Content-Centric Flow Mining for Influence Analysis in Social Streams

Karthik Subbian
University of Minnesota, MN.
karthik@cs.umn.edu

Charu Aggarwal
IBM Watson Research, NY.
charu@us.ibm.com

Jaideep Srivastava
University of Minnesota, MN.
srivasta@cs.umn.edu

ABSTRACT

The problem of discovering information flow trends and influencers in social networks has become increasingly relevant both because of the increasing amount of content available from online networks in the form of social streams, and because of its relevance as a tool for content trends analysis. An important part of this analysis is to determine the key patterns of flow and corresponding influencers in the underlying network. Almost all the work on influence analysis has focused on fixed models of the network structure, and edge-based transmission between nodes. In this paper, we propose a fully content-centered model of flow analysis in social network streams, in which the analysis is based on actual content transmissions in the network, rather than a static model of transmission on the edges. First, we introduce the problem of information flow mining in social streams, and then propose a novel algorithm *InFlowMine* to discover the information flow patterns in the network. We then leverage this approach to determine the key influencers in the network. Our approach is flexible, since it can also determine topic-specific influencers. We experimentally show the effectiveness and efficiency of our model.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Flow Mining; Influence Analysis; Content Analysis;

1. INTRODUCTION

Social networks such as *Twitter* and *Facebook* have tremendous amount of content being posted on them on a daily basis. A lot of this content (or key parts of them) is often re-posted by different users on a daily basis, and this leads to cascades of information flow. Examples of such cascades are as follows:

- When a user posts a news-article or other interesting item on their wall in *Facebook*, this is replicated by the friends of this user on their own wall.
- When a user tweets a message (or a link) in *Twitter*, the followers of that user may either re-tweet, or tweet a message containing the same topic (which is reflected by its *hash-tag*). This corresponds to *topical* propagation.

The constant flow of such user content through a network is referred to as a *social stream* [1], and in conjunction with the network structure it is the *only observable aspect of social network dynamics.* The determination of information flow patterns from social streams, such as *Twitter*, provide useful insights, because repeated tweets can be tracked as discrete event paths, the construction of which is revealing in terms of the importance of different actors. This also provides a natural understanding into the key influencers in the network using frequent epicenters (or originators) of the information cascades. This problem is directly related to that of finding influential nodes [2, 3, 9] in a network, though the latter problem assumes that propagation probabilities are *already available*. On the other hand, the information flow mining model provides a holistic and integrated model, which starts directly from the available content flow in the *social network stream.*

The problem of information flows has been widely studied in understanding epidemic spread [4, 3] and flow cascades in blogs [6]. Most of these related work in information flow analysis have been closely associated with influence cascade models [3, 5, 9], such as linear threshold or independent cascade models [3], as they model the underlying process of information diffusion. The main drawback of these models is the use of static edge propagation probabilities, that do not directly capture the dynamic nature of the content flows. Also, the work on actually modeling these transition probabilities is very sparse and almost never uses the *actual* content behavior in the underlying network. Some recent work on estimating influence probabilities [8] uses a number of application-dependent assumptions about the weights of the different edges in the social network. The results of this paper suggest that such static modeling is quite questionable, because the influence behavior of an edge depends on many factors including the actual content of the interaction, the time period, and other factors, *which can only be tracked using the underlying content stream of the social network.* Our approach addresses these problems by tracking the dynamic content flows in the underlying network structure in order to determine influencers.

We develop an efficient algorithm called *InFlowMine* to discover the information flow patterns using content propagation on the underlying network structure. As the problem of information flow and influence mining are tightly coupled, we leverage this solution for influence analysis. We propose a Network Influencer Discovery algorithm called *NDIF*, to discover influencers using the mined information flows patterns. Our framework can easily be tailored to content- and topic-specific influencer discovery.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505626 ...$15.00.

This paper is organized as follows. In section 2 we introduce the problem of information flow mining. In section 3 we discuss an approach to address this problem. In section 4 we explain an application to find the information epicenters using the mined flow patterns. Section 5 discusses the experimental results. Section 6 discusses the conclusions and summary.

2. PROBLEM DEFINITION

In this section, we introduce the necessary notations, and definitions to formally define the problem of information flow mining.

We assume that we have a network $G = (N, A)$, containing the node set N and the edge set A. For ease in exposition, we assume that the edge set A is undirected, though this assumption can be easily relaxed without loss of generality. Each actor $i \in N$ performs a number of *content-based actions* such as sending tweets, or posting wall posts etc. Since, this process occurs simultaneously over the entire set of actors in the social network, the stream of content created by the different actors can be *globally* treated as a *social stream* of text content, denoted by $\mathcal{T} = T_1 \ldots T_n \ldots$. We refer to each T_i as a *content token*. Each such content-token in the stream is associated with the following meta-data:

(1) The content-token T_i represents the i-th component of the content created by an actor in the social stream. This could correspond to the actual text of the tweet/post, or only specific parts of the post, such as URL strings, keywords, or hash tags. In the event that the cascade behavior is being observed broadly in the form of topics, rather than re-posts, the string T_i could also simply be one of a set of a topical dictionary of keywords (contained in the tweet), or it could be the hash-tag from the tweet. We note that since this content may be copied, re-posted, re-tweeted, or may generally influence the future content posted by the different actors (in the form of appropriate keywords or hash-tags), the value of the different strings T_i (over different values of the index i) will often be the same. It is precisely this propagation of the content over the network, which we wish to track. In general, for a given application, it is first pre-decided what kind of content to track for interesting information flow patterns (e.g. URLs posted, full tweets or keywords from a specific topic), and then all future analysis is applied to this set of content-tokens. An interesting feature of our approach is that it can be easily applied to an arbitrary subset of content-tokens in order to facilitate topic-specific influence in the network.

(2) It is assumed that the content T_i is created by the actor a_i. The value $a_i \in N$ is an index drawn from the actor set N, and it represents the actor who is responsible for the origination of that particular piece of content.

The entire social stream is denoted by $\mathcal{T}$. We note that in many social networks such as *Twitter* and blog networks, the stream of content can be tracked in an automated way. In many cases, the re-posting of the content is a direct result of a particular actor being influenced by specific neighbors, and the cascade effect of this influence behavior, which can be monitored in terms of the URLs posted, the keywords (or hash-tag) in the posts, or the exact content of the tweets. Therefore, it is interesting to determine the frequent information flows, *in which consecutive participants share a neighbor relationship*, and the content of the (tracked subset of the) post is the same.

We define the information flow patterns in terms of *actors* rather than the *content*, as the same sequence of actors may be involved in multiple such content interactions over different portions of the stream. The purpose of our approach is to determine such frequent information flows, where certain content flow may occur frequently in a specific path of the network. In this paper, we assume that flow patterns are essentially sequential paths in the network, however, resulting flow cascades can be obtained by overlaying multiple such frequent flow patterns. Correspondingly, we define the model for determining the information flow patterns in networks. We start off by defining a *valid* flow of information, based on the network relationship of different actors:

DEFINITION 1 (VALID FLOW PATH). *A valid flow path in the network $G = (N, A)$ is an ordered set of nodes $i_1 \ldots i_k \in N$, such that all nodes $i_1 \ldots i_k$ are distinct from one another, and for each $r \in \{1 \ldots k-1\}$, an edge exists between i_r and i_{r+1} in A.*

We note that the purpose of defining a valid flow path is to focus only on those flows which are governed by the *network relationships* between actors, rather than arbitrary links. This is because of the underlying assumption is that network cascades are caused by social interactions, copying behavior, and more subtle influences between neighbor nodes.

For the ease of analysis, we consider only the *first* posting of a particular piece of content by an actor in the social stream. In effect, this implies, the actors not influenced repeatedly by their own posts. In many cases, repeated postings which are exactly identical are more likely to be spam, rather than postings of specific interest, and the issue of *first posting* is usually more critical. Therefore, it makes sense not to include such repeated posts caused by the same content from the same actor. For a given actor $j \in N$, who posts the content $U \in \mathcal{T}$, the index of the first position of the content U in the social stream $\mathcal{T} = T_1 \ldots T_i \ldots$ is denoted by $F(j, U)$. Thus, the content $T_{F(j,U)} \in \mathcal{T}$ is the same as U. In the event that the content U is *not posted* by actor j, we have $F(j, U) = -1$.

Next, we define the concept of *information flow frequency*. This is essentially defined as the number of times that an ordered set of actors posts the same content-token in that order.

DEFINITION 2 (INFORMATION FLOW FREQUENCY). *The information flow frequency of actors $i_1 \ldots i_k$ is defined as the number r of* unique content-tokens *$U_1 \ldots U_r$, such that for each U_j, we have:*

- *Each actor $i_1 \ldots i_k$ has posted U_j. Thus, we have:*

$$F(i_r, U_j) > 0 \; \forall r, j \tag{1}$$

- *Each content U_j was posted by the actors in the order $i_1 \ldots i_k$. Thus, for each U_j we have:*

$$F(i_1, U_j) < \ldots < F(i_r, U_j) < \ldots < F(i_k, U_j) \tag{2}$$

The above definition can be generalized in order to determine the *frequent flow path* for a specified frequency. This is essentially a sequence of actors who share a neighbor relationship in the network, and post the same set of content-tokens in a specific order, possibly as a result of being influenced by the behavior of their neighboring actors in the network.

DEFINITION 3 (FREQUENT FLOW PATH). *A sequence of actors $i_1 \ldots i_k$ is said to be a frequent flow path at frequency f, if the following two conditions are satisfied:*

- *The sequence of actors $i_1 \ldots i_k$ is valid flow path.*
- *The information flow frequency of the actor sequence $i_1 \ldots i_k$ is at least f.*

Algorithm *InFlowMine*(SocialStream: $\mathcal{T}$
frequency: f);
begin
for each node i
$\mathcal{F}_i^1 = \{\}$;
if node i has at least f posts in $\mathcal{T}$ **then** $\mathcal{F}_i^1 = \{i\}$;
end
$k = 1$;
while $\cup_i \mathcal{F}_i^k$ is not empty **do**
for each i generate $\mathcal{P}_i^{k+1}$ from $\mathcal{F}_i^k$ by by appending each frequent neighbor node of i to the paths in $\mathcal{F}_i^k$, and keeping only potential paths with all distinct nodes;
for each i prune any path from $\mathcal{F}_i^{k+1}$, for which at least one of its path of length k is not in $\mathcal{F}_r^k$ (for some r);
for each i generate the path in $\mathcal{F}_i^{k+1}$ which are valid frequent flow paths;
Reorganize the frequent potential paths $\cup_i \mathcal{P}_i^{k+1}$ based on the last nodes of these paths, and generate the corresponding path lists $\mathcal{F}_i^{k+1}$ for each node i;
$k = k + 1$;
end
return($\mathcal{S}(f) = \cup_k \cup_i \mathcal{F}_i^k$)
end

Figure 1: Algorithm for Information Flow Mining

The frequency f expressed here is absolute and can be easily converted to relative frequency by appropriate normalization. The frequency, in essence, indicates the strength of the flow path, as together with the *valid flow path* constraint it ensures the flow obeys the structure of the network. The problem of information flow mining is formally defined as follows.

PROBLEM 1 (INFORMATION FLOW MINING). *Given a graph G, a stream of content propagations $\mathcal{T} = T_1, \ldots, T_i, \ldots$ and a user specified frequency f, the problem of information flow mining is to find the set of all frequent flow paths S in the underlying graph G using the unique content tokens $U_1, U_2, \ldots, U_m$ generated from the content stream $\mathcal{T}$.*

3. FLOW MINING ALGORITHM

In this section, we discuss our algorithm for mining information flows from the social stream $\mathcal{T}$. A simple method for doing so would be to generate all possible information flows in a brute-force manner and test if they satisfy the frequent flow path conditions. However, this kind of approach is extremely slow as it completely ignores network relationships and frequency of interest.

We propose a level-wise approach to efficiently perform the flow mining process. Our level-wise approach expands the flow paths in the network closely in coordination with the knowledge about the network structure. The k-th iteration of this approach generates an information flow path of length k, which satisfies both the network and frequency constraints. In order to achieve this goal, we maintain a linked list associated with each node in the network. At the end of the k-th iteration, this list essentially contains all the frequent flow paths of length k (i.e. containing k nodes) with at least frequency f, *which end at node i*. We denote this *frequent flow path list* at each node i at the end of iteration k by $\mathcal{F}_i^k$.

The algorithm proceeds as follows. In the first iteration, we generate all the singleton actor nodes, which have at least a frequency of f in the social stream $\mathcal{T}$. Thus, in this case, each node i contains a list $\mathcal{F}_i^1$ of either 0 or 1 element, depending upon whether or not that actor has a frequency of at least f in the social stream. In the next step, we examine all neighbors of a given node in order to generate all paths of length 2. Specifically, we concatenate all frequent neighbors of node i at the end of each path in $\mathcal{F}_i^1$, in order to generate *potential 2-length paths* denoted by $\mathcal{P}_i^2$, which have i as the *penultimate node*. From these candidate paths, we determine those for which the frequency is at least f. It is important that the frequent flow paths from $\mathcal{P}_i^2$ cannot be added to $\mathcal{F}_i^2$, since the node i is now the penultimate node for the corresponding flow path, where $\mathcal{F}_i^2$ may contain only paths in which i is the final node. This means that $\mathcal{F}_i^2$ is not a subset of $\mathcal{P}_i^2$, but a subset of the union of the potential lists at its neighbor nodes. Therefore, the frequent paths of $\mathcal{P}_i^2$ are added to the lists $\mathcal{F}_i^2$ corresponding to the final nodes in these paths. This process is repeated until iteration k for node i when there are no more potential paths in $\mathcal{P}_i^{k+1}$.

In order to speed up the process of flow mining, a pruning technique is used. It uses the fact that the frequent flow path satisfies antimonotonicity property, i.e. all k subsets of a $(k+1)$-frequent flow paths are also frequent. We further note that we only care about those k length path, which are *contiguous* in the network; therefore, we only have to check the k length path after dropping[1] the *first* element in a $(k+1)$-candidate. Therefore, we can prune any $(k+1)$-candidate in $\mathcal{P}_i^{k+1}$, if this path of length k is not a frequent flow path. Let us consider a path $Q \in \mathcal{P}_i^{k+1}$, which ends in node r. Then, we prune the flow path Q from the potential list if any of its path of length k (and also ending with r), are not found in $\mathcal{F}_r^k$. The process is repeated iteratively to generate frequent flows of longer and longer lengths. The process finally terminates, when in a given iteration, the potential paths $\mathcal{P}_i^{k+1}$ are empty. All the frequent flow paths satisfying the network constraints are returned at the end of the algorithm. The discovered frequent flow paths is denoted by $\mathcal{S}(s) = \cup_k \cup_i \mathcal{F}_i^k$. The overall algorithm is illustrated in Figure 1.

3.1 Online Re-organization of Social Stream Content via Hashing

The efficiency of the algorithm, listed in Figure 1, depends on efficiently keeping track of the frequency of the information flow paths. An important point to remember is that this process requires determination of content-influence behavior between different posts. As mentioned earlier, this influence is detected in terms of *content-token* propagation. In order to simplify the string matching and frequency tracking process, we use a hash table to track the sequences of actors for each distinct content-token in the stream. Thus, for each *distinct* content-token $U \in \mathcal{T}$, we map it into the hash table index $h(U)$ (typically with the use of a standard hashing technique such as a linear hash function with linear probing for collision resolution). As mentioned earlier, the content-token U may be a URL string, tweet message string, hash-tag, keyword, or another text content string depending upon the underlying scenario, and the particular content that we have pre-decided to track in the social stream. Then, for each distinct value of U in the social stream, we determine all actors j for which $F(j, U) > 0$. This is the set $P(U) = \{j : j \in N, F(j, U) > 0\}$.

For each distinct value of the content U in the social stream, its hash table slot $h(U)$ points to a linked list of actors in $P(U)$. Furthermore, the actors in $P(U)$ are organized by order of their values of $F(j, U)$. Thus, the set $P(U)$ is converted into the *ordered list* $O(U)$, based on the first time of occurrence of the content U for a particular actor. The entire set of lists in the hash table is denoted by $\mathcal{O} = \cup_{\text{All distinct content } U} O(U)$. After this re-organization process is over, we have a set of ordered lists pointed to by the hash table, which contains the list of actors who have posted that content, in order of their (first) occurrence in the social stream. Because of

[1]The path after dropping the *last* element is already known to be in its corresponding frequent path list.

the use of a hashing approach, this conversion can be performed efficiently, and in online fashion with the social stream. Specifically, for each incoming post of content U by actor j, we determine if the actor j occurs in the list pointed to by $h(U)$. If this is the case, then the list is not modified. Otherwise, the actor j is appended to the end of the list. Thus, at any point in the social stream, this data structure is always available for efficient tracking of flow path frequency and maintained in an online fashion.

As the number of valid potentials for the paths of length two is very high for some tokens, we implemented a specialized tracking method for this case to improve the efficiency further. The reason for this is that the some of the content-tokens U have a large number of actors who posted them, which corresponds to a large value of $|O(U)|$. On the other hand, for most of the tokens it was relatively small. So, depending upon the length of the list $O(U)$, we perform the frequency tracking in the following way:

(1) We maintain a global data structure containing counts of all pairs of content-tokens. If the actor sequence for the unique content is less than $\sqrt{|\cup_i \mathcal{P}_i^2|}$, we increment the count of all pairs of content-tokens, which occur in this list $O(U)$. This can be achieved by using a doubly nested loop on the ordered list, and this requires $O(|\cup_i \mathcal{P}_i^2|)$ time per list. (2) For some lists which are very long, we explicitly check the membership of each flow path of length 2 in $\cup_i \mathcal{P}_i^2$ with respect each unique content U_k, in order to increment the support counts in the global data structure. This approach also requires $O(|\cup_i \mathcal{P}_i^2|)$ time per unique content token.

The reason for using this approach is to optimize the frequency tracking process for longer and shorter flow paths, so that the two approaches require the same amount of time at the threshold value, but are optimized within their respective list sizes.

3.2 Topic or Content Specific Flow Mining

An important point to remember is that the information flow mining problem can be made to focus on specific topics or content by calibrating *the kind of content which is tracked for flow paths*. This cannot be easily achieved by cascade and influence analysis techniques which do not directly use the content in the mining process. For example, let us consider the case where a cosmetics manufacturer wishes to determine the content flow patterns corresponding to the topic of *cosmetic products*. The first step is to create a vocabulary D of all content-tokens, such as popular hash-tags, URLs or other strings related to this topic. Then, the occurrence of these content-tokens in the network posts are treated as the strings T_i for content analysis. Specifically, for each post which contains one or more content-tokens from D, we treat each post as a sequence of strings from that user containing all the contained content-tokens from D in that post. We further note that this kind of filtering *greatly reduces* the amount of content (and the number of valid flow paths) which need to be tracked. This can greatly speed up the algorithm *InFlowMine* described in Figure 1, while simultaneously achieving the goal of topic-targeting.

4. INFLUENCE MINING APPLICATION

The information flow patterns provide useful insights about the influence of different entities in the network. In this section we describe the use of such information flows in finding the frequent epicenters or influencers of content propagation.

The influence of a node is generally characterized by the number of nodes that forward or propagate its content further in the network [3]. This is precisely captured by the set of all information flow paths discovered in the previous section. There are two main factors that affect the influence of a node in the context of a flow path: (1) the position of the node in a flow path and (2) the number of frequent flow paths in which it occurs. As the node consistently occurs earlier and in more flow paths it is considered as a promising influencer.

We now, more formally, define the problem of influence mining in the context of information flow paths. For any frequent flow path $P = \{i_1 \ldots i_k\}$, discovered in the last section, any particular node i_r can influence all the actors, which occur after i_r in P (including itself). This corresponds to the set $\{i_r, i_{r+1} \ldots i_k\}$. Correspondingly, we define, the influence set of a node-flow pair.

DEFINITION 4 (NODE-FLOW INFLUENCE SET). *The influence set $Q(i, P)$ for a node-flow pair (i, P) is defined as the set of all nodes in flow path P, which occur after the occurrence of i in P, when P contains i. In the event that node i is not contained in P, this set is empty.*

Intuitively the above definition is the set of all nodes which are *frequently* influenced by a given node i through the frequent flow path P. The definition above can be generalized to the case where we are using the entire set of flow paths derived at frequency f, rather than a single flow path P. First, we define the influence set of a given node i at the frequency f. This is essentially the union of all the node-flow path sets for i for all frequent flow paths P in $\mathcal{S}(f)$.

DEFINITION 5 (INFLUENCE OF A NODE). *The node influence $I(i, f)$ of node i at frequency f is defined to be the size of the union of all the node-flow influence sets for each frequent flow path $P \in \mathcal{S}(f)$. Therefore, we have:*

$$I(i, f) = |\cup_{P \in \mathcal{S}(f)} Q(i, P)| \tag{3}$$

Thus, the above definition determines all the nodes, which are influenced by node i by all frequent flow path in set $\mathcal{S}(f)$. This definition can of course be naturally generalized to a set of nodes, rather than a single node, by computing the union of the influence sets of different nodes.

DEFINITION 6 (INFLUENCE OF A NODE SET). *The influence $I(V, f)$ of a node set V at frequency f is the size of the union of the influence sets of the nodes in V. Therefore, we have:*

$$I(V, f) = |\cup_{i \in V} \cup_{P \in \mathcal{S}(f)} Q(i, P)| \tag{4}$$

Ideally, we would like to determine a node set V (of pre-defined size) in a network, for which the influence set $I(V, f)$ is as large as possible. Therefore, the influence maximization problem may be stated as follows:

PROBLEM 2 (INFLUENCE MAXIMIZATION). *For a given frequency f, determine the node set V with at most k elements, for which $I(V, f)$ is maximum.*

The influence of a given set of nodes V is more likely to be maximized, when we pick nodes from different parts of the network, each of which has a large amount of influence in its locality.

The problem of determining the largest influence set is closely related to that of sub-modular function maximization. A function $g(S)$ is defined as a monotonically non-decreasing sub-modular function over its argument (which is the set S), such that the additional gain from adding an element to the set S reduces by further adding elements to the set. In other words, for two sets V_1, V_2, such that $V_1 \supseteq V_2$, we have:

$$g(V_1) \geq g(V_2)$$
$$g(V_1 \cup \{i\}) - g(V_1) \leq g(V_2 \cup \{i\}) - g(V_2)$$

We make the following observation:

Algorithm *NDIF*(Social Stream: $\mathcal{T}$, frequency f, Number of Nodes: k);
begin
Determine all frequent flow paths at frequency f from stream $\mathcal{S}$ (Section 3, Algorithm 1)
Construct influence sets $I(i, f)$ for each node i;
$V = \{\}$
while $(|V| \leq k)$ **do**
Add the node i to V, such that $I(V \cup \{i\}, f) - I(V, f)$ is as large as possible;
end
return(V);
end

Figure 2: Influence Maximization using Information Flows

OBSERVATION 1. *The function $g^f(V) = I(V, f)$ is sub-modular.*

The above observation is easy to show, because the additional gain of adding an element to a larger influence set can be no greater than that of adding that element to its subset. As in [3], the sub-modularity immediately suggests that the greedy algorithm of starting with an empty set, and continually adding the element with the largest *incremental* addition on the function $I(V, f)$ provides an effective solution both in theory and practice. The overall framework for influence maximization and the pseudo-code for the corresponding algorithm is illustrated in Figure 2. We refer to the algorithm as *NDIF*, which corresponds to Network Discovery of Influencers using Flows.

Based on the results for sub-modular function maximization, the greedy algorithm also yields a tight approximation bound for influence maximization algorithm *NDIF*.

THEOREM 1. *The greedy algorithm for influence set maximization yields an approximation bound of $(e-1)/e$, where e is the base of the natural logarithm.*

Our approach is fairly easy to adopt for any specific goals of influence maximization. For instance, it is possible to target influence analysis in specific topics or areas of expertise, by focusing only on propagation of specific content-tokens. Similarly, to target specific actors L in the social network, by examining the size of $I(V, f) \cap L$ during the application of the influence maximization algorithm.

5. EXPERIMENTAL RESULTS

In this section, we evaluate the efficiency of our approach, *InFlowMine*, in terms of its running time. Then we evaluate the effectiveness of the discovered information flow patterns by comparing the top-k influencers found by our *NDIF* algorithm versus other popular baselines. The results were tested on a 64-bit *Windows* machine, with 8GB RAM and 2.7Ghz Intel i7 processor. Now we describe the measures used for evaluating the flow mining algorithm and the influence mining application.

Evaluation for *InFlowMine*: The proposed *InFlowMine* algorithm addresses a new problem, and hence we have tested its runtime efficiency with respect to a closest baseline, a modified version of a sequence mining algorithm. This is described later in this section.

Evaluation for *NDIF*: Now, we describe how we measured the effectiveness of our flow mining procedure using the *NDIF* algorithm. For this purpose we define a measure called *Influence Score*. We compute the *Influence Score* in terms of how the content in the token set R has spread, for a particular set of influencers K. This is achieved as follows. For each transaction D_i we first mark the seed influencers in K present in D_i as influenced. Then we find each of their neighbors who appear after them in ordered set D_i. We do this, because influence behavior is always assumed to be causal in nature and the ordering in D_i captures this causality. We proceed this iteratively until there are no more neighbors who can be influenced in the transaction D_i. Now, we count the number of influenced nodes for the transaction D_i. We do this for all transactions in D and average out the total influenced nodes per transaction and refer to it as *Influence Score*. This measure is quite meaningful as it captures the causality of influence. The notion of causality here is, if b buys something after a buys it, and if a is a friend of b in the network, then it is highly likely that b's action was influenced by that of a.

Baseline: As our problem definition is new, we use the sequence mining approach as the closest possible baseline. As the sequence mining approaches require a transaction database, we first translated the social stream data $\mathcal{T}$ to transaction database D. Remember, that the sequence mining approach *does not directly address our problem*. Hence, the output of sequence mining has to be further processed to satisfy the *flow path constraints* to construct all valid frequent flow paths. We consider one of the most popular sequence mining algorithm *Prefix Span* [10] as our baseline. We show that our approach extremely efficient in terms of finding flow patterns compared to prefix-span. The efficiency of such an approach is particularly important, because in the case of larger networks, this can restrict our ability to determine important propagation patterns at low frequencies.

The choice of baseline algorithm for comparing against the influence analysis part is very straight-forward. We use the most popular and standard baselines used in several other recent influence analysis work [3, 2, 9, 7]. The first baseline is the *PMIA* algorithm discussed in [9]. This a Prefix excluded extension of the Maximum Influence Arborescence (MIA) model. The model takes a network structure with defined edge probabilities. We calculated the probabilities using the weighed cascade model proposed in [3]. Our next baseline algorithm is *DegreeDiscountIC*, which is the degree discount heuristic of [2] developed for the uniform IC model. The *PageRank* algorithm is the standard algorithm used for ranking web pages in web search and we used power method to compute the page rank values. We used 0.15 as the restart probability and a difference of 10^{-4} in $L1$ norm between two successive iterations as the stopping criteria. The *SP1M* is the shortest path heuristics algorithm of [7] with lazy forward optimization defined in [5].

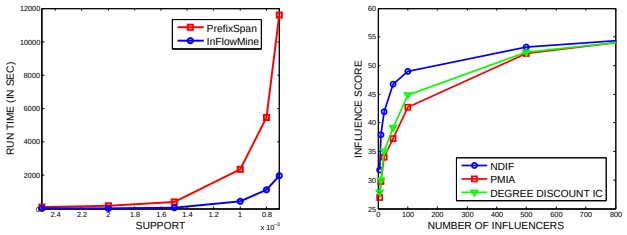
Data Set: We downloaded the publicly available DBLP XML data set [2], and extracted the month, year, title and author details for each of the published document. We cleaned the DBLP data set to remove documents with missing meta-information such as title, author or date. After cleaning, the DBLP dataset had 1,008,883 distinct authors and 1,810,117 documents. The dates on the documents were used in order to create a social stream of content, where the keywords in the titles formed the content-tokens. Temporal ties between documents were resolved using their title string. This data set consisted of 198,760 distinct keywords. We also constructed the DBLP co-authorship network, which contained 1,008,883 nodes and 3,383,570 edges.

5.1 Efficiency Results

As a baseline, we used a modified version of the well-known *PrefixSpan* algorithm[3] for frequent sequence mining [10]. We first find all frequent sequences (irrespective of network structure) at the required support level, and then remove those which do not satisfy the network validity constraints. This is possible only after the appropriate construction of the transaction database described

[2] http://dblp.uni-trier.de/xml/

[3] www.cs.uiuc.edu/~hanj/software/prefixspan.htm



(a) Runtime Analysis (b) Influence Analysis

Figure 3: Runtime and Influence Analysis results.

	Running Time in Secs.		W/O Net. Pruning		W/ Net. Pruning	
Support Level	Prefix-Span	Our Algo.	$\mathcal{P}2$	$\mathcal{P}3$	$\mathcal{P}2$	$\mathcal{P}3$
0.0015	415.23	70	20926050	4060	82509	355
0.001	2373.38	447	153351072	630352	234368	22870
0.0008	5461.77	1123	365823002	4498694	361843	79410
0.0007	11580.93	1984	596751612	13001695	456795	147361

Table 1: Efficiency analysis for *InFlowMine* versus *PrefixSpan*.

earlier. The results are illustrated in Figure 3(a). The actual running times of the $PrefixSpan$ and $InFlowMine$ algorithms are also illustrated in Tables 1 respectively.

It is evident that the running times of our approach were significantly lower than the baseline method, and this advantage increased with lowered frequency. This can easily be seen in terms of the scalability with frequency level, in which our approach scaled linearly with the frequency, whereas the *PrefixSpan* method scaled exponentially. At the lower frequency levels the *InFlowMine* method is *faster by an order of magnitude compared to the baseline*. The advantage of our method is primarily a result of flow path validity pruning, while in the process of discovering the patterns. This can easily be seen from the columns $\mathcal{P}2$ and $\mathcal{P}3$ of Tables 1, which list the number of potential paths of lengths two and three before and after network pruning for each of the data sets. It is evident, that only a very small fraction of the potential paths survive flow path validity constraint, and this significantly contributes to the efficiency of our approach.

5.2 Effectiveness Results

We tested the effectiveness of InFlowMine algorithm with respect to the influencers found using the discovered information flows. We do this by evaluating the influence score measure for NDIF algorithm varying the number of influencers as shown in Figure 3(b). The *SP1M* baseline never completed and the *PageRank* method almost fully overlaps with *PMIA*, and hence they are not shown for clarity. We note that all methods perform very similarly when the number of influencers is too small or too large, because in these cases the problem becomes trivial. However, in the "interesting" range of values on the X-axis, our approach performed significantly better. With less than 100 influencers our algorithm is able to propagate almost all the relevant content-tokens to the different nodes as possible. On the other hand, the best baselines such as *PMIA* and *DegreeDiscountIC* do not even come close at a seed of 100 influencers. This illustrates, the power of using information flows to discover influencers is much more effective compared to the standard approaches.

5.3 Efficiency of NDIF Algorithm

We now analyze the efficiency of the *NDIF* algorithm in comparison with the other baselines. The results for the different methods are illustrated in Table 2. It is evident that some of the methods are so slow, that they can sometimes become hard to use, especially if multiple runs are needed at different times in the social stream scenario. For example, the *PMIA* algorithm was slower by

Method	NDIF	PMIA	DD-IC	PageRank	SP1M
RunTime (Sec.)	472.20	1965.59	21.81	6.29	>86400

Table 2: Efficiency Analysis for NDIF algorithm.

a factor of 10 compared to our *NDIF* algorithm. The *SP1M* algorithm was much slower and never completed after running for more than a day. Hence we report > 86400 seconds for SP1M baseline. In addition to being slower, our results in the previous subsection showed that these algorithms also performed poorly in quantitative influence measures. On the other hand, the running time of *NDIF* was quite modest, and it can easily be applied at any point in the social stream, as long as the online hash tables described are continuously maintained. While *NDIF* is not quite as fast as *DegreeDiscountIC* or *PageRank*, the latter methods performed too poorly to be practical in these scenarios. Therefore, the *NDIF* algorithm provides the best tradeoff between effectiveness and efficiency, and unprecedented flexibility in terms of content-centric analysis.

6. CONCLUSIONS AND SUMMARY

The availability of increasingly larger amounts of content in online networks makes it possible to perform information flow pattern discovery than ever before. In this paper, we introduced the information flow mining problem based on the propagation of content in the network structure. Then, we have proposed an approach to mine the flow patterns, following specific flow validity constraints. We show the effectiveness of the discovered patterns using an influence mining application. Our experimental results demonstrate that our proposed flow mining approach is extremely efficient and is well suited for several application areas such as influence mining.

Acknowledgements

This research was sponsored by the Defense Advanced Research Project Agency (DARPA) agreement number W911NF-12-C-0028 and Army Research Laboratory (ARL) cooperative agreement number W911NF-09-2-0053.

7. REFERENCES

[1] C. Aggarwal, K Subbian. Event Detection in Social Streams, *SDM Conference*, 2012.

[2] W. Chen, Y. Wang, S. Yang. Efficient Influence Maximization in Social Networks, *KDD Conference*, 2009.

[3] D. Kempe, J. Kleinberg, E. Tardos. Maximizing the Spread of Influence in a Social Network, *KDD Conference*, 2003.

[4] J. M. Kleinberg, The flow of on-line information in global networks, *SIGMOD Conference*, 2010.

[5] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. S. Glance. Cost-effective outbreak detection in networks, *KDD Conference*, 2007.

[6] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, M. Hurst. Cascading Behavior in Large Blog Graphs, *SDM Conference*, 2007.

[7] M. Kimura, K. Saito. Tractable models for information diffusion in social networks, *PKDD Conference*, 2006.

[8] C. Wang, J. Tang, J. Sun, J. Han. Dynamic Social Influence Analysis through Time-Dependent Factor Graphs, *ASONAM Conference*, 2011.

[9] W. Chen, C. Wang, and Y. Wang, Scalable influence maximization for prevalent viral marketing in large-scale social networks, *KDD Conference*, 2010.

[10] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, M. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, *ICDE Conference*, 2001.