Meetup Miner

Measuring Event Interestingness on Meetup

Maximilian Grundke, Jaeyoon Jung, Jan Philipp Sachse, Georg Wiese

Hasso Plattner Institute, Potsdam, Germany

Abstract Quantifying event interestingness in Event-Based Social Networks is crucial to filter for compelling events. However, because interestingness is inherently subjective, it is impossible to universally define. We propose a set of features based on the event description as well as the RSVP history of related events that indicate interesting events. Furthermore, we introduce a method to combine them to an interestingness score that is derived from user-specified preferences. We provide details of our implementation for Meetup[1] events and deliver a functioning web application prototype as a proof of concept.

1 Introduction

In recent years, social networks have become an important part of most people's lifes. Widely known networks such as Facebook or Twitter count millions of users, host a huge amount of interesting data and are subject to ongoing research. Additionally, social networks with a more specific target audience have emerged, one of which is Meetup.

Meetup is a so-called "Event Based Social Network" (EBSN), which allows users to gather in groups online and create and manage events of any kind. It has about 20 million members and about 500,000 Meetups take place every month all over the world [4]. EBSNs are centered around interests of people and their demand to meet other like-minded persons face-to-face. Therefore, it is not the online social interaction between friends that is the most important part, but the possibility to create "offline" spaces ("Meetups") for people who share common interests. As Meetup becomes more and more popular around the world, more events are created and it becomes harder to find interesting ones. The platform provides the option to filter events by location, topic, number of members, and date of creation. Furthermore, users have the possibility to connect to other Social Networks and find meetups their friends are attending. Using events attended in the past, interests given at registration and real-world

[1] Meetup. `http://www.meetup.com/`

friends, Meetup also has a "Recommended Events" section, which defaults to random events, if the user has not provided details and never attended an event before. This is commonly known as a cold-start problem in recommendation systems [3].

In order to present users more interesting events without requiring historical information about users, it is important to know what "interesting" means for them. Instead of making assumptions, our work focuses on giving users more tools to define "interestingness" themselves. We crawled Meetup for this purpose and developed five experimental features, based on the data that is already available on the Meetup platform. The concepts of the features and the crawler are explained in Section 3 on the facing page and details of the implementation can be found in Section 4 on page 7.

2 Related Work

In recent years, EBSNs have increasingly gained focus of the research community. Many different approaches have been taken, some of which are covered in this chapter.

In 2012 Liu et. al. [3] described event-based Social Networks as "a co-existence of both online and offline interactions". By analyzing crawled data from Meetup and Plancast[2], some unique properties of EBSNs have been collected. For example, the authors state, that "events present very regular temporal and spatial patterns." Furthermore, some problems with event recommendation have been identified, including the cold-start problem in event participation prediction.

Similar observations were made by de Macedo and Marinho in [1], who conclude that "[EBSNs are] quite different from typical recommendation domains, since there is an intrinsic new item problem [...] and scarce collaborative information". Their work includes in-depth analyses of RSVPs, event lifetime, and the effectiveness of traditional recommendation systems, such as collaborative filtering. The authors propose to use past events, group memberships, and event metadata (such as descriptions and tags) to predict the attendance of future events.

In [6], Xu et. al. examined the influence of the event size on promoting new social connections between users. The data was acquired from Douban[3], a Chinese EBSN similar to Meetup. The authors conclude, that small events are more suitable to make new connections. It is worth noting, that qualitative interviews concerning the topic showed a great "need to understand users' goals in the design of event spaces, sizes, and structures".

[2] Plancast. `http://plancast.com/`
[3] Douban. `http://douban.com`

3 Concept

In order to allow users to find more interesting events, we propose several tools. We developed the following five measures to filter events by, four of them being directly related to the event and the people that are going to be there.

- **Number of People** The number of people that are probably going to go to the event. This is an important feature, as the event size also influences the kind of atmosphere to expect (compare [6]).
- **Trend** The slope of the trendline describing the number of people going to events of a group (i.e., whether there are going to be more, less or equally as many people at future events).
- **Expected Member Loyalty** A measure that describes how many members are going to be at the event that have been at previous similar events of this group.
- **Formality** A measure that describes the level of an event being formal (events on Meetup range from partying and drinking beer together to having a workshop with a regulated schedule and timed speakers).
- **Compactness** Allows filtering by event descriptions that feature more relevant words in a shorter text.

Three of our features are based on RSVPs, as they are a very good source of information about the people that represent an event. Unfortunately, there are few RSVPs given per event on average. Meetup additionally features several big groups containing events with over 100 RSVPs, which results in many events having zero or only one response online (see also [1] and [4]). The two remaining features are based on the description of events and use machine learning and the computation of a TF-IDF value for their prediction. The concepts of these approaches are discussed in detail in the following subsections.

For the sake of precomputing values of upcoming events we need access to as much data of past events as quickly as possible. Therefore an offline dataset is necessary for the computations based on RSVPs and also for the training of the machine learning algorithm. We wrote a crawler that downloads the necessary data for a given city from Meetup and saves it to a database. The architecture and function of the crawler is explained in Section 4.1 on page 7. We initially downloaded available information of groups from big cities in America and Europe, such as New York, San Francisco, Chicago, London and Berlin. Based on these groups we then downloaded information about events, members that went there, their profiles and more. In the end we used the data crawled from Chicago for our development and testing purposes, as the city was small enough to predict all values for the upcoming events and big enough to evaluate the quality of our approach.

3.1 Event Neighborhood

Some of the features that we compute for upcoming events are predicted from past events of the same group. Considering, for example, the size of the upcoming event $e_{upcoming}$: We predict it by computing a weighted average over the event sizes of past events $E_{past} = \{e_1, e_2, ..., e_n\}$, using weights $w^{(1)}, w^{(2)}, ..., w^{(n)}$ with $\sum w^{(i)} = 1$:

$$size(e_{upcoming}) := \sum_{i=1}^{n} w^{(i)} \cdot size(e_i) \tag{1}$$

The general problem we faced with this approach was to find appropriate weights. Phrased differently, we try to determine a measure of how much information a particular past event gives us about an upcoming event. The result is what we call the "Event Neighborhood", which will be described in the following paragraphs.

There are three components from which a particular weight $w^{(i)}$ is computed:

- The time weight $w_t^{(i)}$: This should account for the intuition that events that are close by should be weighted higher than events that are far in the past. It is modeled as a function of the time difference Δt between e_i and $e_{upcoming}$. It falls exponentially and is parametrized by the half time T_{half}:

$$w_t^{(i)} := 2^{-\frac{\Delta t}{T_{half}}} \tag{2}$$

- Similarity weight $w_s^{(i)}$: This should account for the fact that some groups host different types of events. It is a measure that quantifies the similarity between $e_{upcoming}$ and e_i.
- *boost*: As a small optimization, we included a boost if the two events were created at the same time. In the case of Meetup, this indicates that they were created in one bulk. Therefore, the probability for the two events being of the same type is increased.

To compute $w^{(i)}$, we multiply $w_t^{(i)}$ and $w_s^{(i)}$, add the *boost* and make sure that the result never exceeds 1:

$$w^{(i)} := min(1.0, w_t^{(i)} \cdot w_s^{(i)} + boost) \tag{3}$$

An example is visualized in Figure 1 on the next page. As we are weighting all past events of a group (instead of just using events of the same kind) and using them in our calculation, we possibly lose some accuracy, but gain advantages on the cold-start problem, as described in [3].

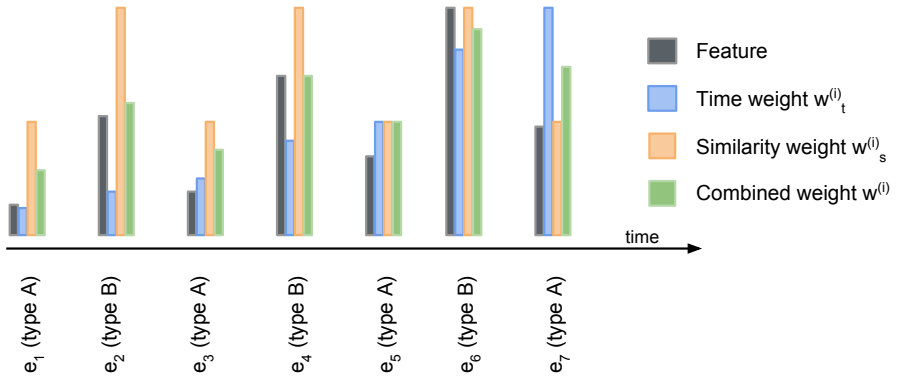

Figure 1: Illustration of the time weight $w_t^{(i)}$ and similarity weight $w_s^{(i)}$ components of the Event Neighborhood weight $w^{(i)}$. $e_{upcoming}$, which is not included here, is in the future and of event type B. Time weight $w_t^{(i)}$ is highest for the latest event e_7 and falls exponentially with half time T_{half} according to Equation (2) on the facing page. Similarity weight $w_s^{(i)}$ is high for all events of type B and low for all events of type A. The combination of all weight components according to Equation (3) on the preceding page results in e_6 having the highest total weight in this example.

3.2 Text-based Features

Every event has a description that contains the basic information such as what is done during the event, what is expected from participating members, etc. Based on the text analysis of descriptions, we introduce two text-based features, formality and compactness, which we consider useful for finding events of interest.

Formality Formality literally refers to the level of an event being formal and is therefore one of the main factors that decides the characteristics of an event. Events on Meetup have a wide range of formality from being highly informal to being highly formal and in general, there are more informal events than formal ones.

Formal events generally have a predefined event schedule, which may contain one or multiple presentations by employees from the industry. Formal events can also specify a dress code and are more likely to be sponsored by one or multiple organizations. In addition, it is likely that they deal with technical topics, such as big data or entrepreneurship. Most of the time, talks are followed by serious discussions, which leads to the events being impersonal, i.e., it is not the person but what they do that happens to be important.

Informal events are more casual and aimed at making new friends and having fun. It is not common that they have any specific dress code. Informal events are also likely to come with food, music and drinks during the events.

Unlike other attributes of events such as location and RSVPs, descriptions are in the form of text written by event hosts. We assumed that the descriptions of formal and informal events would be written in a different way, i.e., using different sets of vocabularies and writing style. That is to say, the style a description is written in might give a hint on whether the corresponding event is formal or informal. In order to classify events based on the descriptions, we use text analysis and machine learning techniques.

Our initial approach was a binary classification, i.e., the differentiation between formal or informal events. In order to obtain a training data set, we randomly selected 200 event descriptions and manually annotated them as either formal or informal. However, we noticed that while some descriptions can be easily annotated, some are hard to tell. Furthermore, even formal- or informal-annotated descriptions have a broad range of formality. Therefore, we decided to use linear regression which predicts the level of being formal rather than binary classification. We annotated the descriptions in a range of zero to ten, with zero being the most informal and ten the most formal. During the annotation process, we wrote down the words from the descriptions that we thought made the corresponding event formal or informal. We call these two features Formal or Informal Content Words, which we used for the machine learning process.

According to Sheikha et al., formal and informal texts are highly likely to be written with different writing styles [5]. They use different vocabularies for the same contents, as shown in Table 1. Examples for words often used in informal texts are "about", "ask for" and "at once", while formal texts rather use words like "concerning", "request" and "immediately". Informal texts also use many contraction words and abbreviation words, while formal texts do not. We used these six writing style features and two content words features, which we mentioned above.

Feature Name		Example
Informal	Informal Words	about, ask for, at once, chance
	Contraction Words	we'll, it'd, don't, can't, isn't
	Abbreviation Words	e.g., Jan., Mon., xmas
	Informal Content Words	karaoke, board game, casual, romance, love, flirt
Formal	Formal Words	concerning, request, immediately, opportunity
	Non Contraction Words	we will, it would, do not, cannot, is not
	Non Abbreviation Words	for example, January, Monday, Christmas
	Formal Content Words	workshop, donation, dress code, seminar, certified

Table 1: A List of Features and Examples for Predicting Formality.

Compactness Although Meetup encourages event hosts to make event descriptions concise, many event descriptions are long and contain a lot of not as useful information. We introduce the concept of compactness, an index of how many relevant keywords are contained in a given event description. We believe that the compactness will enable users to understand an event quickly, thus find an event of interest quickly, especially when there are increasing numbers of events of similar topics and their descriptions are long.

We decide the compactness of a given event by dividing the number of keywords by the number of words in the event description. Every word in an event description has different levels of being relevant to the event topic or what they actually do in the event. Words are defined as keywords if their relevance exceeds a certain threshold. To achieve this, we use TF-IDF values for each word in the event description.

4 Implementation

In order to store the data crawled from Meetup, we used an SAP HANA database and developed a relational database schema, containing 20 different tables for the entities we crawled and the relations between them. To use as many of the available optimizations, we chose to use column table layouts, which allow fast attribute-wise filtering. The following sections will cover the implementation of the developed crawler and our calculation and machine learning approaches.

4.1 Crawler

In order to access and interpret the information available on Meetup, we need to save them to a controlled environment and create an offline data set. The crawler built to achieve this goal is implemented in Python 2.7 and the following section will explain its structure and function.

We chose to develop the crawler in Python, because it requires minimal setup to develop across platforms and to connect to the provided SAP HANA database. It accesses the available information using the Meetup application programming interface (API) v2. Most of the data provided can be accessed through this API version, which provides a RESTful interface with a common JSON response format. Some nodes are only reachable using the first version of the interface, which has another response format, but is also supported by our implementation. For a full list of data nodes that can be accessed and downloaded with the crawler see Table 2 on the next page. In addition to multiple response formats, some response fields are only available for organizers of events or groups and therefore simply not contained in the information sent back from the Meetup server. Other

fields have to be explicitly requested, before they are contained. While most of this is declared in the official API documentation, this information is sometimes missing.

Furthermore, Meetup reserves the right to throttle or block future access to the API to ensure equal quality for all customers. If it is detected, that there are too many requests in a given period of time, the desired response will not be given and instead be replaced by an HTTP 529 error response, which contains further information, whether the access is only throttled or blocked for the next hour.

API Node	Board	Discussion	Discussionpost	Event	Group	Member	Profile	RSVP
API Version	1	1	1	2	2	2	2	2

Table 2: Supported API nodes and their API version

Architecture In order to follow the separations of concern principle, the crawler consists of three main components: fetchers, serializers, and persisters. For a general overview of the architecture of the crawler, see Figure 2 on the facing page. There is a specific fetcher class for each object to crawl from Meetup. It knows its API node, what fields have to be manually requested and if and which additional information has to be sent to the server as well. In the case of failure, accessing the information is tried multiple times in consideration of network errors and the throttle/block response codes sent by Meetup.

Unfortunately, not all information for a single class of objects can be accessed by one call to a corresponding API node, but instead some information is contained in responses to different other requests. Therefore it is necessary to prepare and order the data before it can be saved to a relational database.

This step is done by serializers. They get the JSON-response provided by Meetup and split the information contained according to a relational schema. For example, sponsors are extracted from group information and then saved in a separate table in the database, while the relationship between both entities is conserved using a third table. Each serializer finally produces a set of information containing the table the data has to be inserted into, the list of attribute values for each row and a list of identifying attributes and their corresponding values in order to allow updating existing data sets if the crawler runs multiple times.

This set is then used by a persister to create insert statements for the database. We built two kinds of persisters: one that creates `INSERT`-statements for any database that supports the SQL standard and one that generates SAP HANA specific `UPSERT`-statements, which are then saved in a queue, so that it is

possible to download information independently of actually inserting it into the database. The last object in this chain is a thread, that pulls statements from this queue and uses a database connection to execute them.

The modular structure of the crawler allows it to be adjusted for other database systems, new API versions of Meetup and additional nodes that have to be crawled, as only one part has to be switched out, enhanced or modified.

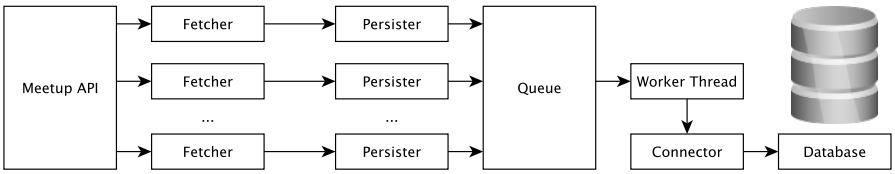

Figure 2: The general architecture of the crawler

Optimizations As most of the runtime of the crawler is produced by waiting for network traffic (so basically I/O), requesting and downloading the information from Meetup is executed using multiple threads. It is freely configurable how many threads should be used, but also limited by two factors.

Firstly, the already mentioned queue is also used to regulate the load that is produced by the crawler on the executing machine. If there are already more than 1000 jobs that wait for execution on the database, the fetchers will wait until the count falls under this threshold once again. This reduces the grade of independence between crawling and saving, but still leaves some buffer between producer and consumer.

Secondly, access to the API is monitored and, if necessary, throttled or blocked by Meetup. Using too many threads quickly leads to the issue of matching or exceeding this artificial border. We expanded our room by using multiple API-Keys for our requests. This is possible, as throttling is based on the combination of IP-address and API-Key of the requesting entity. Therefore, increasing the number of keys also increases the rate in which requests can be sent to the server without getting blocked.

This is why the insertion into the database is the most time-consuming part. To improve the speed, the `UPSERT`-statements are executed as batches instead of separately.

Finally, the crawling is interruptible and can be continued later on, as an additional field is saved to the database that indicates whether a group and all corresponding information has been completely downloaded.

4.2 Data Set Characteristics

We collected offline data from multiple cities all over the world. As Meetup is based in the US, most of its users can be found there. The biggest city worldwide in terms of Meetup-usage is New York City. It is location to more than 770,000 events, which were attended by more than 800,000 different people. In Europe, London is the most prominent city with about half as much members. Additionally interesting to us were German cities. From these, Berlin and Hamburg were crawled to compare them to the rest of the world. Some additional places we crawled, that are not listed in Table 3, are more US-American and Asian cities.

For the analysis and prediction parts of our work, we chose the city of Chicago as enclosed data set, as it has many more events than European and Asian cities, but not as many as New York City. It is also notable, that Chicago-based events have on average more RSVPs per event than comparably-sized locations.

Another interesting point is the analysis of information given by organizers and other users. As Table 3 shows, nearly all of the groups listed on Meetup contain a description. Additional insights into the data show, that these can be quite extensive, with a maximum length of over 28,000 characters in a Chicago-based group. This is one of the reasons, we focused two of our five interestingness features on text analysis. Some of the other user data however has not proven to be useful. For example, members can link their Meetup accounts to other social media platforms. As our statistics show, not only have less than 10% of all users connected to other platforms at all, but also, these connections are to four different social media sites, making the data even more sparse. Given these obstacles, we decided against using data about social media connections for the interestingness features.

	Berlin	**Hamburg**	**London**	**Chicago**	**New York**
Members	34671	9122	414290	236255	807560
Groups	752	191	6499	3294	10943
Groups with a description	744	188	6464	3272	10862
Groups with at least one event	638	72	2333	2513	8440
Events	17589	688	63048	236300	770563
Events per group with events	27.57	9.56	27.02	94.03	91.30
Events with at least one RSVP	12860	135	7939	202541	645678
RSVPs per event with RSVPs	14.71	10.87	7.40	9.05	8.76
Percentage of members with at least one linked social media account	9.96	8.63	7.58	5.98	6.76

Table 3: Data Set Statistics

4.3 RSVP-based features

We implemented the Event Neighborhood concept as part of our Event prediction application. For the half time T_{half} a value of two months is used. The similarity $w_s^{(i)}$ between $e_{upcoming}$ and a past event e_i is computed as the Levenshtein distance between the titles of the event, with a minimum value of 0.2. This rough approximation can be justified by the observation that many groups on Meetup name many events the same which in turn indicates that they are of the same type.

Once the Event Neighborhood is computed, we can use it in a straightforward manner to compute many of the RSVP-based features. The expected size of an upcoming event is a direct application of the Event Neighborhood idea and can be computed as in Equation (1) on page 4.

The trend of an event is computed by doing a weighted regression on the event sizes of the past events E_{past} using the Event Neighborhood weights. The resulting slope quantifies how fast the event is growing or declining. Compared to doing unweighted regression, this method computes a more short-term trend (because recent events are weighted higher) and takes different event types into account (because past events of the same type as $e_{upcoming}$ are weighted higher).

The expected member loyalty is also computed by directly applying the Event Neighborhood idea to the member loyalty values of past events in E_{past}. Member loyalty itself is defined as follows: Let M_i be the set of members that went to event e_i. We define the value $common_{i,j}$ as the fraction of members that went to event e_i that also went to event e_j (see Figure 3 on the following page):

$$common_{i,j} := \frac{|M_i \cap M_j|}{|M_i|} \tag{4}$$

The member loyalty value for event e_i is then defined as the weighted average of all $common_{i,j}$ values using the Event Neighborhood weights with respect to e_i. Note that since the Event Neighborhood only defines weights for events that are past relative to the event in question, the member loyalty value of a particular event e_i only depends on events that took place before e_i.

4.4 Text-based Features

Formality For predicting formality, we use Spark, a framework that provides various machine learning algorithms with high-quality, runs fast and is easy to use. As mentioned in Section 3 on page 3, we use linear regression. This approach requires a set of training data which consists of a double-typed value for a label

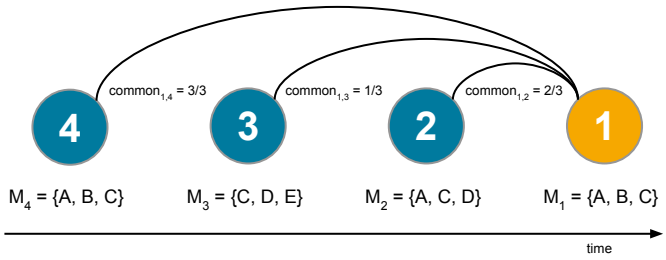

Figure 3: Illustration of the $common_{i,j}$ value calculation from Equation (4) on the preceding page. For instance, $common_{1,3} = \frac{|M_1 \cap M_3|}{|M_1|} = \frac{|\{C\}|}{|\{A,B,C\}|} = \frac{1}{3}$.

and a series of double-typed values for the features as shown in Listing 1.1. This example shows that each data line starts with an annotated value, which in our case ranges from zero to ten, followed by eight features. Each feature has its own set of target words, as shown in Section 3 on page 3. We first calculate the number of occurrences of target words of the given feature appearing in the text. Then, the feature value is decided by dividing the occurrences by the number of entire words.

```
8.0,0.143 0.742 0.424 0.489 0.193 0.495 0.918 0.384
3.0,0.381 0.583 0.934 0.385 0.294 0.583 0.289 0.385
2.0,0.485 0.394 0.729 0.194 0.284 0.193 0.596 0.293
9.0,0.835 0.982 0.193 0.484 0.594 0.293 0.495 0.294
```

Listing 1.1: An example of training data for predicting Formality

Spark linear regression accepts two arguments, the file path of the training data and the number of iterations. In general, the higher the number of iterations, the lower the RMSE. However, there is a certain point where the RMSE practically does not improve anymore. Moreover, as the number of iterations increases, training time increases as well. Therefore, in order to find the optimized number of iterations for our training data set, we ran an experiment of training the data set with up to 7,000 iterations. As shown in Figure 4 on the facing page, the RMSE decreases as the number of iterations increases up until approximately 2,000 iterations. The RMSE did not seem to improve after 2,000 and the training time of approximately one second was acceptable, thus the final number of iterations was set to 2,000.

With regard to feature combination, we ran an experiment of training the data set with all possible combinations of eight features, i.e., 255 combinations without the empty set. As shown in Figure 5 on page 14, the more features, the lower the RMSE. Interestingly, the combination of seven features out of eight

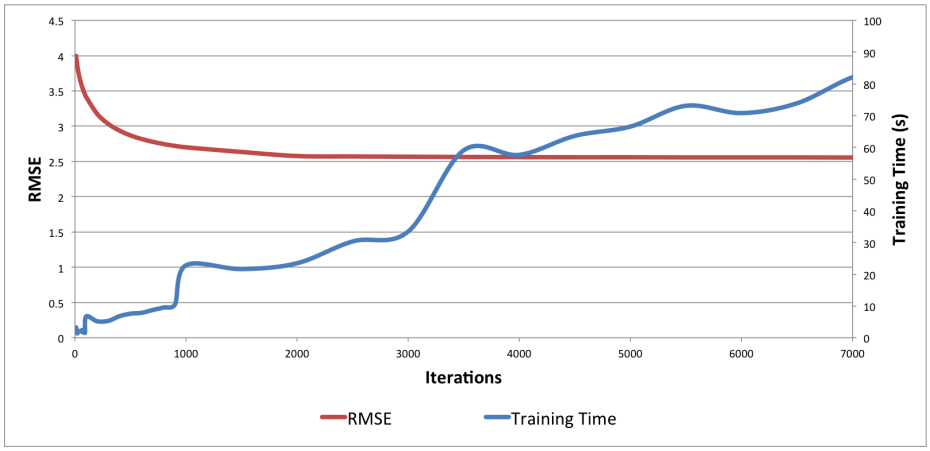

Figure 4: RMSE and training time for the first 7,000 iterations

without Contraction Words gave the lowest RMSE, or 2.57, although there is no significant difference from the RMSE of the combination of eight features.

The usage example of predicting formality is displayed in Listing 1.2. Firstly, a training data set and the number of iterations are passed to `LinearRegression` to start training. Then, a model is created and `LinearRegression` predicts the formality of a given text after converting the text into a series of double-typed feature values as shown above.

```
public static void main(String[] args) throws IOException {

   /** usage example **/

   /** train and save the model **/
   LinearRegression.train("data/Formality_Data.data", 2000);
   LinearRegression.saveModel("data/model");

   /** load the model and predict **/
   String description = "This is a test description";
   LinearRegression.loadModel("data/model");
   double predictedFormality = LinearRegression.predict(description);
   System.out.println(predictedFormality);
}
```

Listing 1.2: Usage example of predicting Formality

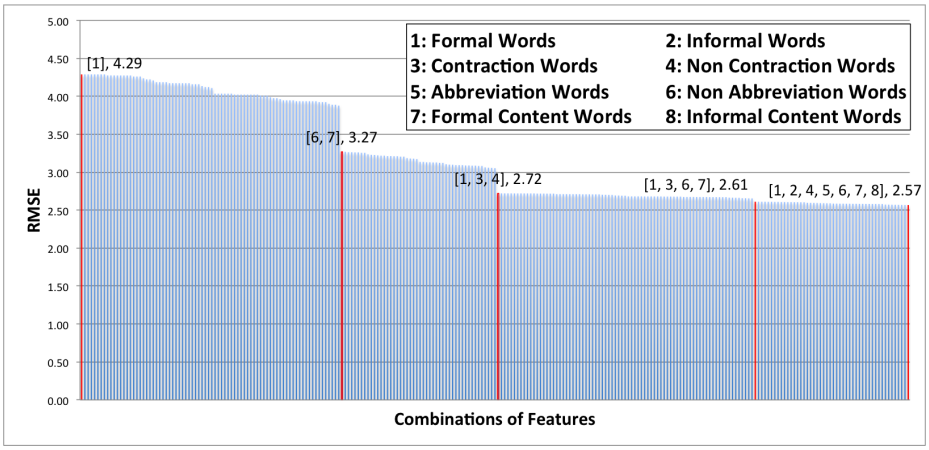

Figure 5: RMSE per unique combination of features

Compactness In order to obtain compactness of an event description, we first created a table in the database containing TF-IDF values of every word in all the event descriptions we crawled. This value is computed with the help of SAP HANA text analysis tools beforehand. Then, we used a global threshold of 0.2 and iterated over all the event descriptions to decide the compactness of a corresponding description by dividing the number of words whose TF-IDF value is above the threshold by the number of words in the description.

4.5 Website Prototype

In order to display all developed filters to users and enable them to use them to narrow down their search, we built a working website prototype. On this website, it is possible to enter a topic and select filters and their desired values, as seen in Figure 6 on the next page. It is a dynamic page that adjusts itself to different screen sizes and built using the Polymer framework[4].

When the user enters a topic or changes the value of a filter-slider, the website communicates this changes immediately to the web server, which is implemented using the SAP HANA XS Engine. This was a requirement for integrating our work into the existing systems of BlogIntelligence[5]. The webserver takes arguments via HTTP GET requests and generates an SQL statement that can be executed on the underlying SAP HANA database. It connects to the database and builds prepared statements in order to prevent SQL injection attacks. After executing the statements the result is sent back to the JavaScript code of the

[4] Polymer. `https://www.polymer-project.org`
[5] BlogIntelligence website. `http://blog-intelligence.de`

website, which modifies the DOM and displays the result. The user interface shows the top-twenty events for the given filtering conditions.

As requests are sent every time the input changes, there is a continuous data flow while the user is entering the wanted topic. In consideration of preventing displaying results for past queries, the server additionally sends back the topic string for which the request was executed. If the string doesn't match the text that is currently entered into the topic field, the response is discarded.

In order to combine multiple features, we calculate the differences between the values set by the user using the website and the ones predicted for all given upcoming events. After normalizing the results, they are summed up and the list of events is ordered in descending order by this rank. This is also the point where weighting could be introduced in order to allow users to define, which filters are more important than others (for more information, see section 6 on page 19).

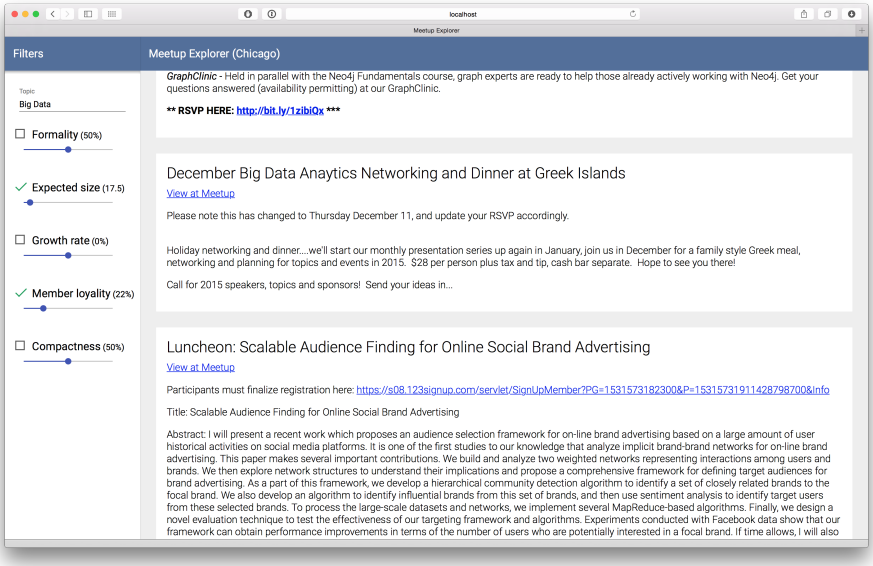

Figure 6: The Meetup Explorer Prototype

5 Results

The following section contains evaluations of the Event Neighborhood on the example of the event size and for the machine learning approach of the predicted event formality. The compactness score and other RSVP features are not evaluated directly, as they are reference implementations of our definitions. We did however conduct a small user study to get an idea of how well our approach and our definitions work. Regarding the execution of a representative user study, see Section 6 on page 19.

5.1 Event Neighborhood

We evaluated the concept and implementation of the Event Neighborhood by predicting event sizes for past events in our data set and comparing it with the actual values. For this, we chose the latest past event from each Meetup Group in Chicago as the prediction event. It includes a total of 1075 events with an average size of 10.24 and a standard deviation in size of 19.27.

With optimal parameters, we achieved a RMSE of 8.57. We consider this error to be sufficiently small in order to get an estimate of what size has to be expected, especially given the very high variation in the evaluation data set.

In order to verify that the parameters were chosen in an optimal way, we did the same evaluation using different parameters and variations of the $w^{(i)}$ equation (see table below).

- Experiment 1: Using Equation (3) and parameters as specified in the implementation section.
- Experiment 2 / 3: Using a lower / higher value for half time T_{half}.
- Experiment 4 / 5: Using only $w_t^{(i)}$ / $w_s^{(i)}$.
- Experiment 6: Using Equation (3) without the boost.
- Experiment 7: Using the average to combine $w_t^{(i)}$ and $w_s^{(i)}$ instead of the product.

As Table 4 on the facing page shows, the parameters we used (Experiment 1) yield the best results.

5.2 Text-based Features

Based on the optimized set of features and the number of iterations we found in Section 4.4 on page 11, we split the 200 annotated event descriptions into 10

Experiment	weight / half time	RMSE
1	$w_s \cdot w_t + boost$ / 2 months	8.56
2	$w_s \cdot w_t + boost$ / 1 month	8.62
3	$w_s \cdot w_t + boost$ / 4 months	8.74
4	w_t / 2 months	8.83
5	w_s / -	10.59
6	$w_s \cdot w_t$ / 2 months	8.57
7	$(w_s + w_t)/2 + boost$ / 2 months	10.07

Table 4: Resulting RMSE under different variations and parameter sets of the Event Neighborhood method.

subsets. Then, we trained using nine subsets, tested the remaining subset and iterated over all different subsets. We display the result of this ten-fold cross validation in Figure 7.

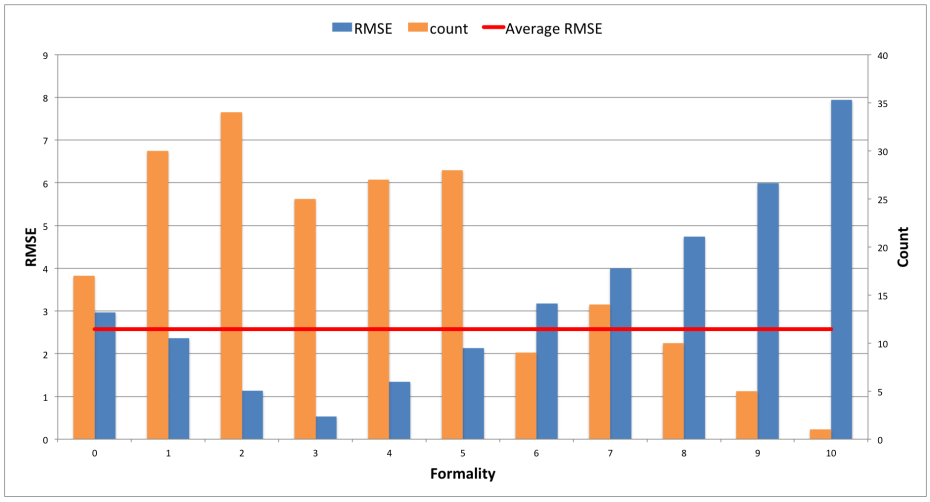

Figure 7: RMSE, the number of annotated events of each formality label from 0 to 10 and average RMSE

This graph shows the RMSE value and the number of annotated events of each formality label from zero to ten. The overall RMSE is approximately 2.57. As shown in the graph, the RMSE values of the most informal descriptions are less than the overall RMSE or similar, whereas those of all the formal descriptions exceed the average. We assume that this was caused by the unbalanced number of annotated formal and informal event descriptions. Most of the annotated descriptions were written in an informal style, and therefore our training set contained only few formal annotated texts. In reality, the majority of Meetup

events are informal. Nevertheless, the result implies that there is a significant difference in using the target words of the features we selected among Meetup event descriptions and the difference is roughly linear, if not very exactly linear.

We then ran an experiment of training only informal annotated descriptions in an attempt to see if having a big enough number of annotated events would give a low RMSE value. The experiment was done in the same way as the ten-fold cross-validation above. As shown in Figure 8, the overall RMSE has improved from 2.57 to 1.65. This indicates that it is essential to have many annotated texts and ideally the same or similar number of formal and informal descriptions.

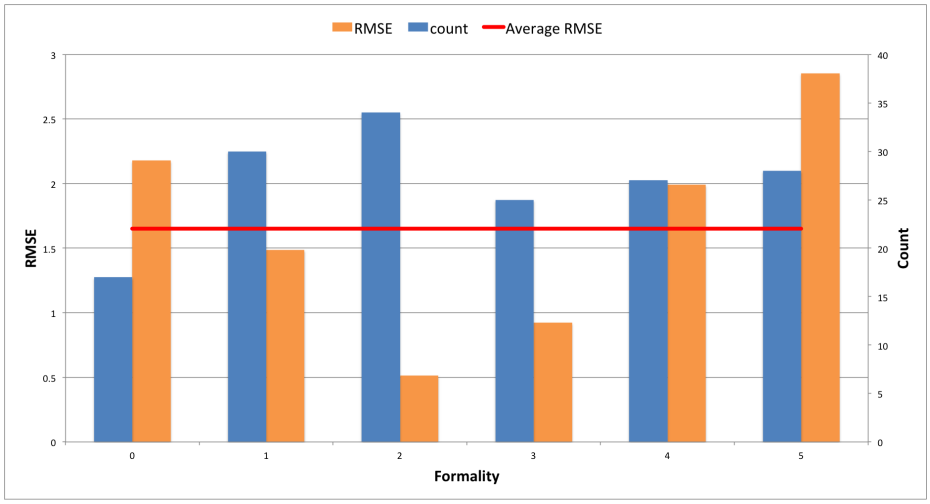

Figure 8: RMSE, the number of annotated events of each formality label from 0 to 5 and average RMSE

5.3 User Study

In the interest of getting an idea how well our approach works, we conducted a small user study. The following section will state the results of this study, which is not to be seen as a representative one. For ideas of how to conclude such a study, see Section 6 on the facing page.

We altered our website prototype so that users would set their preferred filters and the website shows either a list of events ranked using their filters or an unranked list filtered just by the selected topic. It is then possible to switch between those lists using a color-coded toggle, which enables the person conducting the study to see, which list is currently displayed, but not the participant. This

is necessary in order to prevent the users from being biased (as they know that the list using the filters is expected to perform better).

As visible in Figure 9, the usage of filters slightly increased the number of events users would attend. This hints that our approach is valid and able to improve the search on Meetup.

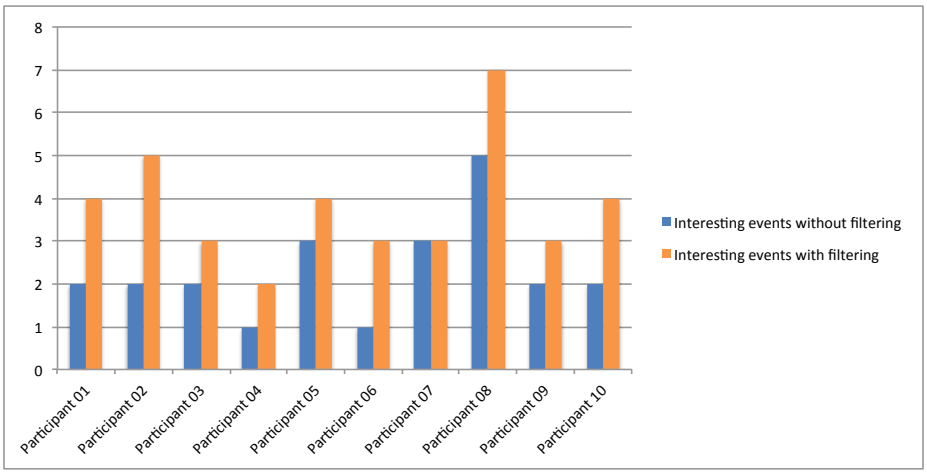

Figure 9: Number of events the participant would attend (out of 10)

6 Conclusion and Future Work

We explored two text-based attributes of events that Meetup currently does not provide: Formality and Compactness. Formality can give users a hint on overall characteristics of a given event, as it defines the atmosphere, what is done and what kind of people come to the event. Using linear regression, we trained on 200 manually annotated Meetup events and can predict the formality with an error range of 2.57 on a scale from 0 to 10. However, because of the unbalanced number of formal and informal annotated descriptions in our training set, this result could be improved. We confirmed this with the subsequent experiment of training and testing informal descriptions (i.e., events with a formality score of 5 or less) only. The overall RMSE was improved from 2.57 to 1.65. This implies that the overall RMSE could be even further improved by having a higher number of annotated descriptions and ideally balancing the number formal and informal descriptions in the training set. On top of this, we confirmed that there was a significant difference among Meetup descriptions in using the target words of each features we selected and the difference was approximately linear, if not

perfectly linear. In addition, compactness of a given event was calculated based on the TF-IDF values of each word in all descriptions of the Meetup events we crawled. We believe that compactness can help users find an event of interest quickly and easily among a number of events of the same or similar topic. The compactness feature could be even further improved by displaying only keywords or highlighting them.

For the RSVP-based features, we introduced the concept of the Event Neighborhood. This method allows us to estimate features for upcoming events that are already known for past events, such as the expected size. It works by computing a time weight and a similarity weight for past events of the corresponding group, which are then used for a weighted average computation. Using this approach, we achieved a RMSE of 8.56 for the size prediction of the event. This result could be improved by investigating more sophisticated means to compute the similarity between events, for which we currently only consider the event titles.

Finally, we combined these features into a ranking of events. This was used to build a prototype application, allowing users to adjust the filtering to their demands. A small user study showed promising results.

Aside of optimizing the existing features, other components could be enhanced in future work as well. Even though the crawler can successfully access and download information for any given city, it is still possible to expand its functionality. One way to improve it to best match the use case of the Meetup Explorer is to enable it to save data for whole countries, continents or even the whole planet. To achieve this, it would be necessary to test the boundaries set by the Meetup API for accessing large regions instead of single cities. A second improvement could be an automated incremental crawling for existing data sets in the database. This would greatly minimize the size of the downloaded block of information, even though it is currently already possible to set timeframes and therefore prevent downloading old data again. Currently, crawling and data mining steps are isolated from each other. However, it would be of interest to only update event predictions and calculations after new data is inserted into the data set and only for this new chunk of data, as it would as well greatly improve the performance.

Additionally, the features developed in this work are only a fraction of the possible features, as the facets of interestingness are diverse and subjective. Therefore, many more than the discussed five features are conceivable and, as described in [1], more event metadata could be taken into consideration. One could even think of using Meetup account information as a basis for new features based on community detection and linked data (see [2] and [3]).

In addition to using more features, it would also be necessary to improve the website prototype and add features such as ranking the importance of selections

or improving the usability of the sliders. Instead of using a linear scale, it would be better to adapt them to the underlying data, so that users get a visual hint what to expect when they change the value. While currently all enabled features are weighted equally, it could also be important to let users of the Meetup Explorer decide, what is more important to them. This could be done during the summation step of the differences between prediction and user selection, as described in Section 4.5. Instead of just summing all normalized values, some could be multiplied by a factor, giving them more weight in the calculation of the order of the events. Finally, it would greatly improve the usability to only show the next event of its kind from a group. To achieve this, it is important to reliably automatically recognize recurring events and group them together. Our Event Neighborhood works in a similar way and could be used as a basis to pool events of a group.

In order to fully evaluate the reference implementations of features like the description compactness or the event trend and the weighting algorithm on the website, an extensive user study would be necessary. This study could take place online with written instructions and a follow-up survey for the participants or offline. An online survey would provide a wider range of people being able to take part in the user study, but is also harder to monitor. In both cases the study could be organized like our small test-study described in Section 5.3 on page 18.

7 Appendix

7.1 Target Words of Each Feature for Predicting Formality

Informal Words a bit, about, absentminded, absorb, abundant, add, advise, again and again, aim, allot, allow, and, anybody, anyplace, apathy, appeal, applause, application, approval, around, arty, ask, ask about, ask for, assert, assign, at first, at once, assume, attribute, authority, avoid, aware, awful, basic, be going to, beach, beg, beginner, belittle, bellyache, better, big, bigger, bitterness, blabbermouth, blame, blessedness, bloody, boozer, boundless, brag, brilliant, bring, bring up, broad-minded, broke, bug, build, bully, busy, but, buy, cancel, carry, catch, catch on, cease, chance, change, chat, cheap, check up, chew, childish, choose, chubby, chump, clean, clear, cleave, climb, clothing, comfort, command, conceit, concern, conduct, conference, confusion, conscious, consider, console, control, convert, copy, cowardly, coworker, crony, crowd, cry, cuddle, curse, cute, dad, daily, deal with, decay, decent, dedicate, delete, delicious, determine, difficult, digest, diligent, dim, disable, disapproval, disaster, discussion, disease, disgusting, do, dog, doubt, doubter, douse, dread, drive, drop, drunk, dry, dub, dumb, dunk, duplicate, earlier, eat, edgy, embarrassement, empty, encourage, end, endless, enough, erase, everlasting, every year, everybody, everyday, evil, excuse, explain, facts, fair, fall, famous, farming, farsightedness, fast, fat, feeling, fib, field, filmy, find, fix, flabbergasted, flashy, fleshy, flimsy, foretell, forgive, fragile, free, fridge, friendly, frisky, funny, gabby, gap, gardening, gather, generous, get, get out, get smaller, gist, give, give out, glasses, gleaming, go, go down with, go through, go up, goal, good, goodwill, goof, gourmet, great, greedy, grill, gripe, grown-up, guess, guy, happiness, hard, harshness, have to, heavy, hefty, help, helper, high, hint, hire, hoard, hobby, home, house, hug, huge, humanity, humorous, hurry, illness, imply, important, improve, in charge, in the end, inbred, incidental, include, indirectness, inhabit, interject, jam, jaundiced, job, jolt, keep, kid, kind of, kindness, lack, lady, lay back, laziness, learn, learner, learning, leave, leftover, lessen, let, letter, letup, lighten, like, lit up, live, lively, loaded, loneliness, lonesome, look up, loud, lucky, lukewarm, mad, mainly, make sure, many, maybe, mean, means, meant, mend, method, middle, modify, mom, moral, move, mushy, nab, need, neighboring, next, nice, nitpicking, nobody, nosy, numb, numbskull, obscure, offer ok, old, older, old-fashioned, on and off, oppose, optimistic, originate, outcome, outstanding, own, pale, parched, participate, pay, peak, phone, photo, piddling, pigeonhole, plan, plane, plucky, portion, power, praise, preacher, premonition, present, pretty much, prize, project, promise, prompt, promptness, pushy, put up, quick, quit, quotation, raunchy, really, reasoning, rebirth, redundant, relentless, remain, remember, replace, request, resemble, resolution, rest, rile, ripen, risk, rob, rot, sanction, say no, scanty, scold, seem, send, send back, sentiment, setup, shameful, sharp, shining, shiv, shock, shorten, show, show up, sickness, sight, skimpy, slander, slapdash, slushy, small, smooch, snatch, sneaky, so, sociable, somebody, sort, sort of, so-so, sot, sourness, speech, speed, spread,

spud, stab, start, stick, stickup, stint, stipend, stop, story, strong, stuff, surround, swamp, swap, sweat, swing, tactful, take on, takeoff, tasty, teach, tell, thing, think, timeless, times, tip, tired, tomb, too, totally, touching, trim, trip, truthful, try, tune, unchangeable, understanding, unhappy, unruly, unselfishness, upset, uptight, use, various, very, want, watch, wealthy, whole, willing, wisecrack, wordy, workable, worry, worse, wrong, yardstick

Formal Words a little, approximately, concerning, abstracted, ingest, copious, affix, counsel, repeatedly, intend, allocate, permit, furthermore, in addition, anyone, anywhere, anomie, petition, acclamation, requisition, commendation, mannered, enquire, request, aver, designate, postulate, initially, immediately, characteristic, jurisdiction, eschew, cognizant, ill, fundamental, will, littoral, plead, novice, minimize, whine, superior, ameliorate, major, large, greater, acerbity, informer, reprehension, beatitude, sanguinary, drunkard, illimitable, vaunt, resplendent, convey, vomit, complaisant, insolvent, exasperate, construct, terrorize, occupied, however, purchase, eradicate, transport, apprehend, understand, desist, opportunity, transform, alter, dialogue, inexpensive, investigate, masticate, immature, select, portly, fool, immaculate, transparent, unmistakable, sunder, ascend, apparel, condole, directive, vanity, solicitude, deportment, assembly, disarray, mindful, deem, solace, govern, transmute, replica, craven, associate, friend, throng, wail, fondle, anathema, pretty, father, diurnal, handle, decompose, ethical, consecrate, expunge, flavorful, ascertain, arduous, imbibe, assiduous, indistinct, incapacitate, aspersion, calamity, colloquy, malady, repugnant, perform, hound, dubiety, skeptic, submerge, foreboding, impel, decline, intoxicated, desiccated, obtuse, immerse, facsimile, previous, dine, restless, discomposure, vacant, gladden, finish, unending, sufficient, efface, annually, everyone, quotidian, nefarious, remit, elucidate, data, disinterested, decrease, renowned, agriculture, prescience, swift, corpulent, emotion, lie, specialization, diaphanous, locate, discover, rectify, astounded, gaudy, mesomorphic, unsubstantial, augur, pardon, frangible, release, exempt, refrigerator, amiable, sportive, comic, talkative, aperture, tillage, convene, magnanimous, obtain, acquire, leave, crux, donate, contribute, distribute, spectacles, luminous, depart, contract, increase, objective, beneficial, generosity, mistake, gastronome, reputable, avaricious, interrogate, complain, adult, believe, man, felicity, laborious, acrimony, must, burdensome, ponderous, assist, assistant, elevated, insinuation, employ, preserve, avocation, residence, dwelling, caress, enormous, humankind, jocular, expedite, infirmity, connote, consequential, responsible, finally, innate, adventitious, comprise, circumlocution, reside, interpose, cynical, occupation, impact, retain, child, somewhat, benevolence, deficiency, woman, relax, indolence, detect, scholar, pedantry, residue, allay, authorize, correspondence, respite, alleviate, such as, illuminate, energetic, prosperous, disaffection, lonely, research, clamorous, fortunate, tepid, insane, principally, ensure, numerous, possibly, perhaps, in other words, instrumentality, denote, repair, procedure, midst, mother, virtuous, transfer, maudlin, arrest, require, contiguous, subsequently, agreeable, pedantic no one, prying,

anesthetized, dullard, arcane, proffer, satisfactory, aged, senior, archaic, intermittently, gainsay, sanguine, emanate, denouement, paramount, possess, wan, dehydrated, partake, compensation, summit, telephone, photograph, negligible, categorize, scheme, aeroplane, valiant, passage, sway, laud, minister, presentiment, gift, essentially, award, undertaking, assure, motivate, alacrity, ambitious, manage, rapid, resign, quote, risque, quite, ratiocination, renaissance, pleonastic, inexorable, abide, recall, supersede, appeal, parallel, fortitude, repose, annoy, mature, jeopardy, extort, spoil, approbation, reject, exiguous, chide, appear, transmit, return, affect, dishonorable, acute, effulgent, knife, reduce, demonstrate, evince, ailment, vision, meager, cursory, mawkish, diminutive, kiss, seize, underhand, therefore, consequently, social, someone, type, rather, mediocre, alcoholic, asperity, oration, velocity, propagate, potato, penetrate, begin, adhere, larceny, assignment, emolument, cease, halt, narrative, stalwart, materials, things, items, circumscribe, deluge, barter, perspiration, oscillate, diplomatic, mimicry, palatable, educate, inform, recount, matter, issue, cogitate, eternal, multiply, gratuity, fatigued, sepulcher, also, completely, poignant, ornament, voyage, veracious, endeavour, song, immutable, comprehension, dissatisfied, intractable, altruism, disturb, nervous, consume, sundry, highly, desire, wish, observe, affluent, complete, entire, compliant, joke, verbose, feasible, apprehension, inferior, incorrect, criterion

Contraction Words ain't, aren't, can't, couldn't, didn't, doesn't, don't, hadn't, hasn't, haven't, he'd, he'll, he's, i'd, i'll, i'm, i've, isn't, let's, mightn't, mustn't, shan't, she'd, she'll, she's, shouldn't, that's, there's, they'd, they'll, they're, they've, we'd, we're, we've, weren't, what'u, what'll, what're, what's, what've, where's, who's, who'll, who're, who've, won't, wouldn't, would've, you'd, you'll, you're, you've, that'll, it's, we'll, it'd

Non Contraction Words am not, are not, cannot, could not, did not, does not, do not, had not, has not, have not, he had, he would, he shall, he will, he is, he has, i would, i had, i shall, i will, i am, i have, is not, let us, might not, must not, shall not, she had, she would, she will, she shall, she is, she has, should not, that is, that has, there is, there has, they would, they had, they will, they shall, they are, they have, we would, we had, we are, we have, were not, what shall, what will, what are, what is, what has, what have, where has, where is, who had, who would, who will, who shall, who are, who has, who is, who have, will not, would not, would have, you had, you would, you will, you shall, you are, you have, that will, it is, we will, we shall, it would, it had

Abbreviation Words e.g., i.e., etc., ok, jan., feb., mar., apr., may., jun., jul., aug., sep., oct., nov., dec., sat., sun., mon., tue., wed., thu., fri., plane, lab., asap, usa, tons, undergrad., grad., hr, prof., ai, ur, &, abt, mt, mt., shd, shd., wch,

wd, wd., wh, wh., yr, yr., yrs, yrs., 2d, 2n, 3d, acct, acct., advt, aftern, aftern., aftn, aftn., am, a m, anti-cathc., arc., ass., b.m., bkfst, bkfst., brkfst, c.b.e., cent, chap, chap., chaps, depart., edn, eng., h.oflds, hist., hrs., in., ld., ldy, ldy., lovg, ly, ly., max., mem., min., min:, phys., prob., recvd, secty., temp., trans., univ., vol., xmas, xtian, yestdy, acad., adm., bib.

Non Abbreviation Words for example, that is, and so on, okay, january, february, march, april, may, june, july, august, september, october, november, december, saturday, sunday, monday, tuesday, wednesday, thursday, friday, airplane, laboratory, as soon as possible, united states of america, tonnes, undergraduate, graduate, human resources, professor, artificial intelligence, your, and, about, might, should, which, would, yours, second, third, account, advertisement, afternoon, ante meridiem, anti-catholic, archaeological, association, british museum, breakfast, commander of the order, century, chapter, chapters, department, edition, english, house of lords, history, hours, inches, lord, lady, loving, maximum, memoires, minutes, physical, probably, received, secretary, temperature, to equal, transactions, university, volume, christmas, christian, yesterday, years, academic, administration, bible

Informal Content Words karaoke, eating, fun, board game, game, mcdonalds, hangout, no location, player, casual, picnic, laugh, enjoy, movie, run, runner, jazz, free hug, single, dance, skate park, dancer, salsa, comedy, chat, skydiving, hanging out, beach, party, atmosphere, hey, flirt, nightlife, romance, friendship, friend, !!, let's, tons of, groove, club, hikes, share in the moment, all you can eat, entertainer, a dime, thanx, :-), lover, valentine's day, love, dating, nuts, ?!, dreamer, knitting, comfy, awesome

Formal Content Words socialising, workshop, registration, payment, fee, charge, business, donation, donate, admission, museum, coach, schedule, street action, discuss, ticket, dress code, entrepreneur, badge, session, course, insight, instruction, program, technique, seminar, scripture, dress, intelligent, attire, constructive feedback, business card, presentation, class, teacher, technically, approval, political, lecture, demonstration, leader, professional, development, services are led, bible, guideline, certified, program

References

1. A. Q. de Macedo and L. B. Marinho. Event recommendation in event-based social networks.
2. X.-L. Li, A. Tan, S. Y. Philip, and S.-K. Ng. Ecode: event-based community detection from social networks. In *Database Systems for Advanced Applications*, pages 22–37. Springer, 2011.
3. X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. Event-based social networks: linking the online and offline social worlds. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1032–1040. ACM, 2012.
4. Meetup. Meetup about page. `http://www.meetup.com/about/`, Feb. 2015. [Accessed 19.02.2015].
5. F. A. Sheikha and D. Inkpen. Learning to classify documents according to formal and informal style, Mar. 2012.
6. B. Xu, A. Chin, and D. Cosley. On how event size and interactivity affect social networks. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 865–870. ACM, 2013.