Querying Sentiment Development Over Time

Troels Andreasen, Henning Christiansen, and Christian Theil Have

Research Group PLIS: Programming, Logic and Intelligent Systems
Dept. of Communication, Business and Information Technologies
Roskilde University, Denmark
Email: {troels,henning,cth}@ruc.dk

Abstract. A new language is introduced for describing hypotheses about fluctuations of measurable properties in streams of timestamped data, and as prime example, we consider trends of emotions in the constantly flowing stream of Twitter messages. The language, called EmoEpisodes, has a precise semantics that measures how well a hypothesis characterizes a given time interval; the semantics is parameterized so it can be adjusted to different views of the data. EmoEpisodes is extended to a query language with variables standing for unknown topics and emotions, and the query-answering mechanism will return instantiations for topics and emotions as well as time intervals that provide the largest deflections in this measurement. Experiments are performed on a selection of Twitter data to demonstrates the usefulness of the approach.

1 Introduction

Social media are becoming more and more popular in humans' daily lives and networking on social media is an increasingly important social activity. With the evolving social networks and increased media activity the already massive amount of media data is rapidly growing and so is the potential value of these data as sources for analysis of structure, relationships and structural trends. There is also a huge potential for the derivation of valuable and reliable indications of trends in development of opinions and sentiments from the immense messaging on social media, but there is still a lack of systematic approaches for these purposes. However, messaging is target for various new and promising approaches to data mining and sentiment analysis and this area is undergoing a rapid development. The present paper emphasizes a new direction for this kind of analysis where opinions on topics can be investigated, not only for appearance but also for development over time.

We introduce a language, EmoEpisodes, for hypothesis testing and querying. Hypotheses about sentiments and emotions and their development over time can be formulated in the language. The language allows for querying on emotions about specified topics, for instance "fear of asteroids" and the degree to which an emotion is expressed about a given topic. An emotion about a given topic can be queried for a specific period, or the period can be left open to investigate peeks. In addition, consecutive sequences of periods can be enclosed in queries

to investigate development of emotions over time, for instance to query peeks in shifts from negative to positive attitudes about a given topic. A query can focus on a specific topic, such as "how did the fear of astroids develop during Christmas" or refer to correlated topics, such as "did the School shooting influence the opinion against Obama's policy". We present a general and flexible language for formulating hypothesis – and thus queries – about detailed patterns of the evaluation of emotions over time. This is in contrast to earlier efforts which have mostly been concerned with measuring specific trends in social media data.

The validity of a given hypothesis in some time interval is measured by a satisfaction degree which is a number in the unit interval, and the grading is applied for ranking of best matches for a given hypothesis. Specifically, we consider here streams of timestamped messages in social media and hypotheses about fluctuations of emotions related to given topics.

Emotions are characterized and measured for some fixed **time granule** adapted to the application at hand. For applications in literature, one year or one decade may be chosen, for historical studies perhaps centuries. For the application chosen here – evolving emotions in social media – we have chosen one day as the granule. This choice abstracts away variations caused by the rotation of the earth and the uneven distribution of users around the globe, but allows to characterize relatively fast changes. In our current implementation, we arbitrarily chose 24h periods starting at 00:00:00 GMT+1, i.e., CET without Summer Time. Within each time granule we characterize topics by **level of emotion** where emotion is a classification of expression and level can be chosen from a finite set of fuzzy linguistic expressions.

Different applications may require different ways of combining satisfaction degrees for the constituents of complex hypotheses and as indicated we may combine topics and evaluate emotions as they appear simultaneously as well as consecutively over time. We discuss aspects of **aggregation** of constituent emotions and introduce a parameterizable function to adapt the different kinds of aggregations needed as well as for user preference.

The present paper is structured as follows. In section 2 we introduce the language EmoEpisodes and in section 3 we describe the chosen semantics for our application at hand – mining trends in social media. In section 4 we describe experiments and evaluation based on an implementation of the language and on an application on Twitter data from about 1.5 month from late December to early February 2013. In section 5 we discuss related work and finally in section 6 we conclude.

2 A Proposal for an Emotional Episode Language, EmoEpisodes

In the following, a *time point* refers to a specific time granule having a fixed position along a time line, and a *time interval* is a contiguous set of time points. The symbol $\mathcal{TI}$ refers to all such time intervals; below we use letter d to refer to time intervals (as t will be used for topics; below).

2.1 The Basic Emotional Episode Language

An arbitrary set of *topics* $\mathcal{TI}$ is assumed, e.g., $\mathcal{T} = \{\mathsf{Xmas}, \mathsf{love}, \mathsf{beer}, \mathsf{asteroids}, \ldots\}$ and set of *emotions*, e.g, $\mathcal{E} = \mathsf{fear}, \mathsf{happiness}, \mathsf{anger}, \mathsf{sadness}, \ldots\}$. As mentioned above, the *level* of an emotion is described by a finite set of symbols $\mathcal{L}$, ordered by magnitude; we use for our main example $\mathsf{high} > \mathsf{medium} > \mathsf{low}$, but more or less (at least two) steps may be used.

An atomic emotion *statement* for a given topic associates an emotion and a level to that topic. Example:

asteroids: fear(high)

The data semantics is given by a satisfaction degree function $SD : \mathcal{AS} \times \mathcal{TI} \to [0; 1]$ where $\mathcal{AS}$ is the set of such atomic statements, i.e., a measurement of how well a given statement characterizes a given time interval.[1]

A *compound* statement consists of a collection of atomic ones written with curly brackets; when more than one atomic statements concerns the same topic, we may group these also by curly brackets. Examples:

{asteroids: fear(high), doomsday: fear(high)}
asteroids: {fear(medium), excitement(high)}

The second example is a shorthand for a compound consisting of two atomic statements about asteroids.

A compound statement is understood as a sort of conjunction, i.e., all contained atomic statements influence the satisfaction of the compound; $\mathcal{S}$ will denote the set of all statements, with $\mathcal{AS} \subset \mathcal{S}$. The satisfaction degree function is extended to go all statements, $SD : \mathcal{S} \times \mathcal{TI} \to [0; 1]$, as follows where $\bigotimes^{com}$ is an aggregation function $[0; 1]^* \to [0; 1]$.

$$SD(\{\phi_1, \ldots, \phi_n\}, d) = \bigotimes_{i=1..n}^{com} SD(\phi_i, d)$$

A *scene* is a statement with an associated *time constraint.* Examples:

asteroids:fear(high)[5 days]
asteroids:fear(high)[> 5 days]
asteroids:fear(high)[2 days, from 2013-02-15]
asteroids:fear(high)[2013-02-15]
asteroids: {fear(medium), excitement(high)}[> 5 days]

The detailed language for time constraints is not specified further; we assume a natural definition of whether a given time interval is matched by a constraint. For example, [5 days] matches any interval of exactly 5 days. A *time assignment* for a scene is any interval of days that satisfies its time constraint, so, e.g.,

[1] As it appears, *SD* measured over an interval is not a mere aggregation of *SD* for each granule in the interval; this provides a freedom to let *SD* measure, say, relative frequencies over the entire interval.

the interval consisting of the days from 2013-02-14 to 2013-02-18 can be an assignment for asteroids:fear(high)[C], where, say, C is '5 days', '$>$ 2 days' or '5 days, after 2012-31-12'.

Finally, an *episode* is a sequence of consecutive scenes, indicated by semicolon. Example:

asteroids:fear(medium)[5 days] ;
{meteorites:fear(high), doomsday:fear(high)}[2 days, after 2013-02-15]

A time assignment for an episode is a sequence of assignments of consecutive intervals for each of its scenes; it is also referred to as a *match*. An episode is called *inconsistent* if no possible match exists, otherwise it is *consistent*. In order to allow 'holes' in episodes, we introduce the empty statement $\{\}$ having $SD(\{\}, -) = e$ where e is a neutral element (not specified further).[2]

The satisfaction degree of an episode with respect to a time assignment is an aggregation $\bigotimes^{eps}$ of the satisfaction degrees for the individual scenes in their respective, assigned time interval. A *best* match in a given context is a match with the highest satisfaction degree.

Notice as a consequence of our definitions that the satisfaction degree of, say asteroids:fear(medium)[5 days], in a given time interval is independent of the emotions for asteroids in the days before or after the chosen period.

2.2 EMOEPISODES as a Query Language

The semantics of EmoEpisodes can produce a measure of the satisfaction degree of a given episode for a specific time assignment, and this is the basis on which we can form a query language. Here we discuss how EmoEpisodes can be used as a query language; actual experiments performed on Twitter messages are shown in section 4, below.

It gives good sense to query with a given episode for *the* best match within a larger time interval, i.e., for the best time assignment together with its satisfaction degree. Such a query mechanism can be extended with an *a priori* defined threshold and only report a match better than this threshold.

While for many other information retrieval tasks, it is relevant to ask for a sorted list of the k best matches, this is more dubious for episode matching due to overlapping matches. To see this, consider the following episode

asteroids: fear(medium)[$>$ 10 days] ; asteroids: fear(high)[$>$ 10 days]

and a data set in which the frequency of observations of asteroids: fear grows slowly over a period of 1000 days. Here we may expect one optimal match about two thirds into the data, and it will be surrounded by a cloud of near optimal matches. It will be even worse in case asteroids: fear varies in very long waves, having peaks of different magnitudes. Here the sorted list approach may likely

[2] A proper formalization of the empty statement would require an extension of the domain for satisfaction degree to $[0; 1] \cup \{e\}$ and a specification of how the different aggregation operators treats e. These details are not interesting and left out.

report only the highest peak and the cloud around it, and having the second and third highest peaks outside the k best matches shown to the user, which we do not consider to be desirable.

If graphical output is an option, the best way to present the result of a query for an episode E may be as a curve showing, for each time point t, the satisfaction degree for an occurrence of E starting at t. When textual or symbolic output is expected, it may be suggested to report islands (defined as contiguous unions of matches better that a threshold) together with the best match in each island (the peaks), sorted according to the latter.

We can extend the EmoEpisodes language for queries involving variables that stands for unknown parts of an episode. The expected answers to a query with variables is a list of alternative instantiations of the variables, sorted according to the satisfaction degrees for their respective best match in the data set. In an interactive query system, a mouse click may, for each such instantiation, open a window with a satisfaction degree curve of a list of islands and peaks as explained above.

In order to obtain efficient query-answering algorithms, we allow only variables in positions where a topic, an emotion or a degree is expected (and not for, say, entire statements or time constraints).

EmoEpisodes as a query language can also be used for giving alert when certain patterns are observed. A journalist may, for example, want a report whenever the attitude towards the topic president changes. He can do this by setting up a tenant that sends a report whenever one of the following two queries have a match in the current data stream.

president: X(high)[3 days] ; president: X(low)[1 day]
president: X(low)[3 days] ; president: X(high)[1 day]

The three day interval indicates a certain stability, but the journalist allows only one day to see a possible shift, so he may be the first to write about it if it happens to be interesting.

3 EmoEpisodes Semantics for Mining Trends on Twitter

In this section we show two different semantics for EmoEpisodes by different choice of data semantics, both considered relevant for the sample applications, mining trends in Twitter data. We suggest also relevant choices for aggregation operators, that fit with both data semantics.

The data semantics for atomic scenes $(t : e(\ell), d)$, with t being a topic, e an emotion, ℓ a level, and d a time interval, is defined in terms of the number of messages tagged with t and classified as e in d. For scenes with compound statements, $t_1 : e_1(\ell_1), t_2 : e_2(\ell_2), \ldots$ the aggregation is based on counting messages tagged with all of $t_1, t_2, \ldots$.

3.1 Atomic Statement Semantics

Let T be the set of topics and D the set of duration specifications (time intervals). Apart from marking some messages with detected emotions, currently among $E = \{anger,\ disgust,\ fear,\ joy,\ sadness,\ surprise\}$, the given prototype also provides a sentiment classification for each message from $S = \{negative, neutral, positive\}$. We want to cover both by the language, but since each of these is a disjoint classification of messages (where E is partial and S is complete), we generalize to calculate satisfaction degrees for a specific classification among a given set $\mathcal{C}$ of classifications, where $\mathcal{C} = \{E, S\}$. For a classification $C \in \mathcal{C}$ and $d \in D$, we define $\delta_C(d)$ as the set of all messages during d that are classified by C, while, for topic $t \in T$ and class $c \in C$, $\delta_C(t, d)$ and $\delta_C(t : c, d)$ denotes the set of all messages during d on topic t and the set of all messages during d on topic t classified as c respectively.

Based on cardinalities of these sets, we define the relative satisfaction R_C of a statement $\phi = (t : c)$ during d for class (emotion or sentiment) $c \in C$ (with $C \in \mathcal{C} = \{E, S\}$) by

$$R_C(\phi, d) = R_C(t : c, d) = \frac{|\delta_C(t : c, d)|}{|\delta_C(t, d)|}, \qquad c \in C$$

and measure the satisfaction degree of sentences by way of compliance with simple fuzzy linguistic terms $low, medium$, and $high$ over relative satisfaction. Membership functions for these terms are defined individually for each classification such that the membership function for $medium$ is symmetric around $1/n$ where $n = |C|$. Figure 1(a) shows definitions of relative satisfaction level terms $low, medium, high$ for classifications E and figure 1(b) for classifier S (notice that $|E| = 6$ and $|S| = 3$). For the relative satisfaction level $level \in \{low, medium, high\}$ the satisfaction degree of a statement $\phi = (t : e)$ during d is defined by:

$$SD(\phi, d) = SD(t : c[level], d) = \mu_{level}(R_C(t : c, d)), \qquad c \in C$$

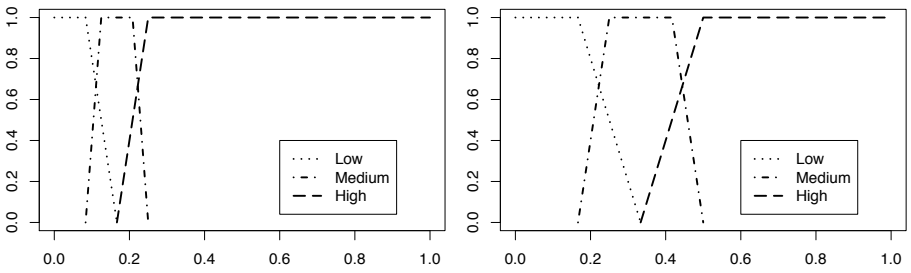


Fig. 1. Membership functions μ_{level} for fuzzy linguistic terms over relative satisfaction with $level \in \{low, medium, high\}$. Shown in (a) for classifier E and in (b) for classifier S (6 and 3 classes respectively).

3.2 Elitist and Populist Data Semantics

Queries are evaluated over time intervals and satisfaction for an individual scene enclosed in a query is based on the fraction of tweets, in the given time interval, satisfying the scene. We consider two data semantics that differ in the way this fraction is derived: elitist and populist data semantics.

When applying **elitist semantics** $SD(t : c[level], d)$ is measured relative those tweets that refer to topic t during d. This corresponds to applying the relative satisfaction as introduced above, that is:

$$R(\phi, d) = R(t : c, d) = \frac{|\delta(t : c, d)|}{|\delta(t, d)|}$$

An alternative is to apply **populist semantics** where $SD(t : c[level], d)$ rather is measured relative to all tweets during d, thus based on a relative satisfaction, which is:

$$R(\phi, d) = R(t : c, d) = \frac{|\delta(t : c, d)|}{|\delta(d)|}$$

As indicated elsewhere we have chosen the elitist for our preliminary experiments. However the choice of semantics could be left to the user as a preference parameter.

3.3 Compound Statement and Episode Semantics

To complete the semantics of the language we must specify aggregation principles for compound statements (aggregating multiple simultaneous statements for a given time interval) as well as for episodes (aggregating statements over continuous time intervals). Intuitively both aggregations should reflect a conjunction of the statements, but we need to take into account the graded satisfaction of statements so an obvious choice is to go for a single, but parametrizable, graded averaging aggregation function. We adopt the Order Weighted Averaging (OWA) function [13] and introduce a simplification of the parameterization of this – scaling with a single parameter a class of averaging functions with min and max as extremes. OWA aggregates n values $a_1, \cdots, a_n$ by means of an ordering vector $W = [w_1, ..., w_n]$, applying w_1 to the highest value among $a_1, \cdots, a_n$, w_2 to next highest value, etc. Thus OWA is defined by:

$$F_W(a_1, \cdots, a_n) = \sum_{i=1}^{n} w_i b_i; \qquad w_i \in [0, 1]\,; \qquad \sum_{i=1}^{n} w_i = 1$$

where b_i is the i'th largest among $a_1, \cdots, a_n$ and $b_1, \cdots, b_n$ is thus the descending ordering of the values $a_1, \cdots, a_n$. By modifying W we can obtain different aggregations, for instance, $W = [1, 0, 0, \cdots]$ corresponds to the maximum, $W = [1/n, 1/n, \cdots]$ becomes the average, and $W = [0, 0, \cdots, 1]$ the minimum. Order weights can, independent of n, be modeled by an increasing function $K : [0, 1] \to [0, 1]$ such that:

$$w_i = K\left(\frac{i}{n}\right) - K\left(\frac{i-1}{n}\right)$$

Assuming $K(0) = 0$ and $K(1) = 1$, aggregations such as *max* can be modeled by $K(x) = 1$ for $x > 0$, *min* by $K(x) = 0$ for $x < 1$, *average* by $K(x) = x$ and for instance a restrictive aggregation (closer to *max* that *min*) by $K(x) = x^3$. Using this principle of prescribing weights by increasing functions, order weight specification can be further simplified using a single parameter $\beta \in [0, 1]$, as in:

$$K(x) = G_\beta(x) = \begin{cases} 0 & \text{for } \beta = 0 \\ x^{\frac{1}{\beta}-1} & \text{for } \beta > 0 \end{cases}$$

where values $0, 1$ and 0.5 for β corresponds to *min*, *max* and *average* respectively, while values closer to zero corresponds to more restrictive aggregations and values closer to 1, to less.

OWA aggregation conveniently adapts intuitive definitions of "linguistic quantifiers". Using the single-parameter approach above we can define $EXISTS$ by $G_\beta(x)$ with $\beta = 1$ and $FOR\ ALL$ with $\beta = 0$, while quantifiers such as $A\ FEW$, $SOME$, $MOST$, and $ALLMOST\ ALL$ can be modeled by β-values such as 0.8, 0.5, 0.2 and 0.05 respectively.

While compound statement as well as episode aggregation intuitively are conjunctive, we consider the former as indicating more restrictive quantification than the latter. We chose, for application in the prototype the $MOST$ aggregation, setting the corresponding β-value to 0.2.

4 Experiments and Evaluation

We illustrate applications of our query language to the social network Twitter using our prototype implementation of the query language.

The implementation is realized in Prolog and R (bridged through the R..eal Prolog library [2]). The system has a grounding component which for each variable and flexible duration generates all possible *query variants* in which query variables are replaced by possible values and durations are given as a fixed number of dates. We refer to a query variant and its associated score as a *match*. A query result set contains all matches to a query sorted by their score. Query evaluation is a recursive procedure which uses fuzzy membership functions for atomic statements and Order Weighted Averaging for compound statements as described in section 3. A search for a match of a specific episode runs in time linear in the size of the data. The number of variables in a query may have drastic influence on the number of query variants and hence on time complexity, but may be controlled by heuristic score cut-offs. Furthermore, both the independent scoring of query variants and the recursive matching procedure is well-suited for parallelization using a map-reduce strategy [5].

To test our implementation we have gathered almost 500GB of data from Twitter[3] over a period of about one month (From December 23, 2012 to February

[3] A tweet including meta-data takes about 1 kilobyte.

7, 2013). The data were collected by monitoring the sample firehose [12] – a service provided by Twitter which gives access to a random subset of Twitter messages (tweets) as they are produced in realtime. Twitter also provide a search interface, but unlike the firehose, the returned data are not purely random but are filtered based on criteria known only by Twitter. Each tweet is provided in a JSON format, which in addition to the tweet text, provides metadata such as the language of the of tweet. We consider only tweets for which the language is indicated to be English.

For each tweet collected, we identify topics and perform sentiment analysis using the `sentiment` package for R [6]. We utilize Twitters hashtags as topic classifications, which has the advantage that we do not have to decide on a prospective set of topics in advance. Hashtags are words in a tweet that are prefixed with the # character and serves as a way for users to volunteer a sort of topic classifications of their tweets. For our purposes, it is not sensible to consider all hashtags since some are so infrequent that it is not possible to measure a trend within our time limit. We consider only hashtags which occurs in at least 500 of the collected tweets (corresponding to an average of at least ten tweets per day). In our dataset, only 3494 out of 2045318 unique hashtags occur in at least 500 messages.

4.1 Example 1: Stock Prices and Surprising Events

As a case to illustrate the query language, we are interested in finding sudden events involving a company which either angers or pleases the public. We use the company Apple (tag #apple) in this example. Reactions to sudden events may involve an expression of surprise as well as an indication of attitude towards the event. To characterize attitude we consider joy and anger rather than positive or negative sentiment because these are more extreme emotions and more likely to pertain to surprising events. We expect that sudden events of these kinds can have an effect on the stock price of a company.

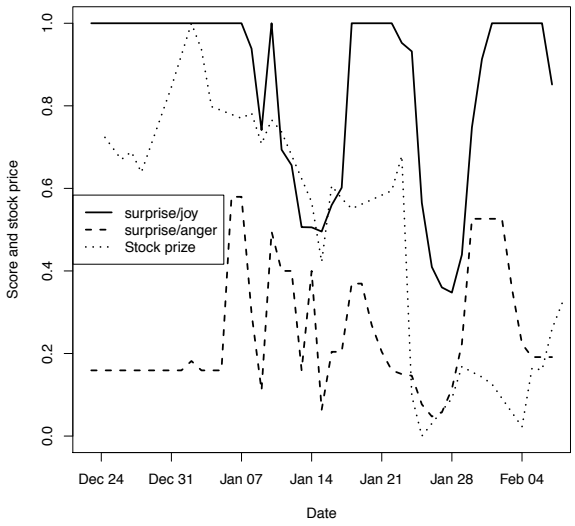
We detect such events with EmoEpisodes using the following two queries:

Q_1: apple:{surprise(high), joy(high)}[$\geq$ 1 day]
Q_2: apple:{surprise(high), anger(high)}[$\geq$ 1 day]

The first query matches scenes of arbitrary duration for which surprise and joy for #apple is high and the second matches scenes for which surprise and anger is high. For each of these two queries we collect all matches.

The matches to a query are likely to overlap. For instance, a match with a period of three days may be overlapped by (similar) matches with periods of more than or fewer than three days, but covering some of the same three day period. Figure 2 shows curves of the maximal scores of any match to Q_1 and Q_2 together with a normalized stock prize. Even if the stock price curve is not a perfect match to either of the query curves, it is easy to spot correlations.

Fig. 2. The solid curve displays the maximal score for query Q_1 (surprise and joy) and the dashed curve displays the maximal score for query Q_2 (surprise and anger). The dotted curve displays Apples stock price normalized to the range $[0, 1]$.



4.2 Example 2: Changes of Attitude

The previous example does not make use of variables which are useful to reason about unspecified tags, sentiments and emotions. We consider here matching topics for which a change in attitude has occurred. In particular, we consider a change from a positive attitude to a negative attitude.

Q_3: X:positive(high)[10 days] ; X:negative(high)[10 days]

Four tags achieve perfect score for this query. The most interesting of these is the tag #14jan ($\approx$ Jan 14–Feb 2) which refer to a march in Pakistan led by Canadian-Pakistanian Tahir-ul-Qadri in protest of the government. While the protest march led to an agreement of reforms signed by the government, the negative sentiments period (Jan 24–Feb 2) contains many critical tweets about the dual nationality of Tahir-ul-Qadri and about his agenda.

The rest are non-topical tags: #emblemfollowspree ($\approx$ Jan 10–Jan 29), #twitterhastaughtme ($\approx$ 29 Dec–17 Jan), #dontbemadatmebecause ($\approx$ 29 Dec–18 Jan). These are Twitter-specific tags for which there is a high volume of these tweets for a relatively short duration (trending tags). Tweets using one of these tags seem to use a sarcastic tone, which explain negative sentiment classifications. In the first few days of these trends, they receive a lot of positive attention. This is apparent from tweets using the tag to comment on the tag/trend to indicate, e.g., that it is funny. This explains a great deal of positive sentiment classifications.

5 Related Work

Our terminology is inspired by the seminal work of [7] who suggested a way to define episodes in sequences of discrete events (from a finite alphabet of such) and gave algorithms to search for a sort of association rules among such episodes. Before that, [1] described algorithms for mining frequent, sequential patterns in a transaction database. See a recent survey [8] on later work inspired by [1, 7].

Our work differs from the referenced work in that we present 1) a logical language EmoEpisodes for specifying scenes and episodes sequences, and 2) these episodes refer to measurements over large sets of timestamped objects (exemplified by tweet messages) with associated multi-dimensional features (exemplified by hash tags and emotions), rather that a finite alphabet. Furthermore, our episodes can be parameterized in arbitrary ways, we can include very general time constraints, and we can search for (ranked lists of) instances of the parameters that provides the best match. Our declarative episode specification language has a well-defined, graduated truth semantics, that is parameterized in a way that allows different interpretations of the data.

An SQL-based query language for specifying *search* for sequential patterns of simple events is introduced in [11]. The queries inherit the generality of SQL, but examples in the paper gives an impression that formulation of queries may be a non-trivial task. There is no account for a priority between different answers, although this may possibly be encoded with aggregate functions and SORT BY in SQL. A generalization of the query language of [11] to handle episodes specified in EmoEpisodes does not seem feasible, as all details of the underlying data semantics and aggregations would need to be unfolded in an SQL style within each query.

Sentiment analysis of Twitter messages over time has previously been demonstrate to correlate with public opinion measured by Gallup polls in [9] which, however, only measures positive/negative sentiment. Another study correlates tweet sentiments and emotions to socio-economic phenomena [4]. Correlation of the sentiment of Twitter messages to stock prices is studied in [3], though not in relation to surprising events. In contrast to the specific nature of these studies, EmoEpisodes provides the flexibility to adapt to a variety of use-cases.

6 Conclusions

We have presented a language for querying and mining the development of sentiments and emotions over time and illustrated its use with Twitter data. The language gives the ability to answer questions on how and when opinions change. It is also useful as a data mining tool to discover sequential patterns of sentiments and emotions associated to topics which may not be known in advance. Being able to answer queries as those supported by our language can have important implications for, e.g., social, socio-economical and political science as well as for market analytics. As a monitoring tool it can discover unexpected events and be used to alert media and decision makers to events worthy of attention. To our knowledge, a query language with these capabilities has not been seen before.

Our implementation can be improved in a number of ways, particularly with regard to the method of sentiment analysis and topic classification. Using hashtags as topic classifications is convenient but it is possibly ambiguous and may be insufficient if tags are non-topical or missing. The use of NLP techniques and an ontology of tag meaning, i.e., MOAT [10], can be used mitigate the issue. Similarly, it would be more accurate to classify sentiment in reference to topics rather than classifying the overall sentiment of a tweet.

Social media data is a multi-faceted, global phenomena which take many forms. Besides the temporal aspect, social media data also contain a geographical dimension which provide relevant information to include in a sentiment query language. In addition, integrating different sources of temporal data such as, e.g., stock prices and news reports, may further increase utility of the language.

References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. L. P. Chen, editors, *ICDE*, pages 3–14. IEEE Computer Society, 1995.
2. N. Angelopoulos, V. S. Costa, J. Azevedo, J. Wielemaker, R. Camacho, and L. Wessels. Integrative functional statistics in logic programming. *Proc. of Practical Aspects of Declarative Languages*, 7752, 2013.
3. J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.
4. J. Bollen, A. Pepe, and H. Mao. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 450–453, 2011.
5. J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
6. T. P. Jurka. sentiment: Tools for Sentiment Analysis. Version 0.2. http://github.com/timjurka/sentiment.
7. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, 1997.
8. C. H. Mooney and J. F. Roddick. Sequential pattern mining – approaches and algorithms. *ACM Comput. Surv.*, 45(2):19:1–19:39, Mar. 2013.
9. B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 122–129, 2010.
10. A. Passant and P. Laublet. Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. In C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee, editors, *LDOW*, volume 369 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
11. R. Sadri, C. Zaniolo, A. M. Zarkesh, and J. Adibi. Optimization of sequence queries in database systems. In P. Buneman, editor, *PODS*. ACM, 2001.
12. Twitter. The streaming APIs. https://dev.twitter.com/docs/streaming-apis.
13. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.