Final Report: The Effects of External Stimuli on Influence Estimation

Kent Blake, Stephanie Palocz, Matthew Volk
Stanford University, CS224W Final Project

1 Introduction

The proliferation of large-scale social networks in recent years has provided companies with new marketing opportunities through targeted advertising. In particular, the marketing industry has devoted significant research towards what has become known as the *influence maximization problem*. The problem is as follows: given a social network with many highly-interconnected users, if you can advertise to only k people, who make up a small fraction of the total people on the network, how do you pick the k people who will share and spread your advertisement such that the largest possible number of people on the network see it?

The greedy influence maximization algorithm introduced by Kempe et al. [6] has become a standard tool for approaching this problem. However, this model fails to account for how a strong external stimulus, like a historic event or groundbreaking news story, may change influence outcomes in a network. We hypothesized that such an event could have a dramatic effect on the network's structure and event propagation patterns. In order to test this, we tested the effectiveness of Kempe et al.'s algorithm on a Twitter retweet network of tweets regarding the discovery of the Higgs boson particle from the three days prior to the announcement of the discovery versus a similar retweet network for the three days following the discovery.

For this project, we worked on generating and testing a model that can learn propagation probabilities given a list of actions and a network of unweighted relationships (here Twitter "follow' relationships). We started by implementing a baseline weight-learning algorithm and a basic testing framework based on the work of Goyal et al. [4] We first trained on tweets from before the Higgs Boson discovery to generate a weighted graph of influence probabilities and tested the accuracy of our probabilities using tweets from after the discovery. Then, we trained and tested using different subsets of tweets from after the discovery. We also did this with two before- and after- lists of randomly-generated tweets. We anticipated that training on tweets from after the announcement would be more effective than training on tweets from before the announcement, but interestingly, the two approaches had approximately the same performance, and both performed worse than the model run on randomly-generated data. There are multiple conclusions that could be drawn here, but in this case, we believe that the lack of a difference is descriptive of the graph structure of our data and Twitter as a whole.

2 Literature Review

Although there is not much literature discussing influence maximization specifically in connection to external events, there is significant literature discussing influence maximization in general. As such, we based our preliminary model on the following existing frameworks, while noting possible factors to leverage in our own improvements.

2.1 Definitions

Kempe et al. [6] present a framework based on submodular functions that generalizes well to the influence maximization problem, as they define it. Further, they prove that a relatively straightforward greedy algorithm of their own design will always yield a solution within a factor of $\left(1-\frac{1}{e}\right)$, or about 63%, of the optimal solution. They also prove that, in practice, their greedy algorithm outperforms the previously-ubiquitous heuristics that looked at degree centrality and distance centrality alone. The following notation is derived primarily from [6] and [2]. To build up the arguments of Kempe et al., we begin with a few

definitions.

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a graph consisting of a set $\mathbf{V}$ of vertices and a set $\mathbf{E}$ of edges. Depending on the network being analyzed, these edges can be directed or undirected. For the specific application to a social network presented in this project, each node $e \in \mathbf{E}$ can be thought of as a user account and each edge $v \in \mathbf{V}$ can be thought of as a connection.

Next, we have a timestamped list of actions performed on the network, a relation *Actions(User, Action, Time)*, which contains a tuple (v, a, t) indicating that node v performed action a at time t. From these two definitions, we can define propagation of actions.

Definition 1: Action Propagation. An action $a \in A$ propagates from user u to user v iff: (i) $(u,v) \in E$; (ii) $\exists (u,a,t_u), (v,a,t_v) \in A$ with $t_u < t_v$. [1]

In our model of action propagation, for a particular action a, all nodes are considered either *active* (if they have performed a) or *inactive* (if they haven't). Once a node becomes active, it cannot return to the inactive state.

The last definition is our influence function, $\sigma(S)$. σ maps finite sets S to integers.The input is a initial set of activated nodes and the output is the final number of activated nodes (after sufficient time has passed that no new nodes will become activated). The goal of this framework is to approximate this function as closely as possible.

2.2 Diffusion Models

There are two basic diffusion models referenced in the literature [4, 6], presented below.

Linear Threshold Model. Under this model, each node v is influenced by each of its neighbors u. The probability that a particular node v will take action a at time t is modeled as a monotonically increasing function of how many of its neighboring nodes have taken a before t. Each neighbor u is given a weight $w_{v,u}$ such that

$$\sum_{u \in \text{Neighbors}(v)} w_{v,u} \leq 1.$$

Each node v has a threshold θ_v for which it becomes active iff the following inequality holds true:

$$\sum_{u \in \text{Neighbors}(v)} f(w_{v,u}) \geq \theta_v.$$

where f corresponds to some function (the identity function in [6] but probability distributions in [4]). Conceptually, then, each of v's neighbors influences them with a weight $w_{v,u}$, and each user v has some willingness to conform θ_v. Time steps in discrete "rounds," and in each successive round we "activate" nodes for which the above inequality holds. By definition, all nodes that have been activated remain activated. Once all actions in the action list have been accounted for, the simulation terminates, yielding a list of activated nodes.

Independent Cascade Model. Under the independent cascade model, we start with a set $U \subseteq V$ of active nodes. Then, we procede through a series of rounds. In each round, every newly active node u has a probability $w_{u,v}$ of activating each of its neighbors v. Unlike in the linear threshold model, each newly activated node has one and only one chance to activate nearest nodes. The process ends when no more activations are possible.

2.3 Approximization Model

The primary contribution of Kempe et al. in [6] is proving that their greedy algorithm can approximate the true solution using the following greedy approach: start with an empty set and repeatedly add the element of V that gives the largest marginal gain. As long as the overall influence function is both monotonically increasing and submodular, this greedy approach can approximate the answer within a factor of $\left(1 - \frac{1}{3}\right)$.

2.4 Optimizations

Goyal, Bonchi, and Lakshmanan [4] build on the work of Kempe et al. Using the definition of action propagation presented above, they generate a directed propagation graph (a directed acyclic graph) for each action a. This graph shows the spread of the action a. From this propagation graph, they generate influence probabilities for each edge in the graph, which identifies the most "influential nodes as well as action influenceabilities and user influenceabilities. They provide pseudocode that we will replicate.

Mathematically, they introduce an instance of the *Linear Threshold Model*. There are two parameters that

[1] NB: This definition is based heavily on work in [4], with our own modifications to generalize it. Specifically, we remove [4]'s tracking of friend / follow requests.

must be learned: (1) the unique influenceability threshold indicating how "stubborn" each user is, and (2) the unique influenceability of each action, which indicates the weight of a action promoting further action.

Interestingly, they build on the idea of weights presented in [6]. They introduce functions $f(w_{v,u})$ that more accurately model the real world. f in this context can be static (some function of the weight) or dynamic (some function of the weight *and* the time). The dynamic case is particularly interesting to us because it allows one to model the "cool-off" period associated with a particular post. After a certain period of time, the probability that neighboring nodes will see the action and repeat it decreases drastically.

Next, Goyal et al. present models that can learn the necessary hyperparameters for an entire graph given a list of actions taken in the network. These algorithms only need to take two linear passes over the action lists to train themselves. They predict trends (once just predicting the end result and once predicting the progression of time to reach the end result) then test these predictions against actual data, using ROC curves as a metric. We will be using adaptations of these models and testing frameworks on a completely different network (Twitter).

2.5 External Stimuli

The existing models assume that activity propagates exclusively from an active node to its neighbors, but in the real world external stimuli can have a strong effect on information propagation. This is especially relevant for networks like Twitter in which a big news story could increase user engagement in general and potentially cause deviations from expected influence patterns with users who have the quickest access to new information becoming much more influential than expected.

For example, in their study of tweets from the seven days surrounding the announcement of the Higgs boson particle, De Domenico et al. [3] observe that spatial and temporal factors play a large role in Twitter activity, with high activity before the official announcement from users within 20km of the event and frenetic tweet activity during the announcement. They also model the information diffusion network of mentions and retweets over the seven day period and are able to reproduce the collective behavior of 500,000 users by employing a memoryless model with variable activation rates in which users are significantly more likely to be active when most of their friends are active at the same time. This supports the idea that temporal factors play a significant role in information diffusion when people are discovering and sharing information about an external event, and that such an external event can have a large impact on social media use in general It also suggests that a linear threshold model which considers the total number of activated neighbor nodes at a given time could be more effective than an independent cascade model in predicting influence within this sort of network.

3 Dataset

We used the Higgs boson dataset from [3], which was built by collecting tweets from July 1, 2012 to July 7, 2012, the days before and after the July 4th announcement of the discovery of the Higgs boson particle. For our problem, we focused on the retweet data provided by De Domenico et al. [3], which consists of the id of an original tweeter, the id of a retweeter, and the timestamp of the retweet.

Since we were interested in comparing influence maximization results before versus after a significant external event, we partitioned the retweet data into all retweets occurring before the July 4th announcment and all retweets occurring after the July 4th announcement. This partition has 111,105 retweets from 46,283 sources before the announcement and 243,827 retweets from 142,046 sources after the announcement. Notably, there are over 3 times as many tweet sources present after the announcement than before. This makes sense given our partition criteria - people are much more likely to tweet about the Higgs boson after the announcement has taken place than beforehand - but it is somewhat problematic for our training algorithm, since we had no way to learn probabilities associated with these extra tweeters based on our "before" dataset and were thus forced to use default values. This issue was mitigated by some further partitioning, but there were still many nodes in our testing sets that were not present in our training sets.

Another important feature to highlight in these retweet lists is that many users (in fact, the vast majority of users) only participated in one retweet event over the time period in question. Given the event-driven context for this data, the low-degree nature of the graph is understandable. When a large external event takes place, many people who do not normally retweet may be motivated to engage with social media, but will likely not engage a large amount, maybe only tweeting or retweeting once. However, this feature makes it difficult to accurately predict the relative likelihoods

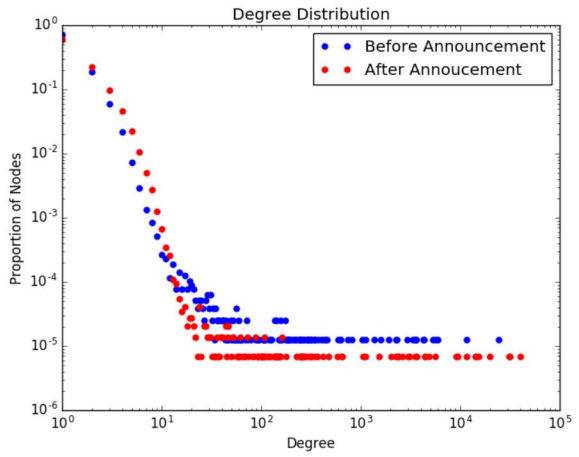

Fig. 1. Degree distributions for the retweet events, partitioned to before and after the Higgs boson announcement

of future retweet events between two given users, since we often have no prior information about the users in question when attempting to make a prediction.

Our model also takes into account the underlying network of follow relationships between the tweeting and retweeting users mentioned in that event list. In this network, each node represents a Twitter user, and a directed edge indicates that the source of the edge follows the destination. The network has 456,626 nodes and 14,855,842 edges, with a clustering coefficient of 0.1887 and a 90th percentile effective diameter of 3.7.

3.1 Null Models

In addition to the Twitter dataset, we generated two null model datasets with which to compare our results. Each of these other datasets is comprised of two lists of retweet events, one corresponding to the retweets from before the Higgs boson announcement and one corresponding to the retweets from after the announcement. Each list contains approximately the same number of retweet events as its corresponding list from the original Twitter dataset partition (within 7%).

These null model lists were generated using a basic infection model. We repeatedly select a node n at random to be the source of a tweet, and each of its followers then retweets it with some probability $P(n)$. Once a follower node has retweeted the original tweeter, their retweet can itself be retweeted by any of their followers, giving the possiblity of an extended retweet cascade just like in the Higgs boson dataset. For one of our null model sets, $P(n)$ is a fixed constant (0.1). For the other, $P(n)$ is proportional to the number of followers n has ($P(n)$ = in-degree(n)/(max degree in the graph)).

3.2 Further Partitioning

We predicted that the structure of our retweet graph after the Higgs boson announcement would be significantly different than before the announcement. To test this theory, we further partitioned our list of "after" retweets into two halves. Because there were about twice as many "after" retweets as "before" retweets, this additional partition gives us 3 retweet lists of roughly equal size. We then denoted one "after" retweet list as a training list, and the other as a testing list. We run each version of our algorithm over our data sets in two different ways: first, we train on the "before" retweet list and test on the testing "after" list, and second, we train on the training half of the "after" retweets and test on the testing half. For our null models (where we generate the before and after lists using the same technique), we did not expect a meanningful difference between these two. For our real-world data, though, we expected the after-after configuration to perform better than the before-after one, assuming that the training and testing data had more structural similarity. If this prediction was correct, it would support our hypothesis that the structure of the network after the external event is fundamentally different from the structure of the network before the event.

4 Calculating Influence Probabilities

Although the referenced approaches to finding the influence-maximizing subset of a graph's nodes are rigorous and relatively robust, they all rely on having a pre-generated weighted graph of influence probabilities. Since a retweet graph over a short time period provides only partial information about how some users influence some of their followers, we needed a method to estimate influence probabilities for each user on each of their followers. We tested two approaches for generating influence probabilities:

4.1 Baseline Probability Function

Our original baseline approach to influence probability calculation was that of Goyal et al. [4], which employs the following probability function: the estimated probability for edge (u, v) (where v retweets u) is simply the number of times u retweeted v divided by the number of times u retweeted any of its neighbors. This is the most common algorithm in the literature for calculating influence probabilities. However, since our retweet graphs were sparse (with nearly 70% of nodes having a degree of 1), this resulted in most of the influence probabilities being set to 1, which means that our

model ends up predicting that a retweet will occur in every possible situation. Clearly, this is a flawed approach for our scenario, so we moved on to a less common but potentially more applicable model.

This model can be defined more rigorously: first, we define the influenceability of a node u by another node v, or $infl_{u,v}$. $infl_{u,v}$ is a score between 0 and 1 inclusive that indicates how likely u is to perform an action if v does. Formally, this is defined as:

$$infl(u,v) = \frac{|\{a|\exists v,t : prop(u,v,a,t) == True|}{|A_u|} \quad (1)$$

where $prop(u,v,a,t)$ evaluates to True if u performed action a at time t after v performed the same action. In other words, this is the number of times u was potentially influenced into performing an action by v divided by the total number of times it performed the action.

From this, we can generate the probability of a node u performing an action based on which of its influencers have. Equation 2 defines the probability of u performing an action given a set of infected influencers S. It is simply the difference of 1.0 and the product of its influenceability by the other nodes.

$$p_u(S) = 1 - \prod_{v \in S}(1 - infl_{u,v}) \quad (2)$$

4.2 Beta Distribution Approach

To address the sparseness of our retweet graph, we used the approach described in Lei et. al [7] to calculate influence probabilities. Since the exact influence probability P_{ij} of user i on user j is not known for all users, we model this using a beta distribution for each edge (i,j) such that $P_{ij} \sim B(\alpha_{ij}, \beta_{ij})$. Initially we set $\alpha_{ij} = 1, \beta_{ij} = 1$ for each of these distributions, indicating that in the absence of prior information user P_{ij} follows a uniform distribution. As we traverse over the list of all edges (i,j) in the follower graph, we update these α_{ij} and β_{ij} as follows:

If j retweeted i: $P_{ij} \sim B(\alpha_{ij} + 1, \beta_{ij})$
If j follows i but did not retweet i: $P_{ij} \sim B(\alpha_{ij}, \beta_{ij} + 1)$

Thus the distribution for each edge counts the number of successful and failed activations passing though that edge smoothed by our sinitial estimates of $\alpha_{ij} = 1, \beta_{ij} = 1$. Finally, for each edge (i,j) we sample from $B(\alpha_{ij}, \beta_{ij})$ to obtain an estimated value for each influence probability. This technique gave us a much better spread in our probability distribution.

4.3 Computational Costs

One roadblock we hit while calculating influence probabilities was the sheer number of nodes in our dataset. Our baseline algorithm in particular was affected by this size, since it operates in $O(n^2)$ time as opposed to $O(n)$ time for the beta distribution model. We used the Stanford Barley machines to mitigate this issue, but the overall scope of all our conditions in combination was still computationally infeasible. Therefore, we ran our baseline probability algorithm on only 25% of our retweet lists for all datasets, to keep the space and time complexities to a more reasonable level.

5 Evaluation Methods

The true influence maximization problem (in which we select the most influential k nodes in the graph, given our calculated influence probabilities) is a well-defined problem, but it generates test output that is difficult to compare and is significantly more computationally costly. Influence maximization generates a list of the k most influential nodes. Given two such lists, it is difficult to compare in a meaningful way how "similar" the two performed at their job, because even if the two sets have no intersect, the two sets might correspond to nodes that have a large spread amongst them. Similarly, changing just a few elements of such a set might drastically change the spread of the overall set. Therefore, for the majority of our data, we evaluated our results using the evaluation algorithm developed by Goyal et al. [4], which we call Predictive Evaluation, seen in the code sample below (TP=True Positive, FN=False Negative, FP=False Positive, TN=True Negative).

5.1 Predictive Evaluation

Predictive Evaluation iterates through each {node, time} pair in the actions list in chronological order. The `performed` flags are updated, where a `perform` value of 0 indicates that u never performs the action but at least one of its neighbors does (it never gets infected), a value of 1 indicates that u performs the action after at least one of its neighbors does (it does get infected), and a value of 2 indicates that it is the first node in its neighborhood to perform the action. We update these flags and probabilities and, finally, compare the computed probabilities against thresholds to generate a confusion matrix.

```
   PredictiveEvaluation Algorithm

performed = {}
probabilities = {}

for each <v, time> in
    chronological order
  if v in performed
    performed[v] = 1
  else
    performed[v] = 2
  for u : (v, u) in graph.Edges()
    if u in performed
      increment probabilities[u]
        as in Equation 2
    else
      set probabilities[u]
        as in Equation 2
      performed[u] = 0

for <u, p> in performed
  prob = probabilities[u]
  if (p == 1 and prob >= thresh)
    TP++
  if (p == 1 and prob < thresh)
    FN++
  if (p == 0 and prob >= thresh)
    FP++
  if (p == 0 and prob < thresh)
    TN++
```

5.2 Traditional Influence Maximization

Additionally, we ran the true influence maximization algorithm for a single setting - our beta distribution-based algorithm on the real-world Twitter data. Since the beta distribution approach was by far the more meaningful influence probability calculation approach, we felt that this was the most important setting in which to use both metrics. For this, we used the greedy approach from Kempe et al. [6] to find the set of k nodes that maximized the spread of the set, as defined by the spread estimation technique from the SIMPATH algorithm [5].

The SIMPATH algorithm defines the spread of a set S, $\sigma(S)$, as the following:

Given a set S of nodes in a graph $G = (V,E)$, is defined as the sum of each node $u \in S$ on subgraphs induced by $V - S + u$. Mathematically, this is:

$$\sigma(S) = \sum_{u \in S} \sigma^{V-S+u}(u)$$

where the superscript indicates the subgraph considered.

6 Results

Overall, our results were not exactly as expected. Our baseline algorithm performed more or less equally across all situations, which is understandable given its simplicity and general inability to cope with a graph structure as sparse as ours. However, our beta distribution approach's results did not confirm our hypothesis that prediction abilities would be stronger when training on data from after the Higgs boson announcement than when training on data from before the announcement - we got comparable results for both scenarios.

6.1 Baseline Algorithm Results

Although our baseline probability algorithm seemed to do all right at first glance, upon further inspection, it actually did quite poorly. For all conditions, its F-scores were consistently high, indicating strong predictive power. However, as it turns out, these F-scores were identical for every threshold value we examined (in the range from 0 to 0.9), and did not change when we chose to train on "after" data rather than "before." In fact, the model's prediction was the same for almost every situation - it almost always predicted that a retweet event would occur. These results held true across both the real data and our null models.

Mathematically, this makes sense, given that the calculated influence probability for a given edge is 1 if a node's only retweet took place along that edge. Given the low degree of most nodes, there are many edges with probability 1 assigned to them. As a consequence of the equation used in the evaluation process to predict the probability of a retweet, the model will predict a retweet with probability 1 if any edge with influence probability 1 is present in the set of edges between the current node and infected nodes. Thus, if the model makes a prediction at all, it will almost always predict a retweet. Since our testing event list is heavily skewed towards the presence of a retweet event, this will give us an artificially high success rate, even though the model is not actually predicting in an intelligent or effective way.

6.2 Beta Distribution Results

Our beta distribution based model gave much more reasonable results than the baseline model. It performed far better for the randomly generated null model data than for the other two datasets, while the next best

Table 1. F-scores for each baseline model configuration run. F-scores were the same for all thresholds in every case.

Training set x data source	Random (p=0.1)	Degree-weighted	Real world
Before	0.9998875	0.99907640	0.94642248
After	0.9998875	0.99907640	0.94642248

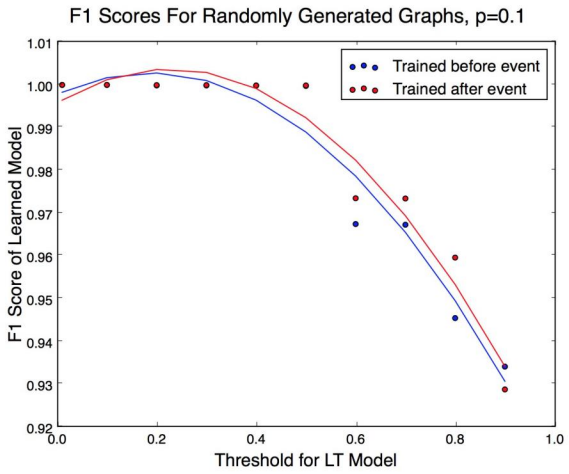

Fig. 2. F1 scores as a function of threshold for the beta distribution based model run on randomly generated data, with quadratic best-fit lines.

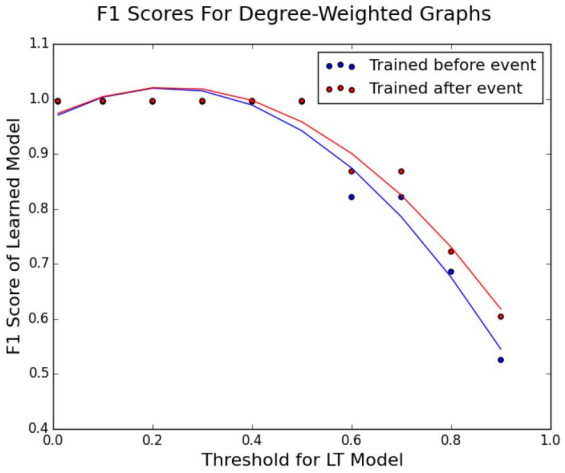

Fig. 3. F1 scores as a function of threshold for the beta distribution based model run on degree-weighted generated data, with quadratic best-fit lines.

results were for the degree-weighted generated graphs and the poorest performance was on the real world data. However, for each overall dataset, there was not a significant difference between training the model on "before" data and training it on "after" data. This lack of a difference is expected for our null model datasets, since the "before" and "after" data were generated in exactly the same way as each other. With the real world data, though, this lack of a difference is not expected - we predicted that the "after"-trained model would perform better than the "before"-trained one.

For all datasets, the precision scores were consistently 1, and the only variation in performance between threshold values came in recall. Thus, we have plotted F-scores as a function of threshold, to give a more clear representation of performance across different threshold values than a typical ROC curve would offer.

We also ran influence maximization on our beta distribution model for our real world data, generating a list of the 15 most influential nodes in our test data both when trained on "before" data and when trained on "after." Unfortunately, there was no overlap between these two lists of most influential nodes. Since both training situations performed comparably for our Predictive Evaluation metric, we have no way of evaluating which list of influential nodes is likely to be more accurate.

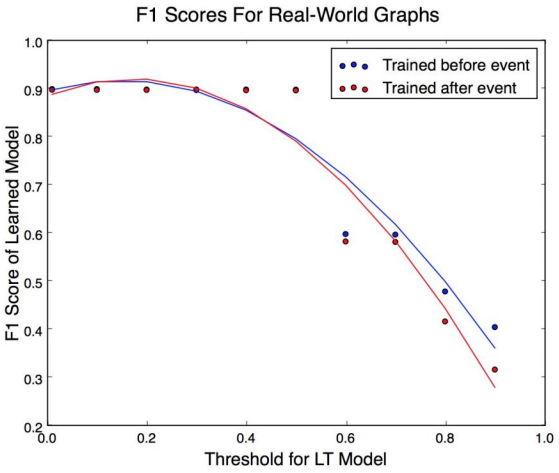

Fig. 4. F1 scores as a function of threshold for the beta distribution based model run on real world data, with quadratic best-fit lines.

7 Discussion

As we have seen, our results do not support the claim that the Twitter graph structures before and after the Higgs boson announcement were significantly different. Our model had roughly the same predictive power no matter which data subset we used for training. Does this mean that our hypothesis was wrong? It's certainly possible. However, there are other possible explanations if we look at the distinctive - and rather

problematic - traits of our real-world data.

In particular, the sparseness of our retweet graph both before and after the Higgs boson announcement presented a real challenge for our model. With the majority of nodes having degree 1, our models have no choice but to assign most nodes the same influence probabilities. Without more edge data in a training set relative to the number of nodes in the set, it is difficult to differentiate between the influenceabilities of different nodes no matter what model we use. The beta distribution approach was better equipped to handle this lack of information, but it, too, was forced to operate using the same limited prior knowledge in the majority of cases. The presence of so many additional nodes in the "after" data that were not present in the "before" data also made things harder, as we were forced to use default values when making many predictions.

As discussed in section 3, these problematic dataset traits are in part explainable by the dataset's event-driven nature. Important external events are likely to trigger small amounts of engagement from large numbers of people on social media, as people learn of the event, engage with it, and then typically lose interest rather quickly. This general trend fits our data well, with its large number of nodes participating in a single retweet event following the Higgs boson announcement, so it might be the case that this sort of data is representative of social media patterns related to external events.

However, these traits rendered our baseline algorithm ineffective immediately, and they may also have caused problems for our beta distribution model even before taking more subtle alterations of the graph structure into account. Without more event-driven datasets with which to compare, it is difficult to know how characteristic these traits actually are, and it is also hard to say how disproportionate their impact on these results may have been. It could be that this dataset structure makes it difficult to calculate relative influence using Twitter data in general and event-related data in particular, without saying anything about differences in influence before and after an event has taken place. In that case, it would take more algorithm refinement before we are in a position to properly consider our original question.

8 Conclusion

Influence maximization is a challenging task in the best of circumstances. When we introduce significant, network-changing events to the picture, it becomes only more complicated. As we have shown, predicting the most influential nodes in a network after some external event is not necessarily more or less effective depending on whether you examine the structure of the graph before or after the event has taken place. However, it is unclear whether our model's disagreement with our hypothesis is truly due to a lack of dissimilarity between the before and after networks, or whether it is simply caused by some inherent flaws in our dataset and in social media data in general.

Overall, lack of data is a frustrating problem when researching these types of scenarios. Although social media is no longer a new concept, it remains difficult to find well-curated datasets describing social media use, especially surrounding large external events. External events can often be hard to predict, which makes gathering such data difficult. Even in the absence of external events, social media exists on such a large scale that obtaining meaningful data that can be analyzed with moderate processing power is a challenge. Without much data, it is hard to make meaningful progress in generating data models. As such, it is not surprising that the current models are not very robust, nor that we were not able to get conclusive results. It is worth noting, though, that this area of research is still rather young. As more data becomes available, we expect that the ability to optimize these techniques will improve as well.

9 Contributions

We're comfortable being graded equally for each part of the project - we feel that we each did equal work.

References

[1] Bakshy, E., Hofman, J. M., Mason, W. A., & Watts, D. J. (2011, February). Everyone's an influencer: quantifying influence on twitter. In Proceedings of the fourth ACM international conference on Web search and data mining (pp. 65-74). ACM.

[2] Chen, W., Wang, Y., & Yang, S. (2009, June). Efficient influence maximization in social networks. In Proceedings of the 15th ACM SIGKDD International conference on Knowledge discovery and data mining (pp. 199-208). ACM.

[3] De Domenico, M., Lima, A., Mougel, P., & Musolesi, M. (2013). The anatomy of a scientific rumor. In Scientific reports, 3.

[4] Goyal, A., Bonchi, F., & Lakshmanan, L. (2010, February). Learning Influence Probabilities In Social Networks. In ACM WSDM. ACM.

[5] Goyal, A., Lu, W., & Lakshmanan, L. V. (2011, December). Simpath: An efficient algorithm for influence maximization under the linear threshold model. In 2011 IEEE 11th International Conference on Data Mining (pp. 211-220). IEEE.

[6] Kempe, D., Kleinberg, J., & Tardos, E. (2003). Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 137-146). ACM.

[7] Lei, S., Maniu, S., Mo, L., Cheng, R., & Senellart, P. (2015, August). Online influence maximization. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 645-654). ACM.

[8] Myers, S. A., Zhu, C., & Leskovec, J. (2012, August). Information diffusion and external influence in networks. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 33-41). ACM.

[9] Ritter, A., Mausam, Etzioni, O., & Clark, S. (2012). Open Domain Event Extraction from Twitter. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1104-1112). ACM.