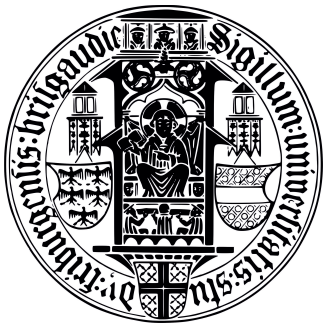
Dissertation zur Erlangung des
Doktorgrades der Technischen Fakultät der
Albert-Ludwigs-Universität Freiburg

Information Diffusion and Provenance in Social Media

Io Taxidou



Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik

2017

Dekan

Prof. Dr. Oliver Paul

Referenten

Prof. Dr. Peter Fischer
Albert-Ludwigs-Universität Freiburg

Abstract

Modern social media like Twitter or Facebook encompass a significant and growing share of the population, e.g. 2 billion users on Facebook, which is actively using it to create, share and exchange messages, e.g. 350K messages/minute on Twitter. This has a profound effect on the way how news, events and all kinds of information are spreading in terms of frequency, reach and speed. Such social media are by no means self-referential and disconnected from the physical world, as they is reflecting and also influencing events there. Given their widespread use on mobile devices, their broad coverage of the world as well as the fast reaction times, social media act as a powerful *"social sensor"*. They detect and publish events in a broader scope with higher speed than traditional media.

Information diffusion is the field which studies the spreading of news, events or other kinds of information over social media. It has attracted attention from many different research areas, for example epidemiology which investigates how diseases propagate over networks. In particular information diffusion over social networks addresses the questions: *how information is propagating on social media and what are its drivers? How fast is it spreading and which are the audiences reached? Which are the paths that information took and how it will be diffused in the future? Which are the influential users which can have an effect over large online populations?* Those questions provide valuable insights for the better understanding of information diffusion on social media.

While the direction to which information is diffused and its impact covers one aspect of analysis, another important one is to identify the sources of information. *Provenance* is the field that seeks the sources, the intermediate steps and any modifications that a piece of information has undergone on the way. Typical research communities that engage with provenance include databases, streaming systems, workflows and the semantic web. Provenance analysis provides valuable answers for the relevance, trustworthiness and quality of information. Consequently, social media can benefit massively from the provenance methodologies and analysis tools: yet provenance has received very limited attention in this context. To sum up, information diffusion studies the *prospective* mechanisms (i.e. where will it end up?),

the provenance provides answers for the *retrospective* mechanisms (i.e. where does it come from?). In this thesis we combine ideas and methods from both communities.

Apart from the contributions of information diffusion and provenance, there has been a significant need in performing fast, scalable and real-time analysis. Being able to analyze information in a real-time fashion, brings its own set of novel challenges including combining volume and speed with complex models and noisy data. Computing information diffusion and provenance in a timely way is crucial for particular use cases, for example digital journalists who need to promptly assess specific information and predict its future spread. Moreover, state-of-the-art for information diffusion and provenance reconstruction evaluates small datasets, after the fact, sidestepping requirements for scale and speed of current social media data.

Social media currently provide insufficient mechanisms for information diffusion and provenance tracking and generation. In turn, state-of-the-art focuses mainly on *explicit* diffusion mechanisms given by social media providers, like retweets in Twitter or reshares in Facebook. Often, even for explicit means, insufficient information is provided in order to reconstruct the complete provenance. The *implicit* diffusion mechanisms, which refer to latent interactions and influence, remain understudied due to the difficulties of being captured and properly understood.

The contributions of this thesis are therefore the following:

(1) First, we provide the common foundations for information diffusion and provenance through a detailed survey which explores the state-off-the-art for both communities. (2) As next, we investigate methods for computing and analyzing the explicit and implicit interactions on social media by recovering the missing or latent provenance. Our findings suggest that explicit mechanisms in isolation (used in state-of-the-art) are insufficient to capture a large part of influence and interactions that take place in social media platforms. (3) Form the technical side, we implement systems that reconstruct the provenance of explicit and implicit information diffusion in an *online fashion*, providing valuable insights in a prompt way. Our approaches can be scaled up to a truly web-scale scenario like major events, without compromising on result quality. (4) Lastly, we propose a formal model for information diffusion and provenance, which covers our findings for online interactions and influence. This model is based on the W3C PROV data model, which has a web-native and interoperable format, thus allowing easy publication of provenance data from different sources.

Zusammenfassung

Moderne Social Media Plattformen wie Twitter oder Facebook werden von einem beträchtlichen und stetig zunehmenden Anteil der Bevölkerung täglich genutzt, wobei Myriaden von Nachrichten erstellt, geteilt und ausgetauscht werden. So hat Facebook mittlerweile über 2 Milliarden Nutzer und auf Twitter werden über 350.000 Tweets pro Minute geteilt. Dies hat weitreichende Auswirkungen auf die Häufigkeit, Reichweite und Geschwindigkeit, mit der sich Nachrichten, Ereignisse und beliebige Informationen verbreiten. Social Media sind dabei keineswegs selbstreferenziell und von der physischen Welt abgekoppelt, sie reflektieren und beeinflussen reale Ereignisse auf der ganzen Welt. Aufgrund ihrer weit verbreiteten Nutzung auf mobilen Geräten, und der damit einhergehenden weltweiten Abdeckung sowie der schnellen Reaktionszeiten, fungieren Social Media Plattformen als ein starker *"sozialer Sensor"*. Ereignisse werden hierbei in einem breiteren Umfang und mit höherer Geschwindigkeit erkannt und veröffentlicht, als von klassische Massenmedien.

Information Diffusion (Informationsverbreitung) ist ein Forschungsbereich, der die Verbreitung von Nachrichten, Ereignissen oder anderen beliebigen Informationen über Social Media, untersucht. Er ist in vielen unterschiedlichen Domänen, wie zum Beispiel der Epidemiologie wo die Verbreitung von Krankheiten über Netzwerke untersucht, auf großes Interesse gestoßen. Die typischen Fragestellungen der Information Diffusion in sozialen Netzwerken sind dabei: *Wie verbreiten sich Informationen auf Social Media Plattformen und was sind deren Treiber? Wie schnell werden Informationen propagiert und welches Publikum wird erreicht? Über welche Verbindungen wurden Informationen weitergeleitet und wie werden sie sich in Zukunft weiter verbreitet? Wer sind die einflussreichsten Nutzer, die einen starken meinungsbildenden Einfluss auf die Onlinebevölkerung haben?* Solche Fragen liefern wertvolle Einblicke, um das Verständnis der Informationsverbreitung in Social Media zu verbessern.

Während die Verbreitung von Informationen und deren Auswirkung einen Teil der Analysen bilden, stellt die Frage nach der Herkunft einer Informatione einen weiteren wichtigen Aspekt dar. *Provenance* umfasst in diesem Zusammenhang einen Forschungsbereich, der die ursprüngliche Informationsquelle sowie alle Zwischenschritte und Modifikationen, die eine Information erfahren hat, versucht zu identi-

fizieren. Typische Forschungsgemeinschaften, die sich mit Provenance beschäftigen kommen aus dem Bereich der Datenbanken, Streaming-Systeme, Workflows und dem Semantic Web. Die Analyse der Provenance liefert hierbei wertvolle Antworten zur Relevanz, Vertrauenswürdigkeit und Qualität der Daten. Social Media Plattformen könnten folglich von bereits existiertenden Provenance Methoden und Analysewerkzeugen enorm profitieren, jedoch wurde der Provenance in diesem Zusammenhang bisher nur sehr wenig Aufmerksamkeit gewidmet. Zusammenfassend untersucht Information Diffusion *prospektive* Mechanismen (wie breitet es sich aus?), während Provenance Antworten auf *retrospektive* Mechanismen (wo kommt es her?) liefert. In dieser Dissertation kombinieren wir Ideen und Methoden aus beiden Forschungsbereichen.

Neben den konzeptionellen Beiträgen zu Information Diffusion und Provenance, besteht ein erhebliches Interesse in schnellen, skalierbaren Echtzeitanalysen. Diese Analysen bringen neue Herausforderungen mit sich, da enorme Datenmengen, die mit einer hohen Geschwindigkeit generiert und weitergeleitet werden, und zum Teil verrauscht sind, mit komplexen analytischen Modellen kombiniert werden. Dabei ist die schnelle Berechnung von Information Diffusion und Provenance für viele Anwendungsfälle, wie beispielsweise Onlinejournalismus, wo bestimmte Informationen zeitnah beurteilt und ihre zukünftige Verbreitung vorhergesagt werden müssen, von entscheidender Bedeutung. State-of-the-Art Lösungen zu Information Diffusion und Provenance verarbeiten allerdings nur kleine Datensätze *im Nachhinein*. Damit werden jedoch die Anforderungen von Social Media Daten umgangen, die insbesondere aus der Größe der Daten und der Geschwindigkeit mit der sie heutzutage erzeugt werden, resultieren.

Weiterhin bieten Social Media zur Zeit nur eingeschränkte Mechanismen an, um Information Diffusion oder Provenance zu untersuchen. State-of-the-Art Ansätze konzentrieren sich vor allem auf sogenannte *explizite* Verbreitungsmechanismen (*explicit Diffusion*), die von den Social Media Anbietern zur Verfügung gestellt werden, wie beispielsweise Retweets in Twitter or Reshares in Facebook. Häufig werden selbst bei den expliziten Vertreibungsmechanismen unzureichende Informationen bereitgestellt, sodass die Provenance nicht vollständig rekonstruiert werden kann. Die sogenannten *impliziten* Verbreitungsmechanismen (*implicit Diffusion*), die sich auf latente Interaktionen und Einflüsse beziehen, sind hingegen bisher kaum erforscht worden, da sie schwierig zu erfassen und zu interpretieren sind.

Die Beiträge dieser Dissertation können wir wie folgt zusammenfassen: (1) Zunächst beschreiben wir die gemeinsamen Grundlagen für Information Diffusion und Provenance durch eine detaillierte Untersuchung, in der State-of-the-Art Verfahren aus beiden Bereichen ausführlich diskutiert werden. (2) Als nächstes untersuchen wir Methoden zur Berechnung und Analyse der expliziten und impliziten Interaktionen in Social Media, wobei die fehlende oder latente Provenance rekonstruiert wird. Unsere Ergebnisse zeigen, dass explizite Mechanismen für sich genommen (wie sie in State-of-the-Art Ansätzen verwenden werden), nicht ausreichen, um einen großen Anteil der Einflüsse und Interaktionen, die in Social Media Plattformen stattfinden,

zu erfassen. (3) Aus technischer Sicht implementieren wir Systeme, welche die Provenance von expliziter und impliziter Information Diffusion in nahezu Echtzeit rekonstruieren, wodurch wertvolle Erkenntnisse zeitnah bereitgestellt werden können. Unsere Verfahren können dabei auf echte Web-Scale Szenarien wie beispielsweise Großveranstaltungen skaliert werden, und das ohne Kompromisse bei der Qualität der Ergebnisse einzugehen. (4) Abschließend schlagen wir ein formales Modell für Information Diffusion und Provenance vor, welches unsere vorgestellten Ergebnisse zu Interaktionen und Einflüssen in Social Media berücksichtigt. Dieses Modell basiert auf dem W3C PROV Datenmodell, ein webbasiertes und interoperables Format welches die einfache Veröffentlichung von Provenance Daten aus verschiedenen Quellen ermöglicht.

Contents

Chapter 1

Introduction

Social media, micro-messaging services, or sharing sites (e.g., Facebook, Twitter, or Instagram) provide the means of interactions among people in which they create, share, and exchange information and ideas in virtual communities and networks. Many real-life situations, such as elections [MM12] or natural disasters [SOM10] are reflected on social media. In turn, social media shape these situations by forming opinions, strengthening trends or by spreading news on emerging situations faster than conventional media. Additionally, activities originating from social media can also have a significant impact on the physical world: protests and demonstrations have been organized via social media during the *arab spring* [BHB13] and *Gezi park arrests* [VFO+14], since traditional media had been under government control. Given the fast and easy access to such platforms, the online population and traffic is impressive: Facebook reported 1.94 billion monthly active users as of March 31 2017, while Twitter reported 40 million daily messages during the 2016 *US presidential election*.

Considering the large share of population participating and contributing actively to such platforms, social media provide large audiences where information can be easily spread and consumed by others. That makes online marketing, and advertisement on social media very appealing since millions of users can be reached in a matter of seconds which brings significant profits. As a result, we observe fast and easy access to information, very large online audiences with a profound effect on information virality and speed, combined with a strong interconnection of the online and physical world.

The rapid and possibly wide spread of information on social media is often exploited in order to propagate negative opinions. The reputation of companies, politicians or celebrities is harmed by such negative views or rumors which might be consumed and re-shared by millions. Furthermore, the spread of disinformation has an impact in the real world with devastating consequences. For example, a tweet in 2013 claiming that two explosions took place in the *white house* caused billions of dollars

to be temporarily wiped off from the *US stock market* [Gua13]. More recently, we observe that social media might influence the outcome of democratic procedures. For example, *bot* [1] accounts were observed to be active only during the period of the *Brexit referendum* in 2016, yet those accounts disappeared after its outcome. While, there was no evidence that those bots spread fake news, they posted user-curated, hyperpartisan and polarizing information. However, such information received a lot of publicity, which altered the public perception of political entities [BM17]. Similar findings were reported for the *2016 US presidential election* with regard to orchestrated bots supporting particular candidates. Consequently, social media bots can have a negative impact on democratic political discussion, which in turn might distort the public opinion and endanger the integrity of the presidential election [BF16].

User interactions leave traces on the online world and provide scientists with a treasure of information for analysis. Given the large share of the online population affected, monitoring social media in a prompt fashion has attracted a lot of interest both by academia and industry, with a strong focus on *event detection* [AMRW12, MK10], *sentiment analysis* [AXV+11, GSS07, WCK+12] and *popularity prediction* [KMF+14, ZSMF15].

An important area of analysis – both on its own and as an underpinning for more complex analysis is – *information diffusion*, i.e. tracing, understanding and predicting how a piece of information is spreading. Such analysis provides valuable insights on who is propagating certain information and who is influencing others. In the literature, information diffusion is modeled as *information cascades*: information cascades are temporal graphs that show *who was influenced by whom* in order to propagate a piece of information.

Provenance is a significant aspect to consider when judging the relevance, trustworthiness and quality of information. Provenance seeks to identify the sources of information, including the intermediate steps and the modifications that a piece of information has undergone on the way. In contrast to typical information diffusion, provenance has received limited attention in the context of social media [GL12]. Likewise, existing models of information diffusion are insufficient to capture provenance [TDNV+15], while social media offer limited or no mechanisms to its users to judge the received information [BFGL13]. Taking Twitter as an example, while in the cases of quotes and replies the direct predescessor is given, the source of diffusion is not. In case of retweets, the source of information is provided in the metadata but not the intermediate forwarders. However, it has been shown that forwarders play an equally important role in the outcome of information diffusion [BBHM13] and, as a result, in the reverse process of provenance. There are also physical limitations to the extent which social media providers can support provenance.

When mechanisms of social media (like *retweet*, *quote* or *reply*) are used, some

[1] A bot is a software that controls a social media account via the API. They can perfom actions like posting messages, liking and forming social connections in an autonomus way.

basic (yet incomplete) provenance is provided and further analysis can be implemented to identify the sources and predescessors. However, we will show that users might sidestep those mechanisms, and use their own conventions for crediting the sources [TFDN+16]. To complicate further matters, users might provide no provenance information, e.g., by copying messages, which renders any further analysis challenging.

Use Case. We will highlight how information diffusion and provenance analysis are useful on an online journalism use case. Since social media exerts a big influence on the way that people are being informed, journalists cannot underestimate this powerful source. In order to take advantage of the rapid update rates and the huge amount of information and cover events while they are happening, journalists need fast and accurate analysis on the fly. Journalists are involved in the diffusion process in three ways:

1. They investigate how information reaches them and through which intermediate steps. This results in an iterative process which validates and updates their sources. The propagation of trustworthy information in a timely way determines their own value as professionals, as a result, the fast detection of rumors and false information is crucial.
2. After contributing their own information, they desire to determine the audience reached. Again, with an iterative process they can update their connections and increase their influence.
3. Lastly, they have to observe current trends and topics with relevant tools (e.g. trend detection systems) in order to increase the efficiency of filtering information and reacting promptly.

As researchers, we thus need to view the information flow in both a prospective way (i.e., its diffusion*: where will it end up?*) as well as in a retrospective way (i.e., its provenance: *where does it come from?*). The main questions we need to address are: *How does information diffuse?* and *What are the drivers of online interactions?* In this thesis, we use the following terms to differentiate among different interactions and influence:

- *Explicit interactions*, based on social media providers' provenance information i.e., mechanisms like retweet in Twitter or reshare in Facebook.
- *Implicit interactions*, based on latent influence indicators, for example users' conventions of crediting their sources, social connections or content similarity.

For explicit interactions, the information that particular messages belong together simplifies our methods and computations. However, for implicit interactions, in lack of any provenance, all previous messages might be relevant for influence computation. The challenge lies in limiting the search space for provenance computation and developing methods to identify latent influence. The speed and virality of information propagation in social media (up to 350K tweets per minute in Twitter),

imposes new challenges for information diffusion and provenance computation in an online fashion. In order to keep up with the stream of messages, incremental and distributed computations are required. In addition, given the need to trace influence, identifying who is connected with whom is a primary concern. This means that the huge social graph (billions of users) needs to be queried in a fast and efficient way.

1.1 Limitations of state-of-the-art and Contributions

There is a plethora of statistical models for information diffusion and other analyses on social media. Such models are trying to approach closer the reality with regard to different aspects of information diffusion. However, they are often developed under the assumptions of specific datasets, which renders those methods not easily generalizable. Additionally, the emphasis is on the whole process and its impact, rather than on the individual diffusion edges. However, the assesment of individual influence edges is of importance to many use cases, especially for relevance and trust computation. Concerning provenance on social media, state-of-the-art is almost non-existing. The limited related work borrows ideas from information diffusion [BFGL13], in particular modeling, to recover any provenance. We also observe very limited research for influence computation at scale and speed on social media that can cover a truly web scenario. Such computations include the combination of stream processing with graph querying, assuming the existence of an underlying social graph.

The *questions* that this thesis addresses are therefore the following:

- *How can we trace information by unraveling user-to-user influence?*
- *Can we identify latent influence which is not provided by social media providers?*
- *Is it feasible to implement such analysis on a real time fashion given the rates of social media and the huge social graphs?*

Our *contributions* that addresses those questions include:

- We identify, model and compute *implicit* and *explicit* interactions on social media. For explicit interactions we leverage several plausible hypothesis to recover the missing provenance information. For implicit interactions, we also form relevant hypothesis; however, since now the support for any provenance is weaker (compared to explicit interactions), we make in addition a thorough investigation to understand and unravel latent user online behaviour and influence. We make use of these observations in order to recover or strengthen the already computed provenance. In any case, the focus is on the reconstruction of individual diffusion edges, rather than on the diffusion processes as a whole.
- For the first contribution we base our concepts, modeling, methods and implementations on two different communities: *information diffusion* and *provenance.* Consequently, we provide a broad and thorough *survey* about both

communities which have been disconnected so far, especially in the domain of social media. We discuss the commonalities and differences of both works and how they solve the similar problem from different dimensions and for different use cases.

- In order to tackle the requirements for fast results we provide two systems that compute *diffusion paths* and influence for implicit and explicit interactions. In particular, those systems are able to perform incremental and online computations at scale and speed. The first system for explicit interactions is based on *social connections* to unravel influence and combines *streams with huge social graphs*. For that, parallel and distributed approaches are needed to handle the large amount of messages and the social graph. The second system is based on *textual similarity* as a means of identifying provenance for implicit interactions. In order to limit the search space for influence computations, we leverage observations from the nature of social media data and online user behaviour.
- The previous contributions considered different types of interactions and two systems that computes them in isolation. The next contribution combines both types of interactions into the same language model. Thus we provide a model for information diffusion that can combine the sources of different interactions and systems.

Structure of the Thesis. In Chapter 1 we provide the introduction and motivation for our work. Chapter 2 investigates the related work extensively for both information diffusion and provenance and makes an attempt to bridge those communities. In Chapter 3 we describe the scope, our methods and design decisions.

In Chapter 4 we examine explicit interactions, in particular the methods to unravel influence given social media providers' indicators. Particularly, in Section 4.2 we reconstruct reply and quote cascades and we discuss the methods and limitations of computing them. In Section 4.3 we provide the methodology to reconstruct retweet cascades and in Section 4.3.5 we construct a system that implements such analysis on a real-time fashion by leveraging concepts of distribution and social graph partitioning.

Chapter 5 investigates the field of implicit interactions. We develop methods that reveal the latent influence by means of semantic similarity and we identify indicators that strengthen the reconstructed provenance. In Section 5.3 we implement a system that scales implicit influence computations to a real time scenario.

For modeling formally information diffusion, we provide the semantics based on the W3C PROV Data Model (PROV-DM), in Chapter 6. PROV is traditionally a provenance model with the advantage of a web-native and interoperable format that allows easy publication of provenance data, and minimizes the integration effort among different systems making use of PROV. Furthermore, we provide insights how different interactions can be computed under the same model. Lastly, Chapter 7 concludes our work and presents directions for future research.

Chapter 2

Survey

2.1 Information Diffusion and Provenance on Social Media: Two Sides of the Same Coin

The goal of this Chapter is two-fold: 1) we extensively survey the related work relevant for this thesis and 2) we compare information diffusion and provenance research fields. In particular, we bring together the concepts of information diffusion and provenance on social media and discuss their goals and scope. In this thesis, we show that these two communities investigate the same problem from different directions and share similar methods. Their main goal is to identify *how information propagates, including the sources and the intermediate steps*. However, they seek to understand different aspects of such analysis and provide answers to different questions. For example, information diffusion investigates information reach and virality while provenance strives to identify the sources of information.

The related work is divided according to information diffusion and provenance fields. Within those large sections, different categories that attract research attention unfold. Orthogonal to those categories, we define a classification scheme in order to describe the different works under the same dimensions. The same classification scheme is provided for all categories. In case some dimensions of the classification do not apply for some categories, we omit them for this instance. In this classification, we focus on social media/network analysis since traditional provenance research cannot be placed under the same dimensions with the rest of the works. For example, provenance in databases focuses on understanding the outcome of particular queries and from where the data was drawn. Another example is provenance of sensor data, where storage and organization of such data is a major concern. Constructing a common classification for those works and typical information diffusion in social media is not possible and out of the scope of this work. However, for the sake of extensively discussing related work, we include representative works for traditional provenance as well. In order to compare related and our own work

under similar dimensions, we construct relevant tables (2.1, 2.2, 2.3, 2.4 2.5, 2.6). Note here that we compare related work only when applicable, i.e. the focus is on information diffusion or provenance in social media, especially when diffusion paths are reconstructed.

2.2 Related Work: Information Diffusion

Here, the representative state-of-the-art is presented and is categorized according to research directions. Information diffusion is an interdisciplinary field combining concepts not only from mathematics and computer science but also from epidemiology and social sciences. Many fundamental information diffusion and influence models are based on approaches that are used in epidemiology and the spread of diseases [Het00, New03]. Likewise, the results of information diffusion analyses are explained under the lenses of social sciences since they involved real users who interact on the online world [Gra83, LT13]. For example the concepts of *weak ties* and *homophily* from sociology are applied also to social networks and information diffusion. Information diffusion is usually modeled with *information cascades*. Information cascades are graphs that reveal how information propagates from user to user, often with the assumption of an underlying social graph.

We observe the following directions for information diffusion in social media which will be discussed in more detail: 1) *modeling information diffusion* in Section 2.2.1 includes statistical and probabilistic models that focus on the diffusion processes as a whole, 2) *inferring diffusion paths* in Section 2.2.2 presents different methods to compute influence by means of modeling or explanatory (no model based) analysis, 3) *analysis of information cascades* in Section 2.2.3, which focuses on the statistical, spatio-temporal and social properties of cascades, 4) *predictions for information diffusion* in Section 2.2.4 presents methods to predict the diffusion process in terms of size, speed, virality, lifetime, etc., 5) *identifying influentials* in Section 2.2.5 investigates the methods to those users in the social graphs, who, if targeted, can spread information to large audiences, 6) *event/trend identification* in Section 2.2.6 focuses on the real time methods for fast computations and detects emerging topics or events.

2.2.1 Modeling Information Diffusion

Modeling of information diffusion on social media has received a lot of attentions and multiple models have been developed considering different aspects and making different assumptions. Modeling information diffusion has two main goals: from the one hand to understand the underlying process and its evolution and from the other side to implement predictions based on such models. There exist some fundandamental models like SIS, SIR, IC, LT [New03, Het00, GLM01, Gra78] which

provide the baseline for further works and research. The models of *SIR (Susceptible-Infectious-Recovered)* [Het00] and *SIS (Susceptible-Infectious-Susceptible)* [New03] are borrowed from epidemiology and group the nodes according to states. SIR models all nodes in one of these states: susceptible (able to be infected), infected, or recovered (no longer able to infect or be infected). At each time step, nodes infected at the last time step, can infect their neighbours who are in a susceptible state with a fixed probability. After that time step, the node that was previously in an infected state makes the transition into a recovered state and is no longer able to infect or get infected. The SIS model is very similar to the SIR mode with the difference that it has only two states (susceptible and infected). Instead of being recovered, a node is again in a susceptible state by meaning that he can infected and get infected. These models make no assumption of an underlying social graph.

Similarly, models like *Indepdentent Cascade (IC)* [GLM01] and *Linear Threshold (LT)* [Gra78] make the assumption of an underlying social graph in order to model information diffusion. The IC models generalizes over the SIR model: instead of a single probability infection, there is a probability of infection associated with each edge. Such a probability can be assigned considering different metrics like the amount of previous interactions or past infections. If a node gets infected it can infect his neighbours according to such probabilities. The LT model associates an influence weight on each edge and an influence threshold for each node randomly. At each step, an inactive node becomes active if the total weight of its incoming neighbors is higher than the threshold.

The authors in [LMF$^+$07] are implementing a large scale blog study based on the SIS model. Their modeling generates cascades that match real cascades in terms of degree distribution and shapes. Two fixed parameters are used which corresponds to the probability of nodes to adopt some information and to become susceptible in the next step. Their study confirms the power law distributions that we observe on the web (size of cascades, size of blogs, in-degrees and out-degrees) and analyzes the structural patterns in blogs.

Another study by [YL10] approaches closer the reality by assuming that the influence of individual nodes are impacting on information diffusion without the assumption of an underlying social graph. The authors develop an approach that models the volume of information diffusion over time as a sum of influences of nodes that propagated the same piece of information in the past.

Other works are focused on particular aspects of diffusion: For example the work of [MZL12] focus on the modeling of external influence. The authors develop a model that assumes information can reach nodes via social links or through the influence of external sources. The observation that information jumps across the network (which is formed by social connections), is an effect of an unobservable external influence. The results indicate that 71% of the information propagation is attributed to social connections, while 29% of the diffusion happens because of external influence.

An alternative way of modeling influence is the hypothesis that "*users implicitly in-*

teract with one another by expressing their preferences on the shared items" [ISG13]. In this work a probabilistic model for finding latent influence is proposed, which supports the identification of influentials, identification of pair-wise influence and prediction of popular items. Information used to infer influence includes shares, clicks, likes and joined communities.

Table 2.1 presents the works in the predefined classification. We observe that these models aim to understand the diffusion processes (coarse-grained analysis), rather than reason about individual edges. Statistical modeling is a common means to infer influence, by using normally parametric methods for simulations. The evaluation is targeted towards understanding and simulating those processes on a coarse-grained level. Also, the focus is not on developing scalable models to process the large amounts of data produced nowadays but rather on approximating the processes as closely to reality as possible. This in turn leads to complex and computationally expensive models, because normally a large number of parameters need to be estimated (e.g. [MZL12, YL10]). Only the latest work of [ISG13] raises efficient issues. Missing data is out of the scope of this works, but such methods can be used for inference.

2.2.2 Inferring Diffusion Paths

Research on inferring information cascades aims in unraveling their underlying structure given a sequence of activation. The models developed here reconstruct the path taken by a piece of information. For example the continuous work of Manuel Gomez-Rodriguez [GRLK10, RBS11, GRLS13] studies this problem focusing on different dimensions: the work in [GRLK10] reconstructs the social graph and cascades over which information propagate. The authors build a model that finds the spreading cascades by maximizing the likelihood of observed data. Every node can activate its neighbour independently based on some probability. Particularly, by observing many different cascades spreading from node to node, the edges of the underlying graph (and as a result cascades) can be inferred. The datasets used for evaluation include memes propagating over blogs and news websites. The results indicate a core-periphery structure of the underlying social graph and large influence from mass media is identified. The authors extend their model in [RBS11] in order to account for the times and rates of transmission of individual edges rather than having a uniform probability for each edge. Such a problem has been addressed before [ML10] but with certain assumptions and heuristics, while the solution of [RBS11] does not require parameter tuning.

In these two works [GRLK10, RBS11] the authors assume that the underlying social graph stays the same, which is not a correct assumption in reality. In [GRLS13] the authors account for the dynamics in terms of structure and timing of the underlying social graph. The underlying social graph is unraveled by observing the infection times of nodes. The authors analyse the dynamic information diffusion paths and

Table 2.1: Modeling of Information Diffusion

		[Het00, New03]	[GLM01, Gra78]	[LMF+07]	[YL10]	[MZL12]	[ISG13]
Goals - Types of Analysis	Predictive			×	×		×
	Exploratory	×	×				
	Modeling	×	×	×	×	×	×
Datasets	Social media/blogs/etc			×	×	×	×
	News				×		
	Other	×					
	Synthetic		×				
Influence Computation	Features						
	Network (social or diffusion)	×				×	
	Content						×
	History						
	Time				×	×	
	Statistical model	×	×	×	×	×	×
	Social	×	×				
	Other						×
Social Graph based	Static	×	×			×	
	Dynamic						
	Inference			×			
Non-Social Graph based	Inference						×
	No inference	×		×	×		
Across multiple media				×			
Diffusion Process	Static	×	×				
	Dynamic			×	×	×	×
Granularity of Results	Fine-grained:edge/node						×
	Coarse-grained:process	×	×	×	×	×	×

concluded that they are more stable for general recurrent topics, while real-world events results in large changes over these pathways. The work in [LZ14] takes a different approach in modeling the dynamics of the social and diffusion graphs: under the assumption that individuals tend to break old connections and connect to the their two hop friends, a link rewiring strategy is implemented. Simulations are under the SIR model and with the developed strategy show that information spreads faster and deeper.

While the previous methods were based on modelling, the work by Cogan et al. [CAB+12] studied user interactions on Twitter by reconstructing the conversational graphs of mentions, retweets, replies by means of observable edges. Their dataset is much smaller compared to what we target: it contains 33K retweets while the largest retweet cascade has size 170 retweets. A similar approach is taken by [KKMV14] to reconstruct retweet cascades from tweets that explicitly mention the source in their tweets (*@username*). That was the old convention of retweeting before the official means released. Parts of our work fall in this category: [TF14] reconstructs retweet cascades with the undelying hypothesis that influence flows through social connections in order to unravel influence. We showed that this is a plausible hypothesis and we provide an extensive analysis of retweet cascades. On [FTLH16] we leverage distributed techniques for streaming reconstruction of the method in [TF14]. The results indicate that our system can keep up with the current social media rates. While the previous works inferred information cascades out of individual activations which were assumed to be complete, the work of [SMLGM11] accounts for missing data. The authors build a model to estimate the overall properties of cascades even in cases up to 90% of data are missing. Instead of inferring the overall properties, the work in [ZWSY12] infers missing nodes by incorporating temporal information. However, the cost of such inference is quite significant, depending on the size of the entire social graph.

Table 2.2 summarizes the works under the lenses of the different dimensions. In particular, research in inferring influence paths includes both statistical modeling and explanatory works. *Explanatory* models infer the underlying spreading cascade, given a (complete) activation sequence. Our work falls also in the category of exploratory models since we always assume an activation sequence of users who are propagating related messages.

The granularity is focused on individual edges while some works target missing information.

2.2.3 Analysis of Information Cascades

The statistical, structural and content aspects of information cascades have been studied in [ZBK+10, KLPM10, HTW+12]. In [ZBK+10] authors investigated the size, shape and decay factors of cascades; the biggest cascade in the evaluations dataset contained 1K messages. In [KLPM10] shape and temporal analysis of

Table 2.2: Inferring Diffusion Paths. The general signal "×" indicates support, and "○" indicates "Dynamic". Own publication are highlighted in bold.

		[GRLK10]	[RBS11]	[GRLS13]	[LZ14]	[CAB+12]	[KKMV14]	[SMLGM11]	[ZWSY12]	**[TF14]**	**[FTLH16]**
Goals - Types of Analysis	Predictive	×					×	×			
	Exploratory				×	×	×			×	×
	Modeling	×	×	×				×	×		
Datasets	Social media/blogs/etc	×	×	×		×	×	×	×	×	×
	News	×	×	×							
	Other										
	Synthetic	×	×	×	×			×	×		
Influence Computation	Features						×				
	Network (social or diffusion)				×	×	×	×	×	×	×
	Content						×				
	History										
	Time	×	×	×					×		
	Statistical	×	×	×				×	×		
	Social				×						
Social Graph based	Static					×	×			×	×
	Dynamic										
	Inference										
Non-Social Graph based	Inference	×	×	○	○						
	No inference							×	×		
Missing data	Social graph										
	Messages								×		
	Estimation of Properties							×	×		
	Quantification										×
Across multiple media		×	×	×							
Diffusion Process	Static	×			×	×	×	×	×	×	
	Dynamic		×	×							×
Computations	Real-time										×
	Incremental									×	×
	Scalability/ Efficiency						×		×		×
Granularity	Fine-grained:edge/node					×	×			×	×
	Coarse-grained:process	×	×	×	×		×	×	×		

retweet cascades were analyzed with the biggest retweet cascade containing 4K messages. The authors of [HTW+12] investigated human interactions on a crisis constructing the corresponding cascades, using a tiny dataset containing 168 retweets.

A more in depth analysis of two very different large cascades is described in [DAF13]: the authors explain how two different cascades of similar size unfold in terms of evolution, depth and influential users on Facebook. Very often the users are sharing the initial source, but the paths here are reconstructed according to any previous clicks to intermediate users who also re-shared, which finally lead them to the source. However, such clicking data are not openly available. The work of [GAHW15] defines structural virality in order to differentiate cascades that are the typical broadcast from cascades that grow more complex and deeper. In particular the authors are based on the wiener index and analyze the fine-grain structure of cascades.

Table 2.3 shows those works under different dimensions. Those works reconstruct information cascades and analyze their properties. Their methods are not based on modeling but rather unraveling the individual influence connections given some activation. The datasets come mainly from one source and any analysis is restricted to those particular datasets. The underlying graphs are mostly static, while missing information is not being considered. The main focus is on the statistical, temporal and social aspects of those cascades and the involved users. Lastly, efficient computations and real-time results are out of scope for this category.

2.2.4 Predictions for Information Diffusion

One of the main goals of information diffusion is to implement predictions about the its future state. Many models have been proposed that predict various aspects of information diffusion like size, virality, users reach, etc There is plenty of literature that implements predictions in social media targeting both on the (i) *micro* and (ii) *macro* level: individual users or communities/social networks). An orthogonal aspect is the methodology of predictions: a) *machine learning* techniques with predictive features, b) *statistical modeling* and processes which can predict the next steps of diffusion. Complementary, *systems* that target real-time predictions have been proposed leveraging concept drift to update the predictions, while targeting speed and scale. Concept drift is particularly interesting when developing real-time systems, because data (users, social connections messages and interactions in general) have a strong temporal component and capturing its evolution is crucial.

In the micro level, [GAC+10] considers predictions of which user will emit which URL, while [POL11] predicts whether a tweet will get retweeted and which features are predictive for this task. Complemenraty, the work of [YGC+10] focuses on predicting the user retweet behaviour -whether particular users will propagate certain content- based on user, message and temporal attributes. Lastly, the work of [KKMV14] predicts the information diffusion based on linguistic and profile features, given one tweet.

Table 2.3: Analysis of Information Cascades

		[ZBK+10]	[KLPM10]	[HTW+12]	[DAF13]	[GAHW15]
Goals - Types of Analysis	Predictive					
	Exploratory	×	×	×	×	×
	Modeling					
Datasets	Social media/blogs/etc	×	×		×	×
	News					
	Other					
	Synthetic					
Influence Computation	Features				×	
	Network (social or diffusion)	×	×		×	×
	Content	×	×	×		
	History					
	Time					
	Social					
	Other					
Social Graph based	Static	×	×		×	×
	Dynamic					
	Inference					
Non-Social Graph based	Inference					
	No inference			×		
Missing data	Social graph					
Diffusion Process	Static	×	×	×		×
	Dynamic				×	
Granularity	Fine-grained:edge/node	×	×	×	×	×
	Coarse-grained:process		×		×	×

Table 2.4: Predictions for Information Diffusion

	[GAC+10]	[POL11]	[YGC+10]	[KKMV14]	[YC10]	[ZHVGS10]	[KOU+12]	[CAD+14]	[NGKA11]	[SH10]	[GMC15]	[WYH+15]	[RXS+17]	[RX17]	**[TAFS15]**
Goals - Types of Analysis															
Predictive	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Exploratory	×			×						×			×	×	
Modeling	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Datasets															
Social media/blogs/etc	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
News															
Other															
Synthetic															
Influence Computation															
Features		×	×	×	×	×	×	×	×			×		×	×
Network (social or diffusion)	×		×			×	×	×							×
Content	×	×	×	×	×	×	×	×	×						
History	×	×	×	×	×	×	×	×							
Time	×	×			×	×	×	×		×	×	×			×
Statistical model	×	×													
Social	×														
Other	×														
Social Graph based															
Static	×		×	×		×	×	×	×	×	×	×	×	×	×
Dynamic															
Inference															
Non-social graph based															
Inference					×				×			×	×	×	
No inference		×													
Across multiple media															×
Diffusion Process															
Static	×		×	×		×			×		×				
Dynamic		×			×		×	×			×	×	×	×	×
Real-time		×													
Incremental								×							×
Scalability/efficiency															
Computations															
Fine-grained: edge/node	×	×	×	×	×	×	×		×						
Coarse-grained: process	×			×	×	×		×	×	×	×	×	×	×	×
Granularity of Results															

In the macro level, predicting scale and speed of information diffusion has been investigated by [YC10, ZHVGS10]; these works shows that both user and message features are informative. Predicting the size of information cascades was analyzed by [KOU+12] by predicting from fixed points in time, considering cascade activity so far. The work of [CAD+14] takes a more realistic approach and can implement predictions from particular points in time, relative to the history seen so far. The authors predict the size of information cascades in Facebook and argue that the final state of the cascade is inherently unpredictable. For this reason, they predict the next stage of the cascade growth, which is a binary classification problem whether it will reach the median size or not. In the same lines, our work [TAFS15] predicts the lifetime of Twitter cascades, again modeled as a binary classification problem. Here the correlation of observation history observed versus different sizes of prediction goals is investigated. For short predictions goals the self-similarity of history and prediction goal is enough to predict the next step, while for longer prediction goals learning features are crucial in order to achieve higher accuracy. The work of [NGKA11] analyzes interestingness of content, which is a good predictor of message virality. A complementary approach is described in [SH10] predicting the long-term popularity of online content based on early measurements; while most of related work uses Twitter datasets, this analysis is implemented in Digg and YouTube content-share portals, using votes and views as a metric of popularity.

There is plenty related work that uses modeling in order to capture the underlying procedures of diffusion. The work of [GMC15] employ complex modeling and focuses on predicting the cascade burst. The authors capture the underlying arrival process of retweets and the user activity variation on the retweeting dynamics in the early stages in their model. In the same lines, the work of [WYH+15] formulates the burst time prediction task as a classification problem. Instead of predicting the exact occurring time of the burst, they predict in which time window the burst will appear. A recent approach with promising results focuses on using self-exiting processes, like Hawkes processes to model the diffusion process. The authors in [RXS+17] investigate into the popularity of YouTube videos according to the content type, network of diffusion, and external promotion. The continuing work of the same authors [RX17] presented two metrics that improve the predictions: popularity gain per unit of promotion, and the time it takes for such effects to unfold. By leveraging those metrics, promotion schedules can be implemented in terms of content virality, timing and economics.

All of the previous works describe analysis offline and the focus is on formulating the problem, modeling and feature engineering and rather than streaming incremental/analysis and concept drift. The work in [ZBPH14] presents a theoretically supported framework for active learning from drifting data streams and develops active learning strategies for streaming data that explicitly handle concept drift. In the same lines the authors in [DP13] propose the integration of their algorithm for concept drift and learning from imbalanced classes. The work of [CT16] trains extreme entropy machines binary classifier which belongs to the family of Randomized

Neural Networks in an online fashion with concept drift. This result is guaranteed to converge to the optimal solution produced by their offline counterparts.

2.2.5 Identifying Influentials

Identifying influential users has also attracted attention since it has particular interest for marketers, online journalists and others. In particular, the influence maximization problem [KKT03] providers answer in the following question: "Which users have to be targeted in a network for maximum spread of an idea, product or a piece of information in general?"

Domingos and Richardson [DR01, RD02] have been the first to study influence maximization as an algorithmic problem: instead of targeting individual users (customers) the social network of such users is leveraged for further spread. Their first work [DR01] is evaluated on movie recommendation datasets and in the second work [RD02] they extend their methods for knowledge-sharing sites. While their methods are probabilistic, [KKT03] have been the first to formulate the problem as discrete optimization problem. Under different cascade models (the independent cascade model, the weight cascade model, and the linear threshold model) and given k seeds (nodes), those k should be selected that maximize the spread over the network. This optimization problem is NP-hard and the authors in [KKT03] present an greedy approximation algorithm with approximation guarantees. However, this algorithm falls short in efficiency. Computing the influence spread given a set of k seeds is a very expensive computation.

Many works that followed focused on scaling the influence maximization problem. [LKG+07] present a method leverages the submodularity property of the influence maximization objective to reduce the number of times that the influence spread has to be computed. [CWY09] improves the algorithm and introduces new degree discount heuristics for influence spread which ameliorates massively the runtimes compared to [LKG+07]. In the same lines, the authors in [GLL11b] improve the work of [LKG+07] in terms of speed by estimating the influence spread using Monte Carlo simulation and heuristics. There is a significant number of works that followed in order to address the efficiency and scalability issues of the influence maximization problem [GLL11a, TXS14, TSX15, NTD16].

Complementary to the scalability considerations, many works proposed variations of the influence maximization problem. For example, [RD02] proposes to optimize the amount of marketing funds spent on each user (customer), rather than making a binary decision on whether to target them or not. The work of [GBL11] considers past history of interactions as an indication of how influence flows in the network. It also considers different levels of influence and temporal aspects of influence. Such information is crucial in estimating the expected spread. The work of [CCC+] considers the propagation of negative opinions and incorporates a quality factor that models the negative behaviour of users towards e.g. a defective product. The time

aspect is important when we consider marketing campaings, for example, what is the influence spread given a deadline? The authors in [CLZ12] address this problem by extending the Independent Cascade model to incorporate time delay of influence in social networks. Considering the time aspect from the perspective of social networks, social connections are dynamic which means that they change over time. The authors in [ZST+13] account for this, by periodically probing some nodes to update their connections. Then they compute influence spread considering the dynamics of social connections from probing which approaches better the current state of the social graph.

Apart from the influence maximization problem, some data-based works have been proposed in order to identify influential users. In [AW12] the authors implement in vivo randomized experiments to identify influence and susceptibility in Facebook. The results account for gender, period of using the platform, relationship status etc and showed significant different for the influence and susceptibility of different groups. Further analysis on influential and susceptible members revealed that influentials are less susceptible to influence than non-influentials and that they cluster in the network while susceptible users do not. This has particular value for the influence maximization problem for which users to target. A different work [BBHM13] focuses on identifying "hidden" beyond the obvious hubs and their impact on cascade properties.

Many works have been proposed that rank users according to their influence. The approach of [ZZWZ13] leverages the result of the influence maximization problem in order to rank users according to their expected reach. The work of [CLS+12b] proposes a new semi-local centrality measure which performs better than degree and betweenness centrality methods and comparable with closeness centrality but computationally less expensive. A variation of PageRank is proposed in [WLJH10] that ranks Twitter users according to their topical similarity and link structure. In a broader community of bloggers the authors in [ALTY08] try to understand what makes a blogger influential and present a model that quantifies various types of influence. While most of the methods to identify influentials use simulations to calculate dynamics and not real information flow in networks. The work of [PMAJ+14] takes the full history of different social networks and computes influentials over that. They show that the k-core metric performs the best and they also provide methods to compute influentials when the global network structure is not available.

2.2.6 Event and Trend Detection

There is also considerable research in the field of trend and event identification with the focus on implementing scalable systems that can cope with the large amounts of streams especially when emergent events occur. This area is particularly interesting for our research due to the real-time methods for social media data streams.

For example, EnBlogue [AMRW12] is a real time event identification system which

detects sudden changes in the popularity of unusually correlated keywords. The TwitterMonitor system [MK10] computes sets of bursty words and groups of words that belong to the same topic in Twitter. A different approach is described in [KRHW12], where the authors extract words from the tweet steam that indicate candidates for events and measure the temporal distance between those candidates. If this distance is smaller than a threshold, the combination of words are considered to be an incident.

Also the geographical distance is considered by [WOOO11]: documents referring to the same event in a small geographical area within a short time describe local events relevant for certain communities of users. The authors in [OPM+12] remove spurious trends that are wrongly detected or of limited interest. For that, they are use views on Wikipedia as a filtering mechanism in order to extract meaningful and useful trends. The work of [XZJ+16] uses topic modeling techniques and develops TopicSketch, that achieves real-time trend detection which can keep up with the Twitter rates. The self-exciting Hawkes process is also used for trend detection [PCA15]. The authors define derive trend indices for each topic, which account for the time between the detection and the broadcasts, the distance between the actual broadcast intensity and the maximum broadcast intensity, and the social network topology. By aggregating all the information about the broadcasts in one-dimensional process the complexity of the model is reduced along with the amount of data needed to make the detection.

2.2.7 Systems for Scalable Influence Computation

In general, no system exists that addresses all the requirements for scalable and real-time influence computation. Those requirements include the ability to process streams with the load of current social media and produce results in a continuous and incremental way. When computing influence the underlying social graph plays a very important role, since users are normally influenced by their connections. This means that we need computations over huge graphs that can handle the billions of connections that social networks maintain.

There exists 1) systems for solving similar problems but fulfill partially our requirements. For example, we observe systems that either address the requirement of fast stream computations or huge graphs, but not both at the same time. There are also 2) general scope systems that can form the foundation for more specialized systems.

Systems that solve particular problems in the domain of social media include trend detection and event identification (see Section 2.2.6). These methods can process social media streams that arrive in fast rates. However, the computational methods are based on topic extraction and not on user interactions, as a result they cannot be directly used to compute influence.

Systems to analyze and process large-scale graphs (which are used to express social networks) have recently seen some interest. These systems tackle a crucial prob-

lem that renders solutions like Map Reduce [DG08] unsuitable: graph algorithms are inherently difficult to process in a scalable and parallel manner due to a larger amount data dependencies. Furthermore most graph algorithms require iterative approaches. A popular approach used in these systems is the vertex-centric graph parallel model, in which a computation is run on each vertex and is iterated until the results converge. Google Pregel [MAB+10] uses this model with a bulk-synchronous processing approach and message passing, while GraphLab [LBG+12] also allows asynchronous execution, permitting a larger amount of parallelization and the opportunity to prioritize more complex vertices over others but this entails complex consistency models. The relatively high-level abstraction allows an algorithm developer to express many common graph algorithms in a concise manner, while leaving enough room for optimization at the systems level. PowerGraph [GLG+12] is an extension of this approach to handle natural graphs with power law distributions in an efficient way, providing different partitioning strategies based on vertex splitting.

However, these system do not provide support for temporal and evolving properties and as an extension typical stream computations. Only recently some systems have come up which support computations on evolving graphs, providing only limited support: GraphChi [KBG12] is a single-node, disk-based derivative of GraphLab that can handle updates in the graph with moderate cost. GraphInc [CLS12a] is built on top of the Pregel model (not system) and handles incremental computations for use cases like shortest paths, PageRank and connected components. Kineograph [CHK+12] is a distributed system that takes a stream of incoming data to construct continuously changing graphs. The advantage of Kineograph is that it captures the relationships that exist in the data feed, and can provide insights of the fast evolving underlying graph. In our case, we not aim for computations on fast evolving graphs, but rather in the combination of streams and (slow evolving) social graph.

There are also more general streaming systems that we can base our analysis and customize them to our problem. Naiad [MMI+13] is system that allows for iterative computation over data streams using timed dataflows. As such, Naiad provides strong coordination means, but it does not provide fine-grained control over the algorithm and state distribution. Storm [sto] is a scalable, distributed data stream processing platform which provides the necessary low-level primitives for distributed stream processing. We mainly use Storm and we compare with Naid.

2.3 Related Work: Provenance

Here we present an overview of related work in provenance and then we focus on provenance on social media.

Traditionally, provenance has been used for databases, workflows and streaming systems. We provide related work to show how these researched fields have been

benefited by provenance analysis. Yet, the main focus is in provenance on social media, where we make our main contributions. We can discern the following categories for state-of-the-art, which include provenance in: 1) *databases* in Section 2.3.1, 2) *workflows* in Section 2.3.2, 3) *web* in 2.3.3, including provenance for trust assessment and 4) *social media* in Section 2.3.4 which has received only limited attention borrowing methods from typical information diffusion.

2.3.1 Provenance in Databases

Provenance in databases has attracted attention in the past, with the aim of providing information about how the data came in its current state, either in the database itself or as a result of a query. For this reasons, methods [CCT+09], systems [Wid05] and formal semantics [GKT07] have been proposed to shed light for provenance of data on databases. The work of [CCT+09] summarizes the methods to express provenance in a database. This includes the "where", "why" and "how" provenance that shows where the data came from in the input, why a particular output was created and how an output record was produced. Trio [Wid05] is a system that serves the purpose of a database, managing also the accuracy and lineage of data in a single model. Such model allows for uncertainty, that is approximate, uncertain, or incomplete data values, including query results. The lineage information provided includes updates, program-based derivations, bulk data loads, and data import from other sources. The work is SQL based and supports extensions for querying uncertain data and lineage. Formal representations have been also proposed for expressing provenance, for example based on semirings of polynomials in [GKT07]. This work also extents those representations to datalog in order to cover incomplete and probabilistic data.

A particular area of interest is storage of the generated provenance. The authors in [MRHBS06] design a storage system that automatically collects and maintains the provanance of data. Sensor data management benefits from provanance storage systems, since these data are massive and need efficient storage and indexing. Those issues have been tacckled in the past: for example the work of [LNH05] solves the naming and indexing considerations and described a methodology for creating distributed, indexed repositories of sensor data. Subsequent works, e.g. [CJR08] try to reduce the amount of storage required by leveraging particular provenance properties.

2.3.2 Provenance in Workflows

Provenance is crucial in any type of scientific work because it supports the reproducibility of methods and understanding of the processes involved and results. Provenance is expressed thought documenting the computational tasks and devel-

oping workflows. While the typical provenance generated for databases focuses on the tuple level, while workflow related provenance treats the steps as black boxes.

The work in [FKSS08] differentiates two types of provenance for computational tasks and describes the components of provenance management. Prospective provenance describes the steps to be taken in order to produce the desired data results. Retrospective provenance captures the actual steps that are executed in practice, combined with information about the environment used to derive a specific result. In practice, this is the produced data log of executing particular steps. Also, this work breaks down the provenance components: Capture mechanisms, representational models, and infrastructure for storage, access, and queries. Capture mechanisms provide information concerning the computational steps, the execution information, the system's state, etc. Representation models describe the provenance generated and provide dependencies among data and processes: most models in the literature support retrospective provenance. The authors review methods and systems that fall into these categories. On the more practical level the work of [MGBM07] proposes methodologies for workflows considering experiments in biology, chemistry, physics and computer science. The authors analyse those particular use cases and determine the requirements for a application independent architecture.

2.3.3 Provenance on the Web

While provenance for databases and workflows deal with closed systems in most of the cases, in the web context we observe an open world where data across multiple systems and different sources are combined. In general anybody can contribute and make links among data, as a result provenance research deals with new challenges. For example, the development of formal provenance models facilitate the intereroperability by allowing different systems to interact and exchange data. Complementary, since data are coming from different sources, users need to make trust assessments about particular data. In practice, not all users might be interested for the complete provenance on data, but rather reply on trust and quality assessments.

The work of [Har09] proposes a provenance model in the context of web data: it captures information about the access of such data as well as their creation. Specific RDF vocabularies are being proposed in order to describe formally such provenance information.

Such provenance data need to be assessed for their quality and trustworthiness. The work of [HZ09] proposes a model for assessing the quality of data by a quantitative approach. The dimensions considered to calculate data quality include accuracy, completeness, believability, and timeliness. The authors provide a concrete example of how to compute timeliness on data and which aspects need to be considered. Their approach in general generates annotated provenance graphs on top of which quality scores are being calculated.

In general, algorithms to compute trust have been proposed in other domains, which can be applied in different provenance use cases. For example, general well known trust models include the TidalTrust and FilmTrust. The authors of [Gol06] present an algorithm (TidalTrust) to infer trust in social networks, based on provenance and annotations and using semantic web technologies. The algorithm claims that shorter paths and higher trust values (annotations) lead to better accuracy. This algorithm is tested in the context of movie recommendations on social media (FilmTrust), which uses trust to compute personalized recommended movie ratings.

Last but not least, the work of [Mor10] summarizes provenance state-of-the-art and highlights the challenges for provenance on the web. Particularly, the author discusses how to generate and publish provenance leveraging semantic web technologies. In addition, accountability and trust are discussed as indispensable parts of provenance on the web.

Since trust is a major concern in provenance, assessing the provenance of information in an online fashion is crucial. The work of [WWP+17] discovers the provenance of information diffusion in an online fashion. The algorithm identifies the provenance paths given some random infected nodes in the network. The problem is modeled as a regression problem while information is gathered on real time. However, there is very limited work that tackles the problem of timely identifying provenance. Online world contains a massive amount of information, and the majority is spam, including false information. Being able to identify the provenance and trustworthiness of information in an online fashion offers great advantages for online users and professionals. In this thesis, this is a problem we tackle.

2.3.4 Provenance in Social Media

Next we proceed describing work about provenance on social media and platforms in more detail. Complementary we present works on online news media clustering, since we use similar techniques in Chapter 5 and meme tracking. In general provenance has been understudied compared to typical information diffusion [GL12]. There is a need for more intensive provenance investigation since the provenance emitted by social media API's is incomplete [GGCM12, BFGL13].

Provenance in social media has received very limited attention, yet the few works that exist borrow standard models and concepts from information diffusion. However, provenance offers rich methods in different contexts which can be adapted and applied to social media. This is what our works aims for: combining methods form both communities and benefiting both of them with our analysis and advances of the state-of-the-art. We can discern the following dimensions of provenance computation: feature, network and content based.

For featured based provenance, normally information about the author is considered [BFGL13, GRFL13]. For example in [BFGL13] information about the location, age, education, gender, occupation, ethnicity provide valuable insights about the author.

These attributes are very valuable in identifying the provenance of some information. Such attributes can be collected from Twitter and combined with information from other sources (e.g. Facebook. LinkedIn, Google results etc). Particular functions have been defined in literature[Bar11] that quantify how much available data exist for a statement of interest (e.g. author). In the same lines, the work of [GRFL13] assigns provenance through profile information collected from different social media. While profile information reveals the popularity, and possibly trustworthiness of the contributor of particular messages, it lacks information about the sources and intermediate steps that information took.

For network based provenance, probabilistic models from information diffusion are being leveraged like SIR, SIS, IC, LT [Het00, New03, GLM01, Gra78, BFGL13]. However, these models are optimized to find influentials in the network or to identify the underlying processes. Thus, such models capture very well the information propagation from the sources to the terminals, but do not work very well for the reverse problem of seeking the paths from known terminals to the sources. However, many works have used these models under different assumptions to identify the sources of information: for example [SZ11] assumes the SIR model, considers one source, while the information recipients are known. The data used are coming from the internet and grid networks. The assumption of known recipients is also used by [LTGM10] where the authors compare the output of the IC model with the observed data in order to infer the sources. In this case, the sources might be more than one, which is a more realistic assumption. While [LTGM10] assume that all recipients are known, in reality, this is not the case. This method is used in the DBLP co-authorship network.The authors in [GFL13] assume that very few recipients are known and with this constraint they recover the provenance paths. With the hypothesis of degree and closeness propensity the intermediate steps of provenance paths are being reconstructed.

Apart from being based solely on probabilistic models, social media offer rich past information about the history of information propagation over a social graph. The authors in [BFGL13] highlight the importance of history of interactions in recovering provenance. Once information has propagated over certain paths, it is highly likely that it will propagate the same way in the future [HRW08]. In order to implement this, the (Twitter) followers of each user are being ranked according to (social) proximity, content and profile features. Then these hypotheses are tested towards real data: users who wrote similar text or propagated the same URLs (content proximity) is the more likely hypothesis [BFGL13].

Next we provide some related work for provenance based on content similarity, which we also used to identify provenance. In such works, the content similarity is leveraged in order to reveal some linkage. Following the work of [DNCVD$^+$12], where clustering was used to identify similar news articles, we developed a method [DNTD$^+$15] that computes the provenance of explicit (retweets) and implicit (based on similarity) interactions under the same model. This way, we enrich the reconstructed provenance which combines both means. However, pure content similarity

methods might be connected with high uncertainty with regard to the reconstructed provenance and influence. External news and trends obscure those methods by overestimating influence. Such type of provenance can be combined with other features or network information to increase its certainty. For example our work [TFDN+16] identifies user interactions and influence indicators on top of the clustering based provenance. Those indicators strengthen the already reconstructed provenance. Lastly, our last work [TLF+17] builds an incremental and efficient system that can sustain the the loads of current online social media messages and news articles and builds provenance based on similarity and additional indicators. We will provide further discussions on Chapter 5.

Similar to our work is the research by [BCJC15] which also computes influence in information diffusion that official mechanisms cannot capture in Twitter. In more detail, the authors reconstruct diffusion paths with message similarity under the following hypothesis: users are being influenced by the last 100 messages from their friends' timelines. In order to compute provenance for a user's messages, all the last 100 messages from their friends should be collected and checked for possible similarity. On the one hand, this is a strict assumption and it is not always true according to our observations. On the other hand, a priori knowledge of friends' past messages is needed in a real-time set up, which stresses the limitations of Twitter's crawling API. Scale also becomes an issue, given the large amount of connections that users maintain (median value: 209, from a 2009 dataset by [BHMW11]).

In general there has been very limited work on provenance on social media and the most complete work is [BFGL13]: The authors cover many types of provenance, as described already, including categorizations, computational methods, quantification and availability. The main methods are taken from information diffusion field, however the focus is different. Instead of focusing only of the processes the authors make steps beyond that. Their work provides methods to quantify how complete is the data and what are the steps taken to complement the requested provenance.

Table 2.5 shows characteristics for these works: we observe a combination of modeling and explanatory works which stems from classical information diffusion methodology. Those works do not address the dynamics and the evolution of these processes, contrary to what one might intuitively expect from provenance analysis (given the scrutiny with which typical provenance analysis treats the evolution of a piece of information). Another unexpected observation is the extend to which missing data are addressed: provenance gives particular attention to the correctness and completeness of any process under investigation. As a result, missing data would be the first issue that one would expect is addressed in provenance analysis.

2.3.5 Meme Tracing

Meme tracing is a typical information diffusion, however we classify it under provenance works, because the questions that are investigated are coming from the prove-

Table 2.5: Provenance on Social Media

		[BFGL13]	[SZ11]	[LTGM10]	[GFL13]	[GRFL13]	[BCJC15]	**[DNTD+15]**	**[TFDN+16]**	**[TLF+17]**
Goals - Types of Analysis	Predictive									
	Exploratory	×				×	×	×	×	×
	Modeling	×	×	×	×					
Datasets	Social media/blogs/etc	×			×	×	×	×	×	×
	News									×
	Other		×	×						
	Synthetic		×							
Influence Computation	Features	×				×			×	×
	Network (social or diffusion)	×	×	×	×		×	×		×
	Content	×					×	×	×	×
	History	×					×			
	Time									
	Social						×			
	Other								×	×
Social Graph based	Static	×	×	×	×		×	×	×	×
	Dynamic									
	Inference									
Non-Social Graph based	Inference									
	No inference	×				×		×	×	×
Missing data	Social graph									
	Messages	×			×					
	Properties-Estimation									
	Quantification									
Diffusion Process	Static	×	×	×	×	×	×	×	×	
	Dynamic									×
Across multiple media		×				×				
Computations	Real-time									×
	Incremental									×
	Scalability/ Efficiency		×	×	×					×
Granularity of Results	Fine-grained:edge/node	×	×	×	×	×	×	×	×	×
	Coarse-grained:process	×	×	×				×		×

nance side. For example, where do memes come from and what are the modifications that have undergone on the way? Methods to unravel such process include clustering and similarity, also widely used for typical provenance research. These works focus on a very specific type of diffusion which can be traced thought online media. For example, the work by [LBK09] tracks memes that stay unchanged over time, clusters their variations and studies their temporal and structural properties. It was followed up by NIFTY [SHE+13], which can cluster and track information flow on a larger scale. Finally, [SAA11] also rely on clustering methods to identify and group memes, but focus more on how these memes – and especially quotes – change as they propagate. At the same time, the properties and temporal variants of the sources where they those memes were first observed, were investigated.

2.3.6 Clustering as a means of Provenance Reconstruction for News Articles

Next we observe a line of works for news clustering as a means of provenance identification, and such methods can be leveraged for provenance reconstruction in social media. While there has been plenty of work for document clustering and its evolution, most of these approaches target an offline scenario and a close-world dataset such as a news archive. For example, the approach by [AS12] clusters news articles with k-means by computing their similarity and identifying groups of news clusters. The clustering is extended to work in an incremental environment where new documents are integrated into existing clusters, and frozen clusters represent inactive clusters. However, this work makes many strict assumptions in the initial number of clusters, supports no cluster re-organization, considers an ad-hoc lower bound for similarity and excludes documents that belong to more than one topical category. The largest dataset tested was 26M documents for the period of one year. Another work that uses k-means to cluster news is that of [KIK08], which also considers incremental updates and deletions of documents. It entails a refined method for documents expiry based on forgetting factors and their datasets consists of 64M documents. The work of [DNCVD+12] reconstructed provenance of a closed news archive news using clustering based on semantic similarity. This method we use as a foundation later in Chapter 5.

More recently, [ADZ+14] presented a multi-funneling approach to provenance reconstruction for offline data. More precisely, they apply three techniques: one based on IR techniques similar to [DNCVD+12], one based on the machine learning and topic modeling, and one based on matching the longest common subsequence. They used Wikinews and Github datasets.

We combine meme tracking and news clustering in Table 2.6 since they share similar methodologies. The methods to compute influence (provenance) is based on similarity (content), that's why those methods are not based on social graphs. We observe that the meme tracking follow a graph data model, while the news clustering

results for [DNCVD+12] are RDF based (PROV-DM in particular, which a provenance model). They are applied across many media (social media and news articles) in most cases. Those works also tackle scalable, incremental and even real-time analysis.

Table 2.6: Memes Tracking/ News Clustering

		Meme Tracking			**News Clustering**			
		[LBK09]	[SAA11]	[SHE+13]	[AS12]	[KIK08]	[DNCVD+12]	[ADZ+14]
Goals - Types of Analysis	Predictive							
	Exploratory	×	×	×	×	×	×	×
	Modeling							
Datasets	Social media/blogs/etc	×	×	×				
	News	×	×	×	×	×	×	×
	Other							×
	Synthetic							
Data Model	Graph	×		×			×	×
	RDF-based				×			
Influence Computation	Features							
	Network (social or diffusion)							
	Content	×	×	×	×	×	×	×
	History							
	Time	×						
	Statistical model							
Across multiple media		×	×	×	×	×		
Diffusion Process	Static		×		×			
	Dynamic	×	×	×		×		
Computations	Real-time			×				
	Incremental				×	×		
	Scalability/ Efficiency			×	×	×		
Granularity	Fine-grained: edge/node	×	×	×				
	Coarse-grained: process	×	×	×	×	×	×	×

2.4 Discussion

Information diffusion has been applied mainly on online *social media data* (Datasets in Table 2.2), while provenance is a much older concept which covers many fields from databases [CCT+09], workflows [DBE+07] streaming systems [GSEFT13] and lately the semantic web [Mor10].

Each community has its own and *types of analysis and methods*. Information diffusion focuses on the analysis for a deeper understanding of such processes and their characteristics: for example how viral is particular information, which users are being influenced and by whom, which influential users have to be targeted for maximum spread (influence maximization problem [KKT03]). This has led to the development of models that approximate information diffusion processes and provide answers to those questions. The deeper analysis of information diffusion properties attracts considerable attention: for example the structural [ZBK+10, LMF+07, KLPM10], spatio-tempemporal [KCLC13, BSW12] and social aspects [WRP+13, LT13] have been analyzed given particular datasets.

Provenance seeks to understand where particular information comes from and how it has been modified on the way. For example, the work in [BKT01] researches the provenance of database queries in terms of data sources and location in the database. Such analysis emphasizes the correctness and integrity of the data and those conditions are crucial for further analysis and processing. While provenance has been an old research field, there are very limited works on social media: in particular those works borrow their methodology from typical information diffusion and social network analysis [BFGL13]. The main goal is to understand how information reached its current state and collect evidence for trust and relevance assessment. Our goal is the first part of tracing the paths and the sources, and any further analysis for trustworthiness we leave for future work.

In terms of *modeling* (see Modeling in Table 2.1), information diffusion methodology is implemented on the coarse-grained level. It focuses on the overall process of spread while messages and users in isolation attract limited attention: that said, statistical modeling of such processes is very popular for information diffusion [GLM01, Gra78, New03, Het00]. On the contrary, provenance researches the individual aspects of such properties, since an assessment about the relevance and trust of information is crucial. Provenance aims at the fine-grained level, turning the attention to (the correctness of) individual influence edges and forwarders [GRFL13].

Missing data (See missing data in Tables 2.2 and 2.5) is investigated from different perspectives for both communities. In general, information diffusion analysis considers assumes *completeness* in the datasets used. This leads to a closed world assumption that does not account for possibly missing data or external influences. Some works that proposed Inferenece of missing data are very costly, e.g. [ZWSY12]. Other works, estimation only particular properties of information diffusion given a fraction of the complete dataset [SMLGM11].

Since provenance analysis focuses on individual edges and trustworthiness assessment, quality of data is crucial to make relevant statements [Mor10, HZ10, GGCM12] As a result, provenance research focuses on quality of data, completeness and accuracy, rendering missing data a major concern. In order to capture provenance more accurately, considering external factors is of high importance [T+07].

Lastly, another important aspect is the *performance* of such systems and computations: as discussed in Chapter 1 identifying provenance in a prompt manner is crucial for many use cases. On the one side, we observe huge amounts of data and graphs that need to analysed (need for*scalability*), and on the other side such analysis should be performed in a real-time fashion (need for incremental and real-time methods). Information diffusion community focuses on the results and analytics rather than scalable methods. The only exception is the event detection field [AMRW12, MK10], where the focus is on developing scalable and incremental methods in order to identify emergent events in an online fashion. However, event detection community has its own methods for on text analysis (topic/event detection and evolution) and their methodology does not apply to typical information diffusion.

For provenance on the other side, scalability and real time results is of a major concern, and there exists a plethora of works for provenance streams [GSEFT13, SMW13, GEFT11, MBK+08]. This derives from the fact that provenance is traditionally computed over systems, e.g. in databases [CCT+09], or systems collecting sensor data [LNH05], etc. In this case, provenance is one aspect of the system to consider and adapt from incremental/real-time results.

To sum up, we observe that the fields of information diffusion and provenance are very diverse and target different dimensions. We leverage aspects of both communities in order to target our research questions from Chapter 1. Particularly, we are based on theories from sociology and social network analysis (used also for information diffusion) in order to explain influence and build our models. For actual computations, we use combined methods from information diffusion and provenance to compute influence at scale and speed. For interoperable results, we are based on provenance models from semantic web.

Chapter 3

Own Work: Assumptions and Design Decisions

In this chapter, we summarize scope and put into perspective our work, also in comparison with related work. We discuss the modeling, the assumptions and the design decisions in the form of research questions and hypothesis. This thesis combines concepts and methods from the fields of information diffusion and provenance and in turn our contributions benefit both communities.

3.1 Methods for Influence Computation

In the next paragraphs, we provide the scoping of our work compared to related work and our rationale for our design decisions. We also provide a summary of the next Sections which are discussing our methodology.

Our main goal is to identify and provide the means to compute information diffusion in social media. We are interested in recovering influence with a broad scope, taking into account *explicit* and *implicit* interactions on social media. By *explicit* we refer to the interactions that happen within the context of social media, for example using operators like retweets on Twitter or reshares on Facebook. *Implicit* interactions include those that are not captured by social media, for example users providing credit to the original contributor by their own conventions. Implicit interactions carry higher uncertainty and are harder to identify, especially in cases that no other indication (e.g. the initial contributor) is provided.

Compared to related work (see Table 2.1 and 2.2), our work is focused on *fine-grained* information diffusion. We do not pursue modeling the process of information diffusion on a coarse grained manner, which is already a well researched topic; the emphasis is on the reconstruction of *individual influence edges*. In addition, we avoid any complex and computationally inefficient modeling because we target

prompt analysis in real-time. In particular, we do not assume any underlying statistical model or any diffusion probabilities and we do not learn from history of past diffusion. We opt for explicit methods that are generalizable and can be applied to other platforms. We also avoid strict assumptions or methods that are tied to particular social media properties.

In order to compute information diffusion, we form plausible hypotheses. Some hypotheses are competing (alternative hypotheses) and they cannot be tested at the same time. Alternative hypotheses can be sorted, ranked, weighted according to their plausibility and use case. Other hypotheses are complementary, which means that they be applied on top of any existing hypotheses. They aim at providing more fine-grained provenance information, while strengthening the already computed influence.

We identify particular cases of explicit and implicit interactions considering the information provided by social media, which also reflects the information that has to be inferred (for explicit interactions). For implicit interactions, we provide more general hypotheses (in lack of any social media provider information) and we make a deep investigation in order to understand how latent influence is being expressed.

Such influence inference is based on generalizable hypotheses that can be applied to any social media that have an underlying social graph (many of our hypothesis do not make the strict assumption of the existence of a social graph). The main problem in testing the generalizability of our methods in different platforms is *availability*. Most platforms do not provide their data due to user privacy constrains. We are based on Twitter since it provides a public API for data acquisition and the majority of the user profiles are public. Twitter is also a representative social media which provides a good coverage of online population and reflects large events and trends. To extend our datasets, we also crawled Vine[1], a platform similar to Twitter, but with short videos instead of messages. Crawling of Vine was cumbersome and non-transparent, and finally it was shut down in 2016.

As next, we provide the background, categorization and hypotheses for our modeling and methods.

3.1.1 Explicit Interactions

For explicit interactions, we observe two main cases:

- *Direct linkage based*, like replies and quotes in Twitter where the previous step[2] is provided.
- *Source based*, like retweets in Twitter where the source[3] is provided

[1] https://vine.co
[2] direct predecessor
[3] origin of diffusion

Figure 3.1 shows these types of interactions. *Single-step edge* means that a message is derived though one step from another, while *any-step edge* means that a message is derived from another through one or more steps.

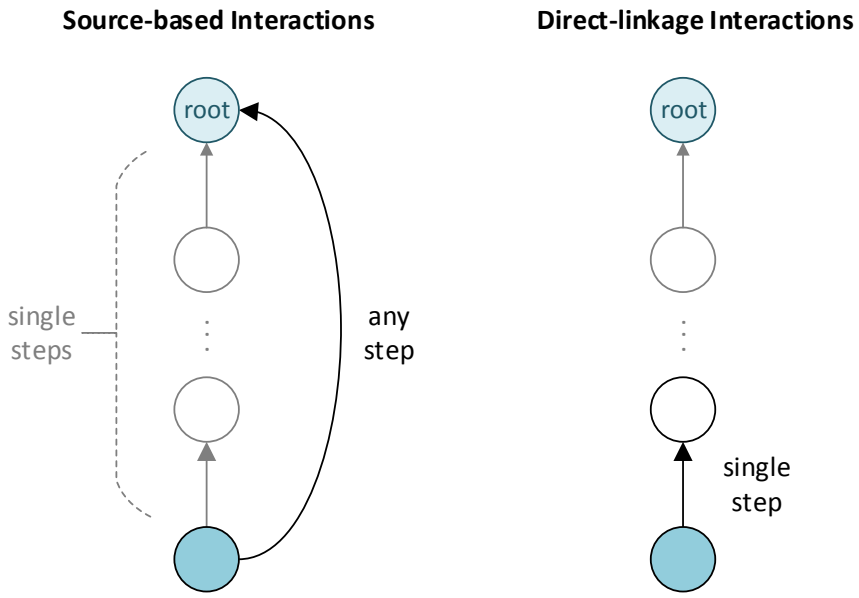


Figure 3.1: Explicit Interactions where nodes depict messages. The information that is given by social media providers is highlighted in black colour. The information that needs to be inferred for fully tracing the diffusion paths is depicted in grey. For the source based interactions on the left, the reference/edge to the root is called *any-step* because the root is being reached through one or more predecessors. The *single-step* edges that lead to the root need to be inferred. For the direct linkage based on the right, a *single step* edge links to the previous message, which is given. The root has to be inferred.

Orthogonal aspects to this categorization are:

- the number of *possible roots*: single vs multiple roots
- the number of *influence edges* that are generated: single vs multiple edges

For the *direct linkage interactions*, since we know the previous step, we have very low uncertainty for the influence edges. However, the source is not provided which has to be found by following back the influence paths.

For the source-based interactions, where the source is provided, we need to infer the influence paths that lead to the source. We also know that those messages belong together, because source is being referenced. However, the previous steps are not being provided and the following research question is emerging:

(Q1) *How can we infer influence edges and paths, when only individual activations are given?*

In order to make the inference, we make the hypothesis that

(H1) *Influence flows through social connections.*

We rely on this hypothesis due to the nature of social media: users are exposed to the content shared by their social connections. As a result, they are most likely to be influenced by them. The same findings also are supported in the literature [GBL10, CMG09]. This hypothesis applies to source-based interactions, and in general to cases where the previous influencer is not provided, like in the case of implicit interactions. We evaluate this hypothesis on Section 4.3.3. These influence edges have medium to low uncertainty, since users might also get influenced from the public timeline, without any obvious connection [MZL12]. Note here, that more than one activated social connections lead to multiple diffusion paths: a large number of alternative diffusion paths, raises respectively the uncertainty of the reconstructed influence edges (and as a result the influence paths).

In general, models of information diffusion support the observation of multiple influencers. For example, the *Linear Treshold* (LT) model [Gra78] maintains a threshold that needs to be exceeded so that users are influenced and as a result "activated". This means that some users have higher influence bounds than others and need higher influence from their neighborhood to get activated. These multiple influence edges can be ordered according to some criteria, weighed or partially eliminated. Related work for influence models provides meaning for multiple influence edges [CHBG10, BHMW11]. Influence models provide *ordering* among the *users* or among the *messages*. *User based influence metrics* include *popularity* (number of followers, number of mentions), *influence* (number of times being retweeted) or *activity* (number of tweets contributed). *Message based metrics* include *popularity* (number of retweets) or *temporal distance* (first or last exposure). Some of these models are evaluated in Section 4.3. We can leverage these models for better understanding the impact of multiple edges, by splitting and weighting the influence according to each model similarly to [BHMW11]. In cases where simplicity is favored, we can keep the most probable edge and eliminate the remaining. We discuss the influence models in Section 3.1.1, and in Section 4.3.2.

In terms of *number of roots* and *number of generated edges*, in cases where the source is provided, we observe a single source and possibly multiple influence edges.

For direct linkage, the root is not provided, but since for every message one influencer is embedded, this will result in a single root. When the direct linkage is provided, we might end up having multiple roots, since there is no explicit single root. The number of influence edges is also one, because it is embedded in every message. Note here that in some cases a message might carry two direct linkage operations (e.g. reply and quote in Twitter at the same time) which results in two embedded edges. In such a case, we encounter two roots. We summarize those findings in Table 3.1, where we always consider the more general case.

Table 3.1: Overview and Properties of our Methods

	Source based	**Direct linkage based**	**No linkage**	**No linkage, with indicators**
General Assumption	credit to previous influencer	credit to previous influencer	Influence without any given credit	User Conventions, Indicators
Source	1, provided	≥1, not provided	≥1, not provided	≥1, not provided
Previous step	≥1, not provided	≥1, provided	≥1, not provided	≥1, not provided
Computational Assumption	Social connections	-	Content Similarity	Mentioning, Social Connection, Interaction with explicit means
Edges generated	≥1 (# of connections activated)	≥1	2 (oldest/similar in each cluster)	≥1
Uncertainty	Medium-low	Low	High	Medium-low
Grouping	Retweet cascade (1)	Reply/Quote cascade (1 or 2)	Topic Clusters (≥1)	Topic Clusters (≥1)

3.1.2 Implicit Interactions

For implicit interactions, we assume that users are influenced by an unidentified source (other users or external sources) but do not express it with the mechanisms of social media. As a result, there might exist some "hidden" provenance with regard to their messages, which we are trying to reveal. Meme tracking [LBK09, SAA11] belongs to this category, similar to reconstruction of hashtags in Twitter [4]. In lack of any explicit influence we rely on the presence of particular hashtags in Twitter for example to infer any influence.

The question here is:

(Q2) *How to identify provenance if no information from social media is provided?*

We make several complementary hypotheses for this question. Our basic hypothesis is that:

(H2-I) *If two messages are highly similar, there is a high probability that they share some provenance.*

which leads us to the additional hypothesis:

[4] A hashtag refers to a metadata tag which is used on online social media, and allows the easy search for other messages referring to the same content or meme. For example the #metoo hashtag was used by popular women who have been harassed at their work environment after the Weinstein scandal in Hollywood in 2017 [Gua17].

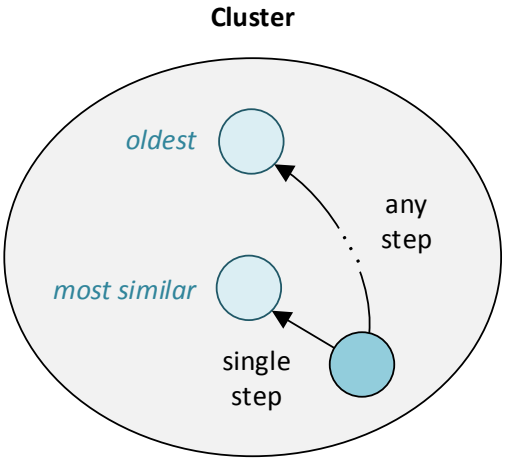


Figure 3.2: Implicit Interactions in one cluster. Since a cluster refers to a topic and the messages within the cluster are more similar with the messages accross other clusters, we assume that all the messages are derived through one or more steps from the oldest messages in the cluster (*any-step*). Every messages is assumed to be derived thought *one-step* from its most similar oldest one.

(H2-II) *The higher the similarity between two messages, the higher the probability that they share some provenance.*

Hypotheses H2-I and H2-II are relevant for cases where no provenance information is provided and we rely solely on the content of the messages. We identify those connections by a clustering algorithm that clusters content-wise similar messages (Section 5.2). The algorithm produces possibly overlapping clusters sharing similar context. As a result, a lightweight topic detection is a by-product of the clustering but out of scope for this work. Within those clusters we try to identify latent influence and provenance. By using hypothesis H2-II we assume that highly similar messages must share some provenance and we connect every message with its most similar that was written in the past (*single-step provenance*). We also assume that all the messages within each cluster are derived through one or more steps from the oldest message in each cluster (*any-step provenance*). This way, we reduce the number of possible roots (single root: oldest message in the cluster) and the number of possible influence edges (singe edge: most similar previous message). However, if two clusters are overlapping the number of roots and influence edges is doubled for the messages in the overlap (one source and one edge for each cluster). This hypothesis still carries high uncertainty, since content similarity does not always imply provenance. In order to further decrease the uncertainty, we will define additional hypotheses. Figure 3.2 depicts the provenance generated in implicit interactions.

The next research question is:

(Q3) *How to decrease the uncertainty of the reconstructed provenance which is*

based on content similarity?

To address this question we make the hypothesis that:

(H3) *Online users use their own conventions to express influence and provenance.*

Hypothesis H3 can be applied to any message in order to unravel latent influence or strengthen the already computed influence. With respect to H2 and H3 we categorize implicit interactions accordingly:

- *Content similarity* based: similar messages that share some provenance.
- *Additional indicators* based: applied on top of the similarity (or as standalone) which lowers the uncertainty of the reconstructed provenance.

Note here that in both cases, we rely on inference, since no provenance information is provided from social media. For the orthogonal categorization on the *number of roots* and *the number of influence edges*, in theory we have multiple roots and influence edges, since no information is provided and we rely on inference. However with our methods presented in Section 5.2 we compute the grouping of messages that belong together under a single root and we compute the most plausible previous linkage edge.

For the *content based similarity* method, the uncertainty is high because our methods are based solely on the similarity among messages. In cases of external (out of social media) influence (e.g. major events) many influence edges are created, which do not necessarily imply influence from other users. In this work, we quantify and model external influence but we plan to properly compute it in future work.

In the *additional indicators* based method, we leverage particular indicators to decrease the high uncertainty of the previously reconstructed provenance. For example, user mentions or the existence of social connections strengthen our hypotheses of the inferred influence. We are going to discuss such indicators in Section 5.2. The generated provenance edges bear medium to low uncertainty according to the strength of each indicator.

In Figure 3.3 we provide the methods ranked according to the uncertainty of results. Additional indicators overlap with the rest, because they borrow methods and assumptions from the previous three categories, which leads in varying uncertainty according to the method used.

In the previously introduced Table 3.1 we summarize the properties of such indicators for comparison purposes. We observe the following:

- We encounter two (or more) roots in the following cases: 1) Direct linkage based: combined reply and quote cascade where we observe one root for each means, 2) No linkage (where we leverage H2 for similarity): overlapping topic clusters where messages that belong to the overlap end up in two roots for each cluster.

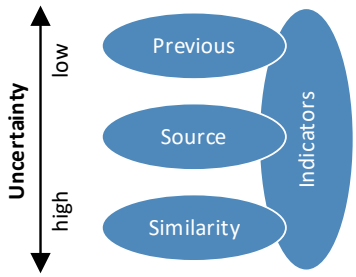


Figure 3.3: Uncertainty of results for different methods

- We observe multiple influence edges in the following cases: 1) Source based: the hypothesis of the social graph might result in multiple edges in case more than one of the user's connections are activated. 2) Direct linkage based: especially in Twitter we encounter cases that a message is a quote and a reply at the same time, which results in two edges. 3) No linkage: in case of an overlap we encounter again two edges for each of the cluster (that applies both for the most similar and oldest message, see Section 5.2). 4) No linkage, with indicators: we encounter many differently annotated edges according to the indicator.

3.1.3 Scoping of our Work using Existing Classifications

In order to put our work into perspective of both the information diffusion and provenance communities, we borrow the existing classifications from [GHFZ13] for information diffusion and [GD07] for provenance.

The classification of [GHFZ13] considers the following categories of information diffusion: detecting popular topics, modeling information diffusion, which includes explanatory models and predictive models, and lastly identifying influential information cascades. Our methods belong to the category of explanatory models: the aim of explanatory models is to infer the underlying cascade, given a complete activation sequence. The classification from [GHFZ13] is reproduced with our methods in Table 3.2. The first dimension is the *evolution of the underlying network*. Our methods considers static underlying social networks. The decision for that is mainly practical due to API limitations.

The second dimension is the *inferred properties*, in particular pairwise transmission probability and pairwise transmission rate. Rate refers to the amount of information that is transferred over time. In a variety of works after observing some data, correlation can be inferred of users (nodes) which interact a lot. Since our methods are not statistical but explicit, they do not infer any probabilities. For future work, we will consider other metrics for influence computation, for example history of past interactions or popularity and learn the probabilities that show the most possible influence paths.

Table 3.2: Information Diffusion: Explanatory Models

Model	Network		Inferred properties			Supports missing data
	static	dynamic	pairwise transmission probability	pairwise transmission rate	cascade properties	
Source based	x				x	
Direct linkage based	x				x	
No linkage	x				x	
No linkage, with indicators	x				x	

The third dimension is *missing data*; as discussed in related work in Section 2 missing data has not seen much attention in related work, which is also reflected in the Survey [GHFZ13]. Our methods do not consider inference with missing data; only the impact of missing data is quantified and analyzed in Section 4.3.

Table 3.3 presents the classification for provenance from [GD07]. Note here that such classification for provenance is very general and considers a system which records, queries and stores provenance. Provenance classifications is divided in three categories: *provenance model*, *query and manipulation functionalities*, *storage* and *recording*. For more details we refer the reader to the paper [GD07]. We are mainly focusing on the modeling and computational part, while typical provenance systems invest their efforts also in storage and query of provenance. Manipulation and querying facilities are out of the scope of our work. Storage is also of limited interest to our work, since we do not crawl continuously and social media messages

and information in general, are rather short (i.e. occupy limited space on disk). We store our provenance on disks for any further analysis and evaluation.

We shortly discuss Table 3.3. According to [GD07], we create both source and transformation provenance. For source provenance, the original author, the original message (both content and identifier) and the timestamp are obtained. Twitter attaches a lot of additional information to the messages (like user profile details) that we store on our disks (more details in Twitter API documentation[5]). Transformation provenance includes the provenance related with the transformation of an item, in a wider sense. We consider the author who has done the transformation, the content similarity of the original and inferred message and the timestamp of the transformation (i.e. message emission). In any case, the inferred influence edges are stored. Our model is an open world model according to [GD07] in a sense that the provenance management system, has no control over the transformations and data items. Identification, which refers to changes in the underlying data are not relevant for our case: deletions and updates of data items do not happen. Any changes in the underlying social network or in the authors' profile are relevant from the time of their actual update and do not influence the previously generated provenance.

We store the generated provenance in files, with data models that are easy to handle. The provenance is stored separately from the actual data, but can be joined using the identifiers of messages or authors. We have not developed a uniform data model to store provenance. Instead we used methods for semi-structured data like XML, JSON. Our last efforts have been focused on generating provenance with an interaroparable model based on PROV-DM [MMW13] (RDF) which can combine provenance statements from different sources (see Section 6.2). Furthermore, the provenance of sources is ignored when creating the source provenance (no explicit propagation model). The provenance is computed by the system, after data collection. Our methods also allow for online and incremental computation of provenance.

3.2 Online Algorithms

While Section 3.1 sets the background for the conceptual modeling and formalization, here we discuss some algorithmic aspects for computing those models. We discuss the hypotheses and design decisions that we followed in order to build online systems for explicit and implicit interaction computations.

3.2.1 Explicit Interactions

For source based explicit interactions, we leveraged hypothesis H1 assuming that influence flows though social connections. When we are reconstructing such interactions, for every new message that arrives we have to check whether its author is

[5]https://dev.twitter.com/overview/api/tweets

Table 3.3: Provenance: Classification dimensions

provenance model	
world model	open world model
identification	entry for every new influence edge, no deletions, network is static
representation of transformation provenance	supports transformations by computing how the message content changed on the way/ metadata: author, time, etc
supported transformations	automatic (computed by the clustering algorithm)
representation of source data provenance	original source data: author, message, timestamp, social connections
source provenance definition	input source: items used to create the transformations
query and manipulation facilities	No
storage and recording model	
storage strategy	no-coupling/stored seperately
storage data model	XML, PROV-SAID, etc
propagation	no-propagation
recording strategy	system-controlled/online computation

connected with any of the previous authors who posted the same message (prefix). Consequently, our problem is not a classical stream processing problem where a single pass over the data is desired. Instead, with every new message that arrives, all previously seen messages are relevant for influence computation. This is an iterative stream processing problem where we need repetitive access to the stream prefix. The stream prefix is our search space for influence computation, which corresponds to all previously seen messages and the users who participated in the cascade so far.

We discussed in Section 3.1.1 that hypothesis H1 might lead to multiple influencers and consequently the need for influence models. From an algorithmic perspective these influence models impose an order to users and messages. In the naive, yet strictest case, all possible orders among the messages and among the users are relevant, resulting in a cubic complexity. The cubic complexity comes from considering two orders (messages and users) and checking the social connectivity for every incoming user (message) with the other users in the prefix. Now, the question is how can we reduce the cubic complexity, i.e., which models or alternative hypotheses can be leveraged? By following simpler hypothesis than H1, which uses the same model

for all edges we can bring the algorithm to linear costs. Alternative hypothesis to H1 include considering that the root is the most influencial (supported also by our observations) and assign all influence to the root. Alternatively we can assume that every message is connected with its closest temporal predecessor and in this case favor temporal proximity. Such models involve linear computational costs but do not consider social connections or more fine-grained influence. In any case, H1 is closer to reality than its simpler alternatives, since the majority of interactions flow through social connections.

This brings us to the question, how can we reduce the cubic complexity while keeping H1? We observe that influence models in the literature [CHBG10, BHMW11, TF14] and Section 3.1 consider only one order (among messages or users), e.g. the most popular user, or the closest in time message. We discuss and compute some of these models in Section 4.3.2, where we show that only one order is necessary for the considered models: the user-based influence models ignore the order of messages, and the message-based influence models ignore the order of users. As a result, by dropping one order, we bring the complexity to quadratic levels. The quadratic complexity comes from regarding the relations (adjacency lists and cascade prefix) as sets with a constant look-up time, since one order no longer applies.

For designing an algorithm that computes influence leveraging H1, we need two or three nested loops (depending on the number of orders): the social connection lookup is at the most inner loop. As a result, performing the lookups extremely fast is of paramount importance. In other words, the main computation is to identify connections which are inside three (in the worst case) nested loops by querying a social graph in the range of terabytes. Since this operation is very frequent (with every new message arrival), it must be computed very quickly, especially for a real-time solution.

In practice, we observe that most use cases require one order, however there are applications where both orders are useful. Even when considering two orders there is room for optimization. We observe that the selectivity of checking the social connections is very low and the average in-degree (i.e. number of influencers) when reconstructing information cascades is slightly above one. This means that in the majority of the cases we can find only one influence edge, which makes any ordering inapplicable. For the cases of multiple influencers where the order matters, the resulting sets are very small. We can leverage this hypothesis and re-apply the order at the end of influence computations. In other words, we can compute all influencers ignoring the order, and re-impose the order at the end but now for much smaller resulting sets. In case we are not interested for all influencers but for the $topK$, we can limit the number of generated influencers: for simplicity, we can keep the $top1$ and ignore the remaining (this model is used in Section 4.3.2).

Another optimization is to make an early stop in the social graph lookups: by ignoring any ordering, we can stop the search as soon as at least k influencers are found. However, we will soon face sparsity for large k's, since a single influencer

is identified in most cases. This has consequences for searching: for retrieving one result (influencer), we still have to consider a large amount of social connection lookups.

For designing our system, we make the following observations that we need to consider for our implementation.

1. As highlighted in Chapter 1, the amount and speed of information on current social media imposes new requirements for timely computations, that renders such analysis usable in practice. Many information cascades tend to be viral, reaching millions of re-shares.
2. The size of current social graphs ranges to billions of users with some of them maintaining millions of connections (occupying some terabytes on disk). Querying such a graph in real-time creates a considerable overhead, as its raw size by far exceeds the RAM available to commodity systems. 1. and 2. combines *high volume* and *high speed* streams with computations on *huge graphs*.
3. We observe *skewed distributions* for the social graph: while the average number of followers is around 200, there are many influential users that maintain millions of connections. The same skew is observed for cascade sizes: while the majority of cascades does not reach more than a couple of messages, there are viral cascades obtaining millions of messages. Those viral messages are the most interesting to reconstruct.
4. The *selectivity* of checking social connections is very low since our experiments show degrees slightly above one. This means we might probe possibly very large lists and the return is only one matching connection on average. For large cascades, where the prefix is also large, the problem gets more pronounced. The overhead is increasing with the number of such social connection lists that we need to probe. At the same time, the skew is rising since we probe an increasing number of large lists while the selectivity is always around one.

Taking into account these observations and the computations required for cascade reconstruction including *graph connectedness tests at the inner most position of several loops (cubic or quadratic), over a TB social graph*, *latency* is crucial.

(Q4) *How can we reconstruct cascades at scale and speed by reducing the latency access to the social graph and computational costs?*

(H4) *A distributed solution both for the data and computations will minimize the access latency on the social graph, and keep the computational cost and number of messages to transfer low.*

At this point, we reason over the alternative hypothesis that current machines can handle the load of the stream and the social graph. For both a cluster or a single machine, data locality is involved, which means that the costs to access remote data are for both cases higher. At this point, we note that the corresponding costs of

communicating from one machine to the other in a cluster is more expensive than the costs to retrieve data from a different CPU socket in the same machine. However, for both cases there is a cost increase if data is not local, thus a distributed solution will benefit both of them. In general, companies tend to use clusters because they are cheaper and easier to scale than a single machine.

As far as caching is concerned, one can argue that with proper caching strategies, distribution is not needed. There is no obvious caching strategy for our problem, since our algorithm makes random accesses to massive amount of data. In other words, our access patterns are not cash friendly because we don't observe locality on our data. In particular, the users that have been already seen are not relevant, but the social connections of these users are used for computations. In turn, for every new user that arrives, his/hers connections might pick up the message and forward it further: as a result *related* data might be relevant, i.e. the neighborhood on the social graph. We implemented experiments to test several caching policies and the results were acceptable only when 90% of the data was loaded. As a result, we assume that partitioning the data in such way that relatedness is favored will improve the efficiency. In other words, we strive to place together related data, which means in our case connected users on the social graph.

Coming back to H4, that supports distribution, we believe that a) partitioning the social graph (set of adjacency lists) in a non-overlapping way will achieve a good scalability in terms of the massive graph sizes. Also, by b) performing all iterative accesses to adjacency lists and checking of set containment *locally*, we avoid the latency penalties of such accesses. This means that c) we need to distribute over messages of the same cascade on different machines.

For a) partitioning the social graph we test several hypotheses. The main question is:

(Q5) *How to partition the social graph so that the communication among different machines is minimized?*

The hypothesis here is:

(H5-I) *If we manage that user interactions and influence remain local, (i.e. users - nodes on the social graph - that interact frequently, are being placed on the same site), then we increase the efficiency of computations. In other words, by increasing data locality we lower the communication cost.*

We will leverage methods from community detection that identify the latent social structures, where most of the interactions take place. Community detection can be applied to both the social graph and the interaction graph, which instead shows who is interacting with whom. It has been shown that the interaction graph captures better influence among users, since not all of the social connections have equal strength [WSPZ12].

However, we have observed massive skew in social graph connections, which means

that we will face this massive skew when distributing the data by such methods. This observation is in line with [For10], showing that most community detection algorithms create imbalanced clusters. This comes in contrast with the principles of parallelism which requires that partitions are balanced.

This leads us to an alternative hypothesis:

(H5-II) *Balancing the load in the partitions in order to speed up computations.*

The placement of nodes in each partition can be based on graph partitioning techniques or randomly. Hypotheses H5-I and H5-II can be applied to any social graph computation problem, where the social graph does not fit in one machine. We are going to discuss such implications on Section 4.3.

By using hypothesis H5-I which favors locality of related data, we have to deal with the massive data skew. The question that rises is:

(Q6) *If we give up locality because of the massive skew, are there any methods that improve the latency?*

(H6) *By hiding the latency we can achieve comparable or better performance.*

We leverage the idea of latency hiding by a broadcast method that sends all messages to remote sites and does the computations in parallel. In this case, we are wasting network bandwidth, but still the benefits are considerable compared to the unbalanced strategy. Our experiments showed that we indeed achieve greater performance by locality (H5-I), but the alternative hypothesis of latency hiding (H6), brings better results (in terms of computational times). We provide more details in Section 4.3.5 and 4.3.6.

The next questions concerns the actual computations of individual cascades:

(Q7) *How can we lower the costs of communication of different nodes when reconstructing a single cascade?*

(H7-I) *We collect all results (and perform the related computation) at the site of the cascade starter*[6] *to minimize the distributed operations altogether. Since the majority of cascades have a star shape, and the root message is the most influential, we assume that the majority of the computations will happen in the cascade starter site.*

(H7-II) *We also combine the low selectivity of lookups with the social graph partitioning strategy to minimize the amount of result data and the time to transfer.*

(H7-III) *We utilize stateful and parallel processing in addition, to reduce the overall response times.*

After we have described our design decisions for a distributed computation, we

[6] the site/machine that contains the cascade root node

will explain our rationale for implementing such an algorithm. First we turn our attention to the skew of cascades size. The majority of cascades are small, attracting a couple of retweets, however the viral cascades are the most interesting ones to reconstruct.

(Q8) *How can we deal with the skew of cascades and how do we evaluate its impact on in the computational cost?*

(H8-I) *We develop an adaptive algorithm with two modes, which iterates either over the prefix or over the adjacency lists depending which set is larger.*

We provide more details in Section 4.3.

(H8-II) *Since in a real time scenario we don't know the end size of the cascade in advance, we can dynamically adapt the algorithm (prefix vs user iteration) accordingly. This adaptation can balance the size of cascades with the size of adjacency lists.*

However, it is not clear whether dynamically adapting the algorithm will produce additional overhead. We will evaluate this hypothesis in future work.

In this work, in order to understand better the behaviour of the reconstruction algorithm for different cascade sizes, we evaluate different classes separately and examine the trade offs.

3.2.2 Implicit Interactions

For implicit interactions we formed hypotheses H2-I and H2-II: we use similarity as a means of incluence computation in lack of any other provenance indicator. We leverage the clustering algorithm in [AHSZ11] which forms natural clusters with a lower bound of similarity, using a feature model with a similarity function (e.g. Tf-Idf and cosine similarity). The advantage of the algorithm is that the number of clusters should not be specified in advance. We discuss more details on that in Section 5.3.

Such an algorithm has a quadratic complexity because we have to compute the similarity among all pairs of messages. In order to deal with evolving streams of messages, we developed an incremental version of this algorithm [AHSZ11]. This becomes soon problematic with the increasing number of messages. While provenance information from social media providers limits the search space for explicit interactions, for implicit interactions all previously messages might be relevant for influence computation. This is not viable with the size and speed of current social media. The question that emerges is:

(Q9) *How to limit the search space for provenance computation?*

We make the following hypotheses to mitigate this problem:

(H9) *We assume that users have a limited attention span and are mainly influenced by recent information. Complementary, news' articles are becoming soon outdated, given the variety and fast updates of online information sources.*

Hypothesis H9 can be applied to any social or news media with different time spans according to the nature of the data. This means, we expire messages in order to limit the computational space, since not all of them are relevant at every point in time. Alternatively an aging function can be considered, that provides recent data with greater relevance. This we leave for future work.

Even with time restrictions, for every new message that arrives we need to compute the similarity with all previous messages into the selected time span from H9 and insert it into the existing clustering.This means that the existing feature model (Tf-Idf) will need to adapt with every new arrival. The question here is:

(Q10) *How can we further reduce the computational costs of building the model and computing the clustering?*

We make the following assumptions and design idea that will reduce the computational costs without compromising in result quality:

(H10-I) *We assume that with every new arrival the existing feature model will change minimally, and small updates will not have any impact on the clustering.*

As a result, we use a threshold that quantifies the difference of Tf-Idf for the existing terms. For small changes the terms are not being updated.

(H10-II) *Given the clustering algorithm with a bound on similarity, pairs of messages that have a lower similarity than the bound, are not contributing to the model, and as a result can be discarded.*

Thus, with increasing similarity bounds the performance is massively improved, given also the observation that the majority of pairs have very low similarity.

3.3 Formal Models

All these observations, methods and analysis are incorporated into an expressive model that integrates explicit and implicit interactions. In Chapter 6, we describe the PROV-SAID model which extends the PROV-DM model, the W3C PROV standard for provenance. The properties that we desire from such a model are the following:

- *Generality*: the model should be general enough to capture influence (i.e. provenance) of interactions on a wide range of social media. It should not constrain the expression of statements under certain assumption or social media specifications.

- *Expressiveness*: It should model the most commonly used interactions, e.g. establishing social connections and exchanging messages, to enhance expressiveness. The goal here is not to minimize the relationships in the model, but to maximize expressiveness. As a result we favor expressiveness over minimality (which is a key goal in databases).
- *Extensibility*: At the same time it should be easily extensible to capture many other types of interactions or social media specification for particular use cases.
- *Reusability*: The model should use when possible concepts already given (from PROV-DM), and extend only when necessary.
- *Interoparability*: The model should leverage the web-native and interoperable format (PROV) that allows easy publication of provenance data, and minimizes the integration effort among different systems making use of PROV.
- *Open world model:* It should consider external influence and allow the modeling of existing but unidentified provenance. In other words it should allow uncertainty with regard to the influencer.

We present such a formal model in Chapter 6.

3.4 Dataset Collection

In this Section we describe our methodology and data collection, which is relevant for all the subsequent Chapters. Performing an online analysis of information diffusion requires access to the relevant messages while they occur as well as an up-to-date instance of the social graph. For both goals, we need to overcome a number of challenges, requiring particular retrieval strategies. Among the popular online social media services, Twitter is the only one that provides an API to access messages and social graph information on the fly, but this API bears significant restrictions.

3.4.1 Messages

For messages, Twitters' Streaming API[7] grants access to a subset of the current stream of messages. This subset can be defined on the basis of user names, keywords (including hashtags) and geocoordinates. There are, however, two kinds of restrictions on this API: On the one hand, the number of user names, keywords and coordinates that can be followed by an account are limited (currently to 5000 each). On the other hand, the number of messages per time produced by such a subscription must not exceed 1% of the total number of messages processed by Twitter at the same time. In cases of heavy traffic - such as a very popular topic at a certain instance - this threshold is exceeded, so we are missing messages (and retweets).

[7] https://dev.twitter.com/docs/streaming-apis

Furthermore, Twitter provides only limited means to retrieve messages after their occurrence.

These limitations have another consequence: we cannot observe all possible retweet cascades, but need to settle for specific subsets before we start to record. Generally speaking, we would need to perform some kind of event or virality detection on the fly in order to determine this subset, which is a research problem on its own. For the time being, we settled for two simple, but still promising approaches to achieve this goal: If we are aware of events that are likely to generate a considerable amount of tweets and retweets (such as Olympics or US elections), we use specific keywords to track cascades referring to such events. This approach bears the drawback that we can request only messages of events known in advance. To overcome this problem and catch also emergent or unpredictable events, we observe the Twitter "sample" stream, containing a small randomly sampled subset of the full message stream. We detect relevant cascades that demonstrate a bursty behaviour in their beginning without knowing the specific topic of them. The beginning of the cascade is then immediately fetched using the Twitter REST API.

3.4.2 Social Graph

For the social graph, Twitter offers methods to retrieve connections for every user, both the list of users who follow this user (followers) and the list of users this particular user follows (friends). Even compared to the limits on message subscriptions, the limits on the social graph are very strict: at most 60 users or 300K follower entries (whatever is smaller) can be retrieved per hour and account. Since we need to deal with high message rates in cascades, on-demand retrieval of current social network information during the reconstruction is not feasible. Instead, we have to retrieve the social graph over time, cache it and refresh it in order to reflect the graph evolution due to following and unfollowing of users over time. Given the sheer size of the social graph (100s of millions of users with their connections), we crawl the social by fetching information of those users that are active in retweets. We favor those users that are retweeted most and/or have the most followers, in order to capture possible popular users that exert influence on others [CHBG10]. When necessary, we can augment this collection by explicit requests on specific users. Since retrieving follower and friend information would provide redundant information, we chose to retrieve only the follower information in general. This is motivated by the fact that followers information provides a better expression of influence and gives a quick way to retrieve all connection information for the starter of a cascade. We retrieve the friend information when they are needed explicitly for some variants of our algorithm. We present more details in Section 4.3.4.

3.5 Discussion

In this Chapter we lay the foundations for our methods and analysis that will follow in the next Chapters by providing an overview of our methods and design decisions. We discuss the different types of interactions under similar dimensions, with the corresponding similarities and differences. We make plausible hypothesis for computing these types of interactions, which comprise our methodology for the next chapters.

At this point we discuss an additional type of interactions, that we leave for future work and falls between explicit and implicit interactions. *Meme based interactions*, like hashtags in Twitter/Facebook/Instagram includes a cultural symbol or social idea that is rapidly propagating on social media. This type of spreading is officially recognized by social media to demonstrate that particular messages belong together. However, no provenance information is provided by social media providers, which makes meme based interactions borrowing characteristic from both explicit and implicit interactions.

For messages that contain hashtags (annotated text), we apply methods from implicit interactions by considering the hashtags as normal text. For future work, we can treat hashtags as additional indicators: if users are sharing similar hastags, then the reconstructed provenance is strengthen. From a computational perspective, we can reconstruct hashtag cascades by leveraging hypothesis like H1 which relies on social connections to reveal influence. In such case, we might have multiple influence edges, if more than one social connections are activated and multiple roots, in case different hashtags co-occur or the same hashtag propagates from different independent sources.

Hashtags reflect real world situations very often, and as a result, there is a lot of external influence involved. In the future we will provide additional support for identifying and computing external influence, as a considerable amount of influence is coming from the real world.

Chapter 4

Explicit User Interactions

4.1 Overview

In this Chapter we describe methods to compute explicit interactions on social media. We are focusing on Twitter as a rich source of datasets and interactions. In Twitter, we observe both types of direct linkage and source based models.

Section 4.2 discusses the direct linkage interactions. Replies and quotes are examples of such interactions. Reply is a very widespread feature in Twitter, and many other social media and platforms. The reconstruction of replies is relatively simple since the previous step is embedded in any reply. However, since reply cascades include messages that are not identical in contrast to retweets, finding messages that belong to the same conversation cascade is challenging. We solve this thought the Twitter API, which is the main bottleneck especially for real-time computations. Quoting is a new feature with relatively sparse use. The linkage provided is similar to replies as well as the method to reconstruct them. A quote gives the possibility to the online user to forward an existing message including his own comment.

Section 4.3 discusses the source based interactions. Retweets is the most prominent example of source-based interactions in Twitter. We provide methodology and a system to reconstruct retweet cascades. Our methods are based on H1 from Chapter 3, where social connections are used to derive influence. We implement a system that supports real-time computations. The main bottleneck is the huge social graph which combined with streams, is a problem that has not been tackled in related work. Those methods can be applied to any other kind of dataset with an underlying social graph.

While retweets refer to typical information diffusion, replies are connected with conversations, which does not imply pure information diffusion. Quotes fall in-between, since they allow for new content. However, all of those means imply some information propagation in a wider sense, since they facilitate an event or topic to

gain further visibility. We mainly focus on source based interactions (retweets) both from the conceptual and technical side, where the methods are disentangled from any API limitations. For direct based lineage interactions (replies and quotes), we rather focus on the methods to reconstruct conversation cascades in Twitter and how to use the Twitter API's to retrieve complete conversations.

This Chapter is partially based on the results of student theses and projects [Lut17, Hub14, LS13, Tri17, Jab16] that were implemented under our supervision in University of Freiburg. Our publications related with this Chapter are the following:

[TF13] Io Taxidou and Peter Fischer. Realtime analysis of information diffusion in social media. In *Proceedings of the VLDB Endowment*, 6(12): 1416-1421, 2013.

[TF14] Io Taxidou and Peter M Fischer. Online analysis of information diffusion in twitter. In *In Proceedings of the 23rd International Conference on World Wide Web*, pages 1313-1318. ACM, 2014.

[TDNF17] Io Taxidou, Tom De Nies, and Peter M Fischer. Provenance of Explicit and Implicit Interactions on Social Media with W3C PROV-DM. In *Lecture Notes in Computer Science, Springer LNCS/LNAI series*, 2017. Accepted, under publication procedure.

[FTLH16] Peter M Fischer, Io Taxidou, Bernhard Lutz, and Michael Huber. Distributed streaming reconstruction of information diffusion. In *In Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pages 368-371. ACM, 2016.

4.2 Direct Linkage Based

4.2.1 Introduction

Online social media offers multiple ways for communication. We observe long conversations for current events e.g. among politicians during the elections [LWBS14]. Here we reconstruct and analyze replies and quotes in Twitter. We observe conversation cascades containing single iterations (replies or quotes in isolation), however, it is often the case that replies and quotes are interleaved in the same cascade. Consequently, we investigate the combined interactions of replies and quotes, which provide new insights for mining online user interactions.

While retweets show aspiration of an author's views and direct influence by forwarding a copy of another message, replies demonstrate engagement and activity. A reply, is a new content-wise message, which entails greater effort to be written than a retweet (which just replicates the content of an older message). Quotes fall in between: they embed another message (like in retweets) but now new content can be added, commenting the embedded message. It does not imply always aspiration

to the original content, as the user can show support or opposition to the quoted message. From a lineage perspective, Twitter provides for both reply and quote the previous predecessor.

There are some challenges connected with reply and quote reconstruction. First, reply and quote are encountered in the same cascade very often and we have to seek their interleaved provenance. Second, Twitter allows that a message is a reply and quote at the same time, leading to two different diffusion paths and sources. Reconstructing cascades with multiple types of interactions raises the corresponding challenges in terms of modeling and computations.

Third, reconstructing *complete* conversations imposes additional challenges. Given the nature of replies, data collection for complete cascade reconstruction complicates matters: replies might bear little or no similarity with their predecessors, yet content similarity has been our assumption for data collection (current events or topics). In particular, the policy for collecting data has been keyword filtering, which collects tweets matching particular keywords like "Olympics" (see Section 3.4). While this method works well for retweets (i.e. identical messages), it is unable to collect complete conversations, because replies might not always match the keywords of their predecessors. Since we strive for data completeness and an open world model, we need to collect additional, unknown to the current dataset replies. For that, we rely on the Twitter API, which has its own limitations and challenges. The use of Twitter API might result in rate limiting in cases of high traffic or for large conversations. This is particularly problematic when implementing real time systems, because they would be bound to API restrictions, as a result their performance cannot be measured reliably.

The contributions of this part are the following: First, we investigate the combined interactions of reply and quote conversations. While conversation reconstruction has seen some attention in literature, e.g. [CAB+12], the combination of different interactions has not been investigated much. By considering combined interactions, we obtain a more complete picture of communication patterns and information diffusion. We expect that the results will demonstrate longer diffusion paths and longer diameters, refuting previous claims that cascades tend to be wide rather than deep [KLPM10]. Second, we investigate how to reconstruct *complete* conversations by leveraging the Twitter API and complementing the initially recorded, but possibly incomplete, datasets. The main challenge is API limitations which is very hard to tackle when developing real-time methods. The developed algorithm presented is incremental and can function in a real time scenario. However, it is not yet optimized to sustain the loads of the current Twitter streams. From an algorithmic perspective and in the absence of API limitations, the suggested algorithm scales well: since direct linkage is provided, no complex computations with large state are needed. Currently, our methods can sustain the load of the selected datasets which contain very large conversations. We leave the technical challenges of this algorithm for future work.

In the following Sections we present our methods for reply reconstruction. In Sections 4.2.2 and 4.2.4 we present the methods and algorithms. In Sections 4.2.3 and 4.2.5 we present an analytical evaluation for reply and quote reconstruction. In Section 4.2.6 we conclude our findings.

4.2.2 Reply-Quote Reconstruction

Here we describe the methodology for reply and quote conversation. As discussed in Introduction, in order to reconstruct *complete* conversations, we reply on Twitter API's to collect additional replies. For reconstructing conversations (reply cascades) in Twitter, we use recursion to identify provenance, since the previous step is given but not the root of the conversation. As a result, reply cascades are trees, since Twitter API provides the previous influencer. This way, we recursively reconstruct diffusion paths until we reach the conversation root. We call this *backward search*. If the previous message, which is embedded in every reply is not available in the dataset, we retrieve it through the *Show Status API*. The Show Status API returns a single tweet given an tweet id (which in our case in embedded in the reply).

As discussed in Section 3.4, our method of collecting datasets queries for particular terms, which works well for a real scenario, where we observe the spread of a particular topic. However, since relevant replies might not contain the particular term are being ignored. In order to collect those additional replies (further branches of the reply tree), we collect the timelines [1] of involved users. The Twitter API (Get User Timeline API) allows us to get further replies from these users who participated in the conversation. This we call *forward search*. The search can go back in the past for approximately 3200 messages for each timeline, as a result for some not very active users a good coverage can be achieved.

Quotes are being reconstructed in the same way as replies with the following difference: For the *forward search* quotes cannot be retrieved from users' timelines and an explicit search for the message id that got quoted is needed. This way, additional quotes are being collected. With this method, data could be retrieved for one or two weeks in the past, which means that such provenance information needs to be reconstructed in a timely way. In order to differentiate among the reply edges and quote edges we refer to *r-edges* and *q-edges* respectively.

Figure 4.1 shows reply and quote cascades in isolation. We observe that reply and quote cascades (with single interactions) are trees since the direct previous lineage is provided. Those edges have very low uncertainty since they are given by Twitter API. However, a reply might be quoted at a later point in time and vice versa, which results in a cascade with a single root, but containing both reply and quote edges (r-/q-edges). Figure 4.2 shows such a case. This is happening much more often in practice.

[1] A user timeline displays the latest tweets ordered from newest to oldest from a user account

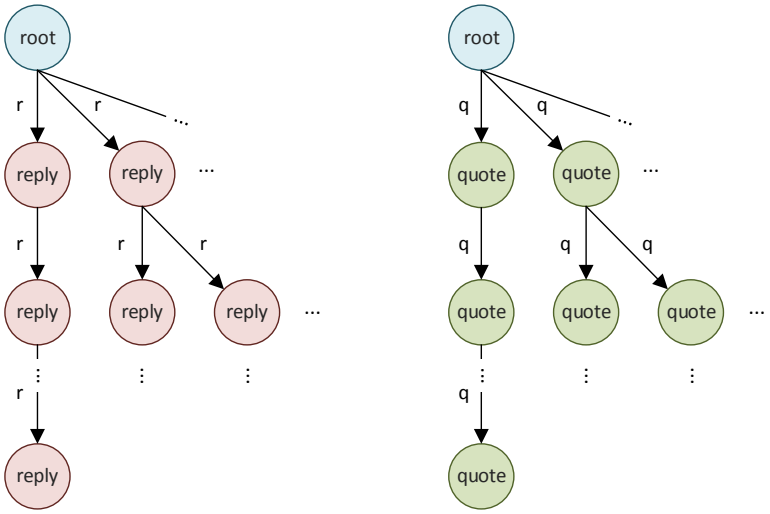


Figure 4.1: Isolated Reply and Quote reconstruction: r-edges denote reply edges and q-edges quote edges respectively.

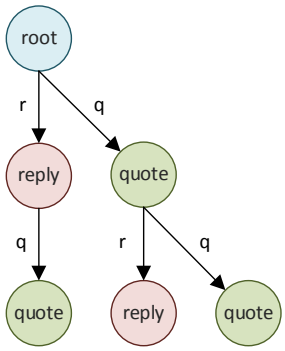


Figure 4.2: Combined Reply and Quote reconstruction

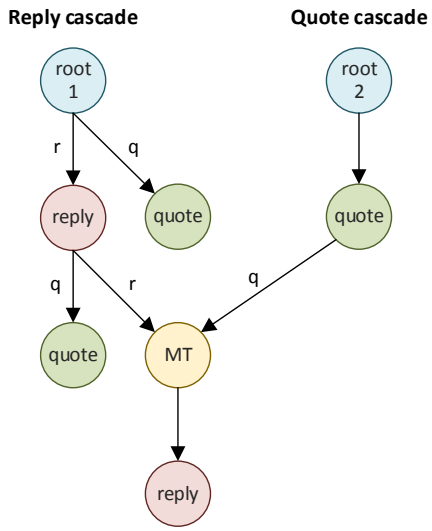


Figure 4.3: Multi-tweet

While in the majority of the cases a message is either a reply or a quote containing the direct predecessors, we note the following empirical observation: a tweet might be a quote and a reply at the same time. In such cases, we observe multiple roots: one for the reply and one for the quote cascade. A tweet that is both a reply and a quote is called *multi-tweet*. Conceptually it is reasonable to quote another tweet as a reply, in other words, to reply by mentioning some previously written content. (Note here, that retweets cannot be combined with others means.) Multi-tweets are not being used widely but we are modeling and computing them explicitly, since they provide interesting insights and connect different conversations. Figure 4.3 shows such a case. We observe that the existence of a multi-tweet will result in a DAG with two roots (one for the reply and one for the quote). Additionally, all the descendants' of the multi-tweet will also end up in two roots.

However, we model a conversation that contains multi-tweets as *separate conversations*. This way, we stay consistent with the tree model which is also a natural model for the direct lineage problem: the provenance of (one) previous step leads to a tree. This results in a multi-tweet participating in two conversations. However, when evaluating those graphs, we connect the separated conversations thought the multi-tweets. This way, we reveal the more complex structures of the combined conversations and we gain a more complete picture of human interactions.

Those influence edges that are generated from replies, quotes or multitweets are low uncertainty influence edges. Since those edges are provided by Twitter API, they are of high precision. Next, we present a conceptual evaluation when we present properties of conversation cascades.

4.2.3 Evaluation

First, we describe the datasets used and then we present an analytical evaluation over conversations. As discussed on Section 4.2.2, dataset collection is challenging for the replies, especially when aiming for *complete* conversation reconstruction.

We have used the following datasets which were recorded with the methods from Section3.4.:

- Dataset for the World Wide Web Conference in 2015, with keyword "www2015"
- Dataset containing well-known users: politician, celebrities, online companies.

The first dataset was recorded during the World Wide Web Conference in 2015, with the keyword www2015. This dataset consists of 7363 messages in total, of which 141 are replies, 22 are quotes and 3819 are retweets. Note here that when reconstructing replies and quotes additional messages might emerge that complement the conversations, as discussed in Section 4.2.2. We reconstructed the reply and quote cascades according to the methods presented in Section 4.2.2.

Replies and quotes constitute a very small fraction and they are often observed interleaved in the same (mixed) cascade. We identified 114 reply cascades with a skewed distribution: the largest reply cascade has 53 replies and the majority obtains 1-2 replies (Figure 4.5). We also plotted the distribution of the number of distinct users participating in the conversation: we observe that the majority consists of two participants, which means that users are highly conversational in this dataset. Quotes consist a tiny fraction of the dataset: the largest cascade contains 8 quotes (interleaved with replies) and the rest consist of 1 or 2 quotes. At that time, quote was a new feature and relatively unknown to Twitter users. We plotted retweet

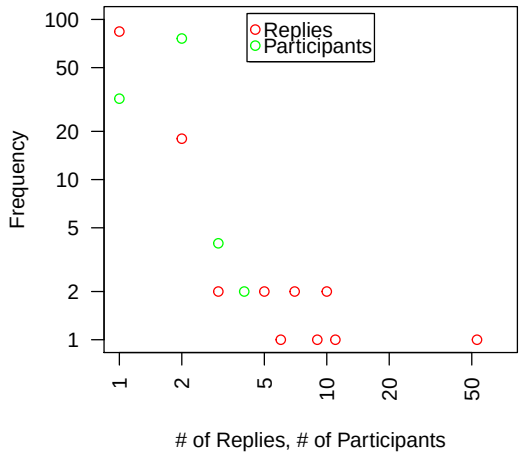


Figure 4.4: Retweet Cascade Distribution

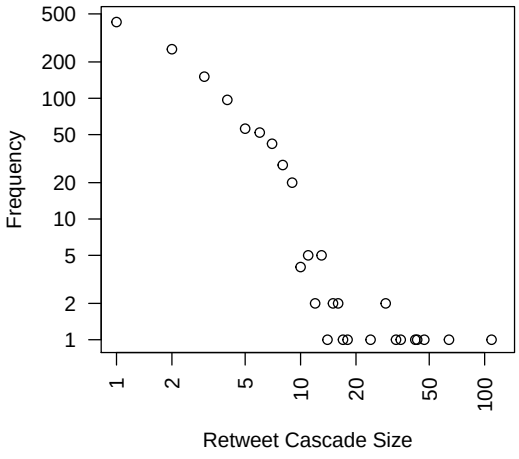


Figure 4.5: Reply Cascade Distribution

cascade size distribution for comparison. We identified 1162 retweet cascades with size distribution presented in Figure 4.4: The are very few large in size (the largest consists of 109 retweets and the high majority contains 1-2 retweets). We observe that retweets dominate compared to replies, as a means of diffusion.

We summarize the second dataset in Table 4.1. For this dataset, well-known personalities were selected and their timelines were recorded from January to May 2017. The user timelines include their tweets, along with the retweets, replies and quotes from other users. Retweets are excluded, since they have been thoroughly investigated in Section 4.3. In order to reconstruct and complement conversations, the backward search was activated for the collected timelines. In turn, the backward search activates the forward search to collect any additional replies or quotes.

Table 4.1: Dataset for reply and quote reconstruction

User-id	**Username**	**Date**
21447363	@katyperry	25.04.2017
813286	@BarackObama	26.04.2017
15846407	@TheEllenShow	28.04.2017
25073877	@realDonaldTrump	01.05.2017
62513246	@jk_rowling	05.05.2017
155659213	@Cristiano	06.05.2017
85741735	@AmazonHelp	07.05.2017
2384071	@timoreilly	09.05.2017

Table 4.2 presents the number of conversation and statistics for each account crawled. In total, we reconstruct 5,731 conversations. Note here that when a multi-tweet occurs we connect the involved conversations. While for the ISWC dataset the number of quotes is very low, we observe that in more recent datasets the amount of quotes massively overtakes replies. This means that a large share of users who retweeted in the past, now they use the quote feature, to add their own comments.

In Table 4.3 we compute some further statistics on these conversations. We observe a very heavy tailed distribution with median value of three tweets (replies and quotes) and one participant. Also the number of multi-tweets is very low (0.99 Quantile = 1). As in the case of ISWC daraset, the number of participants is low (median=2) which means that this dataset is also conversational. We further computed the diameter of those conversations. The maximum diameter is 17,367, and the median value is 759. We observe surprisingly high diameters compared to related work [CAB+12] which is the result of considering combined interactions. Next, we analyze the number of roots for the reconstructed conversations. Given that the number of multi-tweets is very low, the number or roots is also low, with maximun number 127, 0.95 Quantile 2 and 0.99 Quantile 5.7.

Table 4.2: Datasets Statistics

Username	Conversations	Tweets	Quotes	Replies	Multi-Tweets
@katyperry	205	7,433	6,866	283	87
@BarackObama	19	7,207	7,049	127	30
@TheEllenShow	180	24,070	23,250	484	172
@realDonaldTrump	271	89,831	85,578	3,544	685
@jk_rowling	508	68,668	65,037	2,800	448
@Cristiano	8	1,429	1,372	20	31
@AmazonHelp	3,975	16,775	473	12,362	12
@timoreilly	565	108,678	104,917	2,733	708
Total	5,731	324,091	294,542	22,353	2,173

Table 4.3: Statistics for conversation Reconstruction

	Total Tweets	Multi-Tweets	Participants
Minimum	1	0	1
1 Quartile	2	0	2
Median	3	0	2
3 Quartile	6	0	2
.90 Quantile	12	1	5
.95 Quantile	38	1	28
.99 Quantile	874	6	827
Maximum	18,265	126	17,523

4.2.4 Realization-Challenges

In this Section we discuss the challenges of conversation reconstruction, especially the API limitations and how we overcome them. The conversation reconstruction algorithm depends heavily from *Twitter API* which renders any improvements in its performance hardly feasible.

We cost lies in two aspects: 1) Technical: explicit requests to API for reply and quote interactions are time inefficient due to network latencies. 2) Rate limiting: if too many calls are made in a short time, Twitter imposes *rate limiting* and we end up having missing data in the reconstructed cascades. Rate limiting is unavoidable is cases of very high traffic. If rate limiting occurs, we need to resume the algorithm and wait for long periods, before we can retrieve the remaining messages. *User privacy* settings impose further challenges. For those users that are private we cannot retrieve any information, which results in missing messages.

Additional *API limitations*, including the maximum number of messages or the period that is feasible to access messages from the past, renders complete reconstruction of older reconstructions very difficult. In order to overcome this problem, we have to reconstruct promptly any obtained conversations.

The *iterative* nature of the algorithm further complicates matters for an efficient realization. There is no termination condition for both the backward and the forward search. This means that given unlimited API access the algorithm might never terminate. For that, we can impose a termination condition: e.g. in the evaluation (Section 4.2.3) we use three iterations. In order to improve efficiency, concepts from *parallelism* could be used. For example, all the forward searches could be implemented in parallel, since this process is very time inefficient. This we leave for future work, since our current implementation can sustain the work load for the selected datasets.

Another important issue is the *reproducibility* of results. In general, any kind of methodology and implementation should produce identical results given a particular dataset. In other words, the output should be the same in every instance of reproducing any given methodology. However, here is unfeasible to claim reproducibility because of the dynamic Twitter environment, and real-time information we leverage through Twitter API. Messages might get deleted and users might become private, which causes the results of be different when applying the this methodology at different points in time. We can only claim reproducibility when testing our algorithm *offline*, but this means that we are missing additional messages, not captures by the initial data collection.

We address those challenges when possible, following two lines:

- *Batching* of requests towards the API: Instead of making seperate API calls for every message or user, we batch those requests in one API call.
- *Caching* in order to speed up the access to data: We cache in cases that data needs to be accessed multiple times during the reconstruction.

We discuss how we use those optimizations for the backward and forward search. For the backward search, we batch a particular number of messages that we need to search for their predecessor. For every of those messages, we first check in the cache, whether their predecessor is already collected. In the opposite case, we collect the remaining messages and we send a batch-request to the Twitter *Show API* to fetch their sources.

For the forward search, we need to access the timelines of users in order to collect additional replies. In general, when the timelines of users are being fetched from the Twitter *Get User Timeline API* are being stored for future use. First the algorithm checks if the requested timelines already exist on disk. If this is the case, then the most recent user activity that was not recorded, is fetched from API. In other words, we collect the delta of activity that happened after we last recorded those timelines. If the timelines are not on disk, then we fetch them, collecting as many messages as the API allows to retrieve from the past. During reconstruction of conversations, it is highly likely that the same users will appear multiple times in the conversation graphs; this comes from the observation that cascades are highly conversational with a low number of participants (Section 4.2.3). Due to that, we cache the timelines we have seen so far for faster access. Since the cache space might be fast exhausted for large conversations, we implemented an LRU [2] substitution policy, in order to keep always the most recent data available for further use.

4.2.5 Performance Evaluation

In this Section we provide a short evaluation for the performance of the conversation reconstruction. We use the second dataset collected in Section 4.2.3. We provide insights for 1) how many additional messages are collected through the Twitter API during the conversations reconstruction, and 2) what are the run times for the backward and the forward search.

First we compute how many new messages are gathered during every iteration for the forward and the backward search. Figure 4.4 shows these numbers:

Table 4.4: Tweets in every Iteration

Iteration	**Total tweets**
1.Iteration	57,321
2.Iteration	87,132
3.Iteration	154,966

We observe that from the first to the third iteration the number of messages tripled from 57K to almost 155K. That is the reason why we stopped collecting messages after three iterations.

[2]Least Recently Used

The runtimes of forward and backward search for the reconstructed conversations (three iterations) is presented in Table 4.5.

Table 4.5: Total Runtimes in minutes for Forward and Backward Search

	Backward Search	**Forward Search**
Minimum	0.0082	0.52
1 Quartile	0.092	18.16
Median	0.54	164.74
3 Quartile	1.61	1,065.03
.90 Quantile	30.22	3,273.28
.95 Quantile	6.47	2,563.95
.99 Quantile	12.02	2,755.88

We observe that the forward runtimes always dominate. We also observe that the backward search is sustainable even for large conversations. For the forward search the .99 Quanitle is around 46 hours, which means that we encounter large conversation combined with rich user timelines. The median value is 2,5 hours, which is much lower than the time that most of the conversations needs to unfold in reality. This means that the method can work in a real-time scenario.

4.2.6 Conclusion

In the previous sections, we provided the methodology to reconstruct influence paths when the linkage of the previous predecessor is given. We focus on quotes and replies in Twitter and we strive for obtaining complete conversations through the Twitter API's. We observed that quotes and replies are often interleaved and we evaluated cascades with mixed interactions. With our analytical evaluation we showed that interleaved reply and quote cascades result in larger conversational cascades and larger diameters. We also discuss the methodology and technical challenges to overcome in order to obtain and reconstruct complete conversations.

4.3 Source based Interactions

4.3.1 Introduction

On Chapter 1 we have outlined the benefits of computing information diffusion and provenance, particularly in an online fashion [TF13]. In contrast to replies and quotes, retweets provide only the source of the cascade and the intermediate steps

need to be inferred. The advantage is that we know that particular messages belong together under the same root. However, we need to make particular hypotheses and rely on heuristics in order to identify who was influenced by whom. An additional challenge lies in missing and incomplete data that derives from API limitations or unobservable influence. Since the direct predecessors are not provided (as a result, they can not be collected through the API's), we have to rely on approximate indicators for quantifying missing data.

In order to keep up with the stream of messages, incremental and distributed computations are required. Complementary, given the need to trace information diffusion and identify who is connected with whom, processing the social graph in a fast way becomes a necessity. Considering the current sizes of social graphs (330M active users in Twitter), partitioning and distribution need to be performed in such a way that quality of results is not affected.

In this Section, we present methods and results of how information diffusion can be studied in real-time, considering retweets on Twitter. We tackle the problem of determining *influence paths* that express the relationship of *who was influenced by whom*. Our online method relies on an algorithm and supporting system to infer possible influence paths from the stream of messages (tweets) and the underlying social graph (follower and friend graph). Our contributions are the following: 1) We provide insights on how to infer influence without relying on any statistical modeling and provide a broad range of alternatives for influence computation. 2) We provide a system that computes influence paths on real-time in the presence of both very fast streams and huge social graphs. Our methods are evaluated on retweets, but they can be used as a general model of inferring influence paths for any kind of information that propagates over a social network, e.g. URLs or hashtags.

The structure of this part is the following: in Section 4.3.2 the methodology and metrics to reconstruct and analyze information cascades are discussed. Section 4.3.3 evaluates the reconstruction of retweet cascades in terms of properties and missing data. Sections 4.3.4 and 4.3.5 provide the background and methodology for the realization of distributed and real-time reconstruction. Section 4.3.6 evaluates the system for efficiency and applicability in real scenarios, where current streams and large social graphs are involved. Lastly, Section 4.4 concludes the section of source-based interactions.

4.3.2 Methodology

In order to track information diffusion on real-time, we need to extract information cascades out of the message stream. A cascade is formed when users forward the same original message from a user that we call the *root user*. The exact influence path, that shows how forwarding occurs, is not available from the message stream. Under the assumption that the social graph connections (e.g. followers and friends) serve as means of information diffusion and influence, we can derive influence paths

from social connections among users (H1 from Chapter 3). This hypothesis produces medium to low certainty influence edges. Although we will prove in the evaluation Section 4.3.3 that this hypothesis explains our data, there are some online users who are influenced outside of their social graph neighborhood.

A core aspect of modeling information diffusion is the assignment of influence: users might be exposed and influenced by a piece of information by multiple users, hence forming multiple influence paths [BHMW11] (see Chapter 3). When a message arrives that is a retweet, every friend that has (re)tweeted at an earlier point in time has to be considered as a potential influencer, if no constraints are made on the influence model. Specific influence models, however, may include only a subset of these influencers, as in reality users are not influenced equally by all their friends that forwarded the same message.

In order to provide a clear understanding on the interaction of social network connections and messages streams with information cascades, we provide a lightweight formalization: The social graph $SG = (V, F)$ is a directed graph of follower/friend relationships, showing for each user (node) from V who follows this user (F). For simplicity we assume that during the reconstruction this social graph remains static. The message stream is expressed as a sequence of messages M^* in temporal order. Each message M contains several attributes, out of which we just list the ones most relevant for our work: (1) timestamp t (2) user $v \in V$ (3) information item identifier i, e.g. a retweet ID or a hashtag. We say that two messages m_1 and m_2 belong to the same cascade iff $m_1.i = m_2.i$. This model is flexible enough to express many kinds of information diffusion, not just retweets.

Based on these foundations, we define a cascade graph $C(U, E)$ with $U \subseteq V$ as directed graph of influence paths among users. This graph is a subset of SG annotated with (at least) the influence time on the edges. C contains only nodes of those users who actually (re)tweeted, but not those who were exposed to the information without reacting. This way, we can use a smaller graph among the audience reached that focuses on the influence paths. If needed, a full reach computation can easily be achieved by incorporating these passive users that are followers of the (re)tweeters. Among the users in this cascade that tweeted, we designate $u_r \in U$ as the "root", i.e. the user who initially distributes the information item.

The influence paths (edges) need to fulfill the following condition: an edge $(u_i, u_m, t) \in E, u_i, u_m \in U$ may only exist if $\exists m \in M^* : m.u_i = u_m \wedge (u1, u2) \in F \wedge (u_i = u_r \vee (\exists n \in M^* : n.u = u1 \wedge n.t < m.t))$. In other words, a user u_m who spreads information using a message m is possibly influenced by an user u_i if there is a social network connection from u_i to u_m and u_i is either the root or was exposed to this information by a message n which happened before m.

We design the baseline version of our reconstruction algorithm to exhaustively search theses edges in E for all messages in M^*. As our goal is to perform this reconstruction in an online fashion, these edges shall be added to C in an incremental manner whenever a message arrives. Our algorithm therefore checks at every arrival of a

message m if a SG connection from $m.u$ to the user of any message $n \in M^*$ which arrived before m exists. If such a connection holds, it is a possible influence path and will be added to C. For specific influence models, we can utilize a refined version of this algorithm: only a limited amount of influence paths needs to be determined, so possibly only a few of the messages in M^* need to be checked. This may lead to lower average cost, but the specific reconstruction cost will depend on the influence model, the properties of the cascade and the representation of M^* . In Section 3.2.1 we discussed the complexity of such algorithm and the rationale for our design decisions. The naive, yet strictest case, considers all orders among users and messages in the prefix, which will result in a cubic complexity. By removing one or both orders, we can check the existence of an edge in constant time and thus obtain quadratic complexity.

From an algorithmic perspective, we develop two variants for computing the influence edges. For the first variant, for every new user who emits a retweet, we iterate over the prefix, which is the follower lists of users who emitted in the past (*prefix iteration*). Then we check if the user belongs to the follower lists of the prefix. The second variant, iterates over the users in the prefix and checks whether they exist in the friend list of the new user (*user iteration*). We systematically explore the design space in section 4.3.4.

In order to investigate the mechanism of influence and avoid exhaustive search to all follower data that drives complexity, we employ different influence assignment models. We have considered the following models for influence assignment [BHMW11, CHBG10]:

- *Least recent influencer*: Users are influenced by the first exposure even if they do not act immediately.
- *Most recent influencer*: Users are influenced by the last exposure.
- *Most followed influencer*: Users with the most followers tend to be more popular and it is assumed that they can trigger more retweets.
- *Most retweeted influencer*: Users whose messages are forwarded the most, emit interesting content and are considered to be authorities, thus exerting more influence on others.

We present an example in Figure 4.6 in order to better understand the aforementioned models and their impact on influence paths. Nodes represent users who propagate the same tweet while the arrow direction indicates the relationship *"is followed by"*. We apply three of the aforementioned influence models and construct the cascade accordingly. For each user there are two values attached: the first one indicates the temporal order of sending retweets and the second the number of followers. The influence paths in each case are highlighted in red. Note here that influence paths in red form a subset of the influence paths in the cascade, since these models trim the redundant influence paths. We can observe that each model (least recent/most recent/most followers influencer) leads to different structural and conceptual results thus exerting a big impact on the selected paths that information/influence flows.

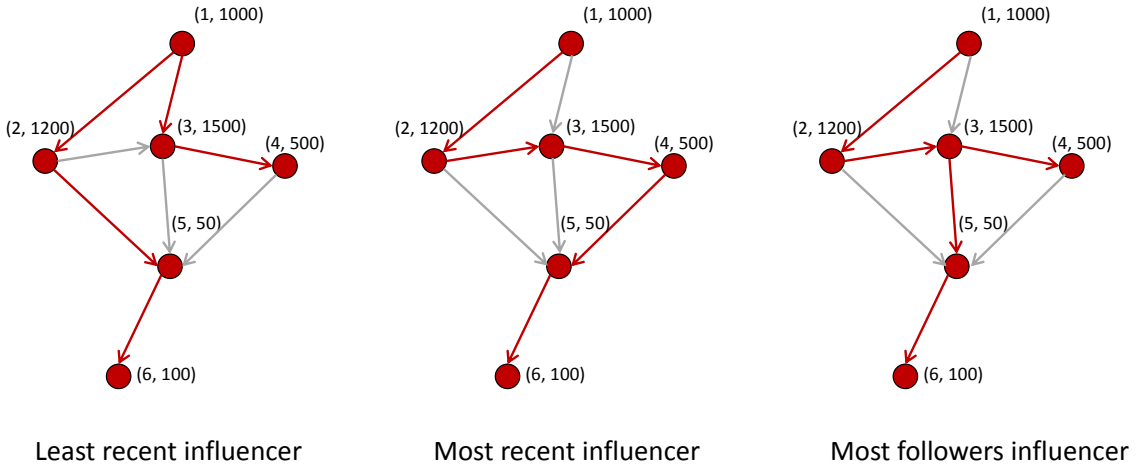


Figure 4.6: Impact of Influence Models

Such influence models do not decrease the uncertainty of the reconstructed influence edges. Instead, they provide some meaning (annotation) when multiple edges exist. In case we desire to keep only one edge for particular types of analysis, we can use one of the influence models to reduce the number or influence edges. In general, uncertainty is not decreased when decreasing the number of edges, because different influence models support different aspects of user behaviour. In general user behaviour cannot be explained by one model for all users at all points in time. When using more personalized models, like the history of past interactions, then the uncertainty of those reconstructed edges could be lower. We leave the development and evaluation of more elaborate models for future work.

When reconstructing information cascades from real data, we encounter either missing messages (nodes) or social graph connections (edges). In turn, this means that we have missing user nodes in C as well as missing influence edges, as these are being derived by the algorithm above. We can never precisely quantify the extend to which the collected datasets are complete. We leverage the ascending message number given by Twitter in a retweet cascades, but this is not always in synchronization. Furthermore, not all influence paths will actually be over explicit social network links. Instead, external influences or overviews on trending topics provide connections that are not captured by our approach. When the algorithm encounters a message that cannot be assigned to a previous influencer, this message becomes the root of a new fragment. As a result, we will not generate a single graph, but multiple fragments that are not connected to the main graph. In case we desire to apply graph metrics over the full reconstructed cascade, we connect those fragments with the cascade root. This means that the fragment roots are connected with the cascade main root forming a connected graph. Assigning the root user as influencer is most plausible hypothesis, since users are influenced by the user of the initial message in the majority of the cases.

In order to implement evaluation in disconnected information cascades and compute metrics on them there are three ways to deal with the problem of disconnected fragments: (1) considering only the large connected component or root component, (2) evaluating the entire forest of all fragments, but not joining them, (3) inferring connections between fragments (nodes). In this case, we need to infer influence edges, which is discussed in the literature [Kos06, ZWSY12], but no scalable methods exist. For our evaluations, we generally chose the second approach. For metrics that require a connected component (e.g. paths) we consider the largest connected component. In the future, we will consider and implement more elaborate models to connect cascade fragments.

For assessing the connectivity of information cascades, we introduced two metrics. Let $C = (U, E)$ be an information cascade graph with U being the nodes and E the edges, u_r the root and M^* a sequence of messages. To evaluate the connectivity of a diffusion graph the two following formulas Connectivity-Rate (CR) and Root-Fragment-Rate (RFR) have been used.

$$\text{CR} = \frac{|\{u|(u', u) \in E \ \vee \ (u, u') \in E\}|}{|U|} \tag{4.1}$$

$$\text{RFR} = \frac{|\{u_j \in U|\text{iff exists a path } u_r, .., u_j \text{ in C}\}|}{|U|} \tag{4.2}$$

The Connectivity-Rate assesses whether there is a connection between two users (nodes) in the cascade. It returns the percentage of users that have at least one connection, and are thus influenced by another user. The Root-Fragment-Rate assess whether there is a path to the root user for every user. It returns the percentage of nodes that are connected with the root directly or via an influence path over multiple users. CR provides a very basic and loose indicator, whereas RFR utilizes a very strict notion. Taken together, they provide sensible bounds for many more advanced metrics.

We implemented those models and algorithms on top of Storm [sto], a scalable, distributed data stream processing platform which provides the necessary low-level primitives for distributed stream processing. Since we need to reconstruct with low latency and high efficiency, there is the requirement to store snapshots of the graph in distributed, main-memory storage components that support the required access patterns. To exploit locality and keep communication cost low, the social graph snapshots should be distributed to the expected computation distribution. Partitioning the social graph snapshots accordingly and investigating suitable systems to store the graphs are discussed in Section 4.3.4.

4.3.3 Analytical Evaluation

In this Section we present the datasets and an analytical evaluation of retweet cascades.

Dataset and Preliminary Analysis Since our focus is to study realtime influence computation thoroughly in a reproducible manner, we still had to record a certain amount of data for evaluations that could be replayed. We used the recording methods from Section 3.4. As a starting point (which we present here), we settled for a dataset that was recorded from August 3rd to September 24th 2012, covering most of the Olympics and the Paralympics 2012. We used the Twitter streaming API to subscribe to the filter terms "Olympics" and "London2012". In total the data set contains almost 11 million tweets, in particular 1.1 million separate retweet cascades - both values are significantly larger than any of datasets studied in the literature [ZBK+10, KLPM10, HTW+12].We performed an initial analysis to understand some of the overall properties of this dataset, encountering a skewed distribution: the largest cascade has more than 60K retweets, around 150 have more than 1K retweets, approximately 5K cascades have more 100 retweets and around 45K cascades contain 10 or more retweets. Twitter includes a *retweet_count* field in every retweeted tweet, so we could compare the number of recorded retweets with the number of reported retweets for every cascade. For most of the cascades we recorded, these numbers showed only minor differences (around 15% on average). For 50% of the cascades we get 90% completeness or more, while for only 15% of the cascades we get completeness less than 80%. That means that our recording policy of tweets through the Twitter API works well. The only major exception happened during the "peak hours" of the Olympics, where the aggregate number of tweets from this subscription exceeded the 1% rate limit of Twitter, and matching tweets were dropped from the system. Our analysis also showed that messages were received in temporal order, so that we can process the message straight away without buffering and sorting.

In order to achieve a good coverage of the social graph, we ensured that the follower information of all the 1.2M users present in these cascades had been retrieved. There were two very distinct subsets of users present: For around 300K users, no follower information was accessible since these users have been blocked by Twitter or made their accounts private. For the remaining users, we fetched their followers, while the number of followers reported in the retweet message at the time of the recording have a high correlation. Since we fetched the followers after the recording of messages the number of users retrieved is slightly greater. This means that users follow more often than unfollow on average.

In this remaining of the section we evaluate our algorithm and models for reconstructing information cascades. The focus of this analysis is on data quality, feasibility, cascade properties and evaluating H1 from Chapter 3. For the real-time reconstruction performance, we present some initial insights: From an execution speed point of view, even a non-optimized implementation of the *complete* influence model finishes the reconstruction of large cascades in a few seconds (e.g. around 4 second for a cascade with 9000 users), if the social graph information is available in main memory on the same machine.

The results presented cover three aspects, using the complete influence model which

considers all influencers. First, we confirm our assumption H1 that influence flows through social connections or in other words that social links are carriers of information. As a second step, we show how the quality of input data affects the reconstruction influence paths. Third, on top of reconstructed cascades we perform an analysis of information cascades and compare it with previous studies on Twitter. As a fourth step, we investigate how different influence models exert an influence on reconstruction of cascades.

Assignment of Influence. First, we evaluate our assumption that information flows through social links. We use a subset of our dataset which contains cascades with more than 100 messages (we call it "full dataset"), due to the fact that the impact of incomplete data in small cascades has more unpredictable results in reconstruction rates. For testing reconstruction rates we used (1) a dataset containing more than 100 messages, i.e. full dataset and (2) a cleaned subset of it. The cleaned dataset contains cascades with more than 90% of their messages acquired and having available more than 80% of follower lists. For the cleaned dataset, we get median connectivity rate 85% and root fragment rate 80%. When we extend our evaluations to the full dataset (that is noisy and incomplete) these rates drop. However, we show that it is possible under data limitations to reconstruct cascades and to obtain meaningful and decent results. For 20% of the cascades we get more than 80% connectivity rate and 70% root fragment rate (Figure 4.7). In ideal cases of message completeness 99% and follower lists 95% we get CR=93% and RFR=90%. As a result, we can conclude that social links are indeed the predominant carriers of information. However, there are still 10% messages that cannot be assigned using social graph information. That means, either those users have no social connections available (corresponding to deleted or private accounts), or they forwarded a message without having a direct link to any of the previous (re)tweeters. It is possible that they forwarded it from the public Timeline where messages of non-followers are also depicted.

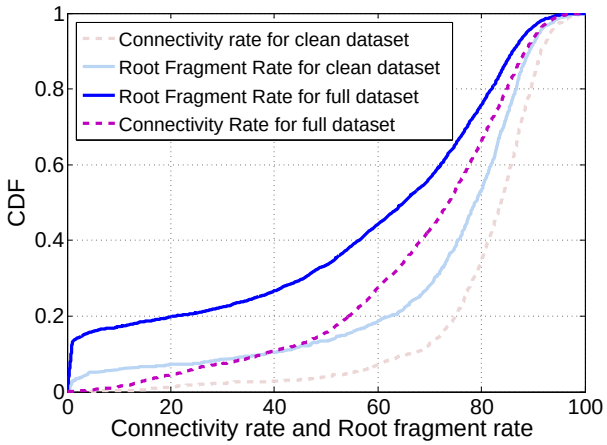


Figure 4.7: Reconstruction rates

Impact of Incomplete Data. Next, we investigate explicitly the impact of different incompleteness parameters on the connectivity rate. Since we target online analysis, either messages are missing or social graph information might be absent or outdated. We take two cascades of size 1000, one star and one with a complex structure, with very good connectivity rates in the presence of full social network data and messages. In order to investigate the impact of incomplete data we removed gradually (1) follower lists, (2) messages. We are presenting these two representative cascades, but results on other cascades are very similar.

Table 4.6: Impact of missing followers - 1000 retweets

		100%	**75%**	**50%**	**25%**	**15%**	**10%**	**5%**
Connectivity Rate	Star Shape	90.9	90.8	90.5	89.8	89.3	88.9	88.4
	Complex Structure	87.7	87.3	85.6	81.7	79.0	76.6	71.0

Table 4.7: Impact of missing messages - 1000 retweets

		100%	**96.8%**	**93.7%**	**87.5%**	**75%**
Connectivity Rate	Star Shape	90.9	88	85.2	79	67.8
	Complex Structure	87.7	80.4	77.1	70.8	58.3

For case (1) shown in Table 4.6, we gradually removed follower lists excluding the root's, keeping the ones with the greater number of followers. Star cascades are expected to undergo lower degradation since most of the users (retweeters) are connected with the root. We can observe that by degrading the follower lists to just 5% of the original data, the connectivity rate drops for the star cascade only 2% and for the complex cascade by 20%. The reason for this is that most users actually don't exert much influence, while multiple diffusion paths compensate for the lost social connections in complex structures.

For case (2), we removed randomly chosen messages in order to investigate the impact on the reconstruction rates. According to Table 4.7 connectivity rate drops significantly when removing random retweets: we encounter a decrease of more than 20% for star cascades and 30% for complex cascade when keeping 75% of messages. That means that connections missing due to absent messages cannot be easily compensated.

Overall, missing messages due to rate limiting results in worse results than missing social graph data. Consequently, retrieval of messages is more important in keeping the cascade connected than social links, which also supports our crawling approach focusing on users with high activity or a large amount of followers.

At this point, comparing with related work for missing data in information diffusion is not possible. While related work [SMLGM11, ZWSY12] develops methods to infer missing data in cascades, those methods are evaluated using *complete* datasets that

are reduced to simulate the missing parts. On the contrary, we do not infer missing data, but we try to quantify their impact in lack of *complete* data.

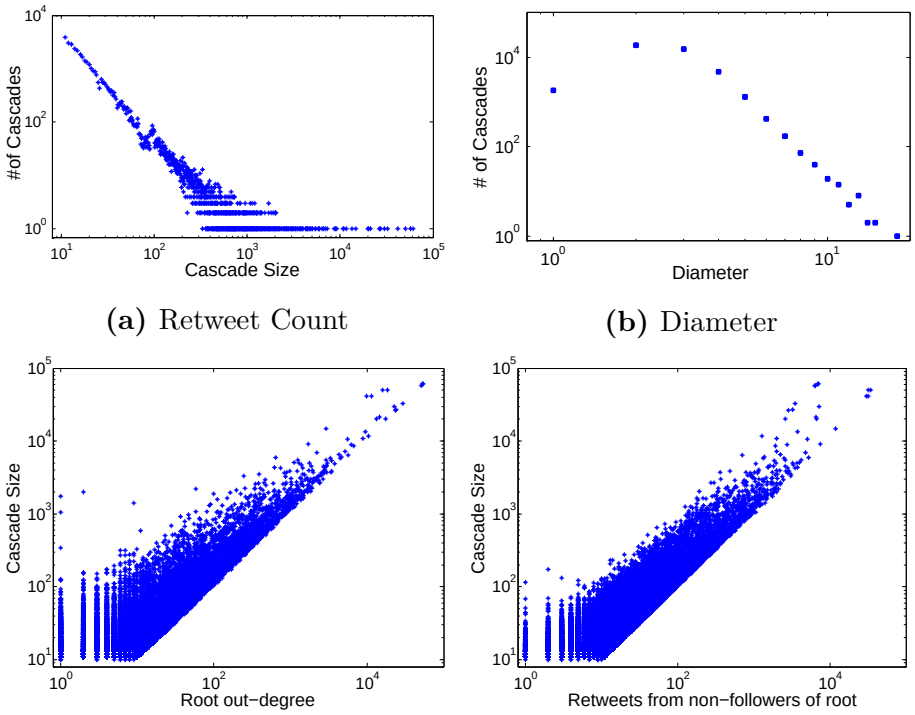
Cascade Properties. A preliminary analysis of properties of information cascades is presented as next. In a first step, we studied basic properties of cascades. We consider cascades with more than 10 retweets since we can find more complex structures on them while small cascades have been already studied [HTW+12, BHMW11]. We can observe a skewed distribution for retweet counts with the biggest cascade containing more than 60K retweets (Figure 4.8a), as discussed. The diameter shows that cascades tend to be deep, with a mean value of diameter 4 (Figure 4.8b), contradicting previous studies even for the big cascades [ZBK+10, KLPM10]. Diameter values up to 18 are observed, indicating that information is being propagated to a large audience much beyond the root's followers. This has an impact on cascades' shapes, that demonstrate complex structures more often than star structures. Another observation is that cascades with diameter 1 are observed with the same frequency as the ones with diameter 5. That confirms again our observations that cascades are more deep than swallow, paving the way for complex analyses.

Since we unravel big and complex cascades with long paths, we study the role of the root node in originating such cascades. Is the root highly influential or cascades tend to be big and deep due to users who forward the tweet of root? Figures 4.8c and 4.8d show that cascade size is correlated both with the direct followers of the root who retweeted (root out-degree) and with the non followers who retweeted. As a result, both the influence of the root and users who forward further the message in combination yield big cascades. On the contrary, there is no correlation (Correlation Coefficient = 0.14) between the size of the cascade and the number of followers of root. Number of followers of a user is not informative of his influence. In this case, our findings are consistent with previous results in [ZBK+10, CHBG10].

Impact of Different Influence Models. Since we get very good reconstruction results in cases of gradually removing influence edges, we concluded and observed that there exist multiple influence paths, hence a large number of possible influences. We study (1) how many of the cascades are actually affected by different influence models, and (2) how cascades metrics and properties are affected, based on a concrete example.

For 10% of the nodes we can observe on average more than 3 influencers, while 20% of the cascades maintain an average number of influencers greater than one. As a result, different influence models described in Section 4.3.2 do matter for 20% of the cascades. This may underestimate the real results, since the datasets are incomplete, in particular possible influence edges may be missing.

Different influence models have a big impact on cascade metrics and properties. First, paths lengths are affected. Since we simplify multiple influence paths to one



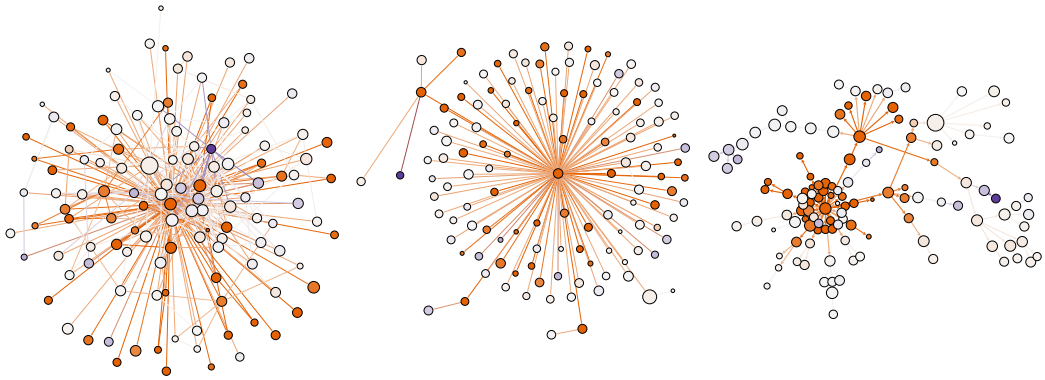
(a) Retweet Count **(b)** Diameter

(c) Size vs Root Out-degree **(d)** Size vs non-followers tweets

Figure 4.8: Cascades' Properties

according to a specific model, the number of edges is decreasing while path lengths are increasing. Also, the temporal distribution of edges changes according to the model: the earlier influencer model produces edges closer to the root's timestamp, while the latest influencer model favors a more stretched distribution of late retweets. In addition, the out-degree of users in the cascade changes according to different model.

In Figure 4.9 three models were used to reconstruct the same cascade with size 124 nodes. In the complete case, all influence paths were considered, while for the earlier and latest influencer models only one edge was selected according to the models in Section 4.3.2. Node colour signifies temporal behaviour: the more red, the smaller the temporal distance from root, the more grey-blue the greater the temporal distance from the root. Node size varies in log scale according to the number of followers a user has, showing how big is the audience that this user can potentially influence. We can observe that the structure and paths change dramatically for different models. For the complete reconstruction and the earliest-retweet model, the diameter is 3 while for the latest-retweet model, the diameter becomes 11. This can be explained by the fact that since we keep the latest influencer on time, we favor longer paths. Moreover, that explains the larger number of grey-blue nodes in the last model (greater temporal distance from root).



(a) Complete **(b)** Earliest **(c)** Latest

Figure 4.9: Impact of different influence models

4.3.4 Realization of Reconstructing Retweet Cascades

Although there is related work on how to model [GHFZ13], reconstruct [CAB+12] and analyze the properties [ZBK+10, HTW+12] of information diffusion in term of cascades', there is almost no research on how to perform such analyses on real-time in the presence of both very fast streams and huge social graphs.

The algorithm in Section 4.3.2, was a single-site, incremental algorithm that can reconstruct information cascades containing retweets, i.e., explicitly forwarded short messages on Twitter. A naive implementation of this algorithm was blocking: this means that all the messages should be collected along with the necessary follower-friend information before the actual reconstruction starts. This is not a viable solution for a real-time scenario. Extended versions of this algorithm are presented here to outline the challenges of reconstruction and possible improvements.

Algorithm Variants. A naive implementation of this idea would have cubic complexity to the size of messages M: $O(M_2)$: For each newly arriving message M_{n+1} we need to check if the user U_{n+1} who created it is connected to any of the users U_1 to U_N who created the previously arriving messages M_1 to M_N, where either the user has K connections or each of these users has J connections. Within the scope of this work we only consider direct connections, but for future research multiple hops may be relevant in order to compensate for missing information. Similar to join processing, we can now exploit indexing and domain knowledge: the order among the prefix or the connections may not matter, so we can treat them as set and get efficient containment checks. This leads to two algorithm variants that were shortly discussed in 4.3.2 are shown in Figure 4.10):

The first variant treats the follower connections of U_1 to U_N as sets and iterates over the prefix of messages seen so far (M_1 to M_N) to check if U_{n+1} is contained in any (or several) of these sets. The complexity is $O(M_2)$, since for every new message/user

we have to check all previous users for connection, which is equal to the number of previous messages.

The second treats the users in the prefix (U_1, ..., U_N) as a set and iterates over the friend connections of U_{n+1} to check if any of them is contained in the set. The complexity of the second variant is $O(M * K)$ where K is the average number of friends. The pseudocode for the two algorithms is presented in 4.3.1 and 4.3.2.

Algorithm 4.3.1 : Prefix Iteration

```
input  : retweets, localEdges, remoteEdges
         // user(retweet): user of retweet
1  for (i = 0 ; i < size(retweets) ; i++) do
2      follower = user(retweets[i]) ;
3      for (j = 0 ; j < i ; j++) do
4          user = user(retweets[j]) ;
5          if (localEdges containsKey user) then
6              if localEdges[user] contains follower then
7                  emit influence edge(user, follower);
8                  continue;
9              end
10         end
11         if remoteEdges containsKey user then
12             if remoteEdges[user] contains follower then
13                 emit influence edge(user, follower);
14             end
15         end
16     end
17 end
```

Algorithm 4.3.2 : User Iteration

input : $retweets$,
$localEdges : user \rightarrow$ set of friends,
$remoteEdges : user \rightarrow$ set of friends

output : $edges(parentId, parentTweet, childId, childTweet)$,
// `user(`*retweet*`)`: user of retweet,
// `time(`*retweet*`)`: timestamp of retweet,
// `retweets(`*retweet*`)`: retweets made by user

```
1  for (i = 0 ; i < size(retweets) ; i++) do
2      currentTweet = retweets[i] ;
3      user = user(currentTweet) ;
4      if (localEdges containsKey user) then
5          friends = localEdges(user)
6      else
7          friends = remoteEdges(user)
8      end
9      foreach friends as friend do
10         friendTweets = retweets(friend) // usually only one tweet foreach
               friendTweets as friendTweet do
11             if (time(friendTweet) < time(currentTweet)) then
12                 emit edge(friend, friendTweet, user, currentTweet)
13             end
14         end
15     end
16 end
```

Clearly, the former works better for small cascades with heavily popular users, while the latter works best with large cascades of not-so-well-connected users. Variants of these algorithm can switch between these iteration types according to the relative size of each of these sets and the iteration targets, since both the cascade sizes and the connection set sizes are heavily skewed in practice. This computation provides incremental results (as each newly arriving message can be handled individually) at high speeds. In Section 4.3.3 we used prefix iteration and observed that the complete reconstruction of all influence paths in a cascade with around 10K messages is completed in less than 5 seconds.

Challenges. Since we want to construct cascades on real time on all possible users we need to address the data scale and rapid rates. In periods of heavy traffic, hundreds of thousands of cascades are taking place in parallel, consisting of up to hundred of thousands or millions of retweets. Complementary, the social graph consists of hundreds of millions of users with dozens of billions of edges; clearly querying such a graph for real-time computations creates a considerable overhead,

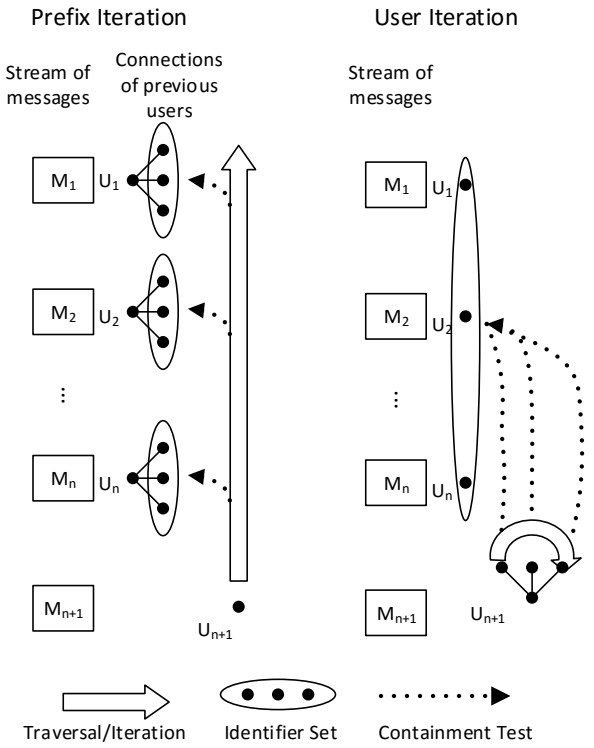


Figure 4.10: Reconstruction with Prefix and User Iteration

as its raw size by far exceeds the RAM available to commodity systems.

In turn, the main bottleneck of these algorithms is the size of iterations and the access latency to the sets. For the latter, we observed that even in the centralized algorithm it is massively affected by increased access latency. Switching from a hash- based approach to a sorted-list-based approach (in order to save space) gave slowdowns by more than an order of magnitude. In turn, a first attempt of distributing only the data by utilizing a request-response based approach to the social graph adjacency lists led to a slowdown by three orders of magnitude.

According to our observations, when computing the influences edges, we encounter very low selectivity in the connection sets: the average degree of a node in the cascade is slightly above one, yet there can be massive skew (such as the outdegree of a very popular user). The repeated access to connection sets at the beginning of a prefix might lend itself to bulk transfer of the entire set, but this skew might also be very costly. In other words, we need to repetitively probe possibly large sets, while the return set is very small. Due to the number of accesses a typical request/response approach will incur significant additional latency, while transferring full adjacency lists can be rarely amortized.

There is not much work that combines classical streams with large graphs; typical graph processing systems like GraphLab [LBG+12] are efficient in iterative graph computations, but do not cater for streams. Complementary, such a problem is not

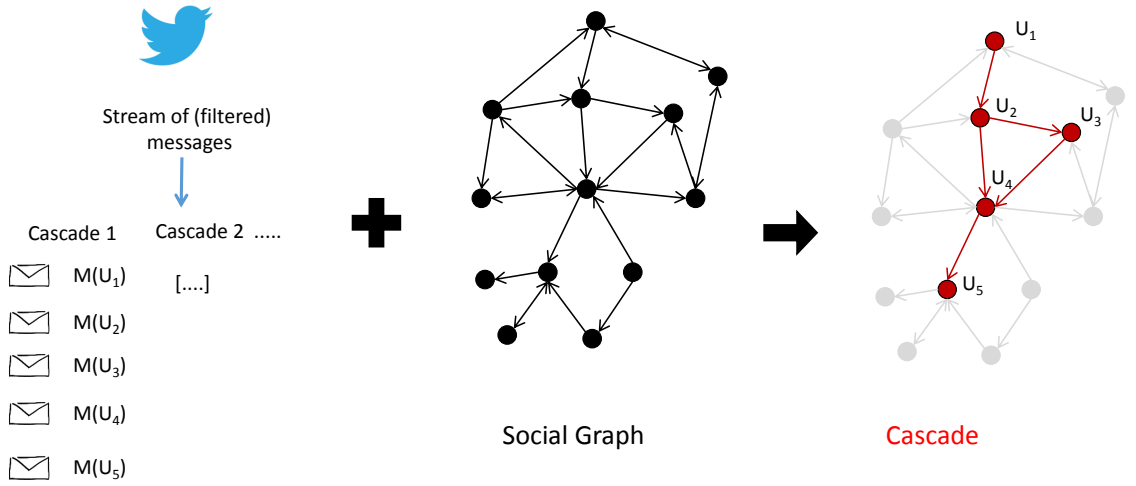


Figure 4.11: Streaming Influence Inference

a classical stream processing problem where a single pass over the data is desired: it is an iterative stream processing problem where we need repetitive access to the stream prefix (set of messages or user connections). The only known approach is Naiad [MMI+13] which allows for iterative computation over data streams using timed dataflows. As such, Naiad provides strong coordination means, something that is one of the next issues to tackle. To the best of our knowledge, it does not provide the fine-grained control over algorithm and state distribution we are trying to achieve here.

4.3.5 Distributed Processing

Given these challenges, we aim for a distributed approach that minimizes the access latency on the social graph, and keeps the computational cost and number of messages to transfer low. A streaming solution is depicted in Figure 4.11: for every new message that arrives its author is checked for social connection with the prefix.

We follow a number of design ideas to express this distribution efficiently:

- The first is to partition the set of adjacency lists in a non-overlapping way to achieve a good scalability in terms of the massive graph sizes.
- The second is to perform all iterative accesses to an adjacency list and checking of set containment locally to avoid the latency penalties of such accesses. Taken together, these two ideas mean that we may need to distribute the iteration over the messages over multiple nodes.

To reduce the latency and amount of data to transfer when processing different messages at different nodes, we take in turn three steps: a) We collect all results (and perform the related computation) at the site of the cascade starter, minimizing distributed operations altogether. b) We combine the low selectivity of lookups with

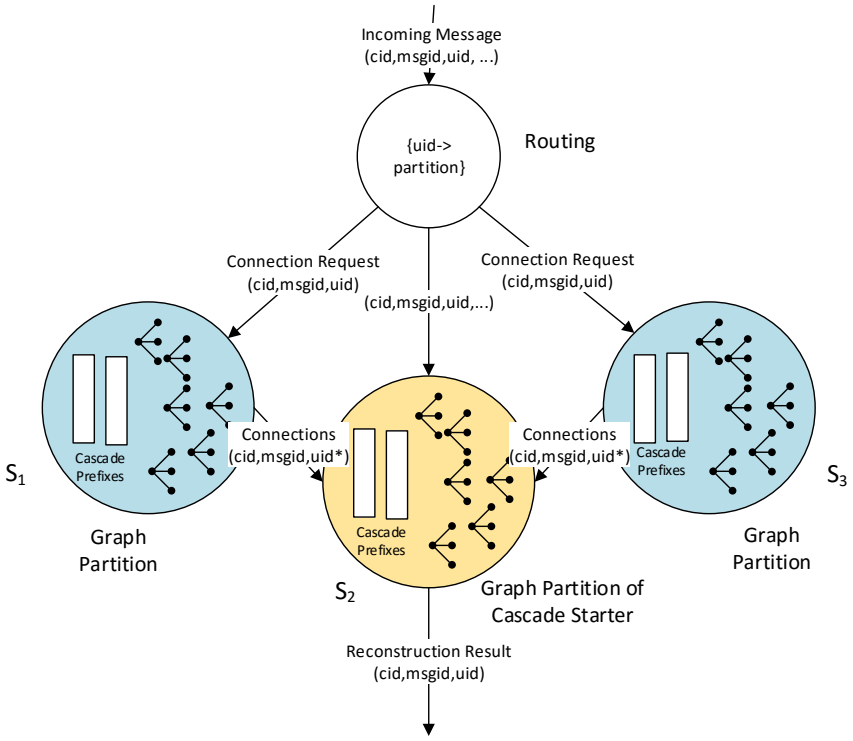


Figure 4.12: Distribution of Graph Data and Processing

an adapted partitioning strategy to minimize the amount of result data and the time to transfer. c) We utilize stateful processing and parallel sending to reduce the size and time of messages' requests at the different nodes. d) Lastly, we perform parallel processing to reduce the overall response times.

For synchronization purposes, we consider that the following assumptions hold:

- Messages are arriving in temporal order in the system.
- No message gets lost.

Without these two assumptions, synchronization would become very costly. If we make the assumption that messages might be unordered or that messages get lost, the system has to wait until any lost or delayed messages are processed. This is problematic for incremental and real-time analysis because partial results cannot be produced with the uncertainty that messages are missing.

Architecture. All theses ideas translate into the following architecture that builds on the infrastructure we developed for the centralized algorithm, as shown in Figure 4.12. We assign the adjacency lists of the social graph to the processing nodes according to the graph partitioning method chosen; we centrally retain information on the adjacency lists locations of this assignment. Utilizing the cascade in Figure 4.11, let us (as an example) assign user U_2 to S_1, U_1 as well as U_5 to S_2 $U3$ as well as U_4 to S_3.

At any point in time, particular partitions are activated, specifically those that contain the adjacency lists of the users that have been seen so far. For newly arriving information in the cascade, we send a triple of identifiers (cascade, message, user) in parallel to the relevant partitions to perform partial computation. This partial computation provides the part of the connections for this user that can be derived at this partition. To gain a full reconstruction of the cascade (containing all edges in the correct order), the partial results are combined at the partition containing the user who started the cascade. This decision is driven by the insight in Section 4.3.2 that such users often have significant influence (and thus many outgoing edges) in the cascade.

The partial prefix information and the computation at each site as well as the information to send to a site depend on the iteration of the reconstruction algorithm: For prefix iteration, each partition will be used to iterate over the intersection of the users in the cascade so far and the user set present at the site. Using the running example cascade and the allocation described before, the connections of message $M(U_5)$ will have to be checked against site S_1,S_2 and S_3, since each of these sites contains users from the prefix of the cascade (U_2, U_1 and U_3/U_4, respectively). In order to avoid sending the (growing) prefix to the affected sites over and over again when probing, we keep the partial prefix (intersection) for each active cascade at each site (e.g., U_3/U_4 at S_3). Generating this prefix can be piggy-backed with connection checking: when we probe for the connectedness of a user, this user is added to the prefix at the site where its own connections reside. This way, messages of (small) constant size are sent to the subset of set of sites that contain users in the prefix of the cascade. For user iteration, all potential connections of a new message can be found at a single site, yet the full set of users active in the cascade needs to be accessed (see Figure 4.12, right). We therefore collect the complete prefix at each site involved in the cascade, and transfer the existing prefix in bulk when a so -far inactive- site is accessed.

In both cases, each partition keeps a partial state of the computation so that the cascade can be processed in parallel. Overall, both approaches will converge to sending all messages to all active sites. Since these requests are performed in parallel to regular processing, they do not significantly affect the overall latency. The selectivity of these requests is very low, resulting in significantly lower response counts (and sizes), providing even less of a bottleneck. We synchronize the reassembly (including reordering and result generation) at the end of a cascade. However, we have developed batching strategies for a finer-grained synchronization, that we are going to explain in the following.

Batching and Synchronization of Remote Requests. The batching of remote requests will increase the performance and reduce the communication costs, since instead of sending one message for every remote request/response, now we can group multiple remote requests/responses in one message.

Batching can be applied a) per cascade or b) globally.

Batching messages does not differ much from reconstructing individual messages. First, the size of the block B for each cascade has to be defined; once B remote requests have been collected, they are being sent to the remote nodes. The remote requests are being sent by one message containing the root node and the user ids requested. Once the remote requests are received, each remote node iterates over the list of user ids and perform the regular check for set containment to identify possible connections. The remote response is being sent in the form of "child id-parent id". Once received by the root node, the reconstructed edges (child id-parent id) are being inserted in the data structure for remote edges. Note here that batching of messages is an orthogonal aspect to batching of remote requests. The batches of messages we call them *blocks* to differentiate them from the batches of remote requests. In general it is intuitive that these two are applied at the same time. For example, the system waits until receiving the full block, then sends all remote requests in one batch at once and starts the reconstruction. Global batching incites more complex communication, since now a request or response is not connected with a particular cascade. Now the batch B refers to the number of remote requests issued by the system independent from cascade grouping. As soon as B remote requests have been collected, the system emits a global remote request to all nodes. This request contains lists with the root node-ids (which is different for every cascade), the user-ids to be checked and the cascade-ids. Again each node iterates over the lists including the user ids. If the user-id is found, the remote response includes the child-id the parent-id, the cascade-id and node-id. It also includes information about the number of answered response requests in order to keep track of the reconstruction process. Once the corresponding root node receives the requested edges (child-id and parent-id) it adds them to the data structure containing the remote edges.

Social Graph Partitioning. Despite these optimizations in processing the cost, we assume that a suitable partitioning would still make a significant difference in the overall performance Since the main bottleneck of the algorithm is the computation of social connections, a proper partition for faster access will possibly increase the performance. Our problem differs from both the traditional graph partitioning as well as social graph partitioning for high variance in out-degree [GLG+12]: 1) The graph partition needs to exploit skew to ensure that interactions of users remain local. 2) The size of a partition is not determined by the number of nodes or intra-partition edges, but by the overall edges outgoing from users.

We aim to solve this problem by using alternative Hypotheses outlined in Chapter 3 for partitioning the social graph. In general we cannot provide a solution that satisfies all conditions (skew and balanced partitioning), and we will test those competing hypotheses.

We will use a key observation and derived hypothesis to perform the partitioning: Past interactions [HRW08] (also past influence) are an important factor to assess

the strength of links in the social graph: Users who interacted in the past are more likely to interact in the future. To exploit this idea, we have partitioned the interaction graph of users [GL12] which reveals who interacted with whom, and use this partitioning to allocate the social graph. In our workload, these graphs are actually a byproduct of computing influence, so we can easily collect the results and learn over time. This corresponds to the Hypothesis H4-I from Chapter 3 . For comparison, we will also partition the social graph of users. This is very useful in cases that we do not have access to past user interactions.

The alternative Hypothesis (H4-II) considers a balanced partitioning, that exploits better parallelism. Note that the size of each partitions is driven by the size of adjacency lists (edges out/in-going from every node). For that, we place every new user to the partition that contains the fewer edges (followers or friends). We call these methods *greedy-followers* and *greedy-friends* respectively. We also provide a pure random partitioning approach for comparison purposes.

4.3.6 System Evaluation

In this section the evaluation is presented for distributed reconstruction, which highlights different aspects of the design decisions and optimizations. First, we describe the datasets used, splitting according to cascade properties for a better understanding of the system's behaviour. Next, we compare the single site reconstruction with our distributed solution and the two variants of the algorithm: prefix vs user iteration. We also investigate the impact of batch size for remote requests for batching per cascade and global batching. Lastly, we compare different graph partitioning techniques discussed in Chapter 3.

Dataset. We use the same dataset from Section 4.3.3 but here we evaluate our methods in different classes of cascades. The goal is to include a variety of options from small, large and mixed cascades to highlight different aspects of the system. Our datasets are the following:

- Large cascades: sizes *1K-30K*
- *Mixed* cascades: taken from three sets containing 10, 100 and 514 cascades with more than 60K retweets.
- Small-Medium cascades: sizes *6-100*
- All Cascades: *bigger4* with at least size of 5.

In Table 4.8, we presents statistics of these datasets.

Centralized and Distributed Reconstruction: Specifications and Metrics. For the centralized case, the reconstruction was performed using an Amazon EC2 instance r3.8xlarge which has 244 GB of main memory. This was particularly necessary

Table 4.8: Cascade classes Statistics

Dataset	Cascades	Retweets	Average retweets	Influence edges
mixed	625	242.266	387.6	1,210,290
1k-30k	350	918.795	2,625.1	2,762,476
6-100	74.569	1,268,607	17.0	962.852
bigger4	99.561	3,683,332	37.0	7,286,620

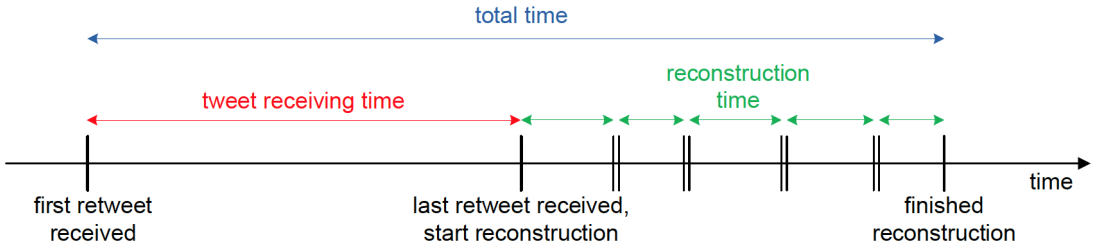


Figure 4.13: Distributed blocking reconstruction times

for the bigger4 set. Only the relevant parts of the social graph were loaded in main memory.

The distributed reconstruction was performed in a cluster of ten nodes, each of them consisting of an Intel Xeon E5-2420 with twelve cores and 32 GB of main memory. The nodes are connected via 1GBit links. Apache Storm was used in version 1.0.1. Again, the nodes load only those parts of the social graph that was needed for the reconstruction of the dataset.

In the experiments that are presented in the next sections we measure the time in seconds and the reconstruction rate. The reconstruction rate shows the number of retweets processed in the time unit (seconds).

$$ReconstructionRate = \frac{|retweets|}{totaltime} \tag{4.3}$$

The total time for the centralized case is depicted in Figure 4.13: first we wait until all the messages are there (in our experiments they are streamed to the system without waiting for the real time lags between messages) and then the reconstruction takes place. The total time includes both processes.

For the distributed blocking reconstruction now the total time includes the time to send and receive the remote requests/responses (which actually corresponds to the tweet receiving time) and the actual reconstruction time. The different time measurements are presented in Figure 4.16.

Lastly, for the distributed incremental reconstruction, the time for remote requests/responses is interleaved with the (partial) reconstruction. Again the total time,

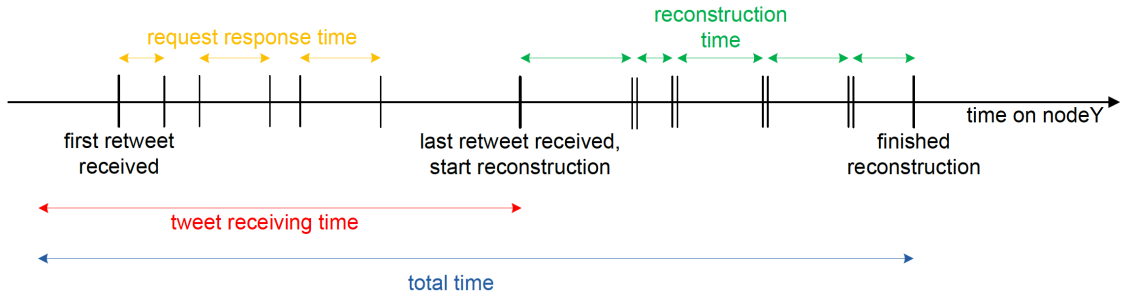


Figure 4.14: Distributed blocking reconstruction times

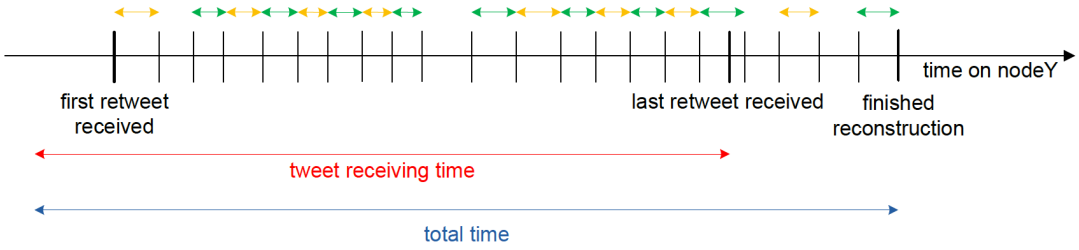


Figure 4.15: Distributed blocking reconstruction times

is the sum of all times from the first tweet that is received until the last cascade is reconstructed. Times are presented in Figure 4.15

The results produced in both cases (centralized and distributed) were compared for consistency: the cascades produced are identical which means that the same influence edges are outputted by both variants.

User Iteration vs Prefix Iteration. The results produced in both cases (centralized and distributed) were compared for consistency: the cascades produced are identical which means that the same influence edges are outputted by both variants.

Next, we compare the results for the two variants of the algorithm described in Section 4.3.4. Here, a centralized set-up is preferable over a distributed, because we want to measure purely the performance of the algorithms.

The reconstruction rate of both algorithms is presented in Table 4.9 for the different classes of datasets:

We observe that the user iteration outperforms prefix iteration in almost all of the cases. Only for small cascades (6-100) prefix iteration outperforms. The reasons for this behaviour are the following: According to a study during 2012, when the Olympics dataset unfolded, the average number of followers is 208 and the average number of friends is 102 respectively [Uda12]. This means that first friend lists are half the size of follower lists, benefiting user iteration which iterates over friends lists. Most importantly the size of the cascade is crucial for determining which

Table 4.9: Centralized blocking reconstruction for Prefix vs User iteration

Dataset	Prefix Iter.	Reconst. rate	User Iter.	Reconst. Rate
mixed	618.4	391	5.4	44.864
1k-30k	729.9	1.258	17.8	51.617
6-100	17.7	71.672	24.6	51.569
bigger4	3,368.7	1.093	85.1	52.619

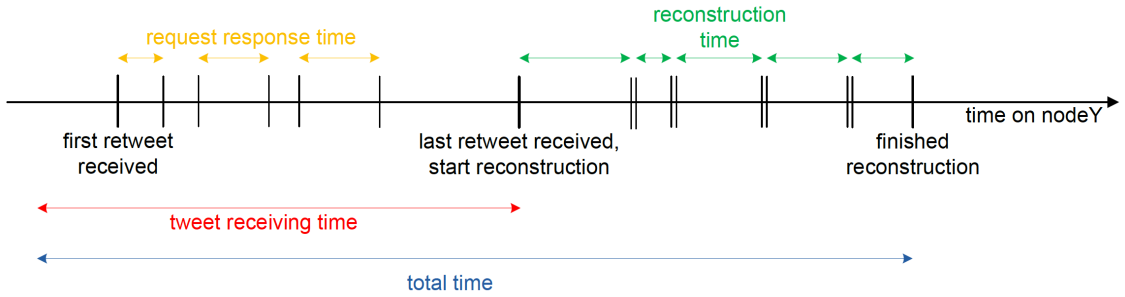


Figure 4.16: Distributed blocking reconstruction times

algorithm performs the best. For large cascades, where the prefix is larger than the (average) friend list, is cheaper to iterate over the users' friends lists. In this case, the set containment tests that need to be performed are as large as the friend list of the incoming user (see Table 4.9). On the contrary, for smaller cascades (than the average friend lists), the set containment tests are equal to the size of the prefix (prefix users' followers lists). This is the reason that the prefix iteration outperforms only for the small cascades, up to 100 retweets, which is also equal to the number of the average friend list. Given these numbers we will use the user iteration algorithm for the rest of the experiments.

Distributed Reconstruction. In this Section, we present results of the different aspects of distributed reconstruction. For the distribution of the social graph we use a random partition for comparison purposes. The impact of different partitioning approaches are going to be discussed in Section 4.3.6. First, the effect of the distribution is investigated and the individual times are analyzed. For that the blocking variant without batching is evaluated. The times are shown in Table 4.10. The results are reasonable and will be used as comparison for the next experiments. The numbers are much lower than the rate at which cascades are unfolding in reality.

Next we investigate the behaviour of incremental reconstruction, which results in making partial results available before receiving all cascade messages. We investigate different block sizes (i.e. message batch sizes) and we try to find a trade off of system throughput and availability of results. Figure 4.17 presents the total time taken to reconstruct with different block sizes.

Table 4.10: Computation of different times

Dataset	Total time	Reconst. time	Request response time	Tweet receiving time	Reconst. rate
mixed	8.5	2.1	5.9	6.5	28.501
1k-30k	26.2	5.9	20.1	23.4	35.068
6-100	34.3	6.6	28.4	28.2	44.986
bigger4	102.0	17.7	78.4	82.5	36.111

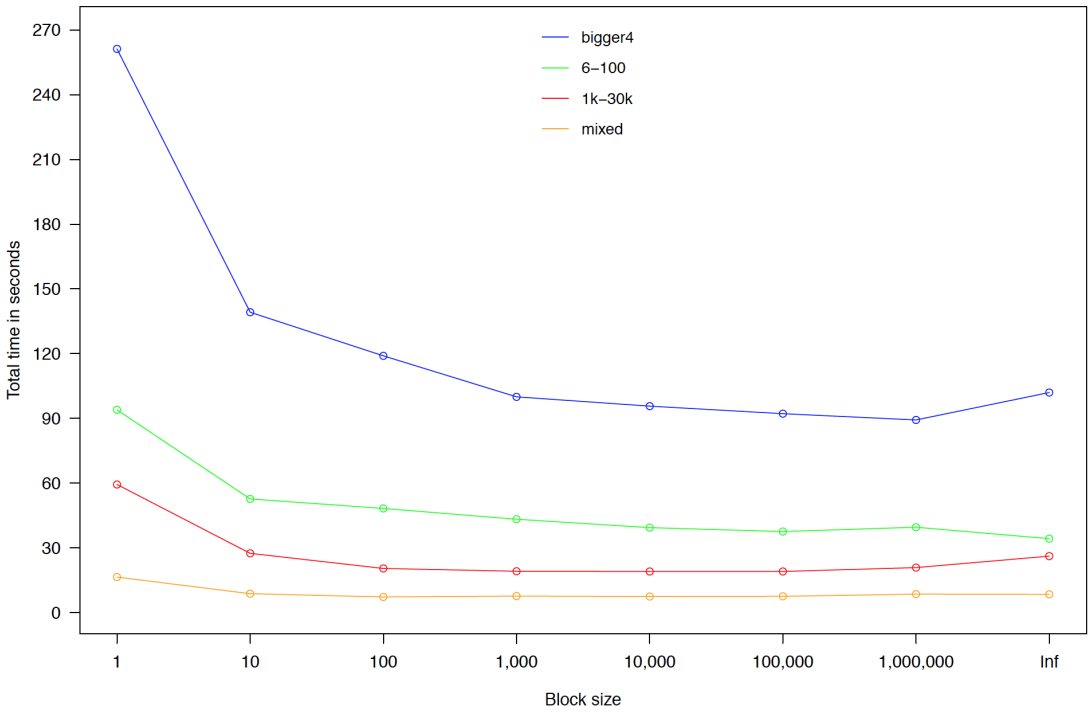


Figure 4.17: Distributed incremental reconstruction with different block sizes

We observe that even for a block size of 10, we have a cost reduction by 40% approximately. After size 100 and 1,000 for the bigger4, the gains are not significant.

Next we investigate the impact of batching the remote requests and responses. Figure 4.18 shows the total time taken by the system to reconstruct with different batch sizes (remote requests/responses), where the batch size is equal to the block size. Here, the batch size is computer per cascade.

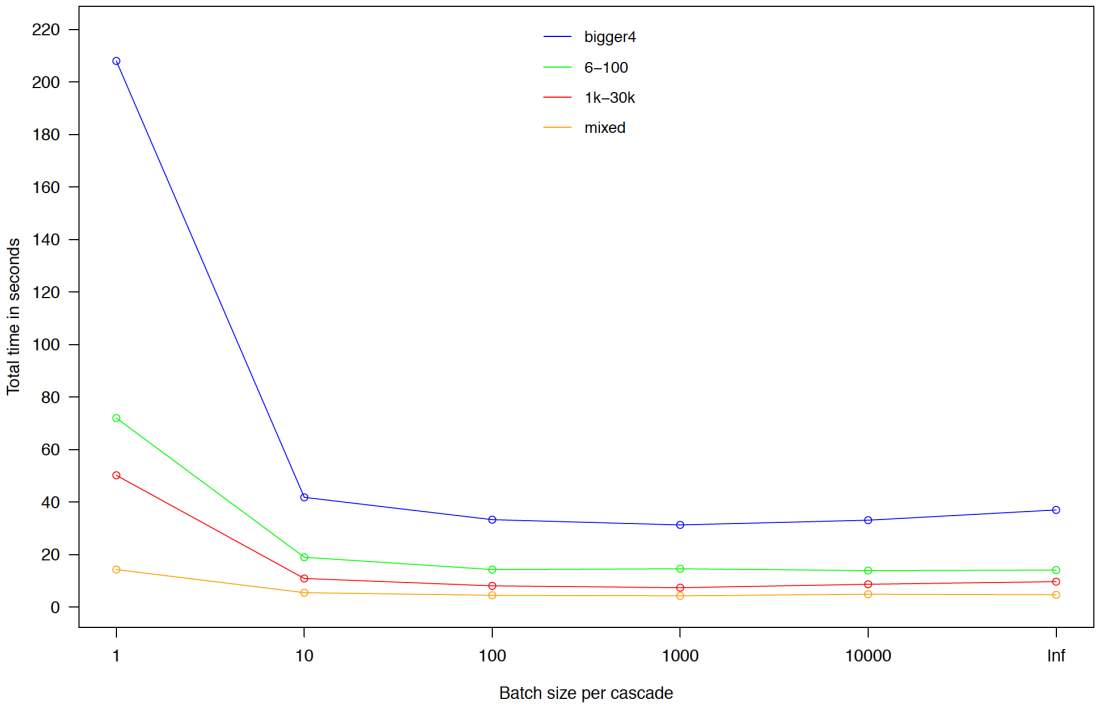


Figure 4.18: Distributed incremental reconstruction with different block sizes

Again, even with size of batch 10, we observe considerable reduction of the processing time. Now, the reduction is 80% compared to the batch of size one, for the largest dataset (biggest4).

Figure 4.19 is similar to 4.18 but now a global model is used. The batching is implemented on the stream model which contains interleaved cascades. Here larger batches can be evaluated, since in the batch per cascade model, the maximum was bounded to the cascade size. We observe that the best throughput is with 10K batch size and the overhead is increased again. This happens because with the increase of the batches, the processing time of the requests increases and after 10K the capabilities of the system are stressed by single (large) messages containing remote requests/responses. Also, we have to pay the more complicated synchronization compared to Figure 4.18, as highlighted in Section 4.3.5.

These experiments show that our system can keep up with the current update rates of Twitter (6,000 tweets per second) and not all of them are retweets. Batching brings

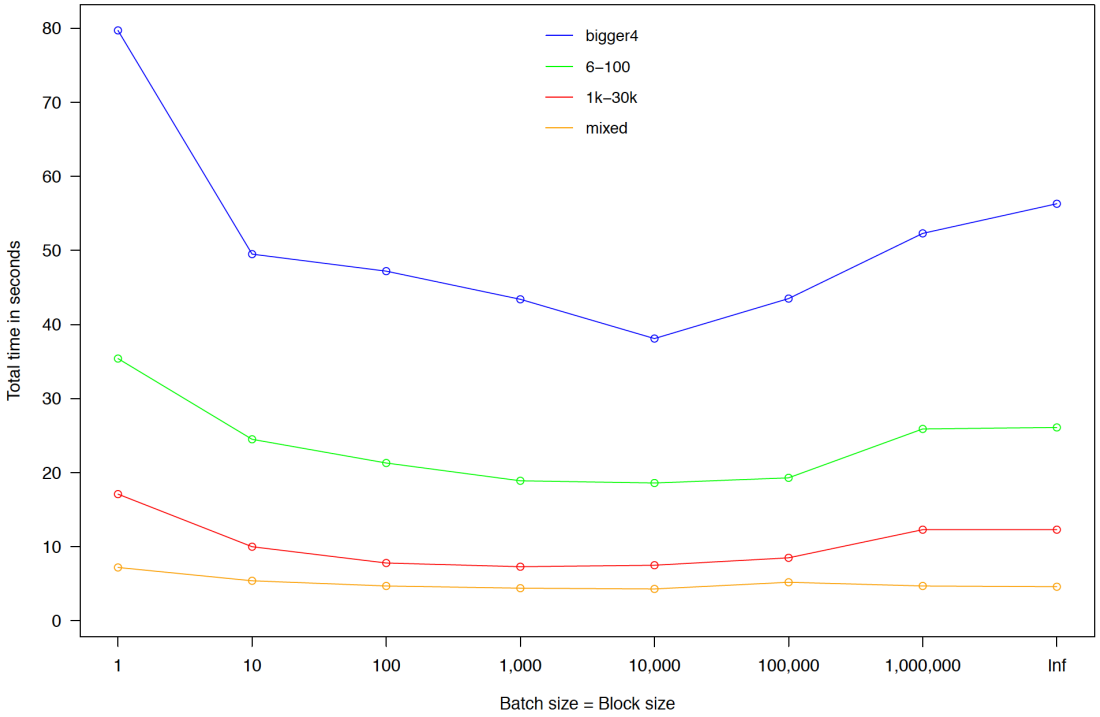


Figure 4.19: Distributed incremental reconstruction with different block sizes

a significant benefit, since the system has to handle less remote requests/responses. Next, we will discuss the effects of different partitioning strategies.

Social Graph Partitioning. Apart from maximizing the performance of the system itself, we investigate into social graph positioning techniques. Since social graph lookups is the main operation to reconstruct cascades, their partitioning plays an important role in the computational times. In Section 4.3.5, we discussed several hypotheses how to best implement that.

During this work we implemented the following partitioning strategies:

- *min cut social graph*: The social graph is partitioned with a min cut strategy using the METIS framework [met13].
- *min cut interaction graph*: As previous, the interaction graph is partitioned.
- *random*: Nodes are placed randomly to the nodes.
- *greedy followers*: Every new user is placed to the partition that contains the fewer edges by considering the follower graph.
- *greedy friends*: As previous, considering now the friends graph.

The total time of reconstruction is presented under different partitioning techniques in Figure 4.20. In general, the different datasets are in line with which strategies

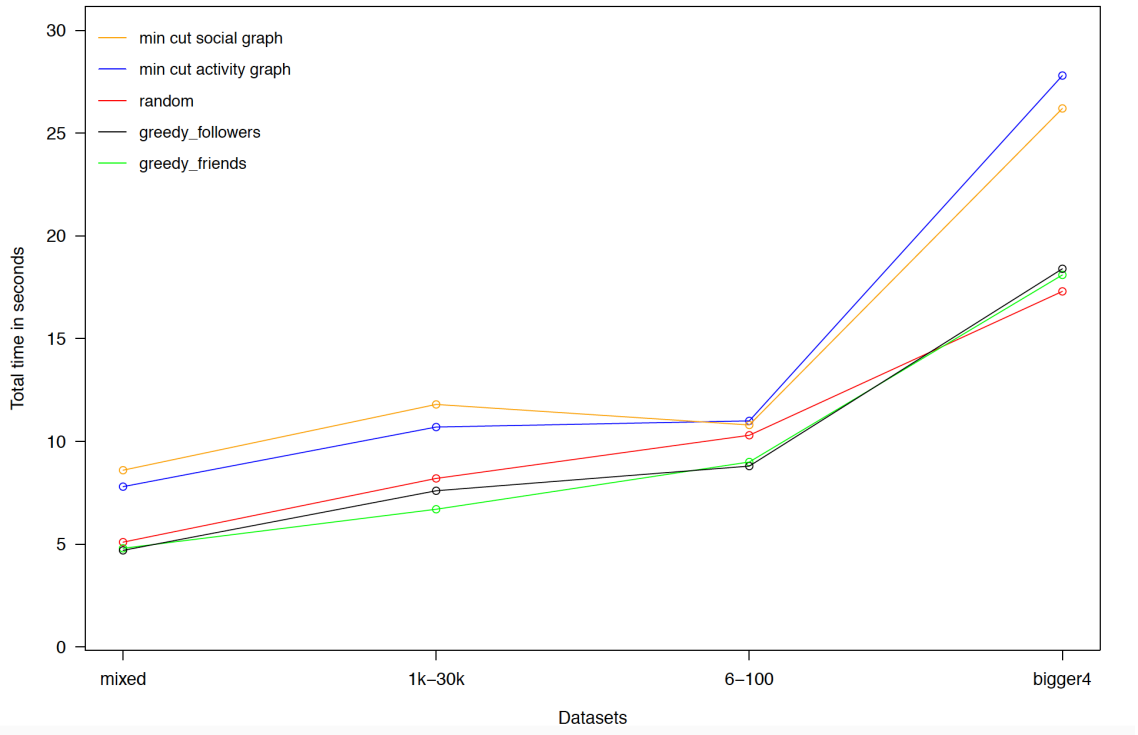


Figure 4.20: Partitioning Strategies

performs the best. Overall, the greedy followers and fiends perform the best, meaning that trying to balance the partitions brings the best results. On the other side, trying to keep the community structure by the min cut techniques, creates overly un-balanced clusters resulting in more computational time. In general, the largest part of the cascade is reconstructed in the root node, and remote requests deliver around 1.5 - 12% of the size. This means that the largest part of the cascade is in any case local. Maximizing further the locality in the detriment of balanced partitions does not bring better results.

4.4 Conclusion

This Section presented the methods and a fully working system for real-time analysis and evaluation of information diffusion. We introduce models and methods to reconstruct information cascades over real-life Twitter data. We focus on retweets in order to highlight the aspects of computing influence when the source is provided. We showed that such a reconstruction is feasible and social links play predominant role in information diffusion. Noisy data does have an impact, but we understand which aspects are most critical. An in depth analysis of information cascades is provided which facilitates the understanding of information diffusion processes. In order to tackle the challenges of real-time computations we have leveraged tech-

niques from streaming and distributed systems, combined with computations over huge graphs. Our system can keep up with the current social media streams and provide influence computations on real-time.

As a note here, we compared Naiad [MMI+13] system for reconstructing cascades [Mey17], which takes an iterative approach in computations (vs the broadcast we analyzed in the previous Sections). The broadcast approach performed better overall, however the pure performance for influence computation was better for the iterative approach with Naiad. The disadvantage of this approach was that the synchronization was implemented by the timestamp and a global synchronization was required. The approach iterates over batches and as a result it worked best for large batches.

In the future, we plan to cover a much broader range of datasets and extend our evaluations to other kinds of information propagation other than retweets. For example, computation of hashtags would leverage the same methods, but we need to overcome the challenge of the lack of an obvious root. Also, we plan to build models in order to infer missing messages and social links. In general, a deeper understanding over missing data is needed by investigating the effect of external influence and social media user behaviour. We provide such insights in Chapter 5 where we discuss more fine grained influence computation.

Chapter 5

Implicit User Interactions

5.1 Introduction

As highlighted in Chapter 1, tracing the provenance of information in a timely way is as challenging as it is crucial, given the information speed, the large audiences it reaches, and the multiple sources that might have produced it: anybody can write anything, without it being verified. The knowledge of information diffusion processes – including the sources, the intermediate forwarders, and the modifications that this piece of information has undergone on the way – provides valuable context to assess its relevance, validity, and trust. In this chapter we focus on implicit interactions, where we need to unravel latent influence.

The first challenge is identifying influence when no explicit information is available. This means in theory that all previously written messages might be relevant. For that, we leverage several hypotheses in order to limit the search space (e.g. similarity, social proximity, limited user attention span, additional user interactions). The second challenge lies in evaluating latent influence without any information from social media providers and absence of ground truth. The third challenge is identifying such influence on an online fashion and keeping up with the fast update rates.

The contributions of this work are the following:

1. We implement methods to unravel and evaluate the fine-grained provenance of implicit interactions in social media, when no information from social media providers is given. By doing so, we provide a deep analysis over human behaviour patterns in information propagation.We show that by considering only explicit means (from social media providers) a large share of implicit interactions and influence is ignored.
2. We provide web-scale, incremental provenance reconstruction of information diffusion to accommodate streaming use cases, such as fast and accurate online journalism. We identify the trade-offs of performance and result quality, and

leverage assumptions including the limited user attention span to limit the space complexity. Our results show that it is possible to reconstruct provenance of messages not captured by social media providers at scale and speed without compromising in result quality.

We evaluate two different datasets to show the applicability of our methods to different scenarios: First, we use Twitter data which consists of very large update rates but with short (and noisy) text and news articles from an aggregation that has lower rates but longer texts. Our methods perform well for both datasets and can keep with the update rates.

The publications related with this Chapter are the following:

[TLF+17] Io Taxidou, Sven Lieber, Peter M Fischer, Tom De Nies, and Ruben Verborgh. Webscale provenance reconstruction of implicit information diffusion on social media. In *Distributed and Parallel Databases*, pages 1-33, 2017.

[TFDN+16] Io Taxidou, Peter M Fischer, Tom De Nies, Erik Mannens, and Rik Van de Walle. Information diffusion and provenance of interactions in twitter: is it only about retweets?. In *Proceedings of the 25th International Conference Companion on World Wide Web, pages 113-114. International World Wide Web Conferences Steering Committee*, 2016.

[TDNF17] Io Taxidou, Tom De Nies, and Peter M Fischer. Provenance of Explicit and Implicit Interactions on Social Media with W3C PROV-DM. In *Lecture Notes in Computer Science, Springer LNCS/LNAI series*, 2017. Accepted, under publication procedure.

5.2 Methodology

We describe our methodology in three parts: first, we describe our approach for similarity-based provenance reconstruction in Section 5.2.1. Second, we further investigate latent influence among users and additional indicators that strengthen the reconstructed provenance Section 5.2.2. Last, the technical contributions follow that lead to the system's scale and speed follow in Section 5.3. Note that although we focus on Twitter here, our method can be applied to any type of text-based social and news media. We validate that by using a news dataset which differs significantly from Twitter data.

5.2.1 From Similarity to Provenance

Here, our main underlying hypothesis is that *"if two messages are highly similar, there is a high probability that they share some provenance"* (from H2-I in Section 3). More specifically: *"the higher the similarity between two messages, the lower*

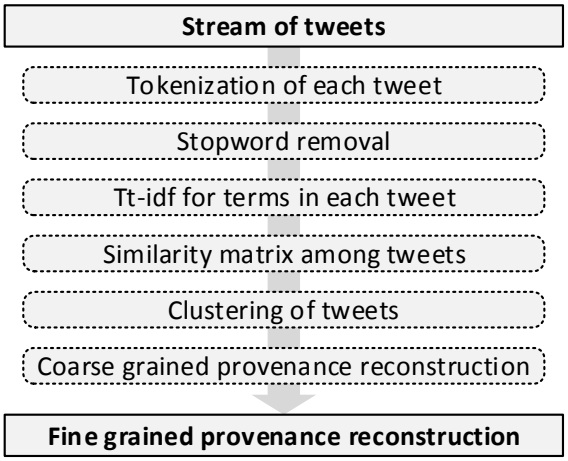


Figure 5.1: Flow of Provenance Reconstruction

the uncertainty of the provenance they share" (from H2-II in Chapter 3). To test these hypotheses, we proposed a provenance reconstruction algorithm, as shown in Figure 5.1.

First, we determine a subset of messages which we want to consider in scope for our provenance reconstruction (e.g., the stream of messages for a conference). Then, we compute similarity between all the messages in this subset. To group similar messages – which we assume share some provenance (H2) – we rely on clustering. The procedure to cluster messages and reveal their provenance is the following. First we tokenize the text of the messages and we remove stop-words. We index the messages using a feature model (e.g., Tf-Idf by [SM86]) and semantic similarity function Sim (e.g., cosine similarity), and we compute the similarity matrix of all messages which are in scope. Next, we apply the similarity-based clustering algorithm SimClus [AHSZ11] to divide the messages into (possibly overlapping) clusters of messages that share some similarity higher than a predetermined threshold T_S. This threshold is dependent on the content type, and should be empirically determined. We provide further discussion on selecting the lower bound of similarity in Section 5.4.1.

In short, the SimClus algorithm works as follows: first, the *coverset (cs)* is determined for each element of the dataset. The $cs(x)$ of a document x includes all documents y for which $Sim(x, y) \geq T_S$. The *uncovered coverset* $ucs(x)$ of a document x then includes all documents $\in cs(x)$ which are not part of a cluster yet.

Next, the document with the largest *ucs* is chosen as the first cluster center. All elements of its *coverset* are removed from the other *uncovered* coversets, and the center is marked to no longer be considered in further iterations. This process is repeated for the document with the next largest *ucs* that is not part of a cluster yet, until there are no more uncovered documents, which is when the algorithm terminates. For example, in Figure 5.2, the algorithm terminates after two iterations, when documents 2 and 8 are chosen as cluster centers.

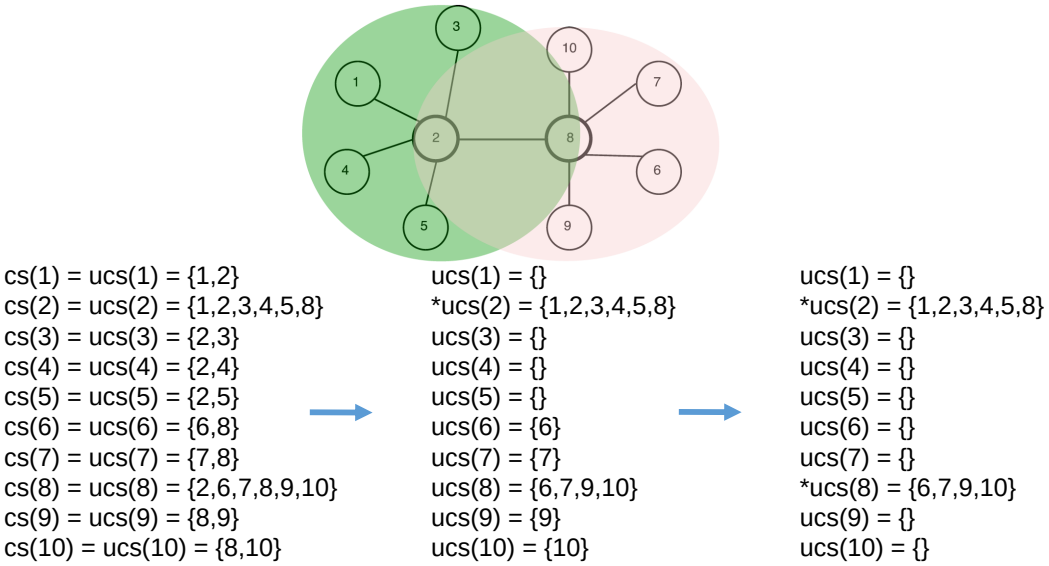


Figure 5.2: SimClus Example

The advantage of this algorithm is its computational efficiency, while it provides a natural way to cluster documents in overlapping clusters where the number of clusters should not be specified in advance.

To reconstruct *coarse grained provenance* from this, we identify the oldest message for each cluster as the *root message* of that cluster. We assume that all other messages are derived from the root through an unknown number of n steps, in other words: that they are *indirectly derived* from the oldest message in the cluster.

Until now, the algorithm does not consider pairwise similarity among other messages in the same cluster, other than the root. Since our goal is to reconstruct *fine grained provenance*, we need to identify the most similar (and chronologically older) candidate for every message, which might not be the root. Therefore, we maximize the similarity in each cluster by connecting each message with its most similar and assume that the newest message is *derived directly* from the oldest through 1 step. By doing this, we decrease the uncertainty of the reconstructed provenance (H2-II). An example of the fine-grained provenance resulting from this process is shown in Figure 5.3.

5.2.2 Additional Indicators

Next, we shed light on information diffusion patterns and users' conventions of credit attribution. We are based on Hypothesis H3 from Chapter 3. These *additional indicators* were identified after extensive observations of user interactions. Note that those indicators can be used as a standalone method for provenance identification or on top of the already reconstructed provenance based on similarity. Table 5.1

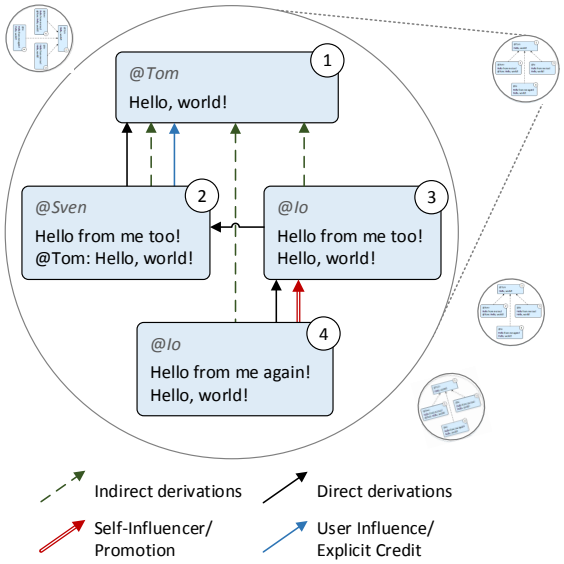


Figure 5.3: Fine grained Provenance Reconstruction. Numbers in messages indicate temporal order. The *dashed arrows* in the figure identify *indirect derivations*, generated by connecting all messages to the oldest. The solid black arrows indicate *direct derivations*, created by maximizing the similarity in each cluster by identifying the most similar pairs of messages, while respecting the temporal order.

summarizes the types of influence indicators that we leverage for reconstructing provenance.

User influence is identified when we can find some possible influencer. This can be expressed by explicitly mentioning another user. In this case, users give *explicit credit* to the initial contributor by mentioning the username within the message text (with "@" or "via"). This behaviour was adopted by users before the retweet feature was released. For example, the blue arrow in Figure 5.3 identifies a message that contains a mention (`@username`), and thus indicates *influence with explicit credit*. For these messages, we identify whether the mentioned user has emitted a similar message in the past by checking the direct derivations between these two users. If there is no user name mentioned, we investigate whether a social graph connection exists among users who propagate similar messages (*without credit*). This is a plausible hypothesis for assigning influence since users are exposed to the messages of their social connections (friends). By looking at the opposite direction of the *mentioning* feature, mentioning a user in order to expose some information to them is a common practice used from many Twitter users. If the mentioned user is emitting a similar message in the future then we observe *influence by mentioning*. We assume that such information is relevant for the mentioned user, and there is a high probability that the latter was influenced and propagated it, thereby further decreasing the uncertainty of the suspected provenance.

User Influence		
with explicit credit	*without credit*	*by mention*
manual mention of user	no user mentioned but social graph connection exists	expose information and user reacts

External Influence	**Self-Influence**	
	delete and rewrite	*promotion*
no user mentioned and no social graph connection	oldest messages get deleted	none of the messages get deleted

Table 5.1: Types of Implicit Influence

In addition to user influence, there might also exist an external event which drives the possibly high similarity among certain messages, without the existence of any additional provenance indicators. In this case, there is no user influence but *external influence*, which is hard to capture without any event identification analysis. Note that there is a grey area of *external influence* and *user influence without credit*, where it is unclear whether users are influenced by their connections or by an external event. Likewise, users might propagate messages without any obvious connection with the original authors, or any other indicator. In Section 5.4, we show that users are indeed mentioning others, without having an explicit connection. It is possible that further research into event identification sheds light on *external influence*, but this is out of scope for this work.

Lastly, we observe that in order to re-expose their audience to their own content, prolific users often promote it again (*self-influence/promotion*). In Figure 5.3 the double red arrow indicates *self-influence/promotion*, and is generated by identifying pairs of messages among the direct derivations that share the same author. In case the oldest message is deleted then it is *self-influence/delete and re-write*. Additionally, certain social media such as Twitter lack an *edit* message function, leaving users with no other option than *deleting and rewriting* their messages when making corrections. Figure 5.4 depicts the workflow we use to discern and compute those influence indicators.

5.2.3 Provenance on Social Media Streams

After we have described the background for provenance reconstruction, we proceed a step further and lay the foundations for provenance computation over streams of messages. We consider here the provenance based on similarity, without additional indicators. The problem we tackle here is *similarity computation over infinite streams of data*.

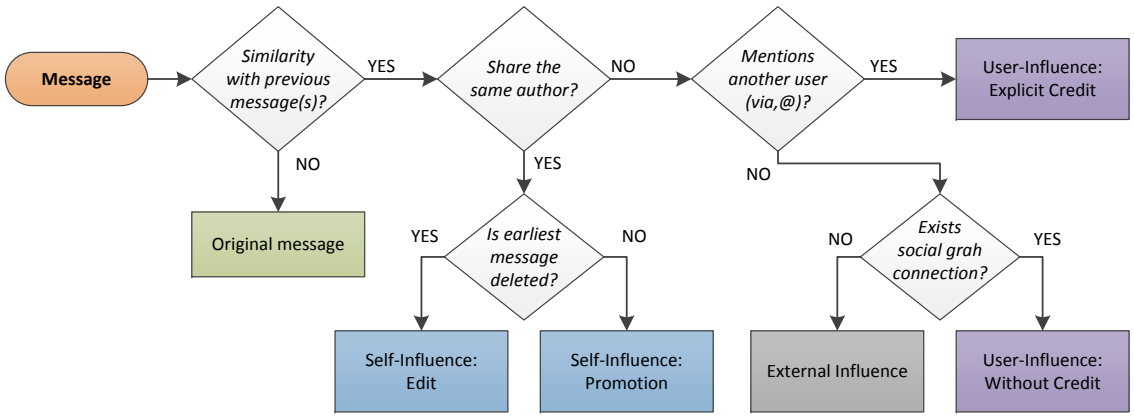


Figure 5.4: Flow diagram for implicit interactions with additional indicators

Besides a massive volume (for which we will provide optimizations in Section 5.3), real-life social media exhibits a strong temporal component that needs to be addressed: Most social media present information to their users in reverse chronological order, both activity of their connections and search results. This has also profound impact on the temporal scope of influence and thus provenance.

As [CCAB12] show for the space of explicit interactions, over 80% of replies and 60% of retweets are reactions to the 50 most recent tweets in a user's feed. For implicit interactions, this observation has been taken into account by [BCJC15]. For the identification of influence, this work considers the 100 last messages from friends presented on a users' timeline, since users generally have a limited attention span to react.

Likewise, complete information diffusion processes (as opposed to individual users) are typically modeled as processes with exponential decay [YL10], exhibiting a long tail of low activity, yet significant differences in overall duration, ranging from few minutes to weeks or months [TAFS15].

We form the following hypothesis to incorporate such a temporal component into our method: *"users are most likely to react on recently seen information or information that is getting enough additional support"* (H5 from Chapter 3). This includes older messages that have gone viral, attracting further attention. H5 entails that we can safely expire messages that are not being seen by users any more. In contrast to [AHSZ11], instead of expiring individual documents, we expire clusters. The underlying assumption is that if a cluster (that corresponds to a topic) gets sufficient support, the oldest messages might become relevant again, because users might explicitly search for them. The recency of messages gets updated if they receive retweets (the time of the most recent retweet is considered), which means that such messages are getting further visibility. Keeping around the entire cluster contents ensures that the long tails of "older" processes get support, while the overlapping nature of SimClus ensures that such messages are not being consumed only by old clusters.

Yet, this stream clustering workload does not fit the bill well on existing stream clustering methods such as [KABS11]: These methods leverage statistical summaries over the data, and provide a hierarchical decomposition into clusters. Their goal is to identify concept drift and adapt to the streaming data speed (leading to a less clear clustering result). For our requirements, we do not need to pay the overhead of computing concept drift, but rather need to carry all the active documents available for fine grained provenance computation. Additionally, a hierarchical clustering decomposition is not sufficient, since we desire to identify meaningful clusters in a natural way that can reveal provenance information. Dynamic topic models such as dynamic LDA [BL06] provide such clusters, but are out of scope for this work, as their main focus is to identify topics and topic evolution by employing complex generative statistical model. Specifically, LDA clusters according to the latent representation of documents, while we consider the explicit Tf-Idf. Latent topics might help for the reconstruction of fine-grained provenance but we desire not to add another layer of abstraction. We leave this for future work to identify whether LDA based clustering contribute to provenance identification. In general, our focus is not on topic detection, but a lightweight topic identification is a by-product of document clustering.

5.2.4 Use Case: Ranking of Influence

Lastly, we describe how provenance reconstruction can be used in practice. Such provenance information with additional indicators complements influence computation analysis, which was in most cases studied with explicit means (e.g. retweets, reshares, replies etc). Online journalists, scientists and online users in general are equipped with a powerful tool that unravels influence, beyond explicit interactions. Apart from the fine-grained provenance reconstruction in section 5.2.1, we provide ranked lists of influential messages, according to different influence metrics. We consider implicit, explicit means and their combinations. We also consider cumulative influence: for that, we compute influence paths by traversing the direct derivations from the leaves to the sources. We compute the following metrics of influence:

- **Retweets** shows how many retweets a message has attracted.
- **Single-hop direct derivations** shows how many messages were implicitly influenced.
- **Single-hop direct derivations with retweets** adds implicit and explicit influence that a message evokes.
- **Multihop derivations** shows the cumulative influence by traversing influence paths.
- **Multihop derivations with retweets** accumulates implicit and explicit influence along influence paths.

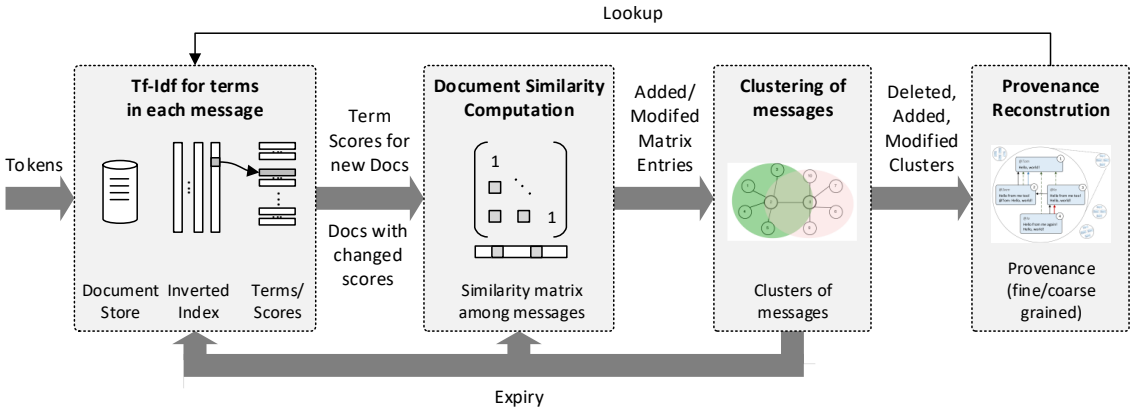


Figure 5.5: Architecture to compute implicit provenance on web-scale data

5.3 Reconstruction on a Web-Scale

5.3.1 Overview

Existing related work like [DNTD+15] and [BCJC15] reconstruct implicit provenance on a finite amount of data by accessing the available dataset in a blocking manner or in reverse temporal order. However, this is of limited use when working with web scale streaming data, since a massive amount of data arrives incrementally in a temporal order with high rates and without an obvious end. While the temporal nature and the decay on relevance dampen the size of the overall working set, long periods (ranging from hours to days) still need to be taken into consideration to find matching candidates (as our evaluation shows in Figure 5.11 for tweets and Figure 5.22 for news). As a result, even when constraining ourselves to specific user relationships and/or events, we end up with a working set consisting of several 10Ks for news media to millions of messages for social networks which gains and sheds hundreds or thousands of messages per second.

The overall pipeline outlined in Figure 5.1 refers to established and well-researched methods for each step, but combining and applying them on such a workload leads to the following challenges:

- The Tf-Idf scores of terms used to determine similarity depend on frequency in the document set, yet this document set is constantly shifting due to arriving and expiring messages. Precisely reflecting these changes will lead to widespread score changes and thus permanent re-computations of the similarity matrix on existing values.
- Determining the similarity between all document pairs is a costly undertaking in terms of space and time. Even without score changes every new document needs to be compared against every existing document in the working set.
- Fully re-computing the results in clustering as well as the intra-cluster PROV derivation (as introduced in Section 5.2.1) is clearly infeasible at the requested

speeds, considering that the computational cost of these methods are also clearly super-linear in size.

To tackle these challenges, we pursue a number of strategies that can be grouped in the following directions:

- Wherever possible, we use or develop incremental methods for the individual steps of the pipeline in order to re-use existing state, turning this into an architecture where every stage consumes and produces incremental results (Figure 5.5).
- We perform semantic optimizations by exploiting the properties of the "downstream" algorithms and the workload to reduce the number of computations and the amount of storage needed.
- Similar to query processing of database systems, we heavily rely on *indexing* to limit the scope of documents to consider when performing operations as well as on early stopping once results could no longer be produced.

The combinations of these techniques enable us to advance significantly beyond the current state-of-the-art and allow for optimizations that speed up the computations by several orders of magnitude.

5.3.2 Architecture for Incremental Evaluation

As outlined above, a key element of a scalable solution on streaming data is the ability to perform incremental computations. We turn the conceptual pipeline outlined in Figure 5.1 into a scalable architecture that is shown in Figure 5.5. Each stage corresponds to a pipeline step, yet refined to fit the requirements of infinite, massive-volume data streams. Instead of computing the full results at each stage and the passing them on, the computation is performed gradually on available data. Each stage stores a certain, limited amount of state, consumes the changes produced by the previous stage and produces new changes that are propagated to the next stage. This state is being pruned by expiring documents that are no longer covered by any cluster. In turn, documents are expired after a certain time has elapsed since last activity and thus cannot generate any provenance, as outlined in Section 5.2.3.

In more detail, this means that we submit batches of newly arrived messages to the system, which are turned in a set of tokens for each added message after stopword removal and, if applicable, stemming. These documents are then stored as long as needed, as some of their contents (including the author ids and the texts) are required to generate the provenance serialization. Furthermore, Tf-Idf scores are computed on the basis of these tokens. New documents as well as documents whose scores change, e.g., by shifts in the term distribution, are forwarded to the similarity matrix computation. In that stage these documents are compared to the set of active documents to compute the cosine similarity on the score-weighted word vectors. The result is a set of document pairs whose similarity values are new (either by addition or value change). For the similarity clustering algorithm, we picked up the

extension for incremental updates working on this set of new and changed similarities described by [AHSZ11]. Most of the conceptual contribution in this space lies in making the intra-cluster influence derivations also incremental. As a foundation, the clustering algorithm was extended to report changes by returning three sets after each computation. (1) The clusters which were updated, (2) the clusters which were added and (3) the clusters which were deleted. Newly added clusters are not only the result of new documents. The possible reshaping of the clustering (former centers become non centers and vice versa) also leads to the deletion and addition of new clusters. In case of addition or deletion of clusters, the incremental provenance algorithm simply computes the derivations of any new cluster; respectively deletes the derivations corresponding to a cluster which no longer exists. For the clusters which were updated, all the direct (single-step) derivations need to be recomputed as the similarity may change or a previous seen message now is more similar to a newly added document (message). If a message was added which is older than the other messages, all the indirect (n step) derivations need to be updated as now all messages within the cluster are derived indirectly by the newly added message.

Overall, this kind of architecture lends itself well to parallelization and distribution, even though we have not exploited this yet. One natural direction is pipeline parallelism, as each stage of our pipeline can be executed individually with little coordination due to the notion of propagated changes. Furthermore, most stages – with the exception of the clustering step – lend themselves well for data parallelism: tokenization, score computation, similarity matrix entry computation, and fine-grained provenance computation in clusters provide clear means for partitioning. Yet, this is a rather brute force approach and in this work we are looking into two means to increase the performance and scalability of this architecture on a more conceptual level: (1) Semantic optimizations to reduce both the amount of state in each stage as well as changes propagated between the stages without impairing provenance results (2) Reducing computational cost at each of the stages using techniques from database implementation. We will now describe these optimizations in more detail.

5.3.3 Semantic Optimizations

A key insight we gained when designing our system to compute provenance is that even when producing precise provenance not all data from the previous stages is needed. As a result, computation and storage in the earlier stages can be avoided, in particular for Tf-Idf score changes and similarity matrix entries.

In order to limit the impact of score/Idf changes due to newly arriving or expiring documents, we only propagate such changes when they exceed a certain threshold. This is driven by the insight that the utmost majority of these changes is small. When documents arrive or expire, updated Tf-Idf values ($\mathit{tfidf}_{current}$) are computed for all documents with the same terms as these documents, utilizing the inverted

index. We then determine the amount of change from the value propagated before ($tfidf_{old}$) as follows:

$$\frac{|tfidf_{current} - tfidf_{old}|}{tfidf_{old}} \geq \epsilon \tag{5.1}$$

The values are only propagated once they exceed the change threshold ϵ. This approach provides several advantages: First and foremost, it guarantees a bounded error that will also not lead to unlimited error propagation. The trade-off on accuracy and cost is tunable, and yields significant benefits even at fairly conservative values, as the evaluation will show. Likewise, the resulting errors are very small even at aggressive settings. Finally, while computing all updated Tf-Idf values does not come for free, this cost is rather moderate and the computation could be easily parallelized.

Likewise, to overcome the prohibitive cost of fully computing and materializing the similarity matrix, our main observation is that the distribution of those values is strongly skewed towards low values, as many documents have very few terms in common. In turn, only similarity values above a certain threshold are actually needed for clustering and fine-grained provenance. After the obvious optimization to store only those values above the threshold to save space, the main pursuit is to avoid as many irrelevant computations as possible while not compromising the results. Significant benefits towards this step stem from the fact that we are dealing with very short documents, often containing less than ten terms after stop-word removal. When applying this technique on datasets with larger texts, the effects are less pronounced, as the distribution of values is more even. In particular, there are many data points that are actually zero.

5.3.4 Indexing and Early Stop

Considering the cost of score and similarity matrix computation, we are also employing techniques from database query processing, namely indexing and lazy computation of results, as outlined in Figure 5.6.

An important observation is that on only documents with common terms will produce non-zero similarities and changes in the Tf-Idf scores. While we were already saving the storage using the optimizations outlined before, these unnecessary values still need to be computed. Given the sparsity of term in a short-text workload, we expect only a small number of matches. Therefore we can effectively utilize the inverted index to retrieve the union of all documents that contain at least one common term (as otherwise the sum and thus the score would be 0). Instead of comparing this document against all documents in the working set, we only compare it against the members of this union which is typically much smaller for workloads with short texts.

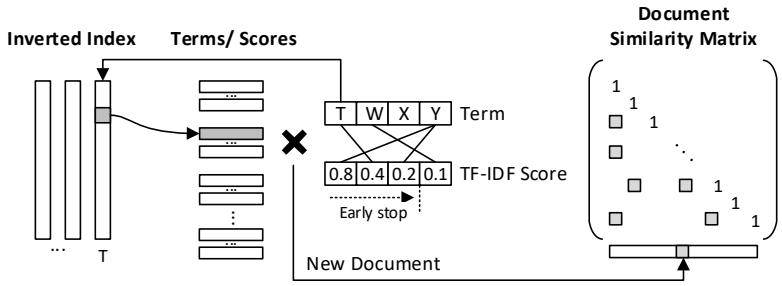


Figure 5.6: Similarity Computation and Early Stop Optimizations

Furthermore, we can also sort the terms by their score and exploit an early-stop approach to only consider those terms with high scores, since documents that have only low-score terms in common will not make it above the threshold. In addition, this will help in pruning out long word lists, as terms with low scores tend to be contained in more documents (due to their lower Idf). This is particularly useful to deal with frequent terms for which no stop-word filtering is effective, e.g., because they have periodic behavior.

To understand the gist of our optimizations, recall the definition of Cosine Similarity we are using to determine the similarity of two documents:

$$\frac{\sum_i^N a_i * b_i}{\sqrt{\sum_i^N (a_i)^2} * \sqrt{\sum_i^N (b_i)^2}} \tag{5.2}$$

where the non-zero parts of the sum in the numerator stem from common terms.

The idea rests on the possibility to estimate the expression $a_i * b_i$ (and also the denominator) from the information in A only, turning it into $(a_i)^2$ and $\sum_i^N (a_i)^2$, respectively. This means that the scores (and Euclidian norms in the denominator) should be roughly symmetric, which tends to be the case since the small number of terms per documents allows only for small variations in the Tf part. Lastly, a good stopping condition for term/score at *currentPosition* is

$$\frac{(a_i)^2 * (|terms| - currentPosition)}{\sum_i^N (a_i)^2} < stopThreshold \tag{5.3}$$

since earlier terms have already been covered and all later terms may at most have the same score as the current term. Choosing *stopThreshold* depends on the (a)symmetry in scores and Euclidian norms. For the short-text workloads, setting *stopThreshold* as θ/k, where k is greater than 2.5 ensured that there were no differences in the results while achieving quite significant saving in cost. When dealing with longer texts, the long terms lists lead to values for *stopThreshold* that are closer to θ, and additionally a filter for the raw score values (e.g., 0.02) to clip of the long tail.

5.4 Evaluation

We evaluate our approach on three levels. In Section 5.4.1, we confirm the conceptual soundness of our provenance reconstruction method on a *small-scale* dataset gathered at the ISWC 2015 conference. Note that automated evaluation is very challenging – if not impossible – here, due to the non-existence of a ground truth. Instead, we evaluate the correctness of the provenance edges by a thorough manual investigation using qualitative methods. We study user interactions in detail, and compute influence according to different metrics from Section 5.2.4.

In Section 5.4.2, we confirm the method's quality and scalability by applying it to a *large-scale social data* dataset collected during the 2012 Olympics, measuring the run-times and consistency of the reconstruction results.

Lastly, in Section 5.4.3, we validate the method on a dataset taken from a news feed, providing a significantly different text corpus with a large number of words/tokens per message. For each section, we will first determine the relevant parameters of the pipeline introduced and discussed in the previous sections:

- ϵ: degree of Tf-Idf deviation allowed before score change propagation;
- θ: minimum similarity value to materialize, needs to be at most as high as the clustering threshold;
- clustering threshold: lower bound of the cluster similarity;
- expiry: duration since the last activity of a cluster before discarding it.

5.4.1 Small-scale Empirical Evaluation

For the empirical evaluation, we used a controlled and complete ISWC dataset in order to compute fine-grained interactions. The dataset contains 3909 messages, consisting of 2068 retweets, 198 quotes, and 93 replies. By excluding messages that carry explicit provenance by Twitter, we end up with 1550 distinct messages.

We applied the pipeline outlined in Figure 5.1 on this data set, relying on non-incremental computation due to the small data size. Therefore, no values for ϵ, θ and expiry needed to be set; we will investigate them in more detail in Section 5.4.2 and Section 5.4.3. In order to find an appropriate clustering threshold for our dataset, we need to define an objective function which assesses the clustering result. Since SimClus allows the overlap of clusters, which is desirable in our set-up (messages might belong to more than one topic and might have multiple sources), we do not need to account for the distances among clusters, as other clustering quality metrics like the Silhouette [Rou87]. We select the RMSSTD (root-mean-square standard deviation) index [KLB05, Sha95] which measures the distance of data items within each cluster. RMSSTD produces values from $[0, 1]$, where values lower than 0.5 imply a good clustering result.

Figure 5.7 shows how the RMSSTD metric and the number of singleton/non-singleton clusters are influenced by different clustering thresholds (shown as numbers at the data points). We observe that clustering thresholds above 0.4 are acceptable, as the quality increases linearly with the increase in the clustering threshold. Furthermore, the number of non-singleton clusters decreases with higher thresholds/lower RMSSTD, making clustering thresholds above 0.8 unsuitable due to the sparsity of results. We quantify sparsity with the number of singletons. We also desire to have larger clusters (more than two messages), so that connections can be developed within each cluster. This also confirms our empirical observations for appropriate clustering thresholds in [DNCVD+12] and [TFDN+16].

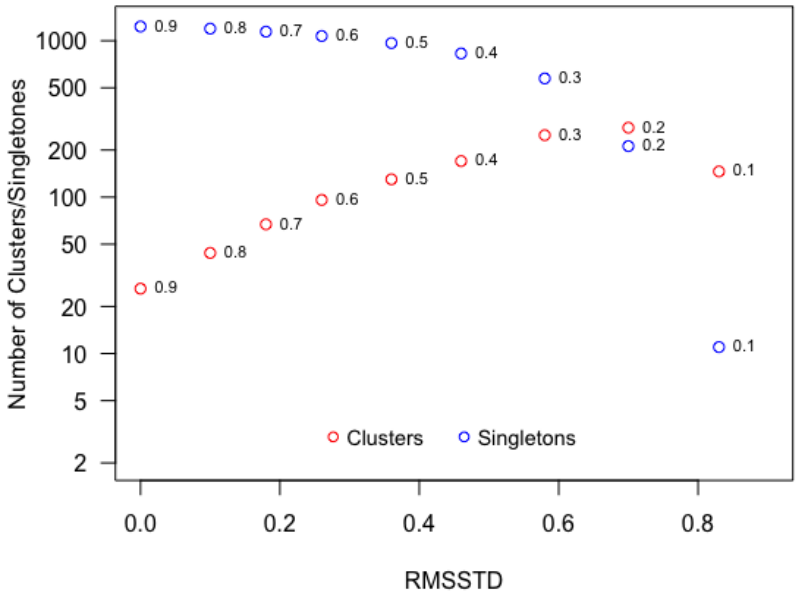


Figure 5.7: Determining suitable clustering thresholds (at data points) via RMSSTD and cluster count for ISWC dataset

In general, there is not a clear lower bound for clustering. This figure is indicative of how the number of clusters and singletons behave for different clustering thresholds, so that each use case can leverage the "*most fitting clustering threshold*". This threshold depends from the amount of external influence that each dataset contains. The higher the external influence, the higher the threshold should be set in order to avoid the "*overeager*" similarity connections that the algorithm makes. Another significant factor to consider is the size of the dataset. Small datasets (in the range of thousands) with very high thresholds yield mostly singletons. On the contrary large datasets (in the range of millions) and low thresholds result in large clusters that are hard to evaluate. Empirically a 0.7 threshold is a good trade off for this

particular dataset that has external but also internal influence. In practice, different similarity thresholds do not affect the fine-grained provenance reconstruction (direct derivations) much. Every document will be connected with its most similar and in the majority of the cases the pairwise similarity within clusters is much higher than the similarity threshold. From the perspective of the light-weight topic identification, evaluating different similarity thresholds provides a hierarchical decomposition of topics. In reality, there is no "*hard clustering*": a corpus of documents can be described by general topics which in turn are divided in sub-topics. It is left to the individual application to decide for the desired level of topic granularity.

By applying the provenance algorithm with lower bound of similarity 0.7, we identified 67 clusters and 81 direct derivations. The cluster distribution is presented in Figure 5.8. We observe that the majority of clusters are singletons or clusters with two messages, given the rather high threshold. Despite possibly missing some lower similarity pairs of messages, we opted for this threshold to simplify the manual evaluation by reducing the number of clusters and edges to be inspected and filtering out possibly irrelevant edges. A quantitative evaluation of this dataset using a less restrictive clustering threshold 0.4 is described in [TFDN+16] for implicit and explicit diffusion means and their interactions.

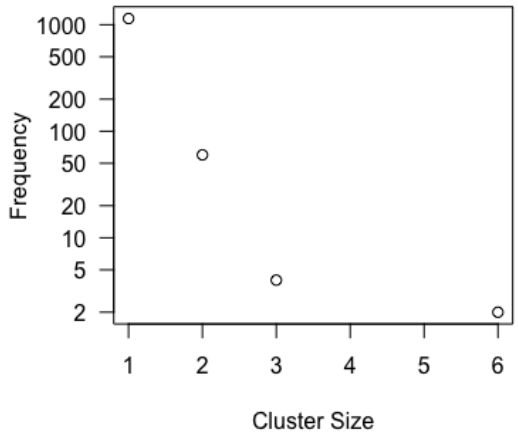


Figure 5.8: Distributions of Cluster Sizes for ISWC dataset. The cluster size distribution has more than 1000 singletons, 60 clusters have 2 messages, 4 clusters have 3 messages, 2 clusters have 6 messages.

In order to verify the correctness and soundness of our approach, we followed two evaluation strategies. First, we manually inspect the results of the clustering algorithm and report observations for human interactions and influence. Such empirical observations shed light on the human behaviour patterns that are not captured by automated tools. Second, we compare the influence of messages with explicit and

implicit means, which also provide indicators for the soundness of our approach.

Empirical Manual Analysis and Observations. We performed a manual, qualitative inspection of the 81 pairs of messages for which our algorithm indicated that they share some provenance. The goal was to understand the truthfulness of the provenance generated and creating a stand-in for actual ground truth. An interesting challenge in this analysis is the role of external influence, i.e., events affecting several users without traveling through the social network.

Several complementary means were employed to aid this analysis – some of which could eventually be turned into rules for automated, deep classification.

1. We considered explicit means of attribution, e.g. user mentions (see Section 5.2.2 and [TFDN+16]) or keywords that show influence, e.g. "via", "RT".
2. We looked at social connections among the authors - their absence might indicate external influence.
3. We also observed shared content in the messages, e.g., memes, URLs or photos. We identified that users might share the same URLs, but also might emit the same photo under a different URL. Unfortunately, the latter cases are indiscernible without image analysis algorithms, which are out of scope for this work.
4. Finally, we considered additional interactions among the authors, e.g. likes, retweets or replies that might strengthen our provenance assumptions (interaction - based observations). For those interaction-based indicators, the authors often imply influence and provenance by interacting additionally through explicit means. We observe messages sharing some provenance, while additionally, the influenced author retweets or likes the source message in order to highlight the influence. Additionally, the source author might identify their influence by interacting with the influenced author in similar explicit ways.

Those indicators are computed on top of the clustering algorithm to strengthen the reconstructed provenance. They are Twitter-specific in a sense that they support our observations mainly from Twitter. We desire that our algorithm is general enough so that it can be applied to any text-based social media, and for that the core of the algorithm remains as simple as possible. The indicators 1,2 that are applicable to other social media like mentions or the existence of social connections are already systematically computed. Indicator 3 is very hard to capture with rules and computationally expensive (image identification or URL dereferencing). Also the amount of messages that demonstrates such provenance is very low, which does not provide enough reasons to fully incorporate such provenance. Indicator 4 is bound to data restrictions. For example, likes in Twitter cannot be crawled (i.e., who liked which message). We identified those cases manually using the Twitter web interface. Furthermore, indicators like additional retweets or replies could be

incorporated given that those messages have been crawled. The systematic analysis of indicator 4 is left for future work.

In terms of truthfulness of provenance, these observations are certainly not perfect since they have been discovered by extensive observations and manual investigations, not formal grounding. Furthermore, they may not apply to other datasets. However, we have been observing some recurring patterns and user conventions and we assume that there must be a qualitative basis to claim that these indicators take place in social media. We also have observed that online users leverage multiple means to convey influence, not always explicit, like the older generations who still use mentioning instead of retweeting. In order to minimize any possible uncertainties over these indicators, we may crowdsource these tasks to a larger group of experts in the future. Alternatively, we can implement surveys where we ask the authors of messages, how they got influenced.

Considering explicit means, interactions and shared content, we could categorize 54 (out of 81) derivations that our algorithm connected as in-network provenance edges, and thus deem them correct. Out of these messages, 29 share the same author: 11 are promoting the same content and 18 show editing behavior. Already by looking only at the user information we can reason for 36% of the implicit interactions (direct derivations). 16 messages mention their influencer by giving some explicit credit to them. 7 messages include particular content or interaction based indicators, which is hard to model and automatically identify.

In terms of the social ties, besides the 29 derivations which are from the same author we observed the following: 4 pairs share information of their friends (users whom they follow) , while 13 pairs of messages demonstrate bidirectional relationships. 4 users are forwarding messages of their followers. This number ties with our observations that sometimes users identify their own influence on others by interacting with their messages. Lastly, 26 messages share no social relationship. These pairs of messages may be accounted to the external influence (the external events of the conference). For 5 pairs of messages the social graph for at least one of their authors could not be retrieved. In general, these results align with our hypothesis that users are influenced by information from their social connections to a considerable extent [TF14].

We classified 27 derivations as demonstrating external influence, meaning that no particular indicators were identified to strengthen the generated provenance. The presence of social links alone is not sufficient for a resolution, as the tightly connected community of people in the ISWC conference knows each other well in real life, yet their behavior does not imply the usage of the social network in those cases.

Expressing external influence as provenance derivations is not "wrong" per se, as we still capture an apparent influence, and can express it as an "unidentified entity" [TFDN+16]. Since our algorithm does not know the actual external events, it connects those messages with their most similar predecessor. In future work, we aim at identifying external events with a topic detection algorithm, create nodes for

such events and connect them with those messages. However, such analysis is out of scope for this work.

In summary, this manual analysis showed that no incorrect provenance edges were generated, demonstrating the high precision of our approach with this cluster threshold. Unfortunately, we cannot assess the recall without a ground truth, as it would be infeasible to scale the manual analysis to include all possible message combinations.

Influence Ranking of Implicit vs Explicit Interaction means. We also analyzed the soundness of our approach by contrasting the generated influence with the explicit means influence, e.g., retweets. In the absence of a ground truth, we rely on the retweet count as a prominent means of attributing influence [CHBG10]. In particular, we are interested in the most influential messages computed by both methods (implicit: our algorithm, explicit: retweets) and we would like to find out whether there are substantial differences in the influence inflicted by both methods.

We compare the ranking of the same documents according to different objectives: explicit and in implicit influence. We are interested in the relative order that each document has in both lists. For the ranked list with implicit means, we counted how many documents are influenced by each document by means of direct derivations. These documents are ordered and the top 10 most influential documents are selected. For those documents, we also retrieve their retweet counts. We used the rank-biased overlap (RBO) which computes how close two (possibly) infinite ranked lists are [WMZ10]. RBO is useful when comparing lists that do not contain the same elements and are not equally sized (disjointedness problem), which is not possible using metrics like Kendal Tau. In Section 5.4.1 we compare lists that might not contain the same elements. The result show that these two lists are indeed very close with an RBO value of 0.92, which confirms the suitability of our provenance reconstruction algorithm for relative influence computation. This also illustrates one of the possible in-use scenarios of our approach: identifying the most influential messages in a dataset.

Influence Ranking of the Combination of Explicit and Implicit means. After confirming that the relative influence computed by our algorithm complies with the "ground truth" influence given by Twitter (Section 5.4.1), we investigate the impact of our implicit influence results. Here, we evaluate the implicit influence computed by our method and we compare it with explicit influence, i.e. retweets. In order to do that, we compare the ranking of messages according to their *implicit* and *explicit* influence. In contrast to 5.4.1 where the two lists contained the same elements and the goal was to compare the same documents with different influence means, here we focus on identifying the absolute ordered lists according to different means. If the rankings are very close, then it means that explicit means in isolation can capture the relative influence of messages, and implicit means can be disregarded.

In contrast, if the correlation of ranked lists is weak, it means that implicit influence has a significant impact on the results, providing additional value over purely explicit influence rankings. Additionally, we want to investigate the impact of accumulating and combining influence with regard to implicit and explicit methods. That will help us to better understand the factors that need to be consider when calculating influence (implicit, explicit, accumulated).

We compare the lists of messages that are ranked according to: (1) explicit influence: number of messages influenced by retweeting, (2) implicit influence: number of messages influenced directly (single-hop direct derivations reconstructed within the clusters, which are formed by the clustering algorithm in 5.2), (3) cumulative implicit influence: number of messages influenced implicitly thought n hops (multi-hop). (4) implicit influence combined with explicit: the sum of single or multi-hop derivations with the number of retweets.

Table 5.2: ISWC dataset Influence Rankings for top30

Comparison of provenance types	**RBO value**
single-hop (2) vs retweets (1)	0.20
single-hop (2) vs multi-hop (3)	0.87
single-hop (2) vs single-hop with retweets (4)	0.38
multi-hop (3) vs multi-hop with retweets (4)	0.48

In order to compute multi-hop direct derivations, we need to construct the influence paths within the clusters. For the ISWC dataset, we observed paths up to length 3. The results are shown in Table 5.2: The first line concerns comparison of ranked lists of messages according to single-hop and retweets. The difference with the evaluation in Section 5.4.1 is that previously we considered the same messages ranked by two different means (single-hop derivations vs retweets). Now we consider the absolute ranking of top 30 messages according to different means. As the results show, the correlation of 0.2 is now quite weak. This means that computing influence solely relying on retweets lacks the impact of implicit diffusion.

In the second line, we observe that extending direct derivations to arbitrary many hops does not change the ranking significantly: messages that are ranked high for direct implicit influence (single-hop), have also similar ranking for cumulative influence (multi-hop). Note here that this relatively high correlation derives also from the small size of the dataset and the short influence paths. However, in the third and fourth lines, when implicit (both single-hop and multi-hop) and explicit influence are combined the shifts in rankings are significant, which further proves the merit of our contribution.

5.4.2 Large-scale Twitter Dataset Evaluation

For the large-scale Twitter data evaluation and also for the news data evaluation, we implemented the system described in the Section 5.3 in Java, utilizing an OpenJDK 8 64 bit JVM on an Amazon EC2 r4.2xlarge instance with a 2.3 Ghz Intel Xeon (Broadwell) CPU and 60 GB RAM. All steps of the pipeline were executed on a single thread. The initial stages of our pipeline build on well-known components: Tweets texts are split using the Twokenize library, stopwords are taken from the Python NLTK toolkit and amended with common Twitter expressions such as emoticons and a sample of frequent, domain-specific words taken from the first batch of messages in the dataset.

The dataset we are using was recorded during the 2012 summer Olympics in London using terms like "`olympics`", "`london2012`". For the evaluation, we used a prefix of this data spanning four days (August 3 to 7th, 2012, afternoon to afternoon) and containing more than 4.6 million messages.

Given the large size of this dataset (3 orders of magnitude bigger than the previous), setting the parameters correctly for incremental processing is of great importance. We used batches of 2500 messages that we fed continuously into the system. Larger batch sizes generally would lead to somewhat higher throughput but also carry higher latency due to the queuing times when building the batches. For the clustering threshold, we selected a slightly higher similarity threshold (0.75) since this data set is much noisier than the previous, also due to the presence of spam. Additionally, there is much more external influence than in the ISWC dataset and we increase the threshold to remove those external influence edges.

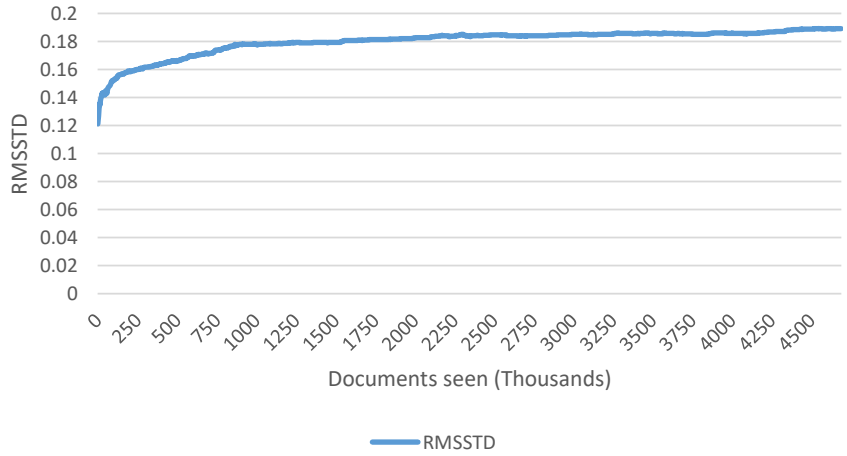


Figure 5.9: Evolution of Clustering Quality for Olympics dataset

With this setting, the RMSSTD value starts off at 0.12 in the first batch, then increases to around 0.18 within the next 250K messages and then stays almost stable (never exceeding 0.19) until the end of the data set (Figure 5.9). In turn, we set

the value of θ to 0.65 so that there is enough headroom for provenance computation in clusters. For ϵ, we settled on a value 0.5. The impact of these values will be studied in more detail in Section 5.4.2. To capture the temporal dynamics and limit the growth of the working set, we enabled expiry with an activity timeout of 12 hours, i.e., a cluster not getting updates to any of its messages for 12 hours will be discarded. To determine an appropriate expiry time, we looked at times between the last and the second-last activity in a cluster, as this would determine if a cluster would have been discarded too early. The distributions of those values showed that doubling the expiry interval would cover about 10-15 percent more cluster "tails", up to a duration of 12 hours, leveling off after. 12 hours was also a convenient time to give room for the roughly 18-24 hours news cycle present in this data set (most events taking place in the afternoon and evening).

Provenance Analysis. Figure 5.10 and 5.11 shows the qualitative characteristics of the clusters produced with these settings.

Figure 5.10 shows the cluster distribution of the last batch. We can observe that the cluster size demonstrates a skewed distribution, with the largest cluster containing almost 21K messages. The median size is 15, which shows that there are many small clusters and a few large ones (corresponding to trending topics). The lifetimes in Figure 5.11 show a more balanced distribution, since small clusters with limited activity expire quickly. Values on the top 25th percentile are more spread, demonstrating a variety of longer lifetimes. The mean value is approximately 8.7 hours (median 8.2).

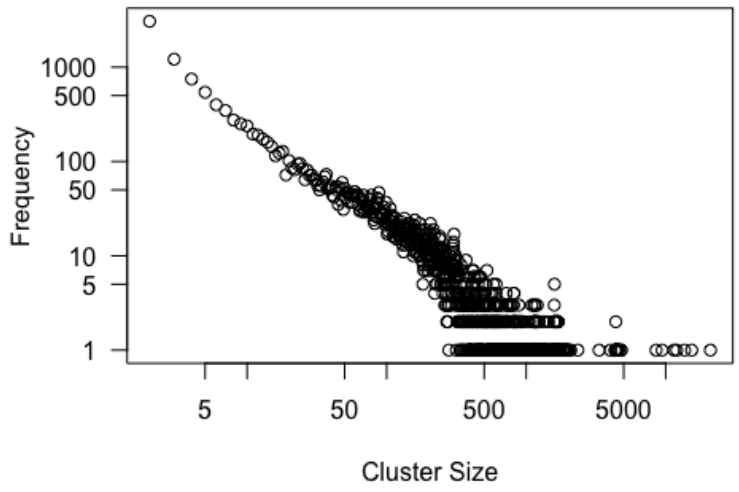


Figure 5.10: Distributions of Cluster Sizes for Olympics dataset

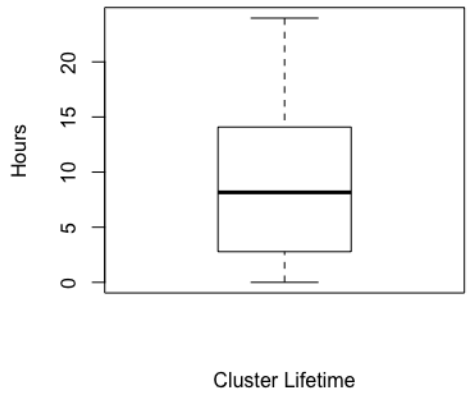


Figure 5.11: Distributions for Cluster Lifetimes for Olympics dataset

The clusters show good coherence, which means that in most cases, the provenance structure will form a single tree; in rare cases, the similarity was too uneven, which will result in several disconnected provenance subtrees. The provenance structure will be quite complex, since we observed paths with up to 14 hops.

Analyzing the social connection on the derivations proved do be difficult for two reasons: In order to not slow down provenance generation, fast access to the full social graph is needed, requiring massive amounts of main memory. Even more limiting for our study was the completeness and quality of the social graph information we had collected for [TF14]: since its purpose was to support analysis of explicit interactions (retweets), it lacked the coverage of user involved in implicit interactions. As a result, we could not get conclusive and reliable results.

We did get reliable results on the rankings of influence, in order to verify and compare the results from Table 5.2, with an additional emphasis on how the rank changes over time. As shown in Figure 5.12, we observe that the single-hop ranks are even less correlated with the retweets, proving again that retweets alone are not a reliable indicator of influence. Single-hop vs multi-hop show medium correlation (compared to Table 5.2). The reason for that is the denser dataset and greater size of clusters which results in influence being aggregated over longer paths. The single-hop vs single-hop with retweets is correlated medium to high: we observe that when considering only single-hop implicit influence, we get a significant share of influence. Similar trends (even stronger) can be observed for multi-hop vs multi-hop with retweets. Figure 5.13 provides some insights into causes of the fluctuation over time. Comparing the number of active non-singleton clusters against the correlation of implicit influence vs retweets, we see that the correlation is stronger on periods of low activity (somewhat obscured by the expiry delays). In such periods the number of retweets remains rather low, thus aligning stronger with implicit diffusion.

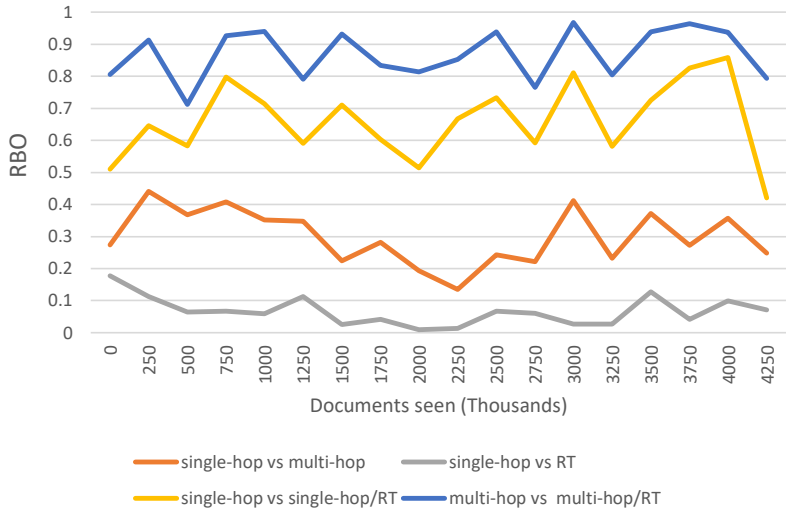


Figure 5.12: Rankings for Different Influence Metrics

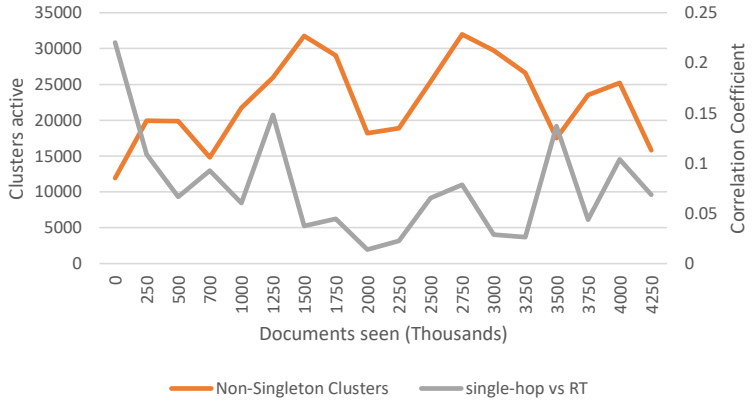


Figure 5.13: Ranking vs Activity over Time

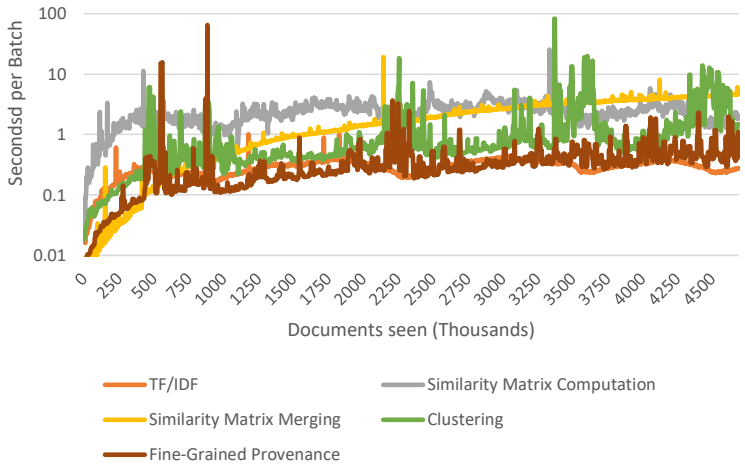


Figure 5.14: Computational Times per # of Documents

Performance Analysis. Our first set of performance experiments presents the timing and cost contributors, determining its suitability to produce quick results. Figure 5.14 shows the cost per batch for the main components of our computational pipeline: Tf-Idf computation, similarity matrix computation, merging of similarity matrix deltas with the main matrix, similarity clustering and computation of fine-grained provenance on all clusters. One can clearly see we are able to maintain a rate off several hundred to thousand messages per second, taking a few seconds to process a batch of 2500 messages. Furthermore, we can observe that the cost for Tf-Idf and similarity computation becomes stable rather quickly, while the cost of merging (similarity post-processing) keeps growing for longer - the reasons will become clear with the next experiment. As expected, the cost for fine-grained provenance dominates since it tries to optimize the connections among each cluster. Figure 5.15 presents the overall costs breakdown, showing that computing the similarity of new documents, incremental clustering and provenance computation dominate, while the cost of updates has been dampened.

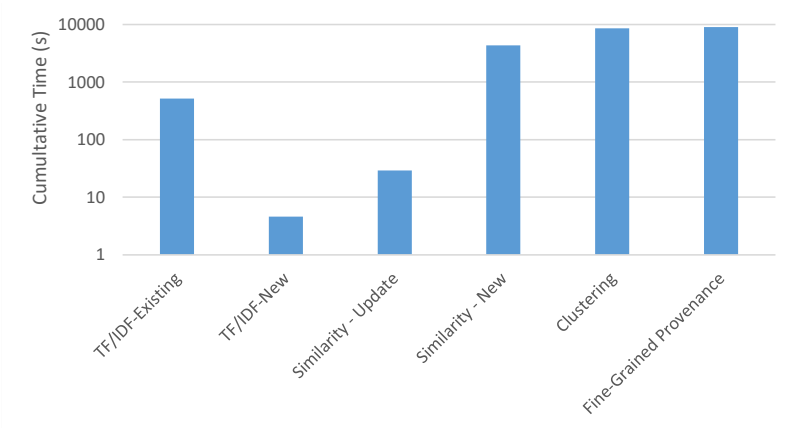


Figure 5.15: Overall Computational Times

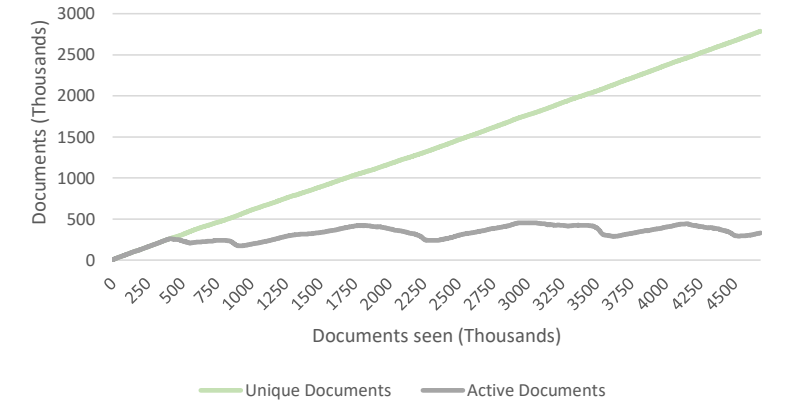


Figure 5.16: Number of Documents over Time for Olympics dataset

The second set of experiments covers the working state of the system to understand where scaling limits exist. Figure 5.16 shows that expirations and changes in trending behavior keep the set of the active documents to a very small fraction of the unique (observed minus retweeted) documents. One can see as in the previous figure that the number of active documents fluctuates over time, as the underlying activity fluctuates. Figure 5.17 provides a deep dive into the state induced and needed by our computations. The number of clusters as well as the number of distinct terms and total index entries in the inverted index become stable soon and fluctuate with the activity, capturing four days of news cycle with slightly different activity per cycle. Overall, the space consumption of the inverted index is rather modest, given the challenges of the "noisy" language use in Twitter. The main limiting factor is the number of similarity pairs used - both in the matrix and the clustering algorithm.

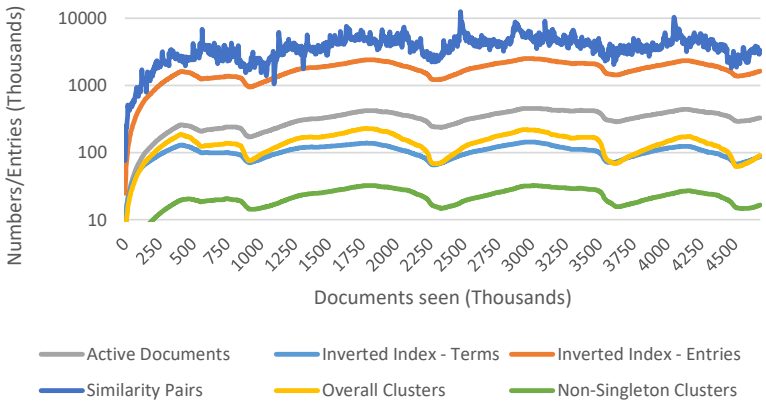


Figure 5.17: Working Set Sizes

Optimizations and Tunables. The final set of experiments highlights the benefits of our optimizations and the sensitivity to the tuning settings. Figure 5.18 shows the benefits of the score change mitigation introduced in Section 5.3.3. Propagating updated scores only when the changes exceed a ratio (epsilon) of the last propagated has a tremendous effect. Without any mitigation, every newly arriving documents triggers more than 20 changes in existing documents. Even when just allowing deviations of 10 percent (0.1), the number of changes produced is reduced to roughly the same number of changes as new documents. Increasing the allowed deviation to 0.3 and 0.5 brings again an order of magnitude reduction, beyond that the gains become very marginal. We computed the mean square error of the similarity matrix and the differences in clustering, both of which turned out to be negligible for the thresholds tested.

The next optimization presented in Section 5.3.3 is covered by Figure 5.19, which concerns the means to store and compute the contents of the similarity matrix. Each optimization introduced brings about one to two orders of magnitude reduction - storing only values above the relevant threshold, indexing and early stop. It should

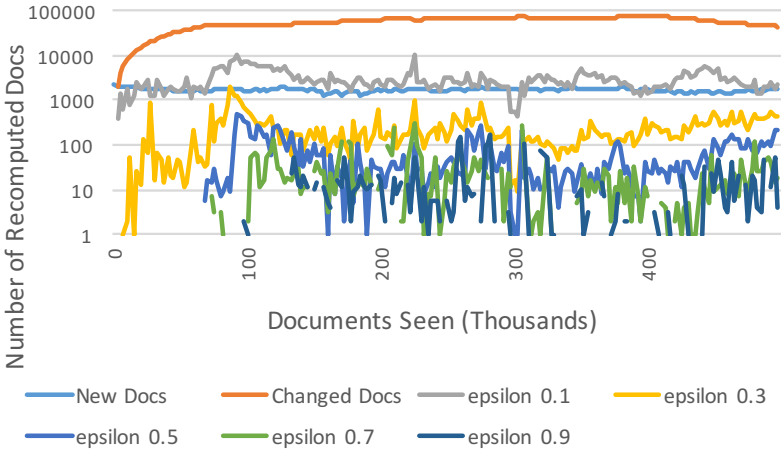


Figure 5.18: Reducing Score Update Frequency by Change Thresholds

be noted that early stop is most effective on short documents like tweets. In turn, most sources with larger texts would not produce the rates we are observing here, side-stepping the need for this final optimization.

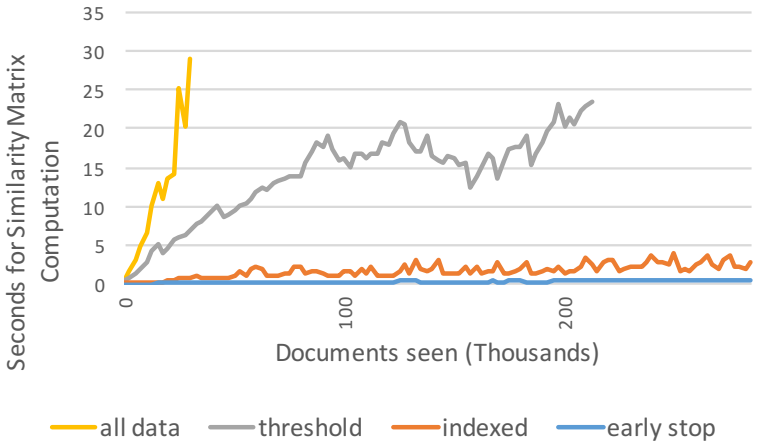


Figure 5.19: Impact of Similarity Matrix Optimizations for Olypics dataset

Last, Figure 5.20 shows the large gains of incremental computations compared to re-computations from scratch for clustering and provenance computations, yielding up to four orders of magnitude speedup. Note here that the quality of results of the incremental computations compared to full re-computation is not being affected. We tested this by computing the Jaccard Index for the clusters produced in both cases. The Jaccard Index measures the similarity between finite sets and is defined as as "the size of the intersection divided by the size of the union of the sample sets". The methodology is the following:

Two variants of the clustering algorithm were computed for equally sized batches:

a) incremental computations, and b) full re-computations in every batch. These two variants resulted in particular clusterings which we need to compare. In particular, the clusters of each variant have to be examined for similarity. In order to identify matching cluster among both variants we computed the Jaccard index for all pairs of clusters among the two variants and selected the most similar for every cluster. This Jaccard index value was averaged for all pairs of similar clusters and was 0.95. As a result, we can safely assume that the clustering results of the incremental case are very similar with the corresponding results of the full re-computation. That lead us to the conclusion that we can trust the results of the incremental variant.

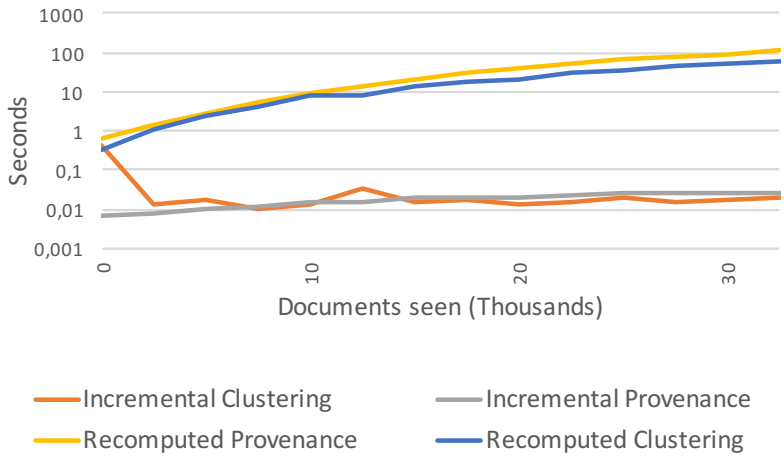


Figure 5.20: Performance Gains for Incremental Computations vs Recomputations for Olympics dataset

5.4.3 News Media Dataset Evaluation

Our second large-scale dataset stems from the IJS newsfeed aggregator[1] and combines the data of 75K RSS feeds, covering more than 40 languages. For this study, we focused on English messages which make about around 50 % of this data set. Compared to the Twitter data evaluated so far, there are two significant differences: 1) The text contents of these messages are significantly larger after tokenization and stemming, on average around 280 terms and up to 30K terms. 2) The message rates are lower, with around a 100-150K messages per day. For consistency, this data set also spanned several days (January 2nd to 4th, 2017, full days) and contained 277500 news articles. The same setup in terms of hardware and software was used as the large-scale Twitter dataset (Section 5.4.2). The only substantial change was a different Tokenizer, replacing the Twitter-specific library with the open-source Lucene Tokenizers and Stemmmers.

[1]http://newsfeed.ijs.si/

Given the lower overall message rates, we now processed batches of 100 messages. Furthermore, we set a lower similarity threshold for clustering (0.5) and θ (0.4) since this data set is much more sparse than the large-volume Twitter dataset.

As a result, the RMSSTD values turned out to be slightly higher, fluctuating between 0.3 and 0.35, which is within the bounds for an acceptable clustering quality.

For ϵ, we increased the value slightly to 0.7 as score changes were more pronounced. In terms of the expiry timeout, we observed that doubling the expiry interval would cover about 15 % more cluster "tails", as more than 85 % of the clusters would have less than half of the time between the last and second-last activity and would not have been expired. Similar to the previous workload we set an activity timeout of 12 hours, i.e., a cluster not getting updates to any of its messages for 12 hours will be discarded.

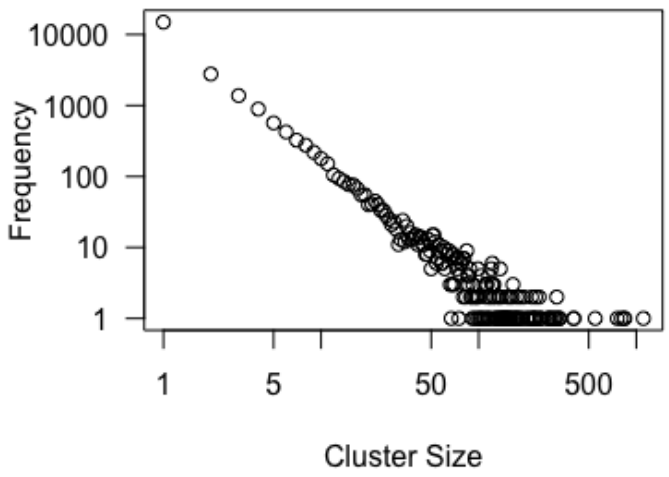


Figure 5.21: Distributions of Cluster Sizes for Newsfeed dataset

Provenance Analysis. In order to evaluate the clustering result, we inspected manually the clusters produced, which accurately cover the news topics of those days. Examples of topics include the North Carolina's transgender bathroom law, the expulsion of 35 suspected Russian spies, the terror attack at Riena night club in Istanbul, and smaller scale events, like sport's matches, etc. After deeper inspection of the fine-grained provenance reconstruction, news updates could be matched to the previously published articles and republishing of the same article could be linked to the original sources. Figures 5.21 and 5.22 demonstrate some characteristics of the clusters produced with the settings from 5.4.3. Figure 5.21 shows the cluster distributions of the last batch. The total number of non singleton clusters is 8808.

We can observe that the cluster size demonstrates a skewed distribution, with the largest cluster containing almost 15K messages. The mean size is 5 (median: 1), which shows that there are many small clusters and singletons and a few large ones (corresponding to emergent news).

The cluster lifetimes in Figure 5.22 show a more balanced distribution, since small clusters with limited activity expire quickly. The median value is approximately 6 hours (mean 7.3), which reflects the large amount of smaller range topical news that are not interesting on a large scale. However, emergent news, like updates for the attack at Reina in Istanbul, follow the typical 24 hours news cycle.

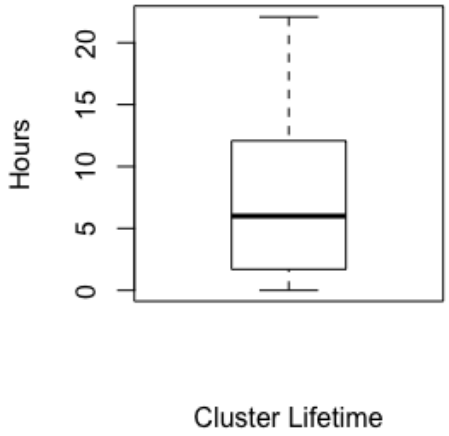


Figure 5.22: Distributions of Cluster Lifetimes for Newsfeed dataset

Performance Analysis. As in the previous dataset, we show the cost per batch for the main components of our computational pipeline Figure 5.23. Again, the cost of all operations becomes stable after a small number of batches, and the processing rates are clearly higher than messages rates: Processing a batch of 100 messages takes up to 10 seconds, yet the arrival rate is around 1-2 messages per second. Comparing the relative costs of the individual pipeline states with those in the Olympics dataset (Figure 5.14) shows how the cost has shifted: The cost of Tf-Idf computation is significantly higher now, as the larger documents contain more terms for computation, which also increases the number of documents with the same term that need to be considered for adaptation. Considering the overall sparsity of the data, clustering and provenance computation are now relatively cheaper.

In terms of working sets (Figure 5.24), a similar effect can be seen. The growth in size levels off quickly for all contributors, starting from active documents to non-singleton clusters. Note that the total number of clusters is very close to number

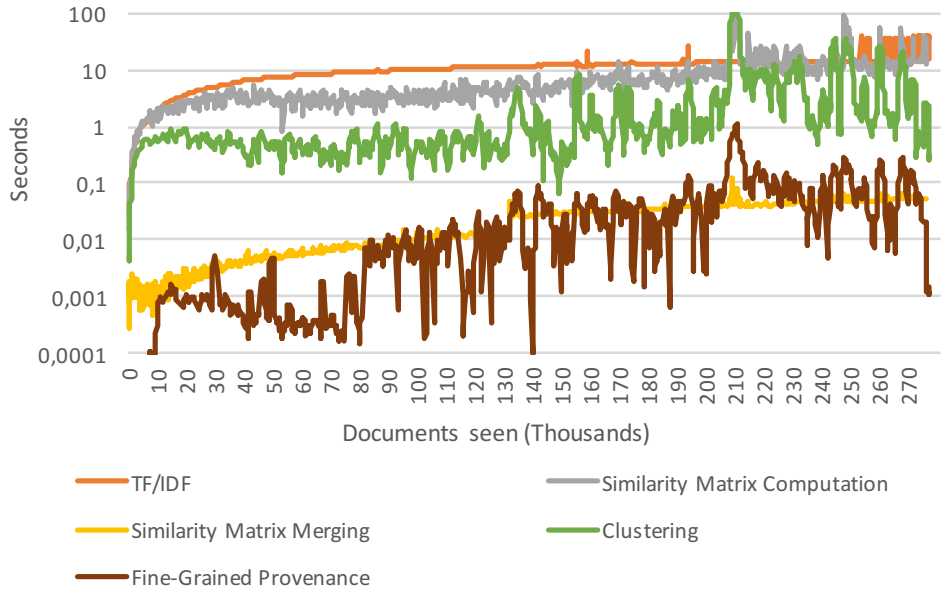


Figure 5.23: Computational Times per # of Documents for Newsfeed dataset

of active documents, about a quarter. A significant difference exists to the Twitter dataset: the inverted index now dominates the space consumption. The number of terms in the inverted index (dictionary) already comes close to the number of similarity pairs, while the number of entries in the document/score lists now exceeds them. Considering that storing a term requires considerably more memory than a similarity score, the actual memory consumption is even higher. This growth can be attributed to the much larger number of tokens per message and the much higher overlap of terms between the messages.

The differences in the data distribution also affect the effectiveness of the optimizations for similarity computation. Indexing has very limited benefits for the news feed dataset, providing only minor improvements for similarity computation or none at all. The reasoning is the following: In contrast to the Twitter dataset many documents share common terms, as the overall number of terms per document is much higher. A a result, when retrieving all the other documents containing the terms in the currently added or updated document, on average about 90% of documents are retrieved.

Early stop, however, does provide more significant benefits, reducing the number of documents for comparison to 35%. As outlined in Section 5.3.4, our strategy is to stop when the remaining terms will no longer be sufficient to lift the document above θ, so the similarity pair is irrelevant. Considering that the number of terms is much higher, their score decreases more gradually and each term will generate more candidate documents. The threshold for stopping was set more aggressively and an additional fixed threshold to clip the long tail of low-score terms is introduced.

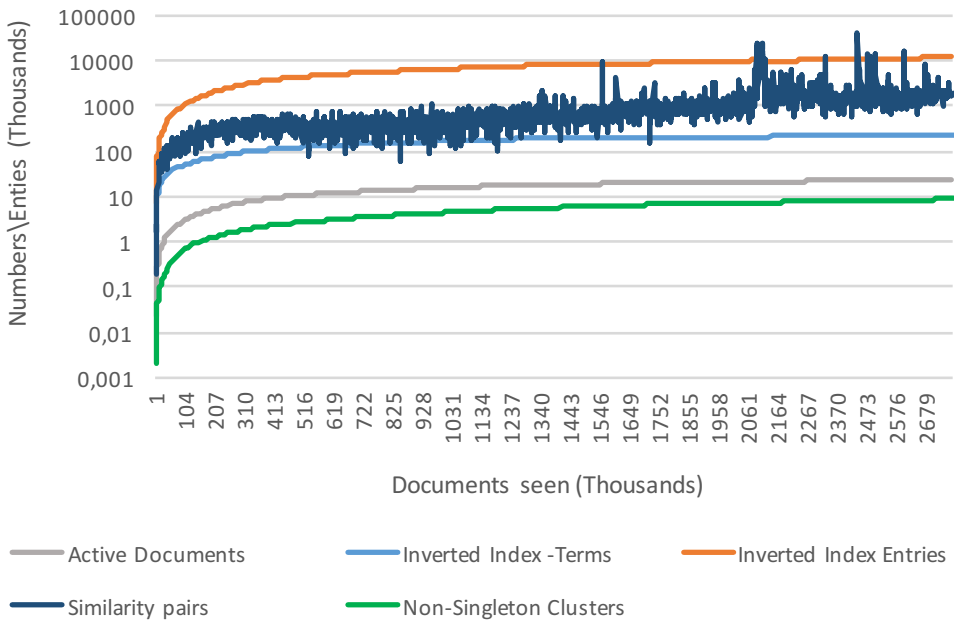


Figure 5.24: Set Sizes over time for Newsfeed dataset

Should high rates of long documents require more optimizations, we are currently considering the following options: (1) More elaborate strategies to determine the early stop such as gradient estimation or more aggressive pruning (2) Means to reduce the number of terms that are considered for similarity, thus reducing both the number of index entries and computation steps. Possible strategies could be more extensive text analysis (e.g., focusing only on certain types of words like nouns or extracted entities) or utilizing different similarity models.

Yet, given these results, we can conclude that our revised algorithm indeed provides the means to scale up our provenance reconstruction approach from small static datasets, to real-world – dynamic and large – datasets, without compromising precision.

5.5 Conclusion and Future Work

The main contributions of the chapter can be summarized in two lines. First, we highlight the importance of implicit interactions and influence that our algorithm can capture by a deep qualitative analysis. Second, we show that is feasible to reconstruct fine grained provenance on social media in an incremental, web-scale way. Our results demonstrate stable computational costs over time, while not affecting the quality of results.

Our analysis indicates that the reconstructed provenance generated by our algorithm is sound and unravels a significant share of online interactions and influence.

By doing that, we offer a more complete picture of information provenance by implicit means which provides insights of how (provenance paths) and why (human interactions) information propagated in particular ways. Moreover, such analysis is implemented in an online fashion, allowing the end user to trace the provenance of data streams in a timely way. Identifying the provenance of information promptly is crucial for online journalists because it enables them to also assess the relevance and trustworthiness of particular messages and news in a timely manner.

Such analysis will open the path for streaming provenance reconstruction on social media and rule-based systems that incorporate observations from our deep qualitative analysis on human interactions. For future work, we plan to further investigate the impact of external influence on provenance reconstruction, and how it can be captured. For our system's optimization, we will exploit techniques for parallelism (possible for many parts of our pipeline) in order to elicit faster computations.

Chapter 6

Modeling Implicit and Explicit Interactions

6.1 Introduction

In the previous Chapters we discussed different types of interactions among users and we analysed them mainly in isolation. In a real life scenario we need to analyze the stream of messages in Twitter containing all interactions which need to be processed at the same time. In a scenario that we trace important events and emergent situations from multiple social media, we need to integrate the output of different social media into one model. In both scenarios, a unified way to express influence edges is needed. This is challenging both from a conceptual and a practical perspective: different platforms have their own functions and ways of expressing influence; different implementations output different formats. In order to process the different output from different interactions and possibly different platforms we need an interoperable model for our data. Such a model should be general enough to cover many use cases but at the same time expressive in order to capture multiple types of influence.

To close this gap, we present the *PROV-SAID* model [TDNV+15], a model to assert the **Prov**enance of **S**ocial medi**A** **I**nformation **D**iffusion based on PROV-DM [MMW13], and its extensions. PROV-DM is the main component of the wider family of W3C PROV documents that defines an interoperable model for provenance.

The PROV specification provides the concepts and supporting definitions to enable the interchange of provenance information in heterogeneous environments such as the web.

PROV-DM [MMW13] provides the means for a generic representation of provenance, but also it caters for specific use cases. PROV-DM is organized in the following components, which represent:

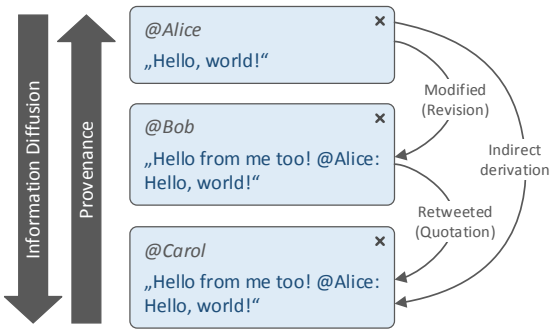


Figure 6.1: Information Diffusion vs Provenance

- entities and activities, and the time at which they were created, used, or ended
- derivations of entities from entities
- agents taking responsibility for entities that were generated and activities that happened
- the notion of bundle, a mechanism to support provenance of provenance
- properties to link entities that refer to the same thing
- collections forming a logical structure for its members

Apart from the main document that describes the model, an additional document specifies the set of constraints that provenance should follow.

PROV-DM has many benefits: concepts of information diffusion and provenance are modeled in a W3C standard that allows subsequent integration and interaction with other tools that make use of PROV. As a result, the cost of integration is lowered since data derived from this modeling are exposed in a web-native and interoperable format. This is very useful in cases where data needs to be combined from different (social media) sources that do not share the same concepts and notations. Additionally, PROV-DM is domain-agnostic, but it has the benefit of extensibility, allowing domain-specific information to be included.

Since PROV-DM is a typical provenance model we clarify the relation between information diffusion and provenance with regard to this modeling in the context of social media. We provide an example in Figure 6.1. Three Twitter users are emitting a similar message: Alice is the source of information diffusion, as she emits an original message. At a later point in time, user Bob modifies the original message and then user Carol copies and forwards (retweets) the message of Bob. In this process, it is important to understand how the message was modified and forwarded. User Carol was indirectly influenced by user Alice, since her message was *indirectly derived* from the source (two-step procedure). This means that the trustworthiness of all three users involved should be judged, since they participate in the diffusion and modification of this message. The backward process of identifying

the sources is what provenance seeks to answer and the forward process of how the message spreads is typical information diffusion.

As our main contribution in this chapter, we introduce a number of new attribute values to extend PROV-DM [MMW13], and relevant extensions to PROV-Constraints [CMW13] to govern the use of these attribute values. In more detail, we provide: (1) a structured ontology for information diffusion and provenance on social media; (2) extensions of entities and activities relevant for information diffusion and provenance; (3) introduction of the use of these new concepts as attributes and roles in PROV assertions; (4) extensions for the vaguely defined concept of *Influence* in PROV-DM. (5) We demonstrate how this model is applied in a scenario of combining different provenance data from Twitter and a news platform.

The publications related with this Chapter are the following:

[TDNV+15] Io Taxidou, Tom De Nies, Ruben Verborgh, Peter Fischer, Erik Mannens, and Rik Van de Walle. Modeling information diffusion in social media as provenance with W3C PROV. In *Proceedings of the 6th International Workshop on Modeling Social Media*, pages 819-824, 2015.

[TFDN+16] Io Taxidou, Peter M Fischer, Tom De Nies, Erik Mannens, and Rik Van de Walle. Information diffusion and provenance of interactions in twitter: is it only about retweets? In *Proceedings of the 25th International Conference Companion on World Wide Web, pages 113-114. International World Wide Web Conferences Steering Committee*, 2016.

[TDNF17] Io Taxidou, Tom De Nies, and Peter M Fischer. Provenance of Explicit and Implicit Interactions on Social Media with W3C PROV-DM. In *Lecture Notes in Computer Science, Springer LNCS/LNAI series*, 2017. Accepted, under publication procedure.

[DNTD+15] Tom De Nies, Io Taxidou, Anastasia Dimou, Ruben Verborgh, Peter M. Fischer, Erik Mannens, and Rik Van de Walle. Towards multi-level provenance reconstruction of information diffusion on social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM'15*, pages 1823-1826, 2015.

6.2 Model Overview

In Sections 6.2, 6.3 and 6.4, we describe our model with its relevant extensions and constraints[1]. PROV has formal semantics [CW13], which cover our model as well, since our extensions and constraints are fully compliant with PROV.

[1]For detailed specification and formal constraints, see `http://semweb.datasciencelab.be/ns/prov-said/`

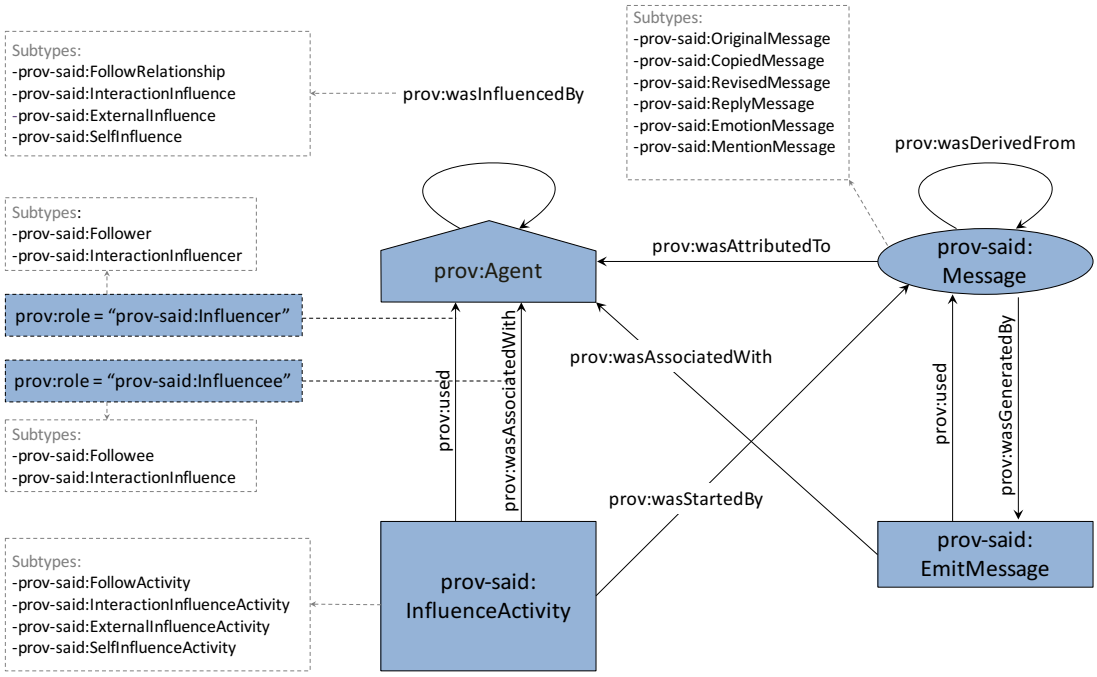


Figure 6.2: Prov-SAID Model

Throughout the text, we provide a full example that covers all aspects of the model. To improve clarity, this example is unfolded incrementally and the reader should take into account information presented in previous examples.

Next we provide an overview of our PROV-SAID model. The PROV-SAID model can be applied to any social network where information propagates from user to user in the form of messages. Messages can be transmitted though *social connections*, but the model is general enough to capture *external influence* as well, as often happens in social media [MZL12]. For example, Twitter users might publish information that has been seen on the public timeline without any direct social connection. Furthermore, our experience with provenance in Twitter shows that information does not flow only from social connections, but there is approximately 20% of external influence [TF14]. The last observation derives from experiments with reweets where diffusion is explicit, while this percentage is much higher for non explicit diffusion (propagation of Twitter hashtags).

Our model includes *activities* and *relationships* connected with information diffusion, such as exchanging *messages*, finding the *source* of diffusion, and expressing which *changes* the message has undergone through this procedure. User *influence* plays a key role in information diffusion since it drives information flow. The concept of influence is only vaguely defined in PROV-DM and it is recommended to use more specific terms when possible, since influence can take many forms in different use cases. However, for our use case the influence relationship has its own merit. Therefore, we define and extend the concept of influence, expressed through different

activities, types and user roles.

Figure 6.2 shows a high-level overview of the PROV-SAID model with its extensions. Throughout this work, the prefix *prov:* refers to the PROV namespace[2] and the prefix *prov-said:* refers to the new PROV-SAID namespace[3]. Users who emit messages on social media are represented by the *prov:Agent* concept. Terms *agent* and *user* are utilized in an alternate way: agent is mostly PROV-SAID model specific, while user is a general term to denote participation in social media.

6.3 Modeling Messages

In order to model messages that are emitted by users, we propose the following extensions that are subtypes of *prov:Entity*:

- *prov-said:Message*: denotes the general class of messages.

  Messages in social media might be original messages, copied messages or revised messages. We define the following categories as subtypes of *prov-said:Message*:

  - *prov-said:OriginalMessage* denotes an original message that is not derived from any other message and the user who emitted it, is the initiator of information propagation for a specific message.
  - *prov-said:CopiedMessage* denotes a message which is based on another message that has been published in the past and was forwarded as an exact copy (such as the *retweet* function of Twitter). Users who emit copied messages comply fully with the content and opinions of the original message.
  - *prov-said:RevisedMessage* denotes a message that is produced by modifying an existing message. This means that the user who emits such a message may or may not share the original opinion of the original message. It is possible that the information carried by the original message is altered.
  - *prov-said:ReplyMessage* denotes a message that is produced by replying to an existing message. A reply is expressed normally by social media providers through an embedded function and often the name of the user who posted the existing message is mentioned. Note here that unless users explicitly annotate replies, it is hard to trace their provenance, since there might be no other characteristics indicating their connection to the existing message.

[2] `http://www.w3.org/ns/prov/#`
[3] `http://semweb.datasciencelab.be/ns/prov-said/`

- *prov-said:EmotionMessage* denotes an empty message that expresses emotions to an existing message. We decided to model it as an empty message because in most platforms the existing message appears in the timeline of the user who expressed an emotion towards it. For example, in Twitter a message is *favorited*, in Facebook *liked*, while recently users can express more emotions like anger, surprise, sadness, etc.
- *prov-said:MentionMessage* denotes a message that mentions another user. Such a message might be complementary to any type of the above mentioned. For example, a *revised message* might mention another user. We have empirically identified the following semantics of a mention: 1) a user mentions another popular user (e.g. celebrity, politician); 2) a user mentions another user to expose information to them; 3) a user tags another user in a message to express a shared participation (e.g. being physically at the same place). We do not model such cases separately due to the difficulty of their detection.

Note here that the types *prov-said:OriginalMessage*, *prov-said:CopiedMessage* and *prov-said:RevisedMessage* are strictly disjoint. *prov-said:ReplyMessage* and *prov-said:EmotionMessage* on the other hand, are only disjoint with the type *prov-said:OriginalMessage*. For example, we have observed on Twitter that a *prov-said:ReplyMessage* can also be a *prov-said:CopiedMessage*. Finally, *prov-said:MentionMessage* can be complementary to any other type of message type, including *prov-said:OriginalMessage*. The focus of the latter type of *prov:Message* lies in the interaction of *prov:Agent*s, rather than *prov:Message*s.

With these six cases, we have covered the main cases of information diffusion and interactions through messages that we see in the empirical data we have gathered.

6.3.1 Message Attribution

A *prov-said:Message* is always attributed to a user *prov:Agent* using the relationship *prov:wasAttributedTo*. Example 6.1 illustrates the use of messages and attribution for the Twitter social network.

Example 6.1 (Message Creation and Attribution).

```
1  prefix twitter: <http://twitter.com/>
2  prefix alice-status: <http://twitter.com/Alice/status/>
3  prefix bob-status: <http://twitter.com/Bob/status/>
4  prefix carol-status: <http://twitter.com/Carol/status/>
5
6  // User @Alice tweeted a message "Hello, world!"
7  prov:entity(alice-status:12345, [prov:type='prov-said:OriginalMessage',
8     prov:label='Hello, world!'])
9
10 // User @Bob modified and re-emitted the "Hello, world!" message
11 prov:entity(bob-status:23456, [prov:type='prov-said:RevisedMessage',
```

```
12     prov:label='Hello from me too! MT @Alice: Hello, world!'])
13
14  // User @Carol retweeted (copied) the revised message
15  prov:entity(carol-status:34567, [prov:type='prov-said:CopiedMessage',
16     prov:label='Hello from me too! MT @Alice: Hello, world!'])
17
18  // User @Dan replied to Alice's message
19  prov:entity(dan-status:45678, [prov:type='prov-said:ReplyMessage',
20     prov:label='Hi @Alice!'])
21
22  // User @Eve favorited Alice's message
23  prov:entity(eve-status:56789, [prov:type='prov-said:EmotionMessage',
24     prov:label='Favorite'])
25
26  // User @Frank mentioned Alice in his message
27  prov:entity(frank-status:67891, [prov:type='prov-said:MentionMessage',
28     prov:label='@Alice just said hello'])
29
30  // alice-status:12345 was emitted by twitter:Alice
31  prov:wasAttributedTo(alice-status:12345, twitter:Alice)
```

□

6.3.2 Message Emission

Next we define the following activity that refers to message emission and is a subtype of *prov:Activity*

- *prov-said:EmitMessage* denotes a generic emission of a message. It must generate a *prov-said:Message*, and may use another *prov-said:Message*.

Note that the subtype of the generated *prov-said:Message* can be inferred from the usage of another *prov-said:Message* by the *prov-said:EmitMessage*. For example, if the content of the generated message is identical to that of the used one, it is a *prov-said:CopiedMessage*. The same applies for the other types of *prov:Message*s.

6.3.3 Message Derivation

Whereas an original message does not have dependencies on other messages, copied, revised, reply and emotion messages can be traced back to their original sources through derivation - i.e., they cannot exist on their own. PROV-DM already provides the basics for the concepts needed to model copied and revised messages, in the form of *prov:Quotation*[4], *prov:Revision*, and *prov:PrimarySource*, as illustrated by Example 6.2. To cover the newly defined interactions, we add the types *prov-said:Reply* and *prov-said:Emotion* as subtypes of *prov:Derivation*.

[4]Note here that a quote in Twitter is a *prov-said:RevisedMessage* and a retweet is a *prov-said:CopiedMessage*

Example 6.2 (Message Derivation).

```
// bob-status:23456 was derived from alice-status:12345,
// which is also its primary source (in the context of Twitter)
prov:wasDerivedFrom(bob-status:23456, alice-status:12345, emit-23456, gen-23456,
   use-12345, [prov:type='prov:Revision', prov:type='prov:PrimarySource'])

// carol-status:34567 was quoted from bob-status:23456
// (which is not its primary source)
prov:wasDerivedFrom(carol-status:34567, bob-status:23456, emit-34567, gen-34567,
   use-23456, [prov:type='prov:Quotation'])

// dan-status:45678 was replied to alice-status:12345
prov:wasDerivedFrom(dan-status:45678, alice-status:12345, emit-45678, gen-45678,
   use-12345, [prov:type='prov:Reply'])

// eve-status:56789 expressed an emotion towards alice-status:12345
prov:wasDerivedFrom(eve-status:56789, alice-status:12345, emit-56789, gen-56789,
   use-56789, [prov:type='prov:Emotion'])
```

□

We observe that *carol-status:34567* was indirectly derived from *alice-status:12345*. To model this dependency, we introduce the concept *prov-said:IndirectDerivation* as a subtype of *prov:Derivation*. This way we can model multi-step provenance and trace how messages are being derived, without being restricted to the previous step only. We illustrate this in Example 6.3.

Example 6.3 (Indirect Derivation).

```
// carol-status:34567 was indirectly derived from alice-status:12345
prov:wasDerivedFrom(carol-status:34567, alice-status:12345,
   [prov:type='prov-said:IndirectDerivation'])
```

□

Note that the *prov-said:MentionMessage* might not be derived from another *prov:Message*. As a result, we do not need to express any additional sub-type of *prov:Derivation*. In case a *prov-said:MentionMessage* is derived from another one, this will be covered by the other defined sub-types of *prov:Derivation* (e.g. *prov:Quotation*).

At this point, we express the following constraints:

- An *prov-said:OriginalMessage* cannot be derived from a *prov-said:Message*.
- A copied, revised, reply, emotion message should always be derived from another message. A *prov-said:EmitMessage* activity that generates a *prov-said:CopiedMessage* and uses a *prov-said:Message* implies that the first message was derived from the latter by means of *prov:Quotation*. Analogously, generation of a *prov-said:RevisedMessage* and usage of a *prov-said:Message* by a *prov-said:EmitMessage* implies that the first message was derived from

the later by *prov:Revision*. The same applies for *prov-said:ReplyMessage* and *prov-said:EmotionMessage* with the derivation types *prov-said:Reply* and *prov-said:Emotion*.

In the next Section we continue with modeling influence with regard to information diffusion.

6.4 Modeling Influence

When users (*prov:agent*s) are interacting by exchanging and forwarding messages, expressing emotions or mentioning other users, influence is created among them. Recall the definition of [MMW13], where influence denotes to have an *effect* on something. Once message *prov:Derivation*s are expressed, we can also express who influences whom by identifying the authors of the messages. Note the exception of *prov-said:MentionMessage*, where influence is created instantly among the two agents involved (the existence of an influencing message is not necessary).

Influence plays a key role since it drives information flow in social media; however, it is challenging to capture its semantics and compute its sources. Users often react to messages by giving credit to the original contributor. However, they are also frequently influenced by external factors such as traditional media, and they react in social media [MZL12]. Influence has many ways of expression: through establishing social connections, interacting or expressing emotions over messages, mentioning other users, or reacting as a result of real events. In this Section, we propose extensions for influence types, influence activities and influence roles on top their generic and rather vague definitions in PROV-DM.

6.4.1 Influence Types

We define a relationship: *prov-said:InfluenceRelationship* to express general influence and four subtypes to specify the ways that such influence can be expressed. Note here that by the term *influence* we do not refer to the classical influence expressed in state-of-the-art for social media analysis, but we rather follow the general definition of PROV-DM [MMW13] *"Influence is the capacity of an entity, activity, or agent to have an effect on the character, development, or behavior of another by means of usage, start, end, generation, invalidation, communication, derivation, attribution, association, or delegation."* This way all social interactions are captured: for example reply messages that do not demonstrate the typical social media influence, but an existing message is *having an effect* on the reply message which shows the PROV-DM influence.

Note that on the one hand, our model allows the representation of social connections among users, since information flows through them in the majority of cases. On the

other hand, the model is generic enough to assert the provenance of information diffusion even without the presence of social connections.

First, we define a generic concept *prov-said:InfluenceRelationship* as a subtype of *prov:Influence*. It denotes an influence between agents in the context of social media. We then define the following subtypes of this generic relationship:

- *prov-said:FollowRelationship* denotes that one agent was influenced by another agent, by establishing a unidirectional (follow) relationship. In the context of social media, that practically means being exposed to the messages emitted by the latter. For example, in Twitter this is the only way of connecting with users, while Facebook apart from the bidirectional *Friendship*, also gives the possibility of unidirectional connection by subscribing to the messages of users. Here, we assume that once an agent starts to follow another agent, he is exposed to his old messages and his future ones (if there are any). This is also the case in Twitter and Facebook. As a result, we do not model the influence that derives from exposure/subscription to messages explicitly, since it is implied by the *prov-said:FollowRelationship*.

  Furthermore, such a relationship entails a certain degree of uncertainty, since it can not be asserted whether an agent has seen the messages of another in reality. Note that on the one hand, our model allows the representation of social connections among users, since information flows through them in the majority of cases. On the other hand, the model is generic enough to assert the provenance of information diffusion even without the presence of social connections.

- *prov-said:InteractionInfluence* denotes that an agent was influenced by another agent through interaction – i.e., by quoting, revising, replying, or expressing emotions towards messages of the latter, or by mentioning the latter. The message types give us more information about the type of interaction occurred. This type of influence can possibly be discovered through the operations of social media platforms, or through message similarity.

- *prov-said:ExternalInfluence* denotes that an agent was influenced by a possibly unknown user and/or external event. Given that events that happen in reality are reflected in social media, we encounter messages that share certain similarity, but are not revised messages. An external event or meme drives the similarity in most of these cases. We decide to model external influence explicitly, since it is very prominent in social media and also to differentiate it from message revision. An indicator that distinguishes these two cases might be the existence of social connection: in case that two messages share certain similarity and their authors are not connected, there is high probability of external influence. However, it is often the case that users forward messages from the public timeline without the existence of a social relationship.

- *prov-said:SelfInfluence* denotes that an agent was influenced by himself. In social media we often encounter cases that a user is emitting similar messages

in order to promote again specific content, or to edit messages.

We illustrate the influence types in Example 6.4.

Example 6.4 (Influence Types).

```
1  // User @Alice followed user @Carol, so a prov-said:FollowRelationship
2  // existed between them
3  prov:wasInfluencedBy(twitter:Alice, twitter:Carol,
4     [prov:type='prov-said:FollowRelationship'])
5
6  // User @Bob revised a message from @Alice, so a
7  //prov-said:InteractionInfluence existed between them
8  prov:wasInfluencedBy(twitter:Bob, twitter:Alice,
9     [prov:type='prov-said:InteractionInfluence'])
10
11 // User @Bob was influenced by an external event
12 // and the influencing agent is not specified
13 prov:wasInfluencedBy(twitter:Bob, -,
14    [prov:type='prov-said:ExternalInfluence'])
15
16 // User @Bob was influenced by himself by propagating
17 // similar messages, as a result the agents of
18 // influencer and influencee are the same
19 prov:wasInfluencedBy(twitter:Bob, twitter:Bob,
20   [prov:type='prov-said:SelfInfluence'])
```

□

By following these influence types, both the social graph and the interaction graph [WSPZ12] can be reconstructed at a certain point in time by using provenance. The interaction graph aggregates interactions (e.g., emission of messages) among users as weighted edges.

6.4.2 Influence Activities

Additionally to the influence types expressed as relationships among users (subtypes of *prov:Influence*), we explicitly model the corresponding activities. This design decision offers greater expressiveness by providing more information about the time the influence relationships started and ended, what triggered them, etc. For these purposes, we introduce the following subtypes of *prov:Activity*:

- *prov-said:InfluenceActivity* is subtype of prov:Activity. It denotes the activity of one agent influencing another with the following four subtypes:
- *prov-said:FollowActivity* denotes the activity of one agent to establish a uni-directional connection with another. Once such an activity starts, the first agent is exposed to the (future and past) message emissions of the latter. This activity has a start time that denotes the time of establishing the connection and an optional end time in case the agent removes the connection with regard to the other agent.

- *prov-said:InteractionInfluenceActivity* denotes the activity of one agent influencing another, so that the latter reacts to the messages of the first. Note here that the *prov-said:InteractionInfluenceActivity* is instantaneous, and thus has the same start and end time (when the influenced message was emitted). In this way, we are able to model multiple interactions of agents by generating multiple instances of *prov-said:InteractionInfluenceActivity*. If we had considered the opposite case when *prov-said:InteractionInfluenceActivity* is asserted only once and has no end time, we would have come to contradiction with the principles of information diffusion, where the significance of past interactions fades quickly over time.
- *prov-said:ExternalInfluenceActivity* denotes the activity of an agent being influenced by an external, possibly unknown entity.
- *prov-said:SelfInfluenceActivity* denotes the activity of an agent influencing him or herself.

Example 6.5 illustrates the subtypes of *prov-said:InfluenceActivity*.

Example 6.5 (Influence Activities).

```
1  // A prov-said:FollowActivity was started at the moment user @Alice followed
2  // user @Carol. Since @Alice was still following @Carol at the time of assertion,
3  // there is no end time for the activity.
4  activity(alice-follows-carol, 2015-01-09T13:00:00, - ,
5     [prov:type='prov-said:FollowActivity'])
6
7  // A prov-said:InteractionInfluenceActivity was started (and ended)
8  // at the moment user @Bob modified and re-emitted the message of @Alice.
9  activity(bob-influencedby-alice, 2015-01-09T13:05:00, 2015-01-09T13:05:00,
10    [prov:type='prov-said:InteractionInfluenceActivity'])
11 wasStartedBy(bob-influencedby-alice, bob-status:23456, emit-23456,
12    2015-01-09T13:05:00)
13 wasEndedBy(bob-influencedby-alice, bob-status:23456,
14    emit-23456, 2015-01-09T13:05:00)
```

□

6.4.3 Influence Roles

Analysis of information diffusion and influence in social media makes use of specific roles for agents [BHMW11]. To model this, we need to specifically define values for the *prov:role* attribute in the context of *prov:Usage* and *prov:Association*. This way, we clarify the roles of agents involved in a *prov-said:InfluenceActivity*. We define the following role-values:

- *prov-said:Influencer* denotes the role of an agent that was used by an *prov-said:InfluenceActivity* that was associated with another agent. This means that the first agent influences the latter.

- *prov-said:Influencee* denotes the role of an agent that was associated with an *prov-said:InfluenceActivity*. This agent is being influenced by another agent used by the same *prov-said:InfluenceActivity*.

Following that, we define two subtypes of *prov-said:Influencee* and *prov-said:Influencer* respectively. Firstly, we model the follow relationship with the roles: *Follower* and *Followee* and secondly, we model the activity of interaction by exchanging messages with the roles: *InteractionInfluencee* and *InteractionInfluencer*. Note that these roles are pairwise complementary by revealing the active behaviour of one agent in to establish social connections and to react to messages (follower, interactionInfluencee) and the passive behaviour of another (followee, interactionInfluencer) who exerts some influence on the first.

- *prov-said:Followee* denotes the role of an agent that was used by a *prov-said:FollowActivity* associated with another agent. This means that the latter followed the first.
- *prov-said:Follower* denotes the role of an agent that was associated with an *prov-said:FollowActivity*. This means than an agent establishes a unidirectional connection with another agent in social media.
- *prov-said:InteractionInfluencer* denotes the role of an agent that was used by an *provsaid:InteractionInfluenceActivity* associated with another agent. This means that the first agent is influencing the latter so that the latter reacts to the messages of the first.
- *prov-said:InteractionInfluencee* denotes the role of an agent that was associated with an *prov-said:InteractionInfluenceActivity*. This means that the agent is being influenced by another agent by reacting to the messages of the latter.

We demonstrate these roles in Example 6.6.

Example 6.6 (Influence Roles).

```
1  used(alice-follows-carol, twitter:Carol, [prov:role='prov-said:Followee'])
2  wasAssociatedWith(alice-follows-carol, twitter:Alice,
3     [prov:role='prov-said:Follower'])
4
5  used(bob-influencedby-alice, twitter:Alice,
6     [prov:role='prov-said:InteractionInfluencer'])
7  wasAssociatedWith(bob-influencedby-alice,  twitter:Bob,
8     [prov:role='prov-said:InteractionInfluencee'])
```

□

Note here that we do not define a specific *prov:role* for the influence types *prov-said:ExternalInfluence* and *prov-said:SelfInfluence*. We argue that the more generic *prov-said:Influencer* and *prov-said:Influencee* suffice to model the roles in these

cases. The type of influence is demonstrated by the type of relationship and activity.

At this point we express the following constraints:

- We ensure that an *prov-said: InfluenceRelationship* always implies a *prov-said:InfluenceActivity*, *prov:Usage* and *prov:Association.* According to the type of *prov-said:InfluenceActivity*, specific *prov:roles* are being used.
- A *prov-said:InteractionInfluenceActivity* starts (and ends, since it is defined to be instantaneous) with the emission of *prov-said:CopiedMessage*, *prov-said: RevisedMessage*, *prov-said:ReplyMessage*, *prov-said:EmotionMessage* or *prov-said:MentionMessage*.
- A *prov-said:SelfInfluence* implies that the *prov:Agents* that influence each other are the same.

With these concepts, we have covered the model of influence with its possible expressions in activities, relationships and roles.

6.5 Towards Combining Different Means of Interactions

The aim of this section is to provide the methodology to reconstruct different means of interactions and output PROV-SAID. We consider two types of interactions in Twitter, which include retweets and similar messages that share some provenance, as a starting point. The proposed methodology can be applied to other influence interactions presented in this thesis. The provenance of such interactions carries different uncertainty levels. As discussed in Section 4.3, retweets (source based interactions) carry medium uncertainty and similarity based provenance carry high uncertainty. For future work, we plan to express statements for uncertainty and even quantify it more precisely. By this methodology, (1) we provide a real-world application and evaluation of the PROV-SAID model. (2) We also show how to map social media data to RDF (Resource Description Framework[5]), so that we can apply the PROV-SAID statements. Additionally, (3) we provide a methodology to build influence graphs of combined interactions. As an example, we combine retweets for explicit interactions and similar messages through clustering for implicit interactions.

6.5.1 Method

We reconstruct and combine explicit (retweets) and implicit interactions (similar messages) on Twitter based on the methodology described in Sections 4.3 and 5.2.

[5]https://www.w3.org/RDF/

In Figure 6.3, we show how we combine two different types of interactions. The general idea is that the roots of the retweet cascades participate in the reconstruction of implicit interactions (clustering) in the upper part of Figure 6.3. In the lower part of that Figure is shown how retweet cascades are unfolded.

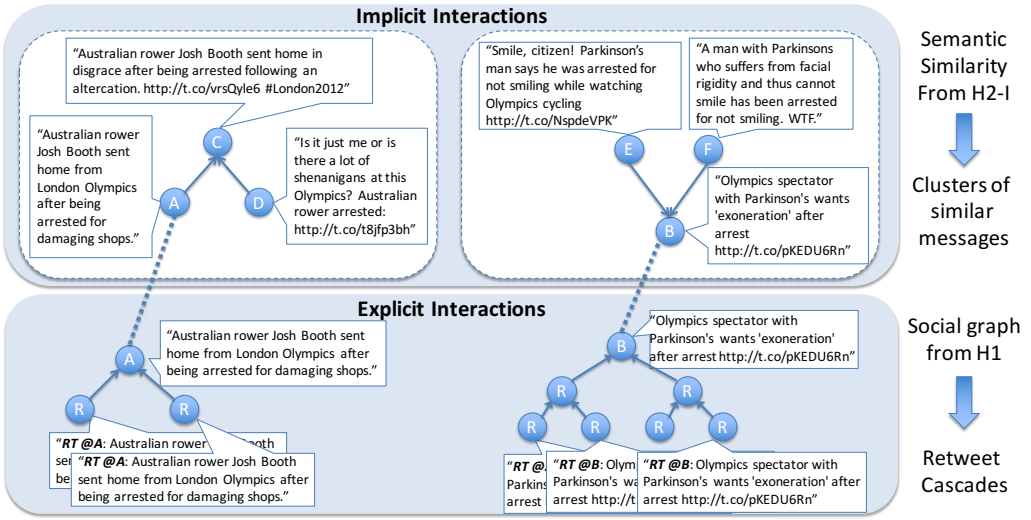


Figure 6.3: Overview of the integrated provenance reconstruction. The arrows for the explicit interactions refer to `prov:wasQuotedFrom` for all copied messages (retweets); for the implicit interactions, they refer to `prov:wasRevisionOf` for all modified messages.

Provenance for Implicit Interactions. We reveal influence paths paths on two levels: (1) *explicit interactions*, based on structural aspects (social graph) as in Section 4.3 and (2) *implicit interactions*, based on content similarity as in Section 5.2. These methods are then combined using the *PROV-SAID* model. The output of retweet reconstruction that we use as an example of explicit interactions is outputed in XML, while implicit interactions output RDF directly. In order to convert the XML-based influence computation from Section 4.3 into PROV-SAID, we use the *RML mapping language* [DVSC+14]. RML is used in combination with a processor to convert proprietary data – such as XML – to RDF. In our case, the data is converted to PROV-O, which is the ontology that expresses the PROV Data Model. Note that by using RML, we ensure that any input can be converted to PROV-O, rendering our method interoperable and reusable in many applications.

Provenance for Explicit Interactions. To obtain explicit interactions, we use as an example retweet reconstruction from Section 4.3. Influence paths are reconstructed according to Hypothesis H1 from Chapter 3 (Users are exposed and influenced by the content of their social connections). For more details, we guide the reader to Section 4.3.2.

The retweet reconstruction algorithm outputs *edges*, directed from a *tweet A* to a *tweet B*. For each tweet, we have access to the *tweet-id*, *timestamp* and *userid*. When we map this to PROV-SAID using RML, we obtain the following PROV-O sub-graph for each edge:

```
status:tweetA_id  a  prov-said:Message ;
  prov:wasAttributedTo  user:tweetA_userid ;
  prov:wasGeneratedBy  _:emit-tweetA_id ;
  prov:generatedAtTime  tweetA_time .
status:tweetB_id a prov-said:CopiedMessage;
  prov:wasAttributedTo  user:tweetB_userid ;
  prov:wasQuotedFrom  status:tweetA_id ;
  prov:wasGeneratedBy  _:emit-tweetB_id ;
  prov:generatedAtTime  tweetB_time .

user:tweetA_userid  a  prov:Agent .
user:tweetB_userid  a  prov:Agent ;
  prov:wasInfluencedBy user:tweetA_userid ;
  prov:qualifiedInfluence  [
    a prov-said:FollowRelationship ;
    prov:agent user:tweetA_userid . ] ;
  prov:qualifiedInfluence  [
    a prov-said:InteractionInfluenceRelationship;
    prov:agent user:tweetA_userid . ] ;

_:emit-tweetA_id a prov-said:EmitMessage .
_:emit-tweetB_id a prov-said:EmitMessage ;
  prov:used  status:tweetA_id .
```

Note that the prefixes "`status:`" and "`user:`" refer to "`https://twitter.com/statuses/`" and "`https://twitter.com/intent/user?user_id=`", respectively, and that the prefixes "`prov:`" and "`prov-said:`" refer to their respective namespaces. This representation of the information cascades as provenance is now suitable to be merged with other interoperable provenance, such as the explicit interactions' provenance described in Section 6.5.1.

To obtain implicit interactions, we use our approach from Section 5.2. We are based in the Hypothesis H2-I from Chapter 3 (H2-I: If two messages are highly similar, there is a high probability that they share some provenance). Since our goal here it to demonstrate how to combine different interaction types, we simplify the method described in Section 5.2 and we omit H2-II, which introduces additional intra cluster edges.

With this method we end up with clusters of similar messages. The reconstructed provenance according to PROV-SAID is the following.

For each cluster:

- identify the oldest message as the root message of that cluster;
- connect all other messages to the root message:
  - if the message is identical to the root message, then use a `prov:wasQuotedFrom` relationship;

- if the message is not identical to the root message, the use a `prov:wasRevisionOf` relationship.

By doing that, we introduce new information to the implicit influence edges: we differentiate edges that show *prov:Quotation* and the ones that show *prov:Revision*, according to the PROV-SAID model. Additionally, based on H2-II from Chapter 3 we decrease the uncertainty of those edges that show *Quotation*. (H2-II: The higher the similarity, the highest the probability that they share some provenance)

Integration of Combined Provenance. Because both algorithms output interoperable PROV, the integration of the two aforementioned levels of provenance consists of simply merging the two sets of RDF statements. However, it is important to understand the new structure that this will give to the data. We clarify how the data is enriched by the combination of the two reconstructed provenance sets as shown in Figure 6.3.

Each level of provenance differs in precision and granularity. The explicit interaction provenance is very detailed, and was constructed with low uncertainty, since it consists solely of copied messages exposed by a social media API (in our case: the Twitter API). The implicit interaction provenance, however, was constructed with much higher uncertainty, relying on semantic similarity to reconstruct connections that were unknown to the social media API. The two levels enrich each other, providing previously unidentifiable connections between messages for data consumers (e.g., social media analysts) to explore.

6.5.2 Evaluation

As a preliminary evaluation, we tested our approach on a dataset gathered using the Twitter Streaming API during the 2012 Olympics from Section 4.3.3. Here, we limited the dataset by only considering tweets with a certain keyword, in our case: *"arrest"*. This simulates a realistic scenario where a social media analyst first searches for a broad keyword (e.g., a trending topic), and then investigates the information diffusion paths among the results. The final dataset consists of 9047 tweets, of which 5174 are copied messages (retweets), and 3873 are original messages according to the Twitter API.

Explicit Interactions. We identified 31 cascades using the low-level reconstruction approach from Section 6.5.1, resulting in a skewed distribution from 5 to 1771 recorded retweets with the root tweet contained in the dataset (out of the total of 5174 retweets). This approach has already been thoroughly evaluated in Section 4.3.3, as a result we can safely assume that the identified cascades are correct.

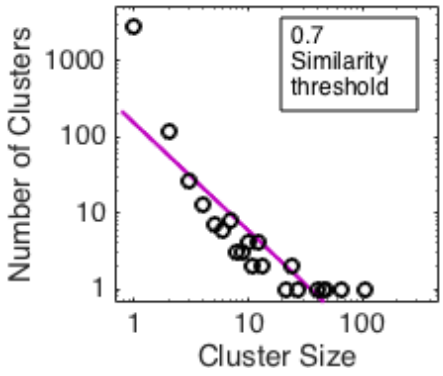


Figure 6.4: Number of clusters for the 0.7 similarity threshold.

Implicit Interactions. Using the approach described in Section 6.5.1, we clustered the 3873 original messages from the dataset based on their semantic similarity. We choose a similarity threshold of 0.7 similar to the evaluation in Section of 5.4.1, where we showed that it is a "safe" threshold. Figure 6.4 shows the distribution of cluster sizes for threshold 0.7. We observe the skewed distribution of cluster sizes with the largest cluster having more than 100 messages and around 1200 singletons. In between there are a lot of small clusters.

Using this threshold, we generated a set of 3094 clusters, and used the 206 non-singletons to reconstruct 879 provenance relationships (32 *prov-dm:Quotations* and 847 *prov-dm:Revisions*).

In other words, when we integrate the 31 cascades discovered with the explicit interactions reconstruction to the implicit interactions, we effectively introduce 879 new connections that were previously unidentified. This creates much larger graphs for the consumers of the provenance data to analyse, and provides an enriched view on the information diffusion process. The entire reconstructed provenance graph can be downloaded at `http://semweb.mmlab.be/ns/prov-said/cikm2015.ttl`

Note that here our goal is not to analyse in depth implicit interactions, as in Section 5.4, instead we are interested to investigate how the output of explicit and implicit interactions could be merged under the same model.

6.6 Conclusion and Future Work

To sum up, PROV-SAID enables systems that analyze social media to incorporate provenance data in their information diffusion analysis. This will benefit the massive human-centric efforts for judging relevance and trustworthiness of information by unraveling its sources and intermediate steps. Particularly, for cross platform provenance reconstruction, such modeling is very useful, because it allows the integration of influence edges coming from different sources into one model.

We use a small dataset to evaluate the PROV-SAID model and the integration of different interactions. We show how different interactions enrich accordingly the reconstructed provenance (influence). For future work, we are going to evaluate more complex interactions' reconstruction and the derived influence graphs.

Chapter 7

Conclusion

We conclude this thesis by discussion our contributions, the advances to state-of-the-art and future work. We provide the methods and systems to trace information diffusion on a real-time fashion. The goal of the thesis is to provide a range of alternatives and fine-grained methods for influence computation. We trace the provenance of explicit and implicit information diffusion, by unraveling latent influence in social media. We strive to implement those computations in an incremental and online fashion, which renders our results usable while things are happening and not after the fact. We have shown that it is possible to handle the speed and rates of current social media without compromising in result quality.

Therefore, the contributions of this thesis are summarized in the following. First, we bridge two different communities, information diffusion and provenance, which look at the same problem from different dimensions and target different use cases. To solve our problem, we leveraged ideas from both information diffusion and provenance and in turn our contributions benefit both communities.

Second, with our work, we bridge the gap of complex analysis for information diffusion and provenance computation with the real-time methods that are focused on specific scenarios (like event detection). We prove that by leveraging our hypothesis, often taken from sociology and in particular from online user behaviour, the quality of results is maintained high.

Third, we shed light on implicit interactions that have not seen much attention in literature: state-of-the-art investigates the diffusion processes in a coarse-grained manner, but our focus is on individual interactions. We provide a thorough analysis into user conventions and latent ways of attributing influence. Our analysis paves the way for better understanding of online behaviour and influence.

Fourth, we solve an untackled problem in the literature, which is iterative stream processing combined with computations over huge social graphs. We leverage several hypothesis form the nature of our data and online user behaviour combined with distributed techniques in order to compute influence in a fast and scalable way.

Fifth, we pave the way towards combining those different interactions on the same data model, based on the W3C PROV-DM specification. By doing that we allow the integrating of different interactions from multiple systems to be expressed under the same model.

Engaging with both conceptual and technical problems taken from different communities, there are many possibilities for future work that will further enhance the tracing of information diffusion in the online world.

We have shown that our solutions provide meaningful results in Twitter, as a representative social platform. We complement our experiments for provenance reconstruction with a news article dataset, but more datasets are needed to further strengthen the relevance of our methods. As a result, a thorough investigation and evaluation is needed in other social media platforms in order to further test our hypothesis and broaden the scope. Our vision is a system that is able to integrate and compute influence from many different online sources in a real time fashion. Research that combines many different social media has not seen much attention, given the diversity and the difficulty to understand the inter-relations.

Another problem which has not seen much attention is understanding the effects of missing data in social media and providing methods for compensation. We observe and quantify the impact of missing data, but developing methods for a full compensation is a very expensive problem (given the huge search space) and out of the scope of this work.

Similar to missing data, external influence is under-researched in state-of-the-art, due to the challenges of being properly captured and understood. Given our methods for implicit interactions, external influence plays a key role in trustworthiness and credibility assessment. Also, external influence identification will help us to improve the precision of our explicit methods: our similarity methods create over-eagerly influence edges, which might not imply influence from other message, but rather external influence.

Another crucial topic is trustworthiness and relevance assessment. In this thesis, we try to provide indications about the comparative uncertainty among our methods. Fully computing trustworthiness and uncertainty is out of the scope for this work and demands the development of fine-grained models and methods.

Lastly, we provide insights how different interaction types can be combined under the same model. Yet, a full evaluation over combining all our methods in a real-time fashion is left for future work. We strongly believe that combined information cascades will update the existing analysis and methods that considered single interaction types so far.

Bibliography

[ADZ+14] Ailifan Aierken, Delmar B Davis, Qi Zhang, Kunal Gupta, Alexander Wong, and Hazeline U Asuncion. A multi-level funneling approach to data provenance reconstruction. In *IEEE 10th International Conference on e-Science*, volume 2, pages 71–74. IEEE, 2014.

[AHSZ11] Mohammad Al Hasan, Saeed Salem, and Mohammed J Zaki. Simclus: an effective algorithm for clustering with a lower bound on similarity. *Knowledge and information systems*, 28(3): 665–685, 2011.

[ALTY08] Nitin Agarwal, Huan Liu, Lei Tang, and Philip S Yu. Identifying the influential bloggers in a community. In *Proceedings of the 2008 international conference on web search and data mining*, pages 207–218, 2008.

[AMRW12] Foteini Alvanaki, Sebastian Michel, Krithi Ramamritham, and Gerhard Weikum. See what's enblogue: real-time emergent topic identification in social media. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 336–347. ACM, 2012.

[AS12] Joel Azzopardi and Christopher Staff. Incremental clustering of news reports. *Algorithms*, 5(3): 364–378, 2012.

[AW12] Sinan Aral and Dylan Walker. Identifying influential and susceptible members of social networks. *Science*, 337(6092): 337–341, 2012.

[AXV+11] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics, 2011.

[Bar11] Geoffrey Barbier. *Finding Provenance Data in Social Media*. PhD thesis, Arizona State University, 2011.

[BBHM13] Raquel A Baños, Javier Borge-Holthoefer, and Yamir Moreno. The role of hidden influentials in the diffusion of online information cascades. *EPJ Data Science*, 2(1): 6, 2013.

[BCJC15] Samuel Barbosa, Roberto M Cesar-Jr, and Dan Cosley. Using text similarity to detect social interactions not captured by formal reply mechanisms. In *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pages 36–46. IEEE, 2015.

[BF16] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 u.s. presidential election online discussion. *First Monday*, 21(11), 2016.

[BFGL13] Geoffrey Barbier, Zhuo Feng, Pritam Gundecha, and Huan Liu. Provenance data in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 4(1): 1–84, 2013.

[BHB13] Axel Bruns, Tim Highfield, and Jean Burgess. The arab spring and social media audiences: English and arabic twitter users and their networks. *American Behavioral Scientist*, 57(7): 871–898, 2013.

[BHMW11] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74, 2011.

[BKT01] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *ICDT*, volume 1, pages 316–330. Springer, 2001.

[BL06] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120, 2006.

[BM17] Marco T Bastos and Dan Mercea. The brexit botnet and user-generated hyperpartisan news. *Social Science Computer Review*, 2017.

[BSW12] Anders Brodersen, Salvatore Scellato, and Mirjam Wattenhofer. Youtube around the world: Geographic popularity of videos. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 241–250, 2012.

[CAB+12] Peter Cogan, Matthew Andrews, Milan Bradonjic, W Sean Kennedy, Alessandra Sala, and Gabriel Tucci. Reconstruction and analysis of twitter conversation graphs. In *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, pages 25–31. ACM, 2012.

[CAD+14] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pages 925–936, 2014.

[CCAB12] Giovanni Comarela, Mark Crovella, Virgilio Almeida, and Fabricio Benevenuto. Understanding factors that affect response rates in twitter. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 123–132, 2012.

[CCC+] Wei Chen, Alex Collins, Rachel Cummings, Te Ke, Zhenming Liu, David Rincon, Xiaorui Sun, Yajun Wang, Wei Wei, and Yifei Yuan. *Influence Maximization in Social Networks When Negative Opinions May Emerge and Propagate*, pages 379–390.

[CCT+09] James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4): 379–474, 2009.

[CHBG10] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and P Krishna Gummadi. Measuring user influence in twitter: The million follower fallacy. *ICWSM*, 10(10-17): 30, 2010.

[CHK+12] Raymond Cheng, Ji Hong, Aapo Kyrola, Youshan Miao, Xuetian Weng, Ming Wu, Fan Yang, Lidong Zhou, Feng Zhao, and Enhong Chen. Kineograph: taking the pulse of a fast-changing and connected world. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 85–98, 2012.

[CJR08] Adriane P. Chapman, H. V. Jagadish, and Prakash Ramanan. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 993–1006, 2008.

[CLS12a] Zhuhua Cai, Dionysios Logothetis, and Georgos Siganos. Facilitating real-time graph mining. In *Proceedings of the fourth international workshop on Cloud data management*, pages 1–8, 2012.

[CLS+12b] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications*, 391(4): 1777–1787, 2012.

[CLZ12] Wei Chen, Wei Lu, and Ning Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 592–598, 2012.

[CMG09] Meeyoung Cha, Alan Mislove, and Krishna P Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, pages 721–730. ACM, 2009.

[CMW13] P. Cheney, J. Missier, Moreau, L.(Eds.), and W3C Provenance Working Group. PROV-CONSTRAINTS: Constraints of the PROV Data Model. W3C, 2013.

[CT16] Wojciech Marian Czarnecki and Jacek Tabor. Online extreme entropy machines for streams classification and active learning. In *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*, pages 371–381, 2016.

[CW13] J. Cheney and W3C Provenance Working Group. Semantics of the PROV Data Model. W3C Note 30 April, 2013.

[CWY09] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208, 2009.

[DAF13] P Alex Dow, Lada A Adamic, and Adrien Friggeri. The anatomy of large facebook cascades. *ICWSM*, 1(2): 12, 2013.

[DBE+07] Susan B Davidson, Sarah Cohen Boulakia, Anat Eyal, Bertram Ludäscher, Timothy M McPhillips, Shawn Bowers, Manish Kumar Anand, and Juliana Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4): 44–50, 2007.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. pages 107–113, 2008.

[DNCVD+12] Tom De Nies, Sam Coppens, Davy Van Deursen, Erik Mannens, and Rik Van de Walle. Automatic discovery of high-level provenance using semantic similarity. In *IPAW*. 2012.

[DNTD+15] Tom De Nies, Io Taxidou, Anastasia Dimou, Ruben Verborgh, Peter M. Fischer, Erik Mannens, and Rik Van de Walle. Towards multi-level provenance reconstruction of information diffusion on social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1823–1826, 2015.

[DP13] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 25(10): 2283–2301, 2013.

[DR01] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, 2001.

[DVSC+14] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. RML: a generic language for integrated RDF mappings of heterogeneous data. In *LDOW*, 2014.

[FKSS08] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3), 2008.

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3): 75–174, 2010.

[FTLH16] Peter M Fischer, Io Taxidou, Bernhard Lutz, and Michael Huber. Distributed streaming reconstruction of information diffusion: poster. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pages 368–371. ACM, 2016.

[GAC+10] Wojciech Galuba, Karl Aberer, Dipanjan Chakraborty, Zoran Despotovic, and Wolfgang Kellerer. Outtweeting the twitterers-predicting information cascades in microblogs. In *Proceedings of the 3rd Wonference on Online social networks*, pages 3–3, 2010.

[GAHW15] Sharad Goel, Ashton Anderson, Jake Hofman, and Duncan J Watts. The structural virality of online diffusion. *Management Science*, 62(1): 180–196, 2015.

[GBL10] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM, 2010.

[GBL11] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. A data-based approach to social influence maximization. *Proc. VLDB Endow.*, 5(1): 73–84, 2011.

[GD07] Boris Glavic and Klaus Dittrich. Data provenance: A categorization of existing approaches. In *In Datenbanksysteme in Business, Technologie und Web (BTW 2007*, 2007.

[GEFT11] Boris Glavic, Kyumars Sheykh Esmaili, Peter M Fischer, and Nesime Tatbul. The case for fine-grained stream provenance. In *BTW Workshops*, volume 11, pages 58–61, 2011.

[GFL13] Pritam Gundecha, Zhuo Feng, and Huan Liu. Seeking provenance of information using social media. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1691–1696, 2013.

[GGCM12] Paul Groth, Yolanda Gil, James Cheney, and Simon Miles. Requirements for provenance on the web. *International Journal of Digital Curation*, 7(1): 39–56, 2012.

[GHFZ13] Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. Information diffusion in online social networks: A survey. *ACM SIGMOD Record*, 42(2): 17–28, 2013.

[GKT07] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 31–40. ACM, 2007.

[GL12] Pritam Gundecha and Huan Liu. Mining social media: a brief introduction. In *New Directions in Informatics, Optimization, Logistics, and Production*, pages 1–17. Informs, 2012.

[GLG+12] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. Powergraph: distributed graph-parallel computation on natural graphs. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*, pages 17–30. USENIX Association, 2012.

[GLL11a] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *2011 IEEE 11th International Conference on Data Mining*, pages 211–220, 2011.

[GLL11b] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, pages 47–48, 2011.

[GLM01] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3): 211–223, 2001.

[GMC15] Shuai Gao, Jun Ma, and Zhumin Chen. Modeling and predicting retweeting dynamics on microblogging platforms. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 107–116, 2015.

[Gol06] Jennifer Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In *International Provenance and Annotation Workshop*, pages 101–108, 2006.

[Gra78] Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, 83(6): 1420–1443, 1978.

[Gra83] Mark Granovetter. The strength of weak ties: A network theory revisited. *Sociological theory*, pages 201–233, 1983.

[GRFL13] Pritam Gundecha, Suhas Ranganath, Zhuo Feng, and Huan Liu. A tool for collecting provenance data in social media. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1462–1465. ACM, 2013.

[GRLK10] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1019–1028, 2010.

[GRLS13] Manuel Gomez Rodriguez, Jure Leskovec, and Bernhard Schölkopf. Structure and dynamics of information pathways in online media. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 23–32, 2013.

[GSEFT13] Boris Glavic, Kyumars Sheykh Esmaili, Peter Michael Fischer, and Nesime Tatbul. Ariadne: Managing fine-grained provenance on data streams. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 39–50, 2013.

[GSS07] Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. *ICWSM*, 7(21): 219–222, 2007.

[Gua13] The Guardian. AP Twitter hack causes panic on Wall Street and sends Dow plunging, 2013.

[Gua17] The Guardian. #MeToo: how a hashtag became a rallying cry against sexual harassment, 2017.

[Har09] Olaf Hartig. Provenance information in the web of data. 2009.

[Het00] Herbert W Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4): 599–653, 2000.

[HRW08] Bernardo Huberman, Daniel M Romero, and Fang Wu. Social networks that matter: Twitter under the microscope. *First Monday*, 14(1), 2008.

[HTW+12] Cindy Hui, Yulia Tyshchuk, William A Wallace, Malik Magdon-Ismail, and Mark Goldberg. Information cascades in social media in response to a crisis: a preliminary model and a case study. In *Proceedings of the 21st International Conference on World Wide Web*, pages 653–656. ACM, 2012.

[Hub14] Michael Huber. Verteilte rekonstruktion von informationskaskaden (distributed reconstruction of information cascades). Master's thesis, University of Freiburg, Chair of Web Science, 2014.

[HZ09] Olaf Hartig and Jun Zhao. Using web data provenance for quality assessment. In *Proceedings of the First International Conference on Semantic Web in Provenance Management-Volume 526*, pages 29–34. CEUR-WS. org, 2009.

[HZ10] Olaf Hartig and Jun Zhao. Publishing and consuming provenance metadata on the web of linked data. *IPAW*, 6378: 78–90, 2010.

[ISG13] Tomoharu Iwata, Amar Shah, and Zoubin Ghahramani. Discovering latent influence in online social activities via shared cascade poisson processes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–274. ACM, 2013.

[Jab16] Anum Jabbar. Reconstruction and analysis of twitter conversations, 2016.

[KABS11] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and information systems*, 29(2): 249–272, 2011.

[KBG12] Aapo Kyrola, Guy E Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a pc. USENIX, 2012.

[KCLC13] Krishna Y Kamath, James Caverlee, Kyumin Lee, and Zhiyuan Cheng. Spatio-temporal dynamics of online memes: a study of geo-tagged tweets. In *Proceedings of the 22nd international conference on World Wide Web*, pages 667–678, 2013.

[KIK08] Sophoin Khy, Yoshiharu Ishikawa, and Hiroyuki Kitagawa. A novelty-based clustering method for on-line documents. *World Wide Web*, 11(1): 1–37, 2008.

[KKMV14] Eleanna Kafeza, Andreas Kanavos, Christos Makris, and Pantelis Vikatos. Predicting information diffusion patterns in twitter. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 79–89. Springer, 2014.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[KLB05] Ferenc Kovács, Csaba Legány, and Attila Babos. Cluster validity measurement techniques. In *6th International symposium of hungarian researchers on computational intelligence*. Citeseer, 2005.

[KLPM10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600, 2010.

[KMF+14] Shoubin Kong, Qiaozhu Mei, Ling Feng, Fei Ye, and Zhe Zhao. Predicting bursts and popularity of hashtags in real-time. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 927–930. ACM, 2014.

[Kos06] Gueorgi Kossinets. Effects of missing data in social networks. *Social Networks*, 28(3): 247 – 268, 2006.

[KOU+12] Andrey Kupavskii, Liudmila Ostroumova, Alexey Umnov, Svyatoslav Usachev, Pavel Serdyukov, Gleb Gusev, and Andrey Kustarev. Prediction of retweet cascade size over time. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2335–2338, 2012.

[KRHW12] Milos Krstajic, Christian Rohrdantz, Michael Hund, and Andreas Weiler. Getting there first: Real-time detection of real-world incidents on twitter. In *2nd IEEE Workshop on Interactive Visual Text Analytics Task-Driven Analysis of Social Media*, 2012.

[LBG+12] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8): 716–727, 2012.

[LBK09] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, 2009.

[LKG+07] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, 2007.

[LMF+07] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie Glance, and Matthew Hurst. Patterns of cascading behavior in large blog graphs. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 551–556, 2007.

[LNH05] Jonathan Ledlie, Chaki Ng, and David A Holland. Provenance-aware sensor data storage. In *Data Engineering Workshops, 2005. 21st International Conference on*, pages 1189–1189. IEEE, 2005.

[LS13] Simon Ebner Lukas Saettler. Social media analysis, 2013.

[LT13] Tiancheng Lou and Jie Tang. Mining structural hole spanners through information diffusion in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 825–836, 2013.

[LTGM10] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1068, 2010.

[Lut17] Bernhard Lutz. Large scale distributed reconstruction of retweet cascades. Master's thesis, University of Freiburg, Chair of Web Science, 2017.

[LWBS14] Haiko Lietz, Claudia Wagner, Arnim Bleier, and Markus Strohmaier. When politicians talk: Assessing online conversational practices of political parties on twitter. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[LZ14] Chuang Liu and Zi-Ke Zhang. Information spreading on dynamic social networks. *Communications in Nonlinear Science and Numerical Simulation*, 19(4): 896–904, 2014.

[MAB+10] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

[MBK+08] Archan Misra, Marion Blount, Anastasios Kementsietsidis, Daby Sow, and Min Wang. Advances and challenges for scalable provenance in stream processing systems. In *International Provenance and Annotation Workshop*, pages 253–265. Springer, 2008.

[met13] Metis graph partitioning tool. [online] available http://glaros.dtc.umn.edu/gkhome/metis/metis/overview, 2013.

[Mey17] Bastian Meyer. Combining graph-iterative and streaming computations for influence graph derivations. Master's thesis, University of Freiburg, Chair of Web Science, 2017.

[MGBM07] Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of using provenance in e-science experiments. *Journal of Grid Computing*, 5(1): 1–25, 2007.

[MK10] Michael Mathioudakis and Nick Koudas. Twittermonitor: Trend detection over the Twitter stream. In *SIGMOD Conference*, pages 1155–1158, 2010.

[ML10] Seth Myers and Jure Leskovec. On the convexity of latent social network inference. In *Advances in Neural Information Processing Systems*, pages 1741–1749, 2010.

[MM12] Panagiotis T Metaxas and Eni Mustafaraj. Social media and the elections. *Science*, 338(6106): 472–473, 2012.

[MMI+13] Derek G Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 439–455. ACM, 2013.

[MMW13] L. Moreau, P. Missier (Eds.), and W3C Provenance Working Group. PROV-DM: The PROV Data Model. W3C, 2013.

[Mor10] Luc Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2–3): 99–241, 2010.

[MRHBS06] Kiran-Kumar Muniswamy-Reddy, David A Holland, Uri Braun, and Margo I Seltzer. Provenance-aware storage systems. In *USENIX Annual Technical Conference, General Track*, pages 43–56, 2006.

[MZL12] Seth A Myers, Chenguang Zhu, and Jure Leskovec. Information diffusion and external influence in networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33–41, 2012.

[New03] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2): 167–256, 2003.

[NGKA11] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. Bad news travel fast: A content-based analysis of interestingness on twitter. In *Proceedings of the 3rd International Web Science Conference*, page 8, 2011.

[NTD16] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 International Conference on Management of Data*, pages 695–710, 2016.

[OPM+12] Miles Osborne, Saša Petrovic, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Bieber no more: First story detection using twitter and wikipedia. In *SIGIR 2012 Workshop on Time-aware Information Access*, 2012.

[PCA15] Julio Cesar Louzada Pinto, Tijani Chahed, and Eitan Altman. Trend detection in social networks using hawkes processes. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 1441–1448, 2015.

[PMAJ+14] Sen Pei, Lev Muchnik, José S Andrade Jr, Zhiming Zheng, and Hernán A Makse. Searching for superspreaders of information in real-world social media. *Scientific reports*, 4: 5547, 2014.

[POL11] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. RT to Win! Predicting Message Propagation in Twitter. In *Proceedings of the 10th International AAAI Conference on Web and Social Media*, 2011.

[RBS11] Manuel G Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 561–568, 2011.

[RD02] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002.

[Rou87] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20: 53–65, 1987.

[RX17] Marian-Andrei Rizoiu and Lexing Xie. Viral potential and the economics of attention for modeling online popularity under promotion. *Proceedings of the 16th International AAAI Conference on Web and Social Media*, 2017.

[RXS+17] Marian-Andrei Rizoiu, Lexing Xie, Scott Sanner, Manuel Cebrian, Honglin Yu, and Pascal Van Hentenryck. Expecting to be hip: Hawkes intensity processes for social media popularity. In *Proceedings of the 26th International Conference on World Wide Web*, pages 735–744, 2017.

[SAA11] Matthew P Simmons, Lada A Adamic, and Eytan Adar. Memes online: Extracted, subtracted, injected, and recollected. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[SH10] Gabor Szabo and Bernardo A Huberman. Predicting the popularity of online content. *Communications of the ACM*, 53(8): 80–88, 2010.

[Sha95] Subhash Sharma. *Applied multivariate techniques*. John Wiley & Sons, Inc., 1995.

[SHE+13] Caroline Suen, Sandy Huang, Chantat Eksombatchai, Rok Sosic, and Jure Leskovec. NIFTY: a system for large scale information flow tracking and clustering. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1237–1248. ACM, 2013.

[SM86] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval*. McGraw-Hill, Inc., 1986.

[SMLGM11] Eldar Sadikov, Montserrat Medina, Jure Leskovec, and Hector Garcia-Molina. Correcting for missing data in information cascades. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 55–64, 2011.

[SMW13] Watsawee Sansrimahachai, Luc Moreau, and Mark J Weal. An on-the-fly provenance tracking mechanism for stream processing systems. In *Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on*, pages 475–481. IEEE, 2013.

[SOM10] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[sto] Storm. [online] available http://storm-project.net/.

[SZ11] Devavrat Shah and Tauhid Zaman. Rumors in a network: Who's the culprit? *IEEE Transactions on information theory*, 57(8): 5163–5181, 2011.

[T+07] Wang Chiew Tan et al. Provenance in databases: past, current, and future. *IEEE Data Eng. Bull.*, 30(4): 3–12, 2007.

[TAFS15] Io Taxidou, Anas Alzoghbi, Peter M. Fischer, and Christoph Schöller. Towards real-time lifetime prediction of information diffusion. In *Proceedings of the ACM Web Science Conference*, pages 61–62, 2015.

[TDNF17] Io Taxidou, Tom De Nies, and Peter M Fischer. Provenance of explicit and implicit interactions on social media with w3c prov-dm. *Lecture Notes in Computer Science, Springer LNCS/LNAI series*, 2017. Accepted, under publication procedure.

[TDNV+15] Io Taxidou, Tom De Nies, Ruben Verborgh, Peter Fischer, Erik Mannens, and Rik Van de Walle. Modeling information diffusion in social media as provenance with W3C PROV. In *Proceedings of the 6th International Workshop on Modeling Social Media*, pages 819–824, May 2015.

[TF13] Io Taxidou and Peter Fischer. Realtime analysis of information diffusion in social media. *Proceedings of the VLDB Endowment*, 6(12): 1416–1421, 2013.

[TF14] Io Taxidou and Peter M Fischer. Online analysis of information diffusion in twitter. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 1313–1318. ACM, 2014.

[TFDN+16] Io Taxidou, Peter M Fischer, Tom De Nies, Erik Mannens, and Rik Van de Walle. Information diffusion and provenance of interactions in twitter: is it only about retweets? In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 113–114. International World Wide Web Conferences Steering Committee, 2016.

[TLF+17] Io Taxidou, Sven Lieber, Peter M Fischer, Tom De Nies, and Ruben Verborgh. Web-scale provenance reconstruction of implicit information diffusion on social media. *Distributed and Parallel Databases*, pages 1–33, 2017.

[Tri17] Dennis Tritschler. Modeling of multiple nodes in conversation graph/dag, 2017.

[TSX15] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554, 2015.

[TXS14] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 75–86, 2014.

[Uda12] Goldee Udani. An exhaustive study of twitter users across the world.[online] available http://www. beevolve. com/twitter-statistics, 2012.

[VFO+14] Onur Varol, Emilio Ferrara, Christine L Ogan, Filippo Menczer, and Alessandro Flammini. Evolution of online user behavior during a social upheaval. In *Proceedings of the 2014 ACM conference on Web science*, pages 81–90. ACM, 2014.

[WCK+12] Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics, 2012.

[Wid05] Jennifer Widom. Trio: a system for integrated management of data, accuracy, and lineage. In *Presented at CIDR 2005*. Citeseer, 2005.

[WLJH10] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[WMZ10] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4): 20:1–20:38, 2010.

[WOOO11] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2541–2544, 2011.

[WRP+13] Lilian Weng, Jacob Ratkiewicz, Nicola Perra, Bruno Gonçalves, Carlos Castillo, Francesco Bonchi, Rossano Schifanella, Filippo Menczer, and Alessandro Flammini. The role of information diffusion in the evolution of social networks. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 356–364, 2013.

[WSPZ12] Christo Wilson, Alessandra Sala, Krishna PN Puttaswamy, and Ben Y Zhao. Beyond social graphs: User interactions in online social networks and their implications. *ACM Transactions on the Web*, 6(4): 17, 2012.

[WWP+17] Haishuai Wang, Jia Wu, Shirui Pan, Peng Zhang, and Ling Chen. Towards large-scale social networks with online diffusion provenance detection. *Computer Networks*, 114: 154–166, 2017.

[WYH+15] Senzhang Wang, Zhao Yan, Xia Hu, Philip S Yu, and Zhoujun Li. Burst time prediction in cascades. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 325–331, 2015.

[XZJ+16] Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. Topicsketch: Real-time bursty topic detection from twitter. *IEEE Transactions on Knowledge and Data Engineering*, 28(8): 2216–2229, 2016.

[YC10] Jiang Yang and Scott Counts. Predicting the speed, scale, and range of information diffusion in twitter. *Proceedings of the 10th International AAAI Conference on Web and Social Media*, 10: 355–358, 2010.

[YGC+10] Zi Yang, Jingyi Guo, Keke Cai, Jie Tang, Juanzi Li, Li Zhang, and Zhong Su. Understanding retweeting behaviors in social networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1633–1636, 2010.

[YL10] Jaewon Yang and Jure Leskovec. Modeling information diffusion in implicit networks. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 599–608, 2010.

[ZBK+10] Zicong Zhou, Roja Bandari, Joseph Kong, Hai Qian, and Vwani Roychowdhury. Information resonance on twitter: watching iran. In *Proceedings of the first workshop on social media analytics*, pages 123–131, 2010.

[ZBPH14] Indre Zliobaite, Albert Bifet, Bernhard Pfahringer, and Graham Holmes. Active learning with drifting streaming data. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(1): 27–39, 2014.

[ZHVGS10] Tauhid R Zaman, Ralf Herbrich, Jurgen Van Gael, and David Stern. Predicting information spreading in twitter. In *Workshop on computational social science and the wisdom of crowds, nips*, volume 104, 2010.

[ZSMF15] Arkaitz Zubiaga, Damiano Spina, Raquel Martinez, and Victor Fresno. Real-time classification of twitter trends. *Journal of the Association for Information Science and Technology*, 66(3): 462–473, 2015.

[ZST+13] Honglei Zhuang, Yihan Sun, Jie Tang, Jialin Zhang, and Xiaoming Sun. Influence maximization in dynamic social networks. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1313–1318, 2013.

[ZWSY12] Bo Zong, Yinghui Wu, Ambuj K Singh, and Xifeng Yan. Inferring the underlying structure of information cascades. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1218–1223. IEEE, 2012.

[ZZWZ13] Xiaohang Zhang, Ji Zhu, Qi Wang, and Han Zhao. Identifying influential nodes in complex networks with community structure. *Knowledge-Based Systems*, 42: 74–84, 2013.