Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Experimental Semester Project

Course number: 402-0215-MSL

Towards Quantifying Self-Excitation in Twitter Messaging

Author: Stefan Rustler
Supervisors: Dr. Vladimir Filimonov
. Prof. Dr. Didier Sornette

Zurich, Switzerland
April 2014

Abstract

This work sets the stage for determining the degree of self-excitation in Twitter by treating tweets triggering retweets as a self-excited Hawkes process. In this context the degree of self-excitation can be regarded as the criticality of the system of users tweeting about a certain topic, in our case the student protests in Austria in 2009. Ultimately, this could serve as a precursor for such protests. The key ingredients of the Hawkes model are the *unconditional intensity* for the arrival of the exo-events and the *memory kernel* for triggering endo-events (retweets). The former could be detrended via a smoothing spline fit per day. Via causally linking tweets and retweets, the latter was found to be of long memory with a power law fit resulting in a scaling parameter of $\alpha \in [1.538; 1.856]$. In the calm aftermath of the protests the memory became shorter, i.e. $\alpha = 2.237 \pm 0.047$. In order to actually be able to quantify the degree of self-excitation, key criteria for the data and further steps have been identified.

Contents

1 Motivation

In complex systems, like human societies or neuronal networks, a large number of strongly interacting subsystems or *agents* is crucial for the emergence of counter-intuitive phenomena from *within* the system. These internal feedback mechanisms are complemented by external impacts on the system, that allow the system to leave its state of equilibrium and to continuously evolve and adapt to those external impacts. This interplay of internal and external or *endogenous* and *exogenous* forces is of great interest in understanding the dynamics of information in complex systems. One of the crucial questions is whether it is possible to separate and quantify the influence of those endo- and exo-factors. Being able to quantify the former, gives insight into the robustness of a system to external influences. A system in which external excitations are coupled to and enhanced by internal *self-excitation* is likely to be in a more fragile or critical state. Hence, the degree of self-excitation can be regarded as a measure of the criticality of a system. An early example of this can be found in the physics of earthquakes. Here aftershocks (endogenous) are triggered by shocks (exogenous). If sufficiently many aftershocks per shock are set off, the system is in a critical state: a series of sudden, strong shocks, i.e. an earthquake, can occur.

In [1] V. Filimonov and D. Sornette proposed a method of quantitative estimation of the impact of the self-excitation, based on the so-called "self-excited conditional Hawkes process", a generalization of a Poisson process which effectively combines exogenous influences with endogenous dynamics. This method was applied on data of financial markets in [1]. As shocks trigger aftershocks, in the financial context *price changes* can trigger price changes. A critical state would then be characterized by sufficiently many internally triggered price changes, i.e. a high volatility of the market, which may lead to financial crashes.

This paper tackles the question of whether it is possible to apply this concept and method to the dynamics of information in *social networks*. Concretely, we apply the method proposed in [1] on Twitter data. Twitter is a main tool of self-organization of social groups, especially during riots and unstable periods. Here messages or "tweets" are publically visible and may be re-posted and thus spread by other users as so-called "retweets." In allusion to the terminology used in the Hawkes model, we will refer to this re-posting as "mother tweets" triggering retweets. According to the scheme above a critical state would be marked by a high retweeting activity. Since we are limiting ourselves to a specific topic, i.e. "hash-tag" of tweets, this criticality would refer to that specific topic only. Loosely speaking, one could then call criticality in this context "emotional ladenness" with regards to a certain e.g. political topic. This emotional ladenness could, if supercritical, manifest in protests or riots outside of Twitter. When speaking of the emotion anger, at some point too much accumulated anger will be released and expressed in violence, for example. It has to be noted, though, that no semantic analysis of the actual (emotional) content of the tweet is applied in this method. Merely the connections of tweets triggering retweets triggering retweets, are traced over time such that the degree of self-excitation or criticality over

time can be investigated. Here it is of particular interest whether there are supercritical phases that could be matched to actual emergent physical protest. Ultimately, detecting criticality with regards to certain topics as necessary ingredient for protests could become a precursor for such.

In this paper, the first steps are performed towards a methodology determining the criticality of the system of users tweeting about a certain topic. A key step in doing so is to determine the memory of the system, i.e. how long information is kept in the system and able to contribute to triggering future tweets. Intuitively, a long memory of tweets would add to the tendency to criticality of the system. The memory of the system enters the above-mentioned methodology as the "memory kernel" of the self-excited Hawkes process, which is described in detail in the next section.

2 Theoretical Background

2.1 Self-Excited Hawkes Processes

The process of tweeting can be regarded as a discontinuous point process, in which events, i.e. tweets, arrive with a certain rate. This arrival rate is generally referred to as the intensity λ of the process and takes different functional forms. The simplest point process is the Poisson process where the intensity is a simple constant. This means that events occur homogeneously, i.e. independent of each other. However, the intensity of an inhomogenous Poisson process has further dependencies, for example time. The self-excited Hawkes process is a generalization of an inhomogeneous Poisson process, which not only depends on time but als on the history of the process [2]. The intensity of the self-excited Hawkes process can be written as follows

$$\lambda(t) = \mu(t) + \int_{-\infty}^{t} \phi(t-s)\,\mathrm{d}N(s) \tag{1}$$

Here the first addend $\mu(t)$ denotes the background or unconditional intensity describing the arrival rate of *exogenous* events (exo-events). Due to the time-dependence, $\mu(t)$ alone would make a simple inhomogeneous Poisson process. The second addend refers to the intensity conditional on the history of the process, ranging from $-\infty$ to the present t via the integrational variable s. The integrand $\phi(t-s)$ denotes the so-called memory kernel which describes the positive effect of past events at s on the current instantaneous intensity at t of the *endogenous* events (endo-events) [3]. Generally, the further events at t and s are apart, the smaller ϕ will be, which corresponds to more distant events being "remembered" less by the system. Hence, the conditional intensity can be regarded as a sum of the instantaneous number of events having happened at s, i.e. $\mathrm{d}N(s)$, weighted by their influence on the present events at t via the memory kernel. Note that, even though we are dealing with discrete events, we are keeping the integral form for more convenient handling in the following. Considering the precision of the data of seconds and a total

duration of weeks and months, this is a sensible approximation. Assuming stationarity of the process as well as requiring that $\mu(t) = \mu$, an average total intensity Λ can be calculated by taking the expectation value on both sides of equation 1 [4].

$$\begin{aligned}\Lambda = \langle \lambda(t) \rangle &= \left\langle \mu + \int_{-\infty}^{t} \phi(t-s)\, \mathrm{d}N(s) \right\rangle = \\ &= \langle \mu \rangle + \int_{-\infty}^{t} \phi(t-s)\, \langle \lambda(s) \rangle \, \mathrm{d}s = \\ &= \mu + \Lambda \int_{0}^{\infty} \phi(\tau)\, \mathrm{d}\tau. \end{aligned} \tag{2}$$

Here the fact that $\mathrm{d}N(s)/\mathrm{d}s = \lambda(s)$ was used. To obtain the last equality the substitution $\tau \coloneqq t - s$ was performed. The interpretation of τ would then be the time it took for an endo-event to be triggered after the arrival of the triggering event. In our case of Twitter, this would correspond to the time between tweet and retweet, i.e. the *response time*. Now using the definition,

$$n \coloneqq \int_{0}^{\infty} \phi(\tau)\, \mathrm{d}\tau, \tag{3}$$

the average intensity can finally be written as

$$\Lambda = \frac{\mu}{1-n}. \tag{4}$$

Since we were requiring that the history of events must have a *positive* effect on the present intensity, this equation is meaningful only for the case of $n < 1$, such that $\Lambda \geq \mu$. When $n = 0$, the process becomes independent of history and thus a simple inhomogenous Poisson process. For the case of $n > 1$, the intensity as defined by equation 1 may explode [3]. The case of $n = 1$ can therefore be regarded as the critical case, separating the subcritical and the supercritical regime.

Now what is the meaning of n? As equation 3 shows, a memory kernel with long memory would result in a larger value for n. As explained before such a memory kernel results in an increased instantaneous intensity. The parameter n must therefore be related to how many events a single event is likely to trigger in the future. The entirety of events triggering further events can be regarded as a *branching process*. In fact, it is well-known that Hawkes processes can be mapped into such a branching processes, in which all events are either exogenous ("mother") or endogenous ("daughter") events of different generations. In this language a mother event would trigger a daughter event and so on, forming a "family tree." Now n would indeed be the average number of first-generation daughters of a single mother [1]. With this interpretation and the stationarity assumption, we can calculate an average *endogenous* (or self-excited or conditional) intenstity Λ_{endo}. The following calculation is mostly adopted from [5]. Generally, the intensity of i-th order events can be written as

$$\lambda_i(t) = \int_{-\infty}^{t} \lambda_{i-1}(s)\phi(t-s)\,\mathrm{d}s,\ i \in \mathbb{N}, \tag{5}$$

where $\lambda_0(t) \equiv \mu(t)$. Now if we require $\mu(t) = \mu$, as we have already done in equation 2, equation 5 would for the case of the intensity of first-generation daughter events simplify to $\lambda_1 = n\mu$. The intensity of the second-generation daughter events would likewise be given by $\lambda_2 = n\lambda_1 = n^2\mu$. Continuing this to inifinity we get for the additional requirement of $n < 1$ the following.

$$\Lambda_{endo} = \sum_{i=1}^{\infty} \lambda_i = \mu \sum_{i=1}^{\infty} n^i$$

$$\Lambda_{endo} = \mu \frac{n}{1-n} \tag{6}$$

In equation 4 we have calculated the total average intensity, for which $\Lambda = \mu + \Lambda_{endo}$ must hold. Taking the result from equation 6, this is indeed the case. Now we can calculate the overall proportion of endogenous events to all events.

$$\frac{\Lambda_{endo}}{\Lambda} = \frac{n/(1-n)}{1/(1-n)} = n \tag{7}$$

Therefore in the subcritical regime ($n < 1$) and in the case of a constant unconditional intensity [$\mu(t) = \mu$], the branching ratio n as average number of first-generation daughters per mother can be interpreted as the proportion of endogenous events among all events. The branching ratio thus is a direct quantitative estimate of the degree of endogeneity, i.e. self-excitation.

As mentioned in section 1 this paper aims to set the stage for determining the criticality in the system of users tweeting about a certain topic. For this, a major part of this work will be concerned with *detrending* the data such that the condition of a constant unconditional intensity is fulfilled. The resulting simpler form of the Hawkes process will allow a more practical numerical treatment of its intensity. The theory behind this step will be explained in the subsection 2.3. In a next step, the other major part will be on determining the memory kernel $\phi(\tau)$. There are in principle many functional forms, the most common being an exponential kernel of shorter memory like it was used in [1], i.e. $\alpha e^{-\beta\tau}$, and a power law kernel of longer memory like it was used in [3], i.e. $\varphi_0 \frac{\tau_{min}^{\epsilon}}{\tau^{1+\epsilon}}$ (Here we do not include a short-time cutoff, though). In both versions α and φ_0 are scale parameters, β and ϵ are shape parameters. Note that in this work we will use a different notation for these parameters as will be explained in the next subsection 2.2. However, τ_{min} as lower bound paremeter will be kept. In order to determine the memory kernel, τ must be extracted and sampled from the data. We need to get these response times by connecting "mother tweets" and retweets:

$$\tau_j = t_{i_{RT}} - t_{i_{MT}}, \tag{8}$$

where $t_{i_{RT}}$ and $t_{i_{MT}}$ denote the time stamps of the retweet and its mother tweet, respectively. Obtaining those is not trivial and will fill a major part of this work. While in the context of financial markets or earth quakes it is highly impractical if not impossible to make a causal link between mother and daughter event, such that the functional form has to be basically guessed, in the context of Twitter there actually is the possibility to make this causal link through various means to be explained in methods section 4. We are therefore in a very good position to determine a well-founded memory kernel.

All in all in this work the following is performed. Firstly, the unconditional intensity $\mu(t)$ is determined in order to turn it into $\mu = \text{const.}$ via detrending. Secondly, the conditional intensity, i.e. the memory kernel $\phi(\tau)$ is determined. This sets the stage for further investigations of the branching ratio n, which is, however, not performed in this work.

2.2 Power Laws

In [1] the memory kernel was chosen to be an exponential distribution. However, in this work we are dealing with a longer memory of the system, which will be empirically shown in section 5.2.1. In this part of the theory section the power law (PL) distribution that will be used to fit the data, will be elaborated on. In principle there are of course many different kinds of PL and other distributions that are heavy-tailled. However, focussing on the tails of the distribution, the PL distribution is a very common and simple one, giving a good first idea for further distributions to be tested.

The following explanations and arguments in this subsection are adopted from [6], which provides a more robust measure to determine PL distributions in empirical data. Though the variable whose distribution we are interested in, i.e. the response time τ, is discrete due to the precision of measurement (seconds), we treat it as continuous variable for three reasons. Firstly, response time is in fact continuous and only the measurement makes it discrete. Secondly, we are considering very large response times up to the order of months in comparison to which seconds are very small. Thirdly, the mathematical treatment is much simpler, though still providing a good approximation.

Now, a continuous PL distribution of τ is described by a probability density function (PDF) $p(\tau)$ such that

$$p(\tau)\mathrm{d}\tau = \Pr(\tau \leq T < \tau + \mathrm{d}\tau) = C\tau^{-\alpha}\mathrm{d}\tau, \tag{9}$$

where α is scaling parameter, T is the observed value of τ and C is the normalization constant. The latter will take care of not allowing the density to diverge at $\tau \to 0$ via including a lower bound τ_{min} for τ. Using the normalization condition of $\int_{\tau_{min}}^{\infty} p(\tau)\,\mathrm{d}\tau = 1$

as well as requiring $\alpha > 1$, the functional of form of the PL investigated here turns out to be

$$p(\tau) = \frac{\alpha - 1}{\tau_{min}} \left(\frac{\tau}{\tau_{min}} \right)^{-\alpha}. \tag{10}$$

Thus in this work some of the main results will be the scaling parameter α and the lower bound τ_{min} along with the respective errors. A larger α corresponds to a faster decay of τ, in the context of the memory kernel a shorter memory. Note that the case of $\alpha \leq 1$ is not normalizable and as such cannot occur in nature. A well-defined mean only exists if $\alpha > 2$. Often times it is useful to consider the complementary cumulatative distribution function (CCDF) of a PDF. The CCDF which will be denoted as $P(\tau)$ is defined as

$$P(\tau) = \Pr(T \leq \tau) = 1 - \Pr(T \geq \tau). \tag{11}$$

In our case with requiring a lower bound, the CCDF turns out to be

$$P(\tau) = 1 - \int_{\tau_{min}}^{\tau} p(\tau') \, \mathrm{d}\tau' = \left(\frac{\tau}{\tau_{min}} \right)^{-\alpha+1}. \tag{12}$$

In this work the fits for the extraction of the PL parameters and errors will be performed on the CCDFs according to the procedure suggested by [6], which were implemented using A. Clauset's codes on [7]. Note that according to this procedure, for a given τ_{min} the scaling parameter α can be approximated via

$$\alpha = 1 + n \left[\sum_{i=1}^{n} \ln \frac{\tau_i}{\tau_{min}} \right]^{-1}, \tag{13}$$

where τ_i are the observed values of the response times and the approximation of the true α becomes better for larger n, i.e. more data points. Equation 13 is equivalent to the well-known Hill estimator [8].

2.3 Detrending per Integration

In subsection 2.1 it was argued that the unconditional intensity μ needs to be made constant via detrending such that we can use the branching ratio n as measure of the degree of self-excitation of the system. Generally, if we were to not perform detrending on the event rate $R(t)$, a variation of this rate would always be attributed to the shape of the memory kernel of the Hawkes process, i.e. exo-events treated as endo-events. This would of course result in a wrong memory kernel with a generally higher memory. Therefore in this section the chosen principle procedure for detrending will be elaborated on.

Detrending "transforms" the time the stamps. That means with unaltered or original time stamps for the events the respective event rate will follow a trend line. Transformed

time stamps for the same events will result in a constant trend line, i.e. no trend. Intensity deviations from this trend line will then be due to endogenous processes. Detrending is a method that works backwards from a detected trend in or order to obtain the transformed times. In this work unaltered time stamps will be denoted as t_i whereas transformed time stamps will be denoted as t_i^*. In many other works the latter is denoted as τ_i. Here, however, this is reserved for the response times τ_j, where j indicates that the number of response times does not generally equal the number of time stamps (different subscript i). Analogously, τ_j^* denotes the transformed response times. This transformation generally looks like

$$t_i^* = \int_0^{t_i} \lambda(s)\,\mathrm{d}s, \tag{14}$$

where it becomes clear that the process of detrending or time transformation is a 1:1-mapping such that the number of elements in $\{t_i^*\}$ is the same as in $\{t_i\}$. It is well-known that the respective counting process $N(\{t_i\})$ will be a homogeneous Poisson process with constant intensity $\lambda = 1$, i.e. the *complete* process has been detrended [4]. Moreover, the distribution of interevent times will behave like $P(\Delta t^*) = \exp(-\Delta t^*)$, where $\Delta t^* = t_{i+1}^* - t_i^*$, for $i = 1...(N-1)$ [3]. Checking whether these two corollaries hold, give insight into whether the chosen form of $\lambda(t)$ can actually produce the observed point process. This has, for example, been applied in [3] and [9].

On the contrary, in this work we do *not* want the complete process to be detrended in order to check those two corollaries. Instead we want to detrend the unconditional part of the process such that $\mu = \text{const}$. Moreover, we want to ensure that the transformed times are matched to the scale of the unaltered time stamps, i.e. $t_{max}^* \stackrel{!}{=} t_{max}$. Therefore, the transformation was chosen to be of the form

$$t_i^* = \frac{1}{K} \int_0^{t_i} \mu(s)\,\mathrm{d}s, \tag{15}$$

where $K = t_{max}^*/t_{max}$ is the normalization factor, such that the respective point process is homogenous with $\mu = f(K) = \text{const}$. The challenge of this procedure lies in finding a good form of $\mu(t)$ which is per definition based on exo-events only. The following section on the data set used in this work will reveal that it is not trivial to discern between exo- and endo-events.

3 The Data Sets

In order to tackle the questions posed in the previous sections, an appropriate data set had to be chosen which not only has a high tweeting and retweeting activity but which also captures a period of dissatisfaction and protests of the Twitter users. The student protests in Austria in 2009 were one of the largest protests on education in recent years

and for the first time mainly organized by social media such as Twitter and Facebook [10]. In times where most relevant information and news are available on and thus shareable via the internet, using social media to connect and coordinate grassroots movements have become very popular [11].

When using Twitter as means to organize a group and provide relevant information, so-called hash-tags are used to label the tweets (multiple hash-tags are possible)[12]. In the case of the student protests of Austria in 2009 the hash-tag "#unibrennt" (engl.: university burns) was primarily used along with others such as "#unsereuni (engl.: our university)". In this work the data sets comprise tweets labelled with the former, more prominent hash-tag. Each tweet can come with a host of meta-data, most notably the author, time stamp, geo-tag, and retweet data.

There are many types of tweets triggered by different mechanisms. First and foremost there are original tweets that may or may not be re-posted by other users as so-called "retweets (RTs)" [13]. In allusion to the terminology used in the Hawkes model with exogenous "mother events" and endogenous "daughter events," we will call original tweets "mother tweets (MTs)." Strictly speaking, though, mother tweets do not necessarily all have to be exogenous events. They themselves can in principle be triggered by truly exogenous events, such as relevant news releases in the media. The definition of exo-events as opposed to endo-events is very crucial and not straight-forward as the system boundaries are not clear-cut. For simplicity, we make a basic assumption in this work: The vast majority of MTs are true exo-events. It is highly impractical if not impossible to trace MTs back to a common source or trigger e.g. in the media. We are therefore in principle able to determine the unconditional intensity as stated in equation 15 well enough, being able to draw conclusions on the exo-to-endo process in Twitter. On the contrary, there is no ambiguity in RTs as they are per definition endo-events. RTs are meant to multiply and spread content quickly to the followers of the retweeting user. There are several ways to perform retweeting, which require different methods of detecting them and linking them back to their MTs. This will be described in detail in the methods section 4.

There are two data sets examined in this work. The data set focussed on in this work dates back to the high-activity period of the student protest, consisting of 51,114 tweets in 33 days. Since this seems to be a rather limited time window that could potentially exhibit finite-size effects, a much longer period of 939 days, was analyzed as well. This second data set, however, is logically of a low-activity period (19,265 tweets), reaching up to present times. Both of these data sets will be further characterized in the following two subsections. Only in the following section 4 thereafter, it will be described how the respective data sets were actually analyzed.

3.1 Past Data

The past data set, being the high-activity set, is of actual interest in this work. It comprises 51,114 tweets of which about 15%, i.e. 7,740, are RTs. How these were detected is described

in detail in section 4.1.1. The time period in which those were tweeted goes from October 23, 2009 until November 24, 2009, which makes 31 days in which it was tweeted through out the entire day. Each tweet in the data contained the following.

- Unique tweet-ID (assigned by Twitter)
- User/author
- Tweet itself (limited to 140 characters)
- Time stamp with a precision of seconds
- Geo-tag, i.e. longitude and latitude (if applicable)

For the self-excited Hawkes model the time stamps of the event are of main interest as stated in section 2.1. The author and his/her tweet become important when establishing the links between MT and RT. The other two pieces of information are not relevant in this work but could be used for further analyses, also including *spatial* on top of temporal dimensions. The unique tweet-ID could be used to obtain further information on the tweet, using the Twitter API.

The way retweeting was performed in this data set, has important implications on the memory kernel of the Hawkes model. This becomes evident, when thinking of the case of a RT being retweeted, i.e.

$$MT_i \rightarrow RT_i^1 \rightarrow RT_i^2 \rightarrow ...RT_i^k, k \in \mathbb{N}$$

Here i denotes a unique identifier of the MT, e.g. the unique tweet-ID. Now in principle, it would be possible to either link RT_i^k to its *immediate* MT, i.e. RT_i^{k-1}, or to the *original* MT, i.e. MT_i. In the latter case this would result in a *renormalized* memory kernel, in the former in a *bare* memory kernel. For the same process both kernels will follow the same law but have different exponents, i.e. bare versus renormalized kernels [14]. Since the renormalized kernel captures higher-order descendants, the memory is larger, which manifests in the altered scaling exponent, i.e. smaller α. In principle, the retweeter is free to alter the MT, e.g. adding personal comments or additional hash-tags. In the past data retweeting was performed by simply copying and pasting the MT and *manually* adding a reference to the original tweeter, i.e. "RT @*username*." However, it can also be the case that several of these references are given, i.e. RTs of RTs, which make it difficult to identify the most original MT. We refer to this type of retweeting via copying and pasting as "manual RTs." Now Twitter is not able to automatically draw this connection between MT and manual RT. However, by text analysis *some* MT can in principle be found but the generation of the MT remains not absolutely certain. Since credit is usually given to the most original tweeter [15], it is sensible to assume a renormalized memory kernel for the purpose of further analyses.

3.2 Present Data

One would intuitively assume that in such a fast-lived platform like Twitter the memory of the system is rather short, i.e. RTs occur in a timely manner, like weeks at most. However, in order to really check for this, a much larger time period of tweets was also analyzed. The "present data" comprises 939 days from November 30, 2010 until May 16, 2013, a very long time span in which virtually all extreme events, i.e. very late RTs, should be captured. There are 19,265 tweets and 8,213 RTs (ca. 43%) in this data set. Obviously, this is a very low-activity with respect to the past data since the main protests have long been finished. The tweet-IDs were obtained manually from the Twitter website by parsing the html-file of the search results of tweets with the hash-tag #unibrennt (`https://twitter.com/search?q=%23unibrennt`). These were then fed into the online Twitter API via a Python script (`t_get.py` by V. Filimonov) in order to extract further information on those tweets. Each tweet obtained this way contained the following.

- Unique tweet-ID (assigned by Twitter)
- Unique tweet-ID of MT (if applicable)
- Time stamp with a precision of seconds
- Unique user-ID (2400 users)
- Number of RTs (if not RT itself)

The form of the data indicates that there is no need to employ a textual analysis for finding MTs of RTs because this information is already provided. As stated in the previous subsection, however, Twitter is not able to capture manual RTs. And indeed, those are not reflected in the data. Yet there is a different kind of RTs recorded, i.e. the "automatic RTs," which uses the in-built function of Twitter to retweet any tweet. By using this function the RT of any generation will always refer back to the original MT. Hence, the resulting memory kernel will strictly be renormalized in this data set.

Both basic types of retweeting, i.e. manual and automatic RTs, are widely used. Which type is chosen by the retweeter depends on the MT. If the retweeter wants the MT to go viral, it is more conducive to use automatic retweeting. If the retweeter wants to add personal comments or actively engage with the mother tweeter, manual RTs are more conducive. In this work we assume that both types of retweeting follow the same mechanism and it is therefore legitimate to use only manual RTs for the past data and automatic RTs for the present data. This assumption, however, needs more rigorous justification and it would be beneficial to combine the approaches of detecting *both* manual and automatic RTs for *each* of the data sets. How these approaches are realized, is explained in the following section.

4 Methods

4.1 Extracting Response Times

The key variable for investigating the memory of the system is the response time, i.e. the time needed for a RT to follow its MT. Unlike in [1], with Twitter data we are able to causally link mother and daughter events, i.e. RTs and MTs. This puts us in a good position to robustly determine the memory kernel of the Hawkes process. The memory kernel will simply be the law underlying the responses time distribution. How this underlying law is found is based on section 2.2, in particular on [6].

The main script `MainScript.m` processes both past and present data. In the following two subsections the respective structures and called functions are explained with the help of pseudo-codes. All codes used can be found on the author's GitHub directory (https://github.com/SRustler/Towards_Quantyfing_Self-Excitation_in_Twitter_Messaging). In all subsections the following notation is employed: t refers to time stamps, while τ refers to response times. Adding an asterix * denotes a variable transformed via detrending.

4.1.1 Past Data

As described in section 3.1 textual analysis of the tweets has to be employed in order to perform the MT-RT-linking and to then obtain the response times. This textual analysis is based on the assumption that the retweeting convention is predominantly followed. In this convention a RT is generally built up as follows:

"[addition 1] RT @[retweeted user]: [MT] [addition 2]" (or space instead of semicolon)

RTs are then identified by looking for the string "RT @" in the tweet. Then the MT, [MT], and its user, [retweeted user], are reconstructed in order to find the MT together with its time stamp in the data. When trying to link the reconstructed MT with the actual MT, the crucial criterion is the so-called *Levenshtein distance*, a variable that quantifies the disagreement between two strings of characters. The Levenshtein distance between two strings of characters is defined as the minimum number of single-character edits required to change one string into the other [16]. So each time a character from the one string has to be removed or added in order to get "closer" to the other string, the Levenshtein distance is increased by 1. For example the Levenshtein distance between 'ABCD' and 'DABC' would be 2 because 'D' would first have to be removed and then added. Since tweets have a maximum character length of 140, the maximal Levenshtein distance is also 140. Now one could expect that there is no need to allow for a slight disagreement, i.e. a nonzero Levenshtein distance, between reconstructed and actual MT. However, as stated earlier, the retweeter can in principle change the MT. This is likely to happen if, for example, only certain parts of the MT are relevant. Hence, allowing for a variable Levenshtein distance allows for detecting more MT-RT-links. It is not obvious what Levenshtein distance is the

optimal one. On the one hand, a too low Levenshtein distance might be too restrictive and dramatically reduce the number of MT-RT-links found and thus response times obtained, such that the statistical analysis of the latter becomes weaker. On the other hand, a too high Levenshtein distance might falsely attribute some later tweet as MT to a detected RT, skewing the response time distribution. Another peculiarity is the following. Consider the reconstructed MT "[Message] [some URL 2]" and its potential MT "[Some URL 1][Message]". Since the actual core message is the same in both tweets ([Message]), the potential MT can in fact very well be the actual MT. However, the Levenshtein distance will be very high as first [some URL 1] would have to be removed and then [some URL 2] would have to be added character by character. So a high Levenshtein distance does not necessarily mean that the linked MT and RT are very different. This is another reason that justifies using not only nonzero but also very high Levenshtein distances.

For all these reasons, when analyzing the past data, the response time distributions of the full range Levenshtein distances $\ell \in [0; 140]$, will be examined in this work in hope to find an optimal Levenshtein distance for MT-RT-linking.

The overall structure of the script part extracting the response times from the past data is as follows. Called functions are explained as pseudo codes in the appendix.

1. Get original time stamps of all tweets, i.e. $\{t_i\}$.

2. For different Levenshtein distances ℓ: Detect and link RTs with MTs to get response times via `get_taus_past.m`:
$$\forall \ell : \{t_{\ell,i}\} \to \{\tau_{\ell,j}\}$$

3. Detrend original time stamps to get transformed time stamps via `detrend_anyFit.m`
$$\{t_i\} \to \{t_i^*\}$$

4. Process transformed time stamps $\{t_i^*\}$
   - For different Levenshtein distances ℓ, acquire response times via `ts2taus.m`:
   $$\{t_{\ell,i}^*\} \to \{\tau_{\ell,j}^*\}$$
   - For different Levenshtein distances ℓ, perform power law fit via `plfit.m` and `plvar.m`.
   $$P_\ell(\tau) \to \alpha_\ell, \tau_{min,\ell}$$
   - Plot and compare response times for detrended versus undetrended and $\ell = 0$ versus $\ell = 140$.

4.1.2 Present Data

Unlike in the past data in the present data no textual analysis can be applied and does not have to be. As described in section 3.2,the MT-RT-linking is unambiguous and always referring back to the most original MT. Consequently, the response time ditribution will be the renormalized memory kernel. In addition, a user specific analysis is performed for this data. The most active tweeters or retweeters can be isolated and their extreme, longest, response times removed. The then re-aggregated response times of those uses can be further analyzed.

The overall structure of the script part handling the past data is as follows. Called functions are explained as pseudo codes in the appendix.

1. Get original time stamps of all tweets, i.e. $\{t_i\}$.
2. Detrend original time stamps to get transformed time stamps via `detrend_anyFit.m`:
$$\{t_i\} \rightarrow \{t_i^*\}$$
3. Process transformed time stamps $\{t_i^*\}$
   - Detect and link RTs with MTs to get response times via `get_taus_pres.m`:
   $$\{t_i^*\} \rightarrow \{\tau_j^*\}$$
   - Perform power law fit via `plfit.m` and `plvar.m`.
   - Get response times of users via `get_user_taus.m`:
   $$\{\tau_i^*\} \rightarrow \{\tau_{user,j}^*\}$$
4. Process original time stamps $\{t_i\}$
   - Again use `get_taus_pres.m` and `get_user_taus.m`, however, on original time stamps:
   $$\{t_i\} \rightarrow \{\tau_j\}$$
   $$\{\tau_j\} \rightarrow \{\tau_{user,j}\}$$
   - Perform power law fit via `plfit.m` and `plvar.m`.
   $$P(\tau^*) \rightarrow \alpha, \tau_{min}^*$$
   - Compare transformed and untransformed response times
   $$\{\tau_j^*\} \leftrightarrow \{\tau_j\}$$
   $$\{\tau_{user,j}^*\} \leftrightarrow \{\tau_{user,j}\}$$

4.2 Detrending Tweet Activity

As stated in section 2.3, the key challenge of detrending lies in determining the unconditional intensity $\mu(t)$ as given in equation 15. For this, ideally the events would be clearly separable into immigrant and descendant events such that we could directly extract $\mu(t)$ from the rate of purely immigrant tweets. However, as explained in section 3 it is only possible to unambiguously identify the descendant events, i.e. the RTs. The MTs, on the contrary, are comprised of both immigrant and descendant events (MTs triggered by true exo-events like common news media releases), with the assumption that the former dominates strongly. So it is not possible to perform a perfect fit on a purely immigrant rate. However, the complete tweet rate $R(t)$ can be filtered from most of its descendant events, i.e. all the RTs. So the remaining MT rate $R_{MT}(t)$ might still have some descendant MTs left but a "low-order" fit to $R_{MT}(t)$ should be able to approximate the behavior of the unconditional intensity. The rationale of this low-order fit is based on the notion that an external variation of the tweet rate, i.e. $\mu(t)$, should happen rather smoothly, i.e. not as quickly as the variation due to endogenous mechanisms. In this sense the smaller number of descendants would be contained in the higher orders of the fit.

This low-order fit was chosen to be a smoothing spline fit (MATLAB function `fit(x,y,'smoothingspline')`) with manual smoothing parameter. A spline fit per se or spline interpolant would fit a piece-wise defined (cubic) polynomial through the data points. A smoothing spline fit, on the contrary, is more appropriate for very noisy data because it does not vary as fast as the data points but to a degree determined by a smoothing parameter $p \in [0; 1]$. For $p = 0$ a least-squares straight-line fit to the data is produced, while $p = 1$ produces a cubic spline interpolant [17]. In a way the smoothing parameter specifies how low our low-order fit will be. The smoothing spline s is constructed for the specified smoothing parameter p. The smoothing spline minimizes

$$p \sum_i [y_i - s(x_i)]^2 + (1 - p) \int \left(\frac{\mathrm{d}^2 s}{\mathrm{d}x^2} \right)^2 \mathrm{d}x. \tag{16}$$

Figure 1 shows an instructive example of this smoothing spline fit of the MT rate $R_{MT}(t)$, represented by the blue bars, with a smoothing parameter of $p = 0.3$. This value was chosen by visual judgment for all days such that the degree of variation of the spline fit could in principle be justified with a varying external daily tweet activity that is in line with people's day time activities (sleep, lunch breaks, after-work). See section 5.1 on the results of $\mu(t)$-fitting for further elaboration on this. The red bars represent the RTs of each bin. The entire stacked bars thus make up the complete process $R(t)$ to be detrended. Now deviations of the tweet activity of the complete process with $\lambda(t)$ from that of the unconditional external process with $\mu(t)$ would be due to endogeneous mechanisms, i.e. self-excitiation.

The pseudo code of `detrend_anyFit.m` for obtaining the transformed times, is based on section 2.3 and addresses all the considerations above. It can be found in the appendix.

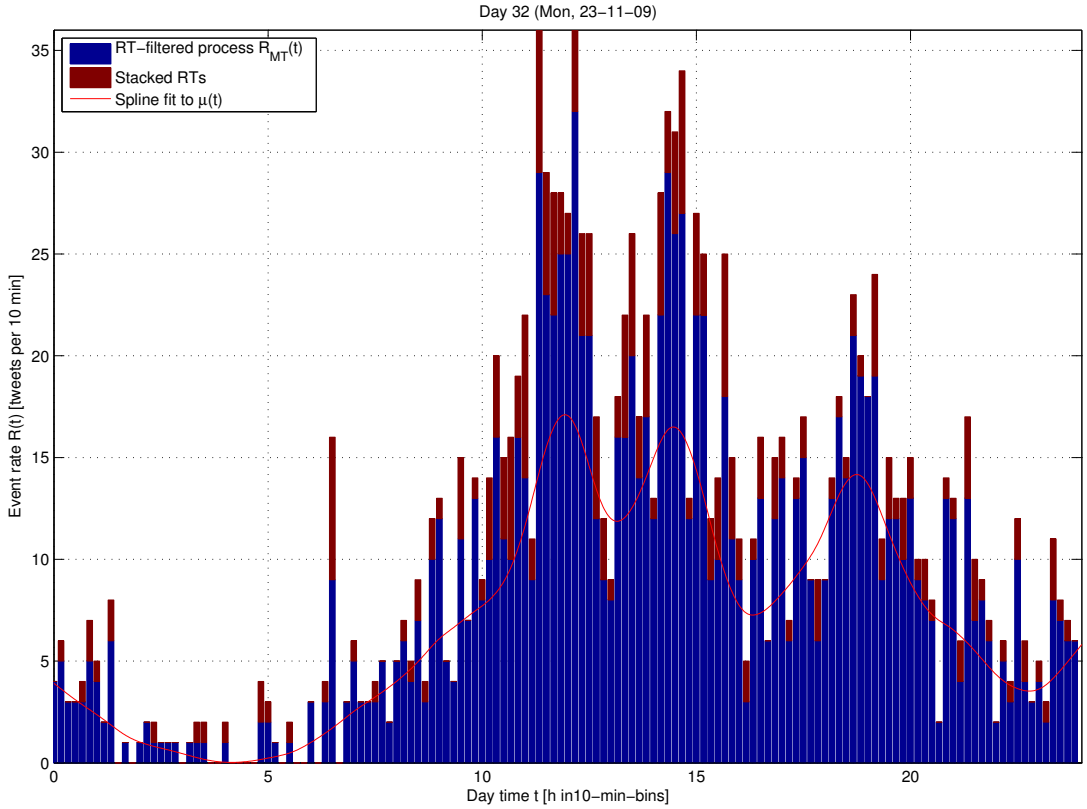

Figure 1: Instructive example of the spline fit of the unconditional intensity $\mu(t)$ of order lower that the complete intensity $\lambda(t)$ of higher variability and order. We assume that the external tweet rate variation is much slower than the internal one. In this sense exogenous events are expressed in the higher order deviations from the spline fit. The red curve depicts the smoothing spline fit with smoothing parameter $p = 0.3$. The fit was performed on the RT-filtered process $R_{MT}(t)$ (blue bars), whereas the detrending was performed on the complete process $R(t)$ including the RTs (red stacked bars).

4.3 Power Law Fitting

Taking the logarithm on both sides of equation 10 shows that plotting $\ln p(\tau) = -\alpha \ln \tau +$ const. of PL-distributed data will be of linear form in a log-log-plot. It is hence tempting to create a histogram of the data and perform a linear fit of the data in a log-log-plot with the slope being α. This, however, has several flaws, resulting in systematic offsets, as explained in detail in [6]. Instead we will determine the PL parameters via a more reliable method of combining maximum-likelihood-estimations with goodness-of-fit tests based on the Kolmogorov-Smirnov statistic and likelihood ratios as proposed by [6]. The results of these tests will be visualized *not* on the PDF but on the respective CCDF.

The MATLAB codes `plfit.m` and `plvar.m` estimate τ_{min} and α, and respectively their errors, according to the goodness-of-fit based method described in [6]. The log-likelihood $\mathcal{L}$ can also be computed. From the description of the codes from [7]: "The fitting procedure works as follows:

- For each possible choice of x_{min} [τ_{min}], we estimate alpha via the method of maximum likelihood, and calculate the Kolmogorov-Smirnov goodness-of-fit statistic D.
- We then select as our estimate of x_{min} [τ_{min}], the value that gives the minimum value D over all values of x_{min} [τ_{min}]."

Here x_{min} denotes the lower bound of the random variable above which the power law is fitted. In our case this is τ^*_{min}. α denotes the power law exponent or scaling parameter as already used in section 2.2. Note that α and its error will be displayed on the "linear" fit of the CCDF-curve, which actually, however, has a slope of $-\alpha + 1$ because of the chosen definition in equation 10.

5 Results

5.1 Unconditional Intensity

As argued in section 2.1, in order for n to be interpreted as branching ratio, we need to make the history-independent part of the intensity, i.e. the unconditional intensity $\mu(t)$, independent of time also. Again, this procedure is referred to as detrending the unconditional intensity with respect to time. Logically, detrending requires to look for trends. In this section we qualitatively examine the daily and weekly patterns of the tweet activity or rate $R(t)$ of the *MTs* only. Since what we aim for, are external variations, i.e. variations that are attributed to natural variations unrelated to tweet activity from within the system. Therefore RTs which are per definition endogenous events have to be filtered out for this analysis. Only external variations can be reflected in the unconditional intensity. We limit this examination to the past data because the present data spans over a much larger time span with much more heterogeneous tweet activity. Hence the present data is much less likely to exhibit overall trends.

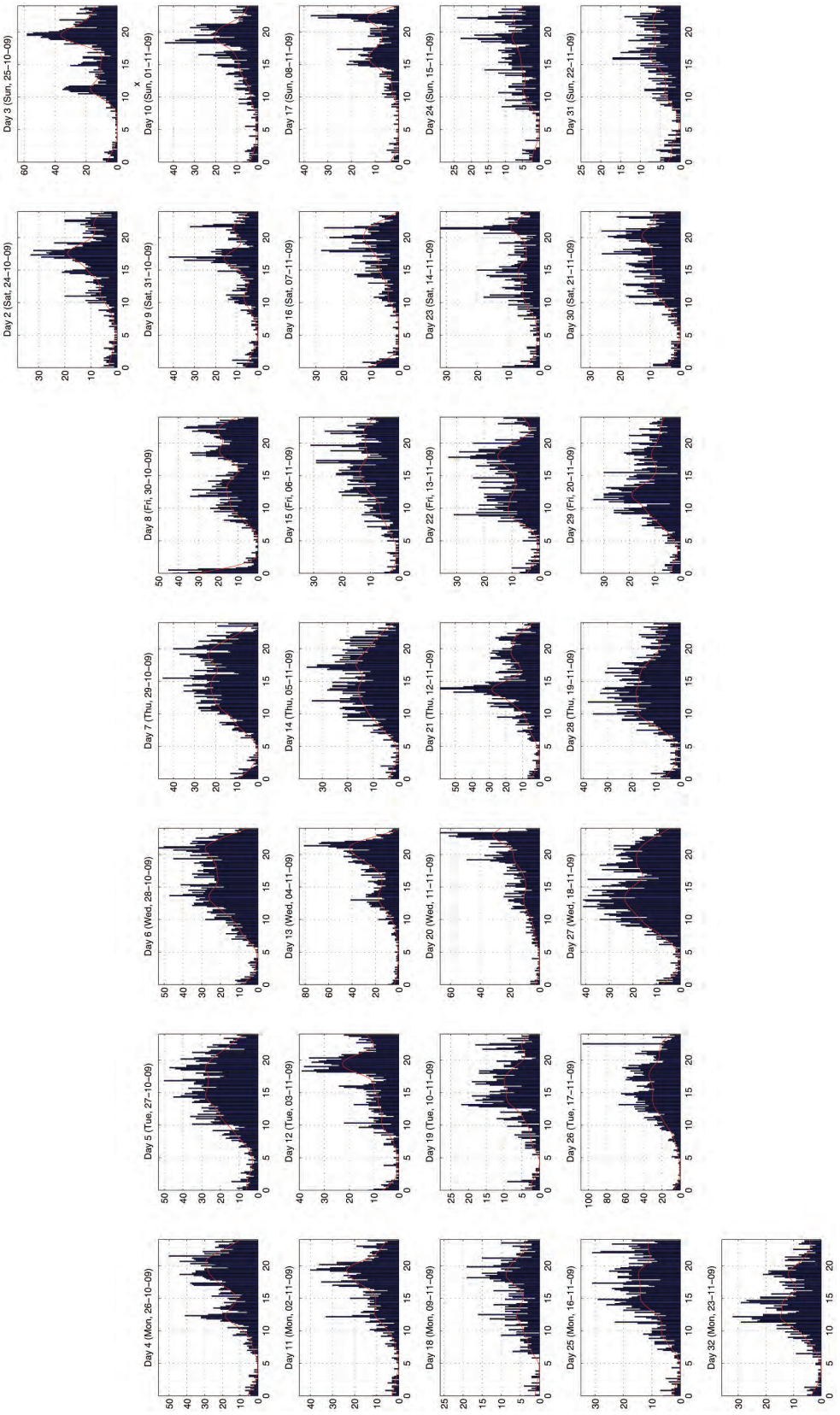

Figure 2: Calendar view of the daily tweet rate $R(t)$ with bins of 10 min and spline fits. For all days the axes dimensions are number of tweets versus day time in hours. Note that the first and last day of the overall period, i.e. day 1 and 33, are omitted because they are only partially filled with tweets. Neither a daily nor a weekly trend is visible.

In figure 2 the daily tweet rates $R(t)$ are displayed in a calendar view. This allows for getting a first impression on the daily as well as weekly trends. The bins of the tweet rate were chosen to be 10 min. This size turns out to both allow for sufficient detail and prevent too many isolated bins, which is what we need for a meaningful spline fit. The spline fits themselves are depicted in the daily plots, too, and were chosen to have a smoothing parameter of 0.3, a value that fits to the detail determined by the bin size. Also note that the range of the tweet rate is oriented at the maximum value per day and varies across different days. When checking figure 2 across days, a rough general trend can be identified. Firstly, the tweet activity is higher during the day and lower during the night, with the minimum being between 3am and 6am. Secondly, there are several peaks throughout the day, above all a lunch peak between 12pm and 2pm and an "afterwork" peak between 5pm and 8pm. Overall it seems that tweet activity coincides with the daytime activity. However, this is very crude and qualitative because the peaks and lows are tough to be attributed to smaller ranges within the day. In fact, in this critical period of student protests, the exact times of the peaks are likely to be also influenced by external factors such as news releases etc. and not merely due to the regular daytime activity. Performing a spline fit against the aggregated daily rates, which would be based on the asumption that the tweet activity is reoccurring with a period of one day, therefore makes little sense. A weekly trend, on the contrary, is less restrictive and sensible considering weekly schedules of students, who arguably were the main contributors to the topic. First one can look at week days versus weekends. What strikes here is that the overall tweet amount, i.e. area under the rate curves, seems to be generally less during the weekends. Moreover, lunch peaks seem to be less pronounced and "after-work" occurring later than during the week. Again, this is crude and there are exceptions. However, it fits the previous observation that tweet activity coincides with daytime activity. It's even less restrictive when looking across certain days of the week, i.e. columns of the calendar. However, again there is not any trend to be found that is more distinct than what has been observed above. Hence, at this point it can be concluded that the external variation of tweet activity is not predominant enough, neither across days nor across weeks, to justify a spline fit of the unconditinal intensity $\mu(t)$ performed on tweet rates aggregated over more than one day. Instead we perform the spline fit on a daily basis, assuming that the daily unconditional intensity follows a low-order fit as described in section 4.2.

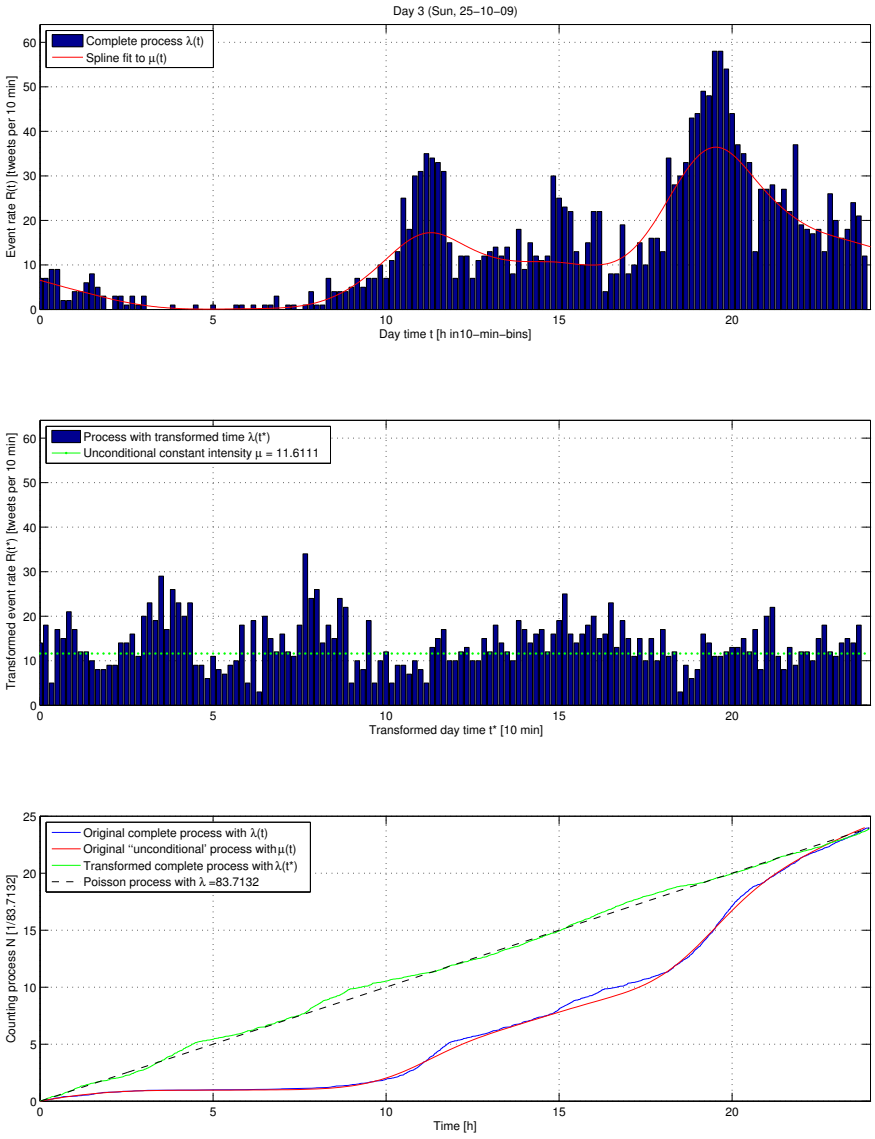

Figure 3: Examplary plots on the detrending of day 3. The first subplot contains the *complete* tweet rate of this day $R(t)$, while the spline fit was performed on the *mother* tweet rate $R_{MT}(t)$. The second subplot shows the tweet rate after detrending $R(t^*)$ and the constant unconditional intensity μ. The third subplot compares the different counting processes: the original complete process corresponds to the primitive function of $\lambda(t)$, the original unconditional process corresponds to the primitive function of the spline-fitted $\mu(t)$, the transformed complete process with the intensity $\lambda(t^*)$, and the homogeneous Poisson process with constant intensity λ.

The results of the detrending performed via daily spline fits are presented with an example of some day out of the entire 31 days. This day (third day) behaves in principle like all the other days but has some features to better illustrate the effect of detrending. Figure 3 on day 3 depicts the unaltered and transformed tweet rate as well as the corresponding counting processes. Note that we do not yet state the effects of the detrending on the response times distributions, i.e. the memory kernel. This will be discussed in the next subsection. Let's turn our attention to the top graph of figure 3. Recalling that the spline fit itself was performed on the MTs only, we first notice that the spline fit does indeed not quite follow the course of the $R(t)$-histogram (entailling both RTs and MTs) especially at the two peaks at about 11am and 7pm. This is in line with the explanations in 4.2. The position of the spline-fit peaks agrees with the general observations across days in the calendar view. However, there are very clear peaks resulting from RTs only, for example at 3pm. Now how does the process of detrending act upon these? Looking at the center graph of figure 3, the three peaks from the upper graph seem to remain, however at different positions: approximately 3-4am, 7-8am and 2-4pm. Recalling equation 15 it becomes clear that the transformed time is proportional to the area under the spline-fit-curve. Consequently, the tweet rate $R(t)$ is spread out more evenly across the day: Periods with low tweet activity will be "filled" with events from higher activity. For example, the first two peaks of day 3 are transferred to earlier in the morning. On the contrary periods with high tweet activity will be "flattened" to periods of lower activity. The third peak at 7pm will, for example, be moved and spread to 2-4pm. The constant unconditional intensity μ obtained from taking the average of the spline fit is plotted, too.

The effect of detrending also becomes clearer when considering the respective counting processes $N_\lambda(t)$ of the different intensities $\lambda(t)$. From section 2.1 we recall that $\mathrm{d}N(t)/\mathrm{d}t = \lambda(t)$ and thus

$$N_\lambda(t) = \int_0^t \lambda(s)\,\mathrm{d}s. \tag{17}$$

Obviously $N(t)$ is and must be monotonically increasing. The bottom graph of figure 3 depicts these counting processes obtained from integrating different intensities and rates. The original complete process with the general intensity $\lambda(t)$ was obtained by simply integrating $R(t)$ according to equation 17 (rate as intensity), while the external process with the unconditional intensity $\mu(t)$ was obtained from integrating the spline fit. What strikes here, is that the regions of greatest deviations between those two curves coincide with the three peaks of $R(t)$ in the upper graph. This is in agreement with equation 17. Now the transformed process after detrending, was obtained from integrating $\lambda(t^*)$, i.e. the complete intensity with the transformed times. It roughly follows the homogeneous Poisson process of constant unconditional intensity, which happens to be $\lambda \approx 83.7$ [tweets/h] for day 3. This value is directly derived from the total number of tweets in that day. The transformed process *roughly* following the homogeneous process, agrees with the former process now having a constant unconditional intensity. Again, we do not want a perfect

match of the two processes. Otherwise the self-excitation would falsely be eliminated. Also note that due to the imposed condition of $t^*_{max} \stackrel{!}{=} t_{max}$ the curves of all counting processes end in the same point at 12pm.

All in all having detrended the unconditional intensity, i.e. $\mu(t) \to \mu$, and hence transformed the time stamps, i.e. $\{t_i\} \to \{t^*_i\}$, we can now turn our attention to analyzing the memory kernel $\phi(\tau^*)$.

5.2 Memory Kernel

5.2.1 Past Data

Analyzing the response times distribution for the critical protest period, i.e. the past data, is the focus of this work. In section 4.1.1 the challenge of causally linking MTs and RTs was outlined. The crucial parameter here is the Levenshtein distance between a RT and its MT, whose exact optimal value is not straightforward to choose. In this section we present the results for different Levenshtein distances on the memory kernel. First of all we recall that a large Levenshtein distance is less restrictive for accepting a potential MT as actual MT of a RT detected. We would therefore expect many more RT-MT-links to be found if the Levenshtein distance is large. In figure 4 this is exactly visible: With a small Levenshtein distance of 15 (top graphs) much less RT-MT-links represented by the red circles are found. When looking at the axes borders, a maximal response time of about 19 days can be seen. In contrast, with a large Levenshtein distance of 135 (bottom graphs) many more RT-MT-links are found. Here the maximal response time was about 25 days. So not only were there more RT-MT-links detected but also a different response time range. This already indicates that there will be an effect of the Levenshtein distance ℓ on the memory kernel, in particular the scaling parameter α.

To more closely examine this, we draw our attention to figure 5, where we include both the log-log-histogram as well as the CCDF of the response times between the RTs and their MTs as given by the detected RT-MT-links. An increment of 15 was chosen to investigate the whole range of Levenshtein distance from 0 to 140. As explained in section 4.3 it is tempting to fit a straight line through the log-log-histogram in order to obtain α. And indeed the data seems to very well resemble a straight line. However, in addition to the flaws explained in [6] it is very visible that the long tail is especially noisy and wide-spread. This is alleviated by plotting the CCDF of the same data. Clearly, increasing the Levenshtein distance has an effect on the response times distribution. It unbends the CCDF-curve, which corresponds to shortening the memory. This is in agreement with intuition: a large Levenshtein distance means that a *later* tweet of the retweeted user will be accepted as MT, artificially shortening the memory. However, when looking closely on the behavior of the CCDF-curves, one can see that the trend of Levenshtein distance negatively correlating with the memory does not hold across all Levenshtein distances. This becomes immediately evident when looking at the two outermost CCDF-curves. They do

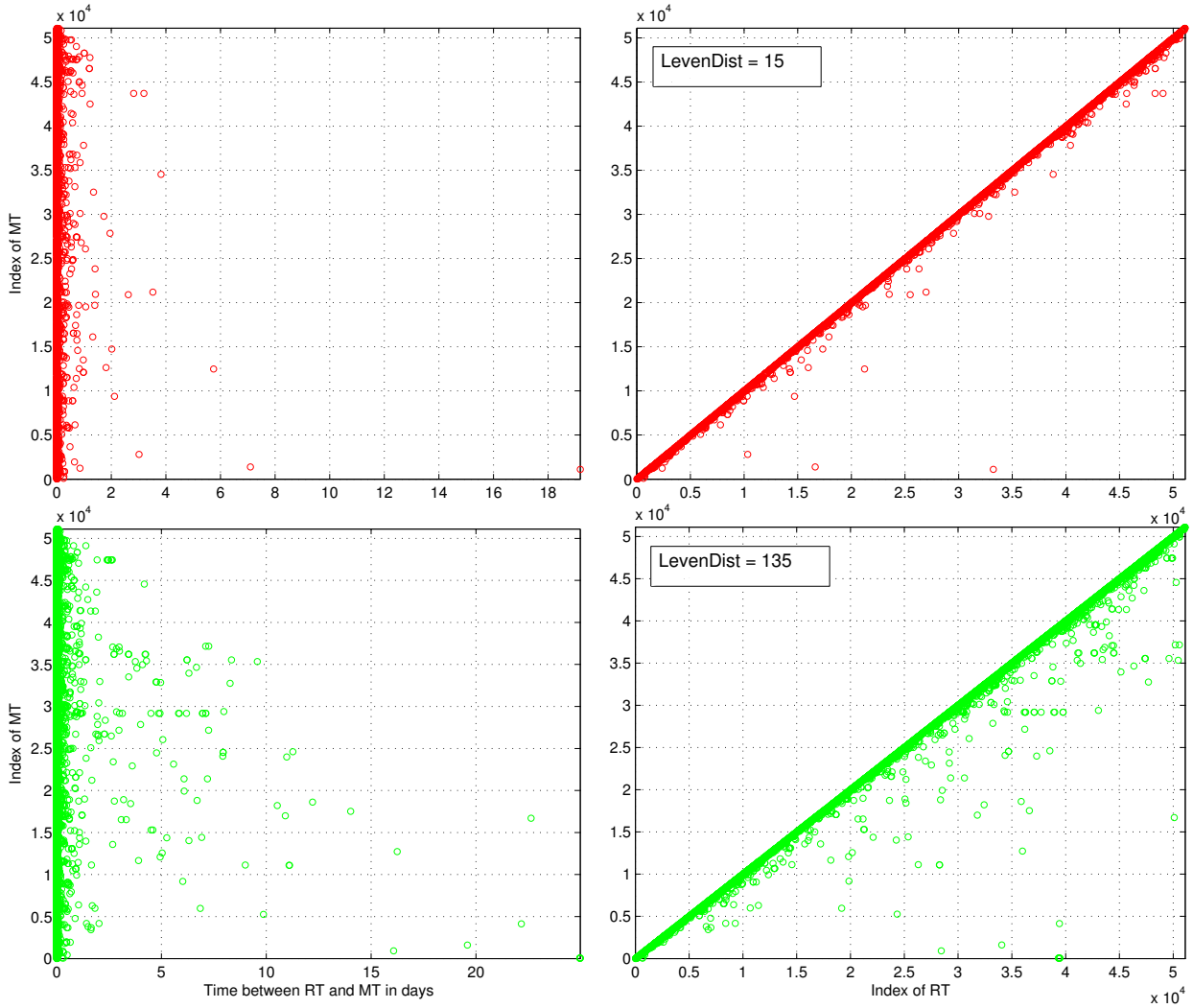

Figure 4: RT statistics for very small and very large Levenshtein distance. What can be immediately seen is that the larger Levenshtein distance is less restrictive, resulting in many more RT-MT-links to be detected. Note that the indices for both the MTs and RTs range from 1 to 51,114, which is the total number of tweets in the data set.

ℓ	α	τ_{min}[days]	τ_{min}[h]
0	1.644 ± 0.034	0.008993 ± 0.000068	0.2158 ± 0.0016
15	1.773 ± 0.020	0.025405 ± 0.000044	0.6097 ± 0.0011
30	1.753 ± 0.018	0.026725 ± 0.000025	0.64141 ± 0.00055
45	1.718 ± 0.015	0.027465 ± 0.000021	0.65916 ± 0.00051
60	1.634 ± 0.014	0.027153 ± 0.000023	0.65167 ± 0.00055
75	1.581 ± 0.011	0.022836 ± 0.000010	0.54806 ± 0.00024
90	1.548 ± 0.010	0.018194 ± 0.000016	0.43666 ± 0.00038
105	1.554 ± 0.010	0.018345 ± 0.000019	0.44028 ± 0.00046
120	1.825 ± 0.031	0.26185 ± 0.00048	6.284 ± 0.012
135	1.798 ± 0.029	0.21909 ± 0.00071	5.258 ± 0.017

Table 1: Scaling parameters α and lower bounds τ_{min} versus different Levenshtein distances ℓ. The values were obtained using [7] and are visualized in figure 6.

not correspond to the most extreme Levenshtein distances, i.e. 15 and 135. This is also confirmed when looking at the α values in table 1. This table was obtained by PL-fitting of each of the Levenshtein distance, which is depicted in figure 6. This will be discussed shortly. There is another peculiarity in the CCDF-curves: the long tails of some of them exhibit truncations, e.g. at 24 days for $\ell = 135$ or at 20 days for $\ell = 90$. This simply corresponds to the maximum response time found in the finite-sized set of 31 days. For the Levenshtein distances for which the CCDF-curve is bent such that this boundary is not touched, e.g. for $\ell = 45$, no such finite size effects, i.e. sharp kinks, are observed. However, in order to eliminate those finite size effects, the present data, which spans over a much larger time span and which therefore will provide more data on the long tail, is investigated in the next subsection on the present data.

To better judge which Levenshtein distance to choose, each of them has been fitted to PLs according to the procedure given in section 4.3. The visualization of this can be found in figure 6 with the respective extracted values for α, τ_{min} and their respective errors given in table 1. Judging by the agreement of the curves with PL-fits as well as the errors of both α and τ_{min}, there is not a single Levenshtein distance that strikes out. Of course this is based on the assumption that a PL is indeed the underlying mechanism, which at this point cannot be confirmed. In fact the agreement of the curves with the PL-fits becomes poorer towards the far end of the tails. This might be due to the application of the KS test which puts more weight towards the head of the distribution. Neither do the scaling parameter values cluster around a small range, which would have indicated that the Levenshtein distance did not matter that much after all. In conclusion we cannot identify a single best Levenshtein distance for the detection of RT-MT-links. Instead we have to present the result of this subsection as a range for the scaling parameter:

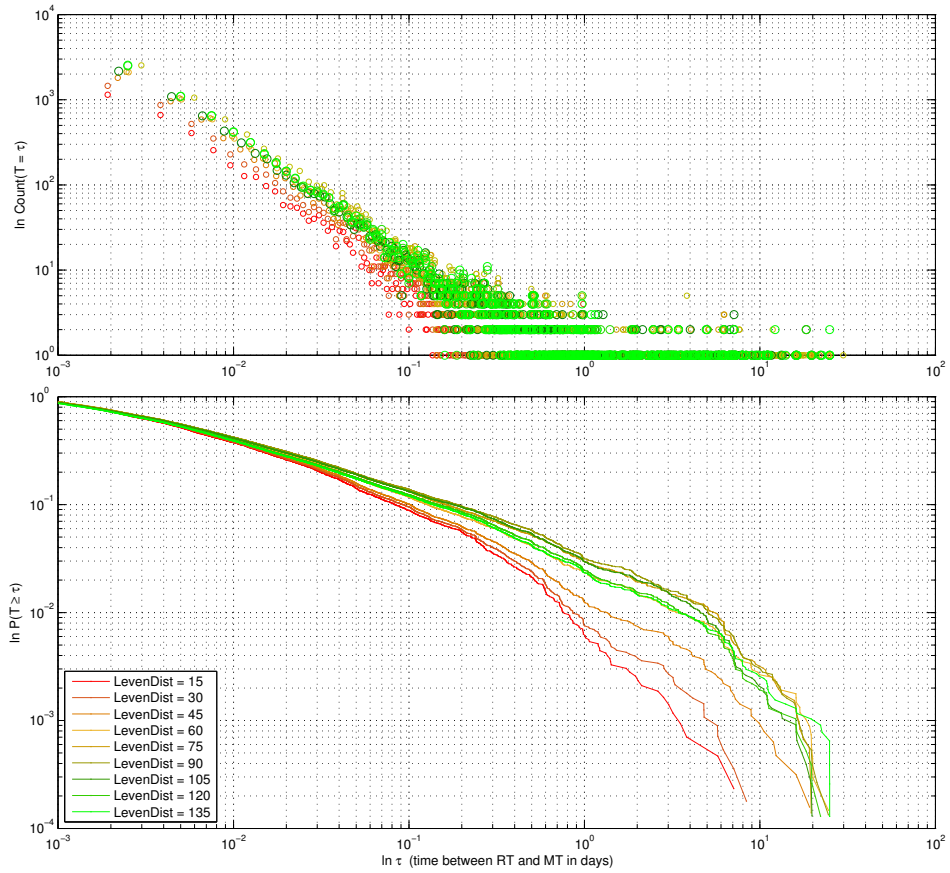

Figure 5: Log-log-histogram and CCDF of response times τ. The former suggests a straight line but is very noisy and wide-spread at the long tail. The latter suggests that the a larger Levenshtein distance unbends the CCDF, shortening the memory. However, when looking closely on the behavior of the CCDF-curves, one can see that the trend of Levenshtein distance negatively correlating with the memory does not hold across all Levenshtein distances. This becomes also evident when looking at the two outermost CCDF-curves. They do not correspond to the most extreme Levenshtein distances, i.e. 15 and 135.

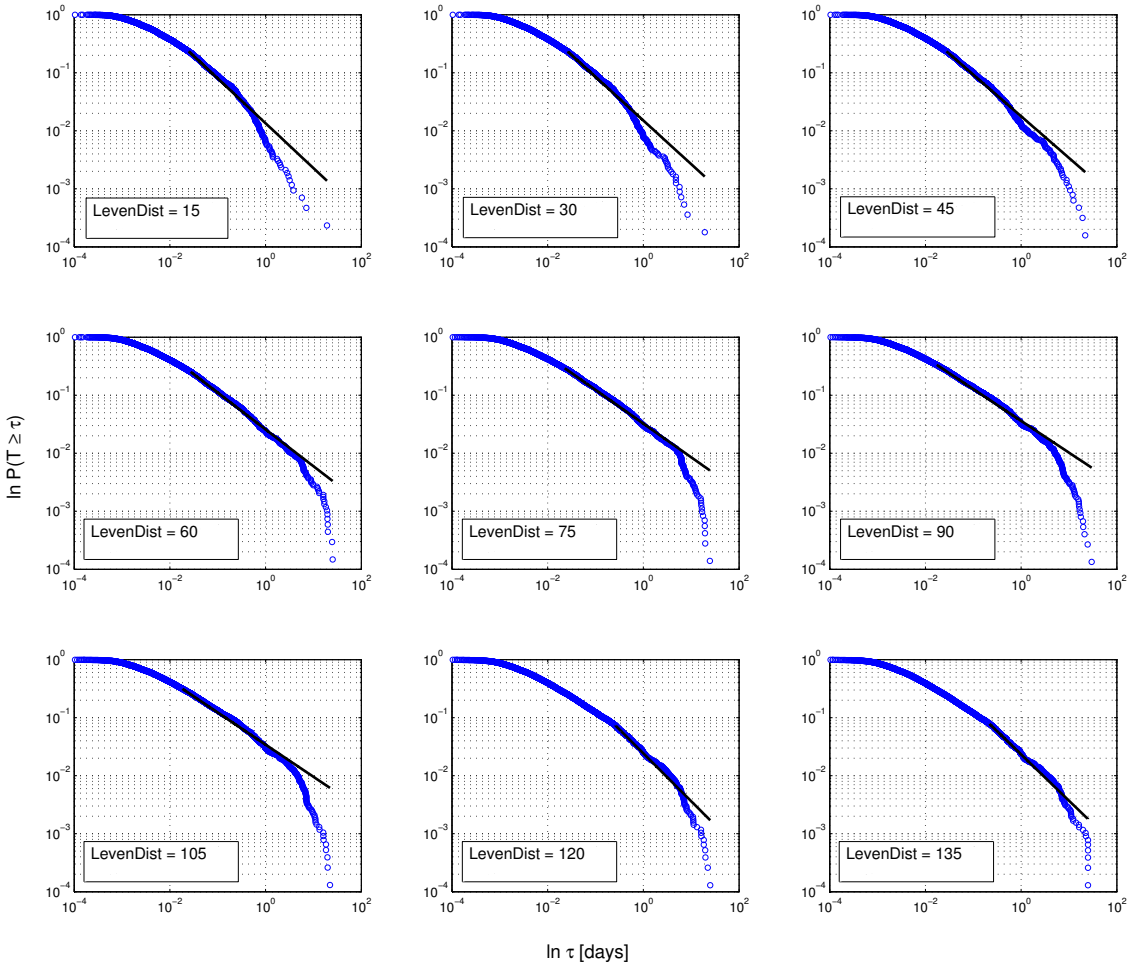

Figure 6: PL fit of response times (manual RTs) distribution for different Levenshtein distances. Recall that the larger the Levenshtein distance the weaker the condition to accept a potential MT as actual MT of a RT detected. The Levenshtein distance of 135 virtually takes the last tweet of the retweeted user. Recall that the maximal tweet length is 140 characters. The PL fits themselves have been performed according to [7]. The respective values for α and τ_{min} can be found in table 1.

$$\alpha \in [1.538; 1.856]$$

$$\tau_{min} \in [0.2142\,\mathrm{h}; 6.296\,\mathrm{h}].$$

So α seems to be definitely less than 2, meaning that a well-defined mean does not exist. Moreover, it is not smaller than 1, a case that was excluded in the theory section 2.2. However, due to the seemingly poor fit at the farther end of the tails, it is likely that the PL fit is in fact steeper, resulting in an α potentially larger than 2. What can be further concluded is that indeed we are dealing with a long memory of the system due to heavy-tailedness of the distribution. This is a non-trivial finding. Intuitively, one would expect a rather short memory of such a fast-paced online platform like Twitter and even more so in such highly active and criticial periods. Newsfeeds of users are limited and display only the most recent tweets on the topic or hash-tag. Yet Twitter users seem to react to older, potentially more important, tweets, too. However, despite the relatively low errors in α it cannot be concluded that the memory kernel indeed follows a PL. In order to corroborate this, further tests of both PL and other statistical distributions that might have better goodness-of-fit-statistics would have to be investigated, too. A better PL test could be based on the Anderson-Darling test, for example.

5.2.2 Present Data

The initital motivation to also analyze the present data which has a much lower tweet activity but a much longer overall time span, was to also allow for more extreme events to be detected, i.e. response that take longer than the maximal time span of the past data. This motivation was corroborated by some of the findings above, where it was clearly visible that for some Levenshtein distances finite size effects occurred in form of truncated memory kernels. In this section the results for the memory kernel of the present data is presented.

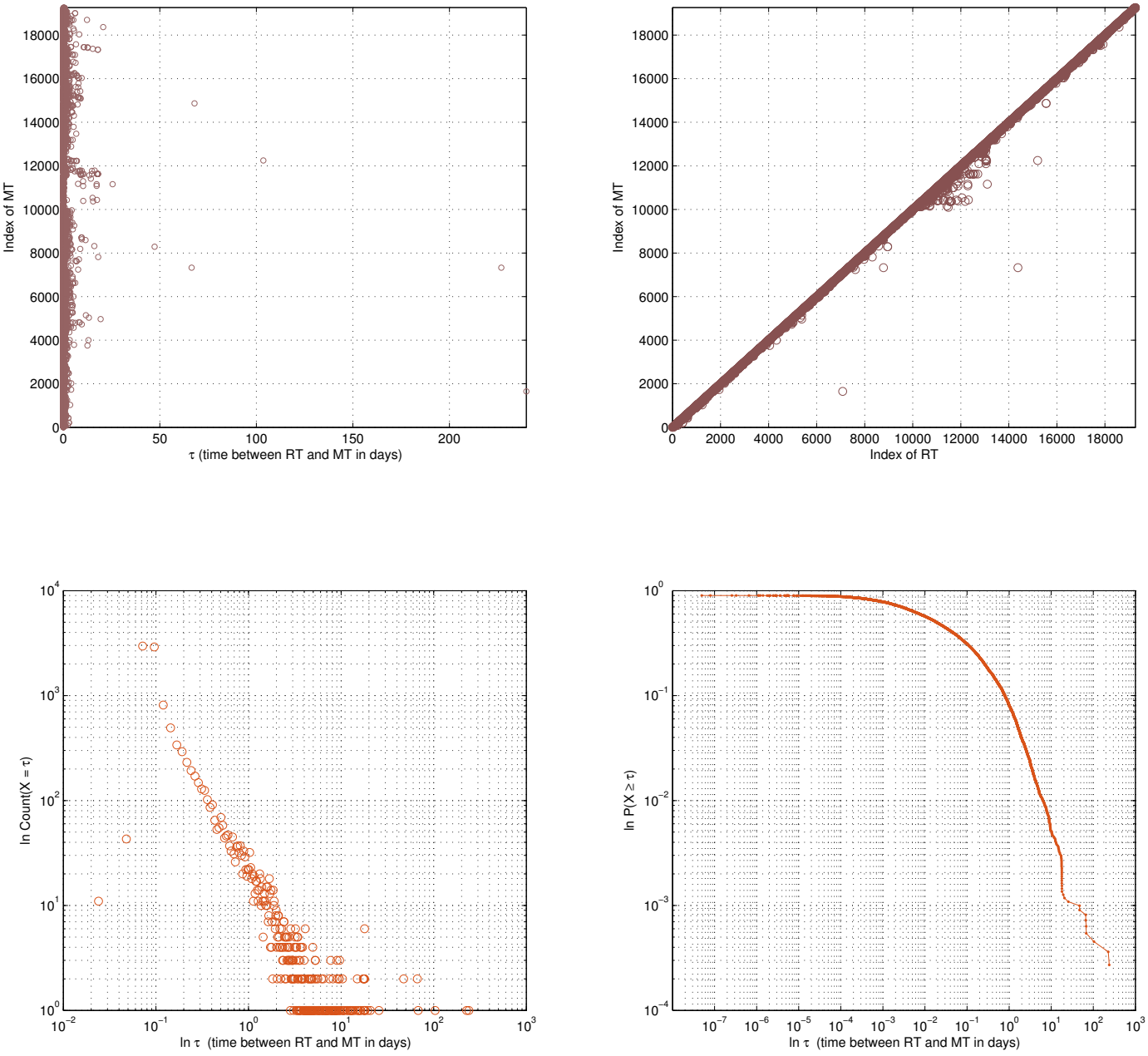

Figure 7: Four subplots visualizing the RT-statistics: MT-index versus τ, MT-index versus RT-index, log-log-histogram and CCDF of τ. The top two graphs indicate the relatively low number of RTs found, when comparing them to figure4. In the CCDF it is visible that the truncation of events due the finite smaller size of the past data set is resolved. Note that the indicies for both the MTs and RTs range from 1 to $19,265$, which is the total number of tweets in the data set.

Recall that in the present data set there is no ambiguity in the RT-MT-linking since this information is contained in the very data. However, this linking is confined to automatic RTs only, i.e. not manual RTs. In analogy to the past data, the top graphs of figure 7 show the RT statistics with the indices of tweets going up to only $19,265$ as opposed to $51,114$ for the past data. First of all, by visual judgement we notice that there are much less RTs in the present data than in the past data. Intuitively this makes sense, considering that we are comparing a low-activity with a high-activity period. Note, however, that here we are comparing the number of *automatic* RTs with that of *manual* RTs. Comparing automatic and manual RTs, respectively, might in fact look different. Unfortunately, the data available did not allow to check this. Second of all, the maximal response time found in the present data exceeds the entire duration of the past data by far: the longest response took about 250 days with the entire duration of the present data being 899 days. The bottom graphs of figure 7 show the log-log-histogram and the CCDF of the response times. We notice that the truncation of the CCDF that occurred in the past data is indeed resolved now. The question is, do those ? Now performing the PL-fit of [7] on the present data results could show whether the now longer tail follows the same law. The results of this are

$$\alpha = 2.237 \pm 0.047$$

$$\tau_{min} = 1.4518 \pm 0.0011 \,\text{days}$$

and are given in figure 8. The scaling parameter is significantly larger than for the past data, which corresponds to a shorter memory. Moreover, this time there is a well-defined mean because $\alpha > 2$. From visual judgement of figure 8 the PL fit extends well into the long tail. Also, the error of the both α and τ_{min} but more importantly the log-likelihood is very low: $-1.2481 \cdot 10^3$ (Likelihood $\mathcal{L} \approx 0.9988$). This seems rather robust at first sight. However, it is unlikely that these retweets happened from MTs being read one the news feed, which is updated on a daily basis. Even when taking the low activity into account, tweets visible in a user's news feed still should not be older than weeks. Yet RTs occured in the order of months. It might be sensible to assume that this must be due to a different way of encountering MTs to be retweeted, e.g. by manual searches of tweets. Therefore, a different mechanism and hence a different regime in the memory kernel might need to be included. This, however, needs further analyses and is not subject of this paper. How does this fit together with the shorter memory of the system in the present data? If there were indeed two mechanisms of encountering tweets to be retweeted, the only explanation would be that in periods of low-activity, users retweet tweets encountered in their newsfeed with a much higher frequency than tweets encountered by manual search. To further investigate this, the different ways of encountering tweets as well as the crude algorithm of which tweets show up in the newsfeed would have to be elucidated. Also the question of whether the memory of the system in the past data is really that long would have to be examined further.

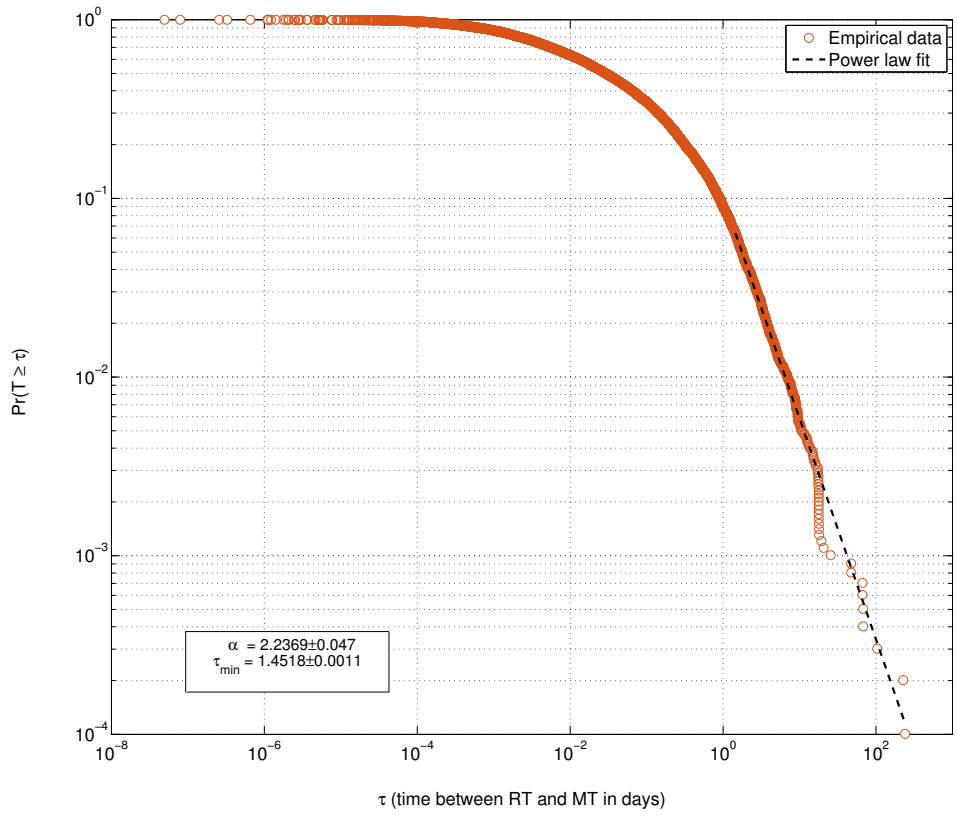

Figure 8: PL fit to the response times in the present data (automatic RTs). We notice that the truncation of the CCDF that occurred in the past data is indeed resolved now. The long tail seems to follow a PL with a small error in the scaling parameter. The PL fits themselves have been performed according to [7].

Memory kernel of ...	α	τ_{min} [days]
All users and responses	2.237 ± 0.048	34.844 ± 0.027
Extreme responses of most active RTers omitted	2.520 ± 0.086	29.092 ± 0.20
Extreme responses of remaining RTers omitted	3.03 ± 0.15	30.093 ± 0.098
Extreme responses of most active tweeters omitted	2.473 ± 0.080	28.53 ± 0.21
Extreme responses of remaining tweeters omitted	3.03 ± 0.15	30.093 ± 0.096

Table 2: Results of the PL fits on the different CCDFs according to [7].

5.2.3 Individual User Data

In the present data it is possible to efficiently discriminate amongst individual users. The question here is how the memory kernels of individual users resemble that of the entire system. Do they follow the same law same distribution? Are they also of long memory as it was concluded for the entire system in the previous subsections? For this we first draw our attention to the most active users in terms of their (re-)tweeting activity as they per definition have lots of responses, i.e. data points in their memory kernel.

The top and bottom graphs of figure 9 depict the same user statistics but with transformed and unaltered response times, respectively. This is simply to show that detrending smoothens the CCDFs as intended. Now looking at the top graph of figure 9, the individual memory kernels of the five most active RTers (at least 300 RTs) are shown. Clearly, each user has their very own memory when looking at the slops and the most extreme events. Only collectively they form a long memory. The CCDFs of the five users might very well follow a PL. However, the last response seems to be a bit off. What happens to the overall memory of the system if these extreme events are cut off and the remaining response times re-aggregated? Are these responses the main contributors to the overall memory? Rather than determining individual user scaling parameters, we determine the scaling parameters of the CCDFs re-aggregated after cutting of the most extreme responses of the most active users. The results for this are presented in figure 10 and in table 2.

We can define active tweeters in terms of number of tweets as well as number of RTs. For both types in figure 10, the five most active users have been cut off of their two most extreme responses. Judging from both the CCDFs in the figure and the scaling parameters $\alpha = 2.520, 2.473$ in table 2, the memory of the re-aggregated system is shorter. Of course this was to be expected when removing the longest responses. The more interesting question is now, how the memory is altered if the two most extreme responses of the remaining 2395 users are removed *instead*. The CCDF-curve of the response times after this removal is also presented in the same figure and table. Again the memory is significantly shortened and - as intuitively expected - even more so than after removing the extreme responses of the much fewer most active tweeters. One could play this exercise further to see at which point - varying the number of cut-off extreme events and the number of most active tweeters - the memory reduction of the most active equals that of the remaining less active users. A

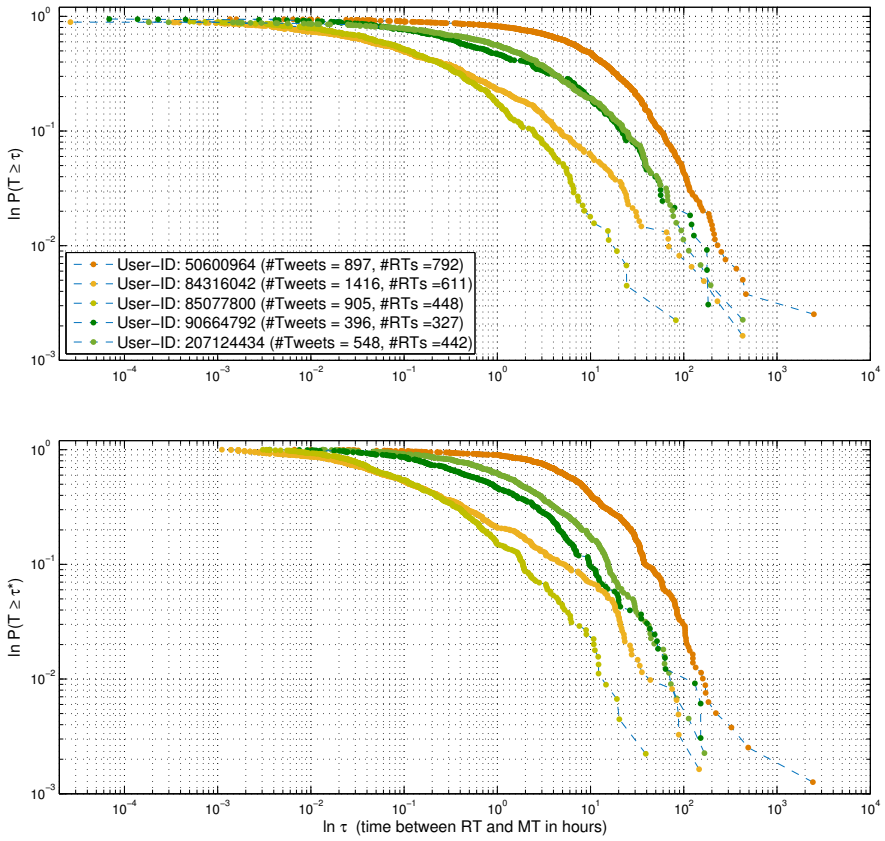

Figure 9: Detrended statistics of users with at least 300 RTs. The top graph was obtained with, the bottom graph without detrending. It is clearly visible that each of the five users have their individual memory expressed in different slopes of their CCDFs. However, the memory for these five users is not related to neither there number of tweets nor RTs. Comparing the top with the bottom graph, it is visible that detrending has the intended smoothing effect.

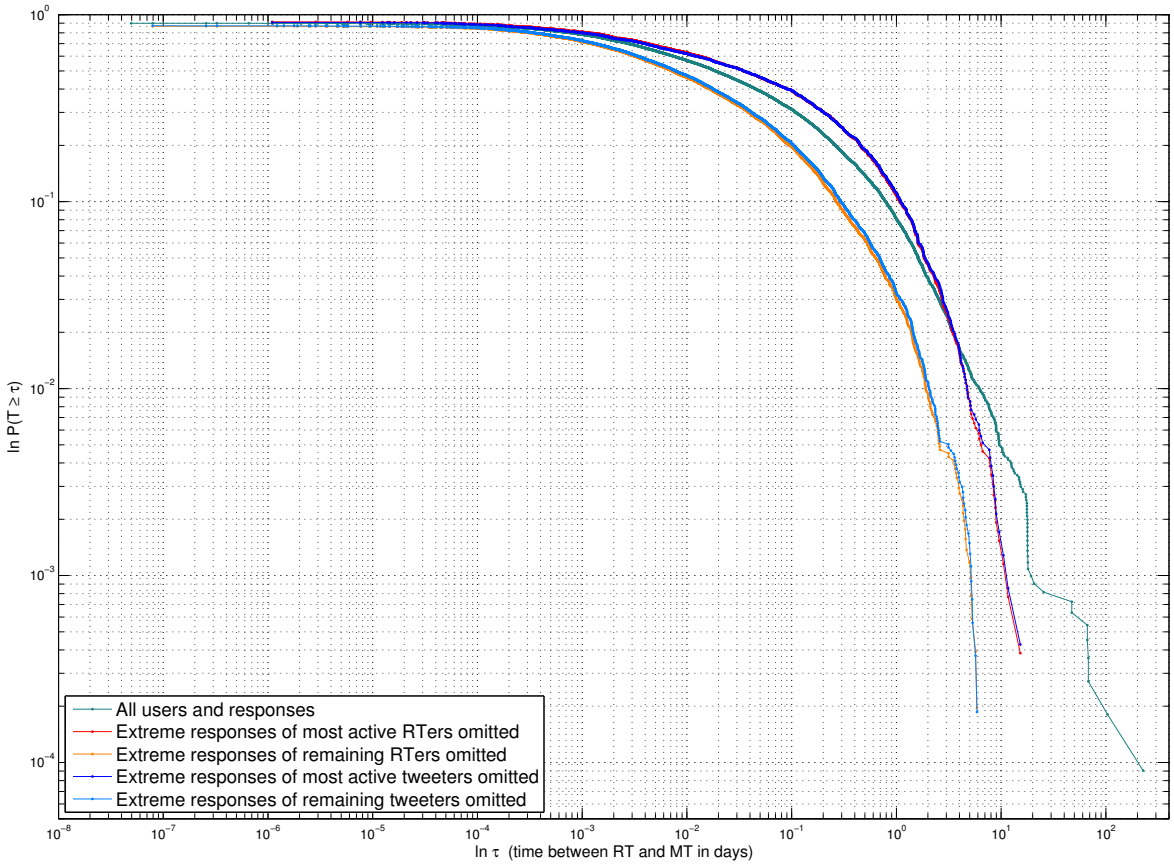

Figure 10: PL fit according to [7] of the response times with user discrimination. Extreme response of most active tweeters and RTers as well as of the remaining users were removed in order to check their effect on the overall memory of the system. The unaltered CCDF of the response times of the present is included for comparison. The respective fitted values can be found in table 2. Generally removing extreme events shortens the memory as expected. Also removing those events from either active tweeters or RTers does not seem to play a big role.

likely outcome is that a smaller number of active users contribute most significantly to the memory of the entire system.

6 Conclusion

The initial motivation of this paper was to set the stage for determining the degree of self-excitation of Twitter messaging, i.e. the emotional ladenness with respect to a certain topic, in our case the student protest in Austria of fall 2009. We have identified the key ingredients to perform this. Equation 1 of the intensity of the self-excited Hawkes process provides these: the unconditional intensity $\mu(t)$ and the memory kernel $\phi(\tau)$.

An unconditional intensity based on reoccurring daily or weekly patterns attributable to external influences, could not be constructed. Instead a daily low-order fit to the RT-filtered, immigrant-dominated MT-intensity had to be performed, assuming that the endogenous events are represented in the higher-order variations of the tweet rate. With this, the unconditional intesity could be detrended such that $\mu(t) \to \mu =$const.

Robustly determing the memory kernel seemed within reach when considering the fact that RT and MT are causally linked. However, the two principal ways of RTing, i.e. manual and automatic, were only individually detectable in the past and present data set, respectively. While for the automatic RTing drawing the RT-MT-links is unambiguous, doing this for the manual RTs is far more difficult. The reason for this is that the Levenshtein distance, the string distance between the MT reconstructed from a detected RT and the actual MT, is in fact a suboptimal criterion for an automized RT-MT-link detection. Nevertheless, in both past and present data a *long* overall memory could be identified, though a short memory had initially been expected. PL fits of the CCDFs, i.e. the memory kernels, revealed that for the past data a range of the scaling parameter $\alpha \in [1.538; 1.856]$ could be determined, for the present a more exact value of $\alpha = 2.237 \pm 0.047$ with a log-likelihood of $-1.2481 \cdot 10^3$. Hence the past data of the high-activity period is in fact of longer memory, which is counter-intuitive. Though the PL fit seems to do justice to the long-memory, further models have to be tested for a more robust conclusion on the underlying laws.

In the following, we want to dive a bit deeper into the shortcomings and limitations, giving suggestions as to how to alleviate them in order to make bigger steps towards quantifying self-excitation in Twitter messaging.

Improve detection of manual RTs

As already discussed when introducing the Levenshtein distance and as the results for the different Levenshtein distances show, choosing this criterion for RT-MT-linking is suboptimal. Simple reorderings of words and parts of the tweet increase the Levenshtein distance, even though the content has in fact not been changed significantly. Generally, reconstructing MTs from detected RTs is a delicate and sensitive undertaking that, how-

ever, bears potential for great improvements in automizing the RT-MT-linking of large data sets.

Combine detection of automated and manual RTs

Twitter with its tweets and causally connected retweets does in principle make a robust analysis on the functional form of the memory kernel very possible. However, this work showed that correctly linking RTs to their correct MTs is of utmost importance. The different ways of retweeting, i.e. automatically and manually, require different methods of detection, which in the best case need to be combined for the *same* data set. The ideal data set would combine the information that was given in the past and present data such that *all* types of retweets can be found in the *same* data set. In concrete that would mean to have data set that includes at least the following:

- Username and user-ID (for individual user statistics)
- Tweet message (for the detection manual RTs)
- Meta-data on RT/MT (for the detection of automatic RTs)
- Time stamp (for the acquisition of response times)

Increase size of data set

The rather limited size of the past data show that due to the possibility of long-memory, the events at the long tail become important, potentially being truncated away. In order to prevent this, larger time spans have to be considered. In our case the two data sets of the past and the present were separated in time. It would be ideal to have one coherent large data set that entails both the high- and low-activity period of a social protest, for example. Also, we have been looking at one particular hash-tag of limited size only. Considering the fact that there may be several hash-tags for one and the same topics, extending the analyses preferably to all of these hash-tags would be a better way for truly capturing the emotional ladenness of this topic. Lastly, generally only about 1% of Twitter data is freely available. By manually crawling data this percentage can be increased. When wanting to connect RTs to MTs potentially in the remaining 99%, many connections might not be drawn. So overall increasing the data set both in time and density is preferable and crucial.

Extend models for fitting the memory kernels

Again, the PL fit seemed to do justice to the long-memory of the tweet system. However, the extreme events in the present data indicate that there might be different mechansims at work in different regimes. Here it is crucial to elucidate the different ways a Twitter user could encounter tweets to be retweeted. Individually checking extreme RTs and their MTs could provide hints for this. In order to account for potential regime transitions, it is possible to treat the PL as family member of a stretched exponential (SE) and/or

log-normal (LN) distributions. Here additional parameters smoothly let PL transition into SE and LN, respectively.

7 References

References

[1] V. Filimonov, D. Sornette. *Quantifying reflexivity in financial markets: towards a prediction of flash crashes*. Phys. Rev. E 85 (5), 056108 (2012).

[2] A. Hawkes. *Point Spectra of Some Mutually Exciting Point Processes*. Journal of the Royal Statistical Society Series B (Methodological) 33 (3), 438–443 (1971).

[3] S. J. Hardiman, N. Bercot, J. P. Bouchaud. *Critical reflexivity in financial markets: a Hawkes process analysis*. The European Physical Journal B 86 (10), 1-9 (2013).

[4] I. M. Toke. *An Introduction to Hawkes Processes with Applications to Finance*. http://lamp.ecp.fr/MAS/fiQuant/ioane_files/HawkesCourseSlides.pdf. Accessed March 27, 2013.

[5] A. Helmstetter, D. Sornette. *Importance of direct and indirect triggered seismicity in the ETAS model of seismicity*. Geophysical Research Letters 30 (11), 1576 (2003).

[6] A. Clauset, C. R. Shalizi, M. E. J. Newman. *Power-law distributions in empirical data*. SIAM Review 51, 661-703 (2009).

[7] A. Clauset. http://www.santafe.edu/aaronc/powerlaws. Accessed September 13, 2013.

[8] B. M. Hill. *A Simple General Approach to Inference About the Tail of a Distribution*. The Annals of Statistics 3 (5), 1031-1188 (1975).

[9] Y. Ogata. *Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes*. Journal of the American Statistical Association 83 (401), 9-27 (1988).

[10] ZEIT Online. *Mit Twitter und Trommeln gegen die Regierung*. ZEIT ONLINE GmbH (2009). http://www.zeit.de/studium/uni-leben/2009-10/studentenproteste-oesterreich. Accessed January 17, 2014.

[11] M. Papic, S. Noonan. *Social Media as a Tool for Protest*. Stratfor Global Intelligence (2011). http://www.stratfor.com/weekly/20110202-social-media-tool-protest. Accessed April 14, 2014.

[12] P. Barberá, M. Metzger. *Tweeting the Revolution: Social Media Use and the #Euromaidan Protests*. Huffington Post (2014). http://www.huffingtonpost.com/pablo-barbera/tweeting-the-revolution-s_b_4831104.html. Accessed April 14, 2014.

[13] Twitter. *FAQs about Retweets (RT)*. https://support.twitter.com/articles/77606-faqs-about-retweets-rt. Accessed October 19, 2013.

[14] A. Saichev, D. Sornette. *Generation-by-Generation Dissection of the Response Function in Long Memory Epidemic Processes*. arXiv:0904.0872 [physics.data-an]. (2009).

[15] D. Zarella. *ReTweet Etiquette*. Personal Blog. http://danzarrella.com/retweet-etiquette.html. Accessed April 14, 2014.

[16] V. I. Levenshtein. *Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady 10 (8), 707–710 (1966).

[17] MathWorks. *Smoothing Splines*. MathWorks Documentation Center (Version R2014a). http://www.mathworks.ch/ch/help/curvefit/smoothing-splines.html. Accessed April 15, 2014.

8 Appendix

Functions called by the main script:

1. `get_taus_past`
    - Function: $\{t_i^*\} \rightarrow \{\tau_j^*\}$
    - Input:
        - `pastC`: Cell of past data that contains time stamp, tweet message, and username.
        - `LevenshteinDist`: Vector of Levenshtein distances to loop through. Minimum distance is zero (no agreement of strings), the maximum 140 (full agreement of 140-character tweets). The distance defines the criterion for a potential MT to be accepted as actual MT of a RT found.
    - Output:
        - `RTMTtau_pastvsLeven`: Cell of past data that for each element of `LevenshteinDist` contains the matrix with RT-index, the respective MT-index and $\{\tau_j^*\}$, i.e. response time between the MT and its RT. Note that *index* refers to index of the matrix, whereas *ID* refers to the actual ID of the tweet.
    - Pseudo code:
        (a) Loop through each Levenshtein distance as given by `LevenshteinDist`.
        (b) For each Levenshtein distance loop through all tweets as given by `pastC`.
        (c) For each tweet complement username and tweet itself by spaces such that string lengths are uniform and thus better comparable.

(d) In each tweet search for the string part "RT @" based on the convention that RTs have the following general form: "[addition 1] RT @[retweeted user]: [MT] [addition 2]" (or space instead of semicolon).

(e) Extract [retweeted user] from RT by taking the characters between the "@" and the semicolon (or space).

(f) Extract and reconstruct [MT] from RT by taking everything after the semicolon. Note that there it is currently not implemented to discriminate between [MT] and [addition 2].

(g) Complement each reconstructed MT with spaces for better string comparison.

(h) Store time stamp of RT in `time_RT` and search for its respective MT in the *previous* tweets:

i. Only check RTs of [retweeted user], i.e. `RTd_user` in the code.

ii. Check his/her tweet and compare it with [MT], i.e. `MT` in the code.

iii. If Levenshtein distance criterion is fulfilled, accept this MT as MT of the found RT.

iv. Subtract time stamp of MT `time_MT` from stored time stamp of RT `time_RT` to obtain response time τ_j^*.

v. Also store the index with respect to the time series, i.e. i of $\{t_i^*\}$, for both RT and MT. E.g. 51,000-th tweet is the RT of the 50,000-th tweet (MT).

(i) Store matrix with RT-indices, the respective MT-indices and $\{\tau_j^*\}$, i.e. response time between the MTs and their RTs, into `RTMTtau_pastvsLeven` with cell index for the current Levenshtein distance.

(j) Return `RTMTtau_pastvsLeven` after looping through all Levenshtein distances.

2. `detrend_anyFit.m`

- Function: $\{t_i\} \to \{t_i^*\}$
- Input:
  - `binsize`: Bin size for tweet rate $R(t)$ in minutes. Default values: 240 [min] for present data, 10 [min] for past data.
  - `unit`: Unit of time in $R(t)$ on which spline is fitted. Default values: 'month' for present data, 'day' for past data.
  - `idx`: Indices of `unit` which are looped through. Default values: 1:31 (31 calendar months) for present data, 1:33 (33 calendar days) for past data.
  - `tMT`: RT-filtered time stamps $\{t_i\}_{MT}$.
  - `tALL`: Unfiltered time stamps $\{t_i\}$.

- `visual`: Boolean for visualizing results.

- Output:
  - `tstartot`: Total transformed time stamps of tweets $\{t_i^*\}$ in units of hours.
  - `gofcell`: Goodness of fit values (Detrendings uses spline-fit). Note that a too high goodness is not aspired (recall subsection 2.3).
  - If visuals on, four graphs for each detrended basic unit: $R(t)$ with spline-fit, $R(t^*)$ with mean of spline-fit, $N(t)$ for different intensities, residuals with quantiles.

- Pseudo code:
  (a) Splits $\{t_i\}$ and $\{t_i\}_{MT}$ into basic units, e.g. $\{t_i\}_{MT,month}$ via function `get_timeperunit.m`
  (b) Check `unit` to define different smoothing parameters and fitting ranges `Tmax` for the spline fit.
  (c) Loop through each `unit`, i.e. day or month, to perform detrending on each.
    - From $\{t_i\}_{MT,unit}$ obtain tweeting rate $R(t)_{unit}$ in units of 1/`binsize` [min].
    - Perform smoothing spline fit on $R(t)$ and integrate the fitted function to obtain $\{t_i^*\}_{unit}$.
    - Store goodness of spline-fit statistics into `gofcell{i}`.
    - Normalize $\{t_i^*\}_{unit}$ such that $t^*_{max,unit} \stackrel{!}{=} t_{max,unit}$
    - Concatenate current $\{t_i^*\}_{unit}$ to the previous $\{t_i^*\}_{1...unit-1}$
    - If visuals are on, plot and save the four graphs mentioned under "Output".
  (d) Return $\{t_i^*\}$ and `gofcell`.

3. `ts2taus.m`

- Function: $\{t_i\} \to \{\tau_j\}$ (or with asterix $*$)
- Input:
  - `tevnts`: Detrended or undetrended time stamps.
  - `RTMTtau_past`: Matrix that contains the RT- and MT-indices w.r.t. the original time stamps vector `tevnts`. It also contains the taus but we are interested in calculating the taus from a different set of tevnts (e.g. tstars).
- Output: Response times, `taus`.
- Pseudo code:
  (a) Loop through all MT-RT-links, i.e. all rows of `RTMTtau_past`.
  (b) Obtain the indices of the RT and its MT with respect to the time stamp vector `tevnts`.

(c) Subtract the respective time stamps as given by `tevnts` to obtain the response times.

(d) Return response times `taus`.

4. `get_taus_pres.m`

- Function: $\{t_i^*\} \rightarrow \{\tau_j^*\}$
- Input:
  - `presC`: Cell of present data that contains MT-ID (if applicable), tweet-ID, time stamp, user-ID, and RT-count (-1 if RT)
  - `tevnts`: Transformed (or original) time stamps of present data in units of hours, i.e. $\{t_i^*\}$ (or $\{t_i\}$).
- Output:
  - `RTMTtau_pres`: Matrix of present data that contains RT-index, the respective MT-index, and $\{\tau_i^* j\}$, i.e. response time between the MT and its RT. Note that *index* refers to index of the cell, whereas *ID* refers to the actual ID of the tweet.
- Pseudo code:

(a) Loop through all tweets to search for RTs which are marked by -1 in the RT-count column.

(b) If a RT is found, store its time stamp and MT-index and look for this its MT in inner loop:
  - If correct MT is found, store its time stamp to get the response time τ_i^*
  - Store RT-index, MT-index and τ_j^* into `RTMTtau_pres`.
  - Break inner loop.

(c) Print progress as fraction all tweets.

(d) Return `RTMTtau_pres` which contains key variable $\{\tau_j^*\}$.

5. `get_user_taus.m`

- Function: $\{\tau_j^*\} \rightarrow \{\tau_{user,j}^*\}$
- Input:
  - `presC`: Cell of present data that contains MT-ID (if applicable), tweet-ID, time stamp, user-ID, and RT-count (-1 if RT)
  - `RTMTtau`: Cell of present data that contains RT-index, the respective MT-index, and $\{\tau_j\}$, i.e. response time between the MT and its RT.
- Output:

  - `user_taus`: Cell in which each row contains corresponds to one user with unique user-ID, number of active RTs, number of tweets, vector of individual response times.

- Pseudo code:

  (a) Get vector with unique user IDs and loop through all users:
    - Set RT and tweet counter to zero.
    - For each user scan whole data set to find and count his/her tweets.
    - If tweet is a RT, find the corresponding response time in `RTMTtau`.
    - Store the following into `user_taus`: user ID, RT count, tweet count, individual response times.

  (b) Print progress as fraction all users (2400).

  (c) Return `user_taus`.