.005
params-4-mods	mods	10	10	5	1	3	10	.0005	.005

Table 1. Experiment parameter definitions.

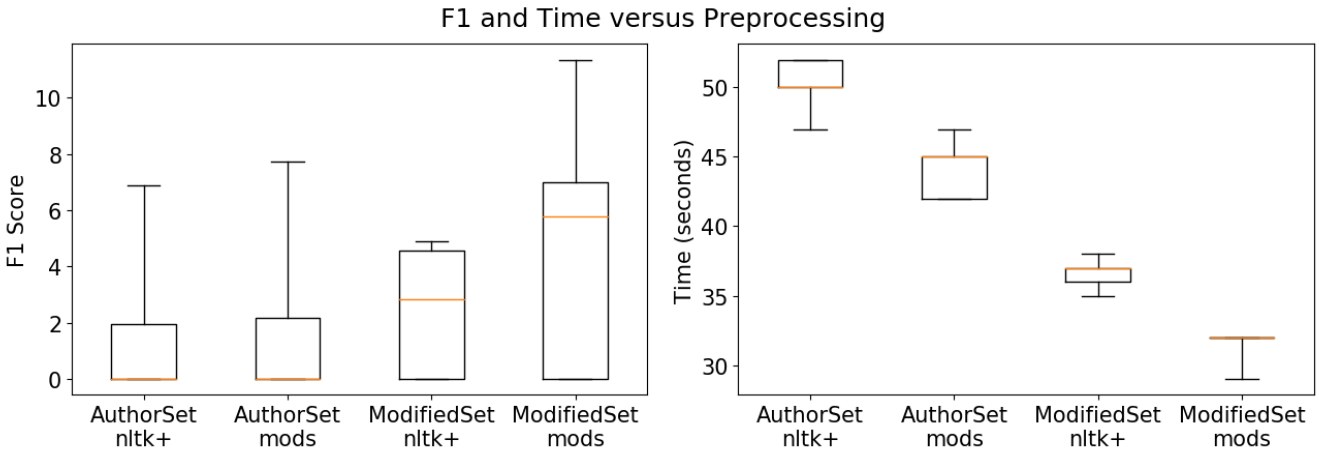

Fig. 3. Results in terms of F1 score and Time for the preprocessing methods that use the original AuthorSet or the ModifiedSet as well as ones that use nltk+ stopwords versus my modified stopword list.

experiment show that changing this threshold alone will not help retrieve any events on my datasets.

With the `params-2` sets we can finally see a bump in the number of final events. The modification was reducing β and χ by a factor of 10. This could suggest that in my data, the phrase clusters have smaller weights than in the authors data, or intuitively the event phrases appear more often than other phrases. This could signify that my data collection method affected the types of tweets I obtained, however, this is cannot be concluded because the authors did not specify their data collection method.

In the `params-3` sets, I show the affect of the ToPMine α parameter. By increasing the parameter to 10 from 5, the number of edges in the graph are greatly reduced, thus lowering the set of possible candidates before the community detection phase. This is because increasing this parameter makes the requirement for a group of words to occur much more often than other groups of words in order to be considered a phrase. Though we still obtain a number of phrases, these phrases are more unique and co-occur less frequently, which affects the output of the FPGrowth algorithm.

In the `params-4` set, I allow for very liberal thresholds, which thus increases the recall. Though the recall improves, the results are not as precise as the `params-2` sets.

To demonstrate the effect of β and χ, figure increases are demonstrated visually in Figure 4 plots these points from these experiments. Though the two sets of stopwords used in these experiments did not vary the F1 scores, the box plots in Figure 3 demonstrate the variation that preprocessing can have during runtime. Even though the *mods* stopwords was a larger set, the overall runtime improves because of the reduced output from ToPMine, and in turn the graphs. However, the results did not generally improve, but on the `ModifiedSet` we see precision improvements.

The effect of these parameters in my experiments go to show the importance of discussing reasoning behind parameters. Moreover, it is important to not leave out parameters used in intermediate steps, as was shown in the case for the ToPMine α parameter.

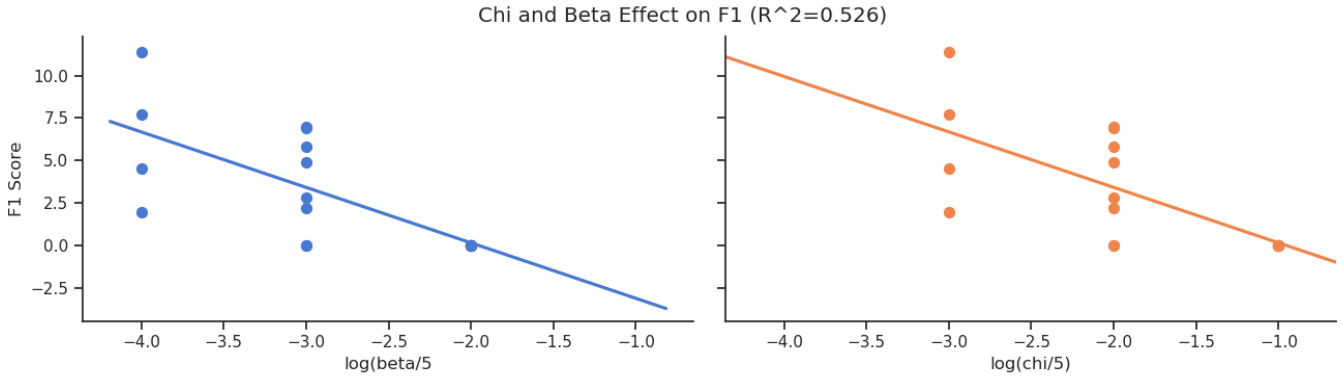

Fig. 4. Effect of changing beta and chi and it's effect on F1 score. We see that smaller values correlate with better results (using $log_10(x/5)$ for ease of viewing).

4 SCALABILITY EXPERIMENT

4.1 Data

To see how the method scales, I created a partially synthetic dataset of 195,714,150 tweets. The LIT Lab[10] has been building this dataset by constantly (with some blips from power outages or machine shutdowns) collecting tweets since March 21st, 2014. There are around 100 million well-formed tweets from many languages. To get an even larger dataset for a scalability experiment, I doubled the tweets and leniently preprocessed them, keeping non-English tweets and bot/automatic tweet sources. The collection program grabs up to 10,000 tweets that that have user description and location fields enabled every four hours, and dumps them into a new text file every four hours. The first few months of its collection it was grabbing fewer tweets per dump. To get progressively different dataset sizes, I started with 100% of the data, and then took smaller percentages (90%, 80%, ... 0.1%) of the tweet files starting from the most recent file. The actual percentages of tweets in each subset are listed alongside their results in Table 2. The reason the percentages are not separated by clean intervals is due to the variance of the number of well-formed tweets in each tweet file, and also because the program was originally collecting fewer tweets.

4.2 Setup

Environment. Each of the `PhraseNet` experiments on the data subsets were run on an allocation on The Great Lakes slurm cluster with 20 CPU threads for the parallelized tasks (ToPMine and FPGrowth).

Metrics. For this experiment, I have recorded in Table 2 the total number of tweets that were in the subset, the number of frequent phrases identified by ToPMine (Freq. phrases), the total number of edges in the set of all timestep graphs that were determined by FPGrowth ($|E| \in \bigcup_{t=1}^{T} G_t$), the total number of phrase cluster candidates that resulted from Louvain community detection and merging similar events across consecutive timesteps (2nd Cand.), the total number of Final Events determined by `PhraseNet` and the Time in seconds it took to execute.

[10]Language Information Technologies Lab at Michigan

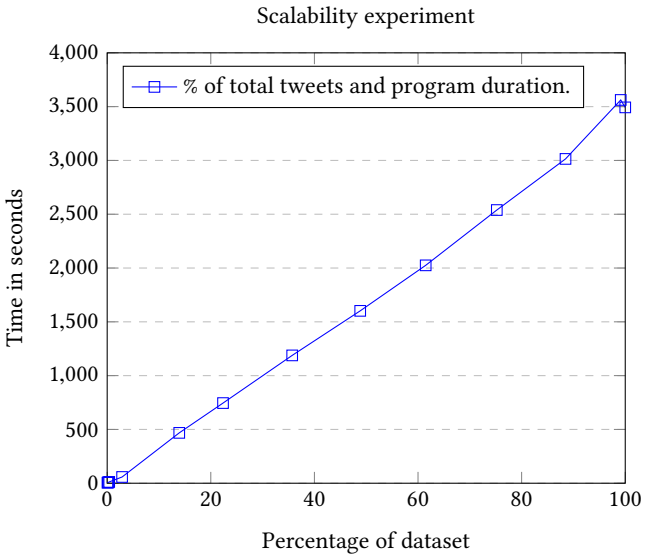

Fig. 5. A plot of the dataset sizes as a percent of the total number of tweets and the time `PhraseNet` took to run on the subset.

4.3 Results

Table 2 shows the time in seconds of the scalability experiments on each subset size of the data, along with the other intermediate output metrics described in the previous section. Figure 5 plots the time in seconds for `PhraseNet` to run for each dataset size as a percentage of the total number of tweets. The figure shows clearly that the method scales linearly with the number of tweets.

5 CHALLENGES

A source of challenges that I faced was the authors' choice of notation and terminology throughout the paper, and their vagueness in method descriptions. Because of the use of many terms for a

Dataset percentage	Total Tweets	Freq. phrases	$\lvert E\rvert \in \bigcup_{t=1}^{T} G_t$	2nd. Cand.	Final Events	Time (s)
100%	195714150	2501305	166936	37588	1329	3493
~99.13%	194006796	2491789	166484	37420	1270	3562
~88.47%	173146360	2189099	117068	33179	964	3014
~75.21%	147202422	1840100	83878	27887	750	2539
~61.48%	120319646	1462103	66804	22715	629	2025
~48.84%	95590100	1132378	52670	17901	529	1602
~35.71%	69895244	795533	39993	13139	424	1188
~22.36%	43755280	446266	22327	7744	316	745
~13.91%	27233238	254270	13781	4497	174	468
~2.88%	5650484	26085	1242	446	50	58
~.39%	768718	4823	281	68	20	12
~.25%	479886	3207	198	42	14	8
~.13%	255352	1700	89	24	12	7
~.058%	113150	769	32	11	4	5

Table 2. Scalability experiment results

concept and the reuse of terms for separate concepts, some of which I described in section 1.1, there were times that I was not certain I was computing everything correctly, for instance whether to use the count frequency or relative frequency (the phrase count to all phrase counts in a timestep) in the sliding window means computation. Though it was not clearly specified in that case, I ultimately decided that where the sliding window means was used, the relative frequency was more reasonable since it they were being compared across different timesteps even though their notation should suggest count frequency. Another case was how they defined dampening coefficient (see the yellow highlight in Figure 2). In fact, there was not a sound definition for this parameter. Since they said that this parameter was up to the user to define and their definition was lacking, I did not use this parameter (or you could consider I used 1). The issue with the authors' definition is that surrounding the equation could suggest that the dampening coefficient is determined by a function with respect to the timestep, since it is marked by the t subscript, but they did not define what that function may be. Also, they said they used .1 for this parameter, and without specifying if this changes over timesteps it would suggest that they used this as a constant. In this case, all of the inner sums would remain proportional to each other and you would not see a change in the standard deviations from the sliding window mean. Therefore, this parameter is either insufficiently defined or, which may sound harsh, is pointless.

The *reproducibility* of this project presented most of the challenges I faced with *reproducing* the work. These challenges and the level of reproducibility of the paper are discussed in-depth in section 6. In short, it is challenging to reproduce research that is vague in its specifications. Some of the methods they described could have been carried out in a variety of ways (i.e., how to merge similar event candidates in neighboring timesteps) based on their description.

6 REPRODUCIBILITY

6.1 Importance of Data Specifications

We saw a dramatic difference between the results using the author's parameters on my dataset and the results that were presented in their paper. On my dataset, the ToPMine minimum support parameters did not allow for the retrieval of any results, even on the `AuthorSet` which was nearly twice as large as the authors' employed little preprocessing limitations following their description. Though both of our datasets were collections of English tweets for the same time period, the minimal output of the ToPMine and FPGrowth stages with their parameters compared to the more liberal parameters could suggest that our tweets may actually look very different.

Our collection methods could have varied significantly which could have affected the data we obtained, thus possibly contributing to this result. For example, the collection method for my dataset was defined by others in my lab who were looking for tweets that included user description and tweet and/or user location attributes. Without knowing if the authors of `PhraseNet` imposed any requirements in their collection method, I cannot attribute my collection method to be the cause of such variance with certainty. Had they included details about their collections methods, one could analyze how subtle or maybe not so subtle requirement differences would change our expectations for an event detection method. In general, this experience shows the importance of describing data collection methods and the data attributes, especially when the data cannot be distributed, in order to have a reproducibile study.

6.2 Specifying Preprocessing

Through this replication, I realized that data preprocessing specifications are especially necessary when the exact dataset cannot be shared. Often times in papers dealing with natural language processing steps, preprocessing steps are not discussed, and if they are they are vague. In `PhraseNet`'s saying that one "removed stopwords" is not informative enough for reproducibility because there is no agreed upon list of stopwords. For example, the original NLTK

English stopword list has 179 stopwords,[11] whereas the source code repository for the ToPMine repository that I used included a list of stopwords that has 524 words.[12] Stopword choices maybe more critical for some applications rather than others. Though I did not show these experiments in the results of this paper, I did see how stopword choices should be made with care for the `PhraseNet` model. For a simple example that does not need a full experimental setup to demonstrate, the ToPMine stopword list includes the word “new," which made the detection of “New Years" challenging and for practical uses, uninterpretable. Another preprocessing step the authors used is the expansion of contractions. List of contractions also vary, depending on their inclusivity to regional language and slang. While implementing `PhraseNet`, I did not find evidence that varying contractions lists would greatly vary the results. However, it could make the paper more reprodicible if their contraction considerations were specified if it were the case that contractions play a significant role to the results of their model.

6.3 Parameters specifications and interpretation

Providing reasoning for parameter decisions is valuable as an analysis of the behavior of a method, but it also can help the study be more interpretable. My `PhraseNet` implementation retrieved no results with their parameters. As I discussed in section 6.1, it may be due to differences in collection methods. However, if the authors had provided an analysis of how their parameter choices affected the output at each stage of `PhraseNet`, it would allow for a better comparison between a blind replication of the model and their implementation.

In my experiments, I recorded the output of each stage to better understand the process that lead to the results I obtained. I do not know how these outputs compare to the authors’ intermediate outputs. Questions that could be answered could be, for example: is the proportion of frequent phrases obtained by ToPMine to the number of tweets we start comparable between the the original work and the replicated study, and does that comparison stay consistent with different ToPMine parameter values? In our implementations, how do the phrase community structures differ depending on different minimum support values in the FPGrowth algorithm?

It is also important to be sure to explicitly specify important parameters or conditions. For example, the authors said that their sliding window was 24-hrs, from midnight to midnight, but the did not specify the size of the timesteps within that window. When they described the initial step of the method, ToPMine, they said that in their model, they said that its purpose was to gather frequent phrases for a timestep, and gave “one hour" as an example of a timestep. This is the only place in the paper where a timestep size is mentioned, so I assume that this is what they used. However, the size of the timestep could greatly affect the results. The standard deviations from the sliding window means are important to their model in determining the final event outcomes. If the timesteps are every four hours, then it is possible that a phrase clusters weight could be very strong if the cluster appears frequently each of the four hours, and maybe even gain strength in by merging with similar phrases after the Louvain community detection stage. Then the phrase could be appear in only one four-hour chunk, and have a weight higher than the sliding window mean that meets the θ threshold for number of standard deviations above the mean. However, the sliding window average would look different across smaller timesteps, and the same phrase cluster may appear more frequently across the sliding window. Then this phrase cluster might not meet the θ threshold in this case.

The previous point also re-emphasizes that parameter specifications should be reasoned based on the model behavior and dataset conditions for better reproducibility. In the previous example, it could be seen that potentially a lower θ value would still enable a phrase cluster to pass the sliding window phase when the timesteps are smaller. In the case of the other thresholds α, β, and χ, these could be chosen based on setting very liberal values to start with, observing which values retrieve more events from the target events that were chosen, and deciding based on a more conservative value that does not have much of a trade-off with lost events. This point leads to the next issue I will discuss, which is how targets that are manually defined by authors for evaluation metrics affect the reproducibility.

6.4 Evaluation metrics

I mentioned earlier a challenge with reproducing this project was that the authors were not explicit in what target events they chose. The authors said that they chose holidays from English speaking country and events listed in certain websites, but did not name all of the websites, and when they did link their website, it was clear that the method they described for choosing the events was too vague. For example, I went to the “On This Day" website and tried to follow their target definition method for those months, but their method of filtering by English speaking countries events would have led me to get many items that I am not sure they would have considered events. For reproducibility of their experiment, it would have been feasible for them to either list the 102 chosen events in their Appendix, or to link to an external website where they are hosting the 102 events that they chose for reference.

While I was replicating this work, it was clear to me how problematic this could be. They compared to a prior method `Twevent` [Li et al. 2012] as their baseline, and `PhraseNet` achieved outstanding metrics compared to `Twevent`, which are shown in Table 6 which was presented in their paper. Since the target events were defined by the authors, it could be the case that these targets were biased in `PhraseNet`’s favor, and potentially `Twevent` was retrieving other valuable results that `PhraseNet` was not and that were not a part of the target set. However, if the authors had included their list of target events, or better yet, additionally including the retrievals side by side for a handful of dates, these results could be better supported. This would also benefit researchers implementing new event detection models for Twitter so that they could compare their event detection models with the original `PhraseNet` implementation.

Another issue with their evaluation is that the authors provided no details about their implementation of `Twevent`, or if they implemented it themselves at all. When I started this replication project, I was hopeful, seeing the number of its citations, that I could find an open-source implementation of `Twevent` so I could easily compare

[11] Can be seen with: from nltk.corpus import stopwords; len(stopwords.words('english'))
[12] https://github.com/anirudyd/topmine/blob/master/topmine_src/stopwords.txt

my own PhraseNet implementation with it. I did not find an implementation of the original Twevent (there have since been other event detection methods that built off Twevent such as [Morabia et al. 2019]) that could easily be run. Even so, variables in implementation and parameter choices for Twevent could affect the results.

	Precision	Recall	F_1 Score
PhraseNet	40%	84%	.54
Twevent	2%	15%	.04

Fig. 6. The author's comparison to Twevent (their Table 3)

7 LESSONS LEARNED

The main takeaways from this experience was that reproducibility requires specificity. It especially critical for evaluation methods and metrics to be detailed precisely in order for experiments to be reproducible, as was seen in that the PhraseNet authors did not list their full set of target events, nor a precise way of getting them exactly. Implementation details are necessary for all stages where any coding is involved, from preprocessing to finalizing results in order for a method to be properly reproduced. Authors should distinguish the parts of their implementation that relied on libraries or open-source code and parts of their implementation that is their from scratch so that the reproducer can know which components of their implementations are comparable. And lastly, I learned that if I choose a parameter, it would increase the reproducibility of my paper if I provide reasoning behind the parameter choices. This will help anyone reproducing my methods to be able to attribute the reasons behind the differences between our results, and if my data is not publishable, for them to be able to interpret possible variances in our data vs. implementation that cause the differences.

REFERENCES

Chi Wang Clare R. Voss Jiawei Han Ahmed El-Kishky, Yanglei Song. 2014. Scalable Topical Phrase Mining from Text Corpora. In *VLDB 8*.

Vincent D Blondel, Jean-Lou Guillaume, Renaud Lambiotte, and E Lefebvre. 2011. The Louvain method for community detection in large networks. *J of Statistical Mechanics: Theory and Experiment* 10 (2011), P10008.

Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *ACM sigmod record*, Vol. 29. ACM, 1–12.

Chenliang Li, Aixin Sun, and Anwitaman Datta. 2012. Twevent: segment-based event detection from tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 155–164.

Sara Melvin, Wenchao Yu, Peng Ju, Sean Young, and Wei Wang. 2017. Event Detection and Summarization Using Phrase Network. In *Machine Learning and Knowledge Discovery in Databases*.

Keval Morabia, Neti Lalita Bhanu Murthy, Aruna Malapati, and Surender Samant. 2019. SEDTWik: Segmentation-based Event Detection from Tweets Using Wikipedia. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. 77–85.

A FULL RESULTS OF 2015 EXPERIMENT

Events:

B STOPWORD SETS

B.1 *nltk+* stopwords

i me my myself we our ours ourselves you your yours yourself yourselves he him his himself she her hers herself it its itself they them their theirs themselves what which who whom this that these those am is are was were be been being have has had having do does did doing a an the and but if or because as until while of at by for with about against between into through during before after above below to from up down in out on off over under again further then once here there when where why how all any both each few more most other some such no nor not only own same so than too very s t can will just don should now

B.2 *mods* stopwords

a about above after again against ain all am an and any are aren aren't as at be because been before being below between both but by can couldn couldn't d did didn didn't do does doesn doesn't doing don don't down during each few for from further had hadn hadn't has hasn hasn't have haven haven't having he her here hers herself him himself his how i if in into is isn isn't it it's its itself just ll m know make me mightn mightn't more most mustn mustn't my myself needn needn't no nor not now o of off on once only or other our ours ourselves out over own re s same shan shan't she she's should should've shouldn shouldn't so some such t than that that'll the their theirs them themselves then there these they this those through to too under until up ve very was wasn wasn't we were weren weren't what when where which while who whom why will with won won't wouldn wouldn't y you you'd you'll you're you've your yours yourself yourselves could he'd he'll he's here's how's much please i'd i'll i'm i've let's ought she'd she'll that's there's they'd they'll they're they've we'd we'll we're we've what's when's where's who's why's would followed following follows follow really become becomes becoming come ever every everything well each even us use used useful uses using usually mean oh like liked look looking looks than thank thanks thanx always though although get gets already hm hmm b c d e f g h i j k l m n o p q r s t u v w x y z way ok okay ha haha please hey hi go goes going gone got gotten give given gives et etc even ever every everybody everyone everything everywhere subscribe so some somebody somehow someone something sometime sometimes somewhat somewhere soon

Experiment	Dataset	Time (s)	Freq. phrases	$\lvert E\rvert \in \bigcup_{t=1}^{T} G_t$	2nd. Cand.	Final Events	Precision	Recall	F1
AuthorParams	AuthorSet	50	6444	3686	738	0	0.00	0.00	0.00
params-1	AuthorSet	52	58035	2921	741	0	0.00	0.00	0.00
params-2	AuthorSet	50	58035	2921	742	18	17.65	4.29	6.90
params-3	AuthorSet	47	58035	90	48	4	0.00	0.00	0.00
params-4	AuthorSet	52	49182	8117	935	342	1.17	6.15	1.96
AuthorParams-mod	AuthorSet	42	4384	2183	518	1	0.00	0.00	0.00
params-1-mod	AuthorSet	45	38844	1596	477	0	0.00	0.00	0.00
params-2-mod	AuthorSet	45	38844	1596	476	27	3.70	1.54	2.17
params-3-mod	AuthorSet	42	38844	35	17	5	0.00	0.00	0.00
params-4-mod	AuthorSet	47	31495	2711	602	272	4.78	20.00	7.72
AuthorParams	ModifiedSet	36	2957	1033	277	1	0.00	0.00	0.00
params-1	ModifiedSet	37	34590	1120	329	0	0.00	0.00	0.00
params-2	ModifiedSet	37	34590	1120	329	17	11.76	3.08	4.88
params-3	ModifiedSet	35	34590	98	45	6	16.67	1.54	2.82
params-4	ModifiedSet	38	30887	4390	588	249	2.88	10.77	4.55
AuthorParams-mod	ModifiedSet	29	1022	74	35	0	0.00	0.00	0.00
params-1-mod	ModifiedSet	32	17630	86	47	0	0.00	0.00	0.00
params-2-mod	ModifiedSet	32	17630	86	47	21	14.29	4.62	6.98
params-3-mod	ModifiedSet	32	17630	16	5	4	50.00	3.08	5.80
params-4-mod	ModifiedSet	32	14733	485	145	111	9.01	15.38	11.36

Table 3. Parameter experiment results

timestep	χ	α	β	event
Feb 14, 2015	0.332	1.000	0.014	happy valentines day, valentines day
Mar 28, 2015	0.320	1.000	0.014	kca vote1duk, kca voteonedirection, rt quote, vote5sos kca, votefifthharmony kca, voteonedirection kca
Mar 20, 2015	0.293	1.000	0.013	kca votelittlemixuk, littlemix girls, littlemix girls love, littlemix kca votelittlemixuk, littlemix love, love kca votelittlemixuk, votelittlemixuk kca
Jan 29, 2015	0.291	1.000	0.013	allybrooke ally, allybrooke askally, allybrooke favorite, allybrooke favorite song, askally allybrooke, eu te amo, happy birthday, love askally
Mar 08, 2015	0.280	1.000	0.012	dream true, eu te amo laurenfollowspree, lauren ilysm, laurenfollowspree laurenjauregui, laurenfollowspree votefifthharmony kca, laurenjauregui lauren, laurenjauregui lauren love, laurenjauregui laurenfollowspree, laurenjauregui laurenfollowspree lauren, laurenjauregui love, love laurenfollowspree, realize dream, vote5sos kca, votefifthharmony kca, womens day
Mar 28, 2015	0.277	1.000	0.012	vote5sos kca, votefifthharmony kca
Feb 14, 2015	0.274	1.000	0.012	happy valentines day, valentines day, youtube video
Mar 27, 2015	0.244	1.000	0.010	kca vote1duk, rt quote, vote1duk kca, vote5sos kca
Jan 14, 2015	0.242	1.000	0.010	act age, broken hearts, change ticket, fools gold, girl almighty, love song, myfourtrackonrepeat onedirection, night changes, onedirection myfourtrackonrepeat, stockholm syndrome, stop listening
Feb 14, 2015	0.241	2.000	0.013	happy birthday, happy valentines day
Feb 23, 2015	0.238	1.000	0.010	american sniper, best picture, eddie redmayne, green card, julianne moore, sean penn
Feb 26, 2015	0.235	1.000	0.010	black blue, blue black, gold white, white gold
Feb 02, 2015	0.225	1.000	0.010	katy perry, super bowl, youtube video
Mar 05, 2015	0.212	1.000	0.009	jai love, jaibrooks1 lagirlmusicvideo, jaibrooks1 love, la girl, lagirlmusicvideo jaibrooks1, vote5sos kca, watch lagirlmusicvideo
Mar 29, 2015	0.201	1.000	0.009	final four, free throws, last night, michigan state, tom izzo
Jan 18, 2015	0.191	1.000	0.008	green bay, last night, russell wilson, super bowl
Mar 10, 2015	0.182	1.000	0.008	chip kelly, free agency, jake locker retiring, jimmy graham, nick foles, sam bradford
Jan 30, 2015	0.181	1.000	0.008	new followers, new unfollowers, people via, today stats, week twitter
Jan 03, 2015	0.174	1.000	0.007	nashgrier nashsnewyearsskit, nashs new video, nashsnewyearsskit nashgrier, new skit, new year, watch nashgrier new video, watch nashs new video, watch new
Jan 10, 2015	0.173	1.000	0.007	big thumbs, check themattespinosa new video, check themattespinosas new, check themattespinosas new video, forget happymatturday, happy matturday, matt ilysm, thumbs happymatturday, vinny block, watch themattespinosa new video, watch themattespinosas new video, youtube video
Jan 25, 2015	0.173	5.000	0.010	happy birthday, youtube video
Feb 15, 2015	0.173	1.000	0.007	allstar game, bill clinton, national anthem, new york, queen latifah, star game
Feb 05, 2015	0.170	1.000	0.007	new followers, new unfollowers, people via, week twitter
Feb 19, 2015	0.160	1.000	0.007	love walkaway5days, madisonellebeer love, madisonellebeer madison, madisonellebeer walkaway5days
Jan 10, 2015	0.160	1.000	0.007	check matts new video, happymatturday themattespinosa, love matt, matts new vid, new vid, new video amazing, themattespinosa happymatturday, vinny block beginning, watch matts new video
Jan 07, 2015	0.160	1.000	0.007	big brother, katie hopkins
Jan 01, 2015	0.157	1.000	0.007	nashgrier nashsnewyearsskit, nashsnewyearsskit nashgrier, watch nashgrier new video, watch nashs new skit, watch nashs new video
Jan 27, 2015	0.155	1.000	0.007	eu te amo streamreflectiononitunes normanikordei, mani love, normani kordei, normani love, normani see, normanikordei streamreflectiononitunes, realize dream, streamreflectiononitunes normanikordei
Feb 22, 2015	0.154	1.000	0.007	emma stone, lady gaga, neil patrick harris, red carpet, vote5sos kca, watching oscars
Jan 11, 2015	0.151	1.000	0.006	asknacks nashgrier, jackjackjohnson jackgilinsky, nashgrier asknacks, new video asknacks, via youtube, watch nashgrier, watch nashgrier new video, watch nashs new video
Mar 21, 2015	0.151	1.000	0.006	check nashs new video, nashgrier nashsnewvideo, nashgrier new video, new video, snowboarding colorado, watch nashgrier new video, youtube video
Jan 12, 2015	0.151	1.000	0.006	college football, ohio state
Feb 14, 2015	0.150	1.000	0.006	happy valentines, valentines day
Jan 27, 2015	0.148	1.000	0.006	dream eu te amo, love reflection, love streamreflectiononitunes, normanikordei love, normanikordei mani, normanikordei mani love, normanikordei normani, normanikordei streamreflectiononitunes normani, reflection amazing
Feb 27, 2015	0.147	1.000	0.006	black blue, blue black, white gold
Feb 11, 2015	0.144	1.000	0.006	new unfollowers, one person, people via, week twitter
Jan 19, 2015	0.142	1.000	0.006	jackgilinsky jackjackjohnson, jackgilinsky jackjackjohnson skatemaloley, likethatvideo jackgilinsky jackjackjohnson, new music video, watch likethatmusicvideo, watch likethatvideo
Jan 11, 2015	0.142	1.000	0.006	bad call, cowboy fans, cowboys fan, cowboys fans, last week
Jan 10, 2015	0.141	1.000	0.006	happy birthday, zaynmalik zayn
Mar 17, 2015	0.138	1.000	0.006	happy birthday, happy st patricks day, st patricks day
Jan 11, 2015	0.138	1.000	0.006	aaron rodgers, dez bryant, football move, green bay, hit ground, super bowl
Jan 16, 2015	0.136	1.000	0.006	first album, proud shawnmendes, shawn love, shawnaccess updates, shawnmendes proud, shawnmendes shawnaccess shawnsfirstalbum, shawnmendes shawnsfirstalbum, shawnsfirstalbum shawnaccess, shawnsfirstalbum shawnmendes, shawnsfirstalbum shawnmendes shawnaccess,shawnsfirstalbum shawnsfirstalbum shawnsfirstalbum shawnsfirstalbum
Jan 29, 2015	0.133	1.000	0.006	good morning, january 29 2015
Mar 26, 2015	0.130	1.000	0.006	full read ebay, one direction
Jan 17, 2015	0.129	1.000	0.006	entering win daily prize fifth harmony fanuary 5hfanuaryday17, luke hemmings 5sos, te amo
Mar 09, 2015	0.129	1.000	0.006	good morning, new followers, new unfollowers, people via, week twitter
Mar 25, 2015	0.128	1.000	0.006	one direction, zayn malik
Jan 26, 2015	0.127	1.000	0.005	good day, great day, happy birthday, harry hope, harrystyles harry, harrystyles hope

Table 4. Results of params-4-mod on ModifiedSet. Sorted by χ, highest peak intensity