


RESEARCH

Open Access



Expressive modeling for trusted big data analytics: techniques and applications in sentiment analysis

Erik Tromp¹, Mykola Pechenizkiy¹ and Mohamed Medhat Gaber^{2*} 

*Correspondence:

mohamed.gaber@bcu.ac.uk

²School of Computing and Digital Technology, Birmingham City University, Curzon Street, Birmingham, UK

Full list of author information is available at the end of the article

Abstract

Background: Sentiment analysis becomes ubiquitous for a variety of applications used in marketing, commerce, and public sector. This has been raising a natural interest within the academic research and industry to develop approaches and solutions for ubiquitous sentiment analysis. However, we can observe that most of the academic research focuses on adopting state-of-the-art machine learning techniques for sentiment classification and elements of natural language processing for feature construction and evaluate them on benchmark datasets not regarding much the actual application settings. In industry the focus is on developing platforms, services and customized solutions for certain applications and for different domains. In this work we propose a generic framework for ubiquitous sentiment classification. We discuss the Rule-Based Emission Model (RBEM) algorithm that we employ for polarity detection.

Results: We show with the experimental results on benchmark datasets and real case studies that the proposed framework and RBEM approach for polarity detection are indeed generic and extendable.

Conclusion: As the state-of-the-art machine learning techniques produce black-box models for sentiment analysis, they are hard to fine-tune, debug and adapt to new domains. The necessity to collect lots of labeled data from a particular domain is another obstacle. Therefore, in industry rather simplistic approaches are adopted resulting potentially in poor accuracy. The proposed framework for sentiment analysis allows to develop different solutions that are scalable, transparent, and easy to maintain.

Keywords: Sentiment analysis, Trust in Big Data, Rule-based mining

Background

The field of sentiment analysis is given more and more attention of the past few years [1, 2]. In commercial settings, brand-awareness and online reputation management (ORM) are crucial factors tightly linked with KPIs that require proper sentiment analysis to be put in place. In these practical settings however, typical sentiment analysis approaches are often too simplistic to successfully perform the task at hand or are overly complex but not transparent. The latter affects the possibility of adapting such sentiment analysis approaches to a certain domain of interest and results in either poor performance or high maintenance overhead. There is hence a big need for a low entry-level approach with high accuracy, easy adaptation and high maintainability¹.

In this work we present a framework for ubiquitous sentiment analysis. The framework facilitates the development of sentiment analysis tools that are reasonably accurate, have low maintenance overhead and easy adaptation to new applications. The framework we provide is designed to develop solutions that are:

- *extendable* to suit an application's needs;
- *scalable* to process large amounts of data, for example originating from social media;
- applicable in a generic setting yet *easily adaptable* to specific domains or application areas;
- *transparent* in their use due to the ease of human-understanding of the rule-based learners and their models; hence the developed solutions are also more *trustworthy* and much *easier to improve and maintain*;
- *portable* to run on devices with variations of computational resources including resource-constrained devices (e.g., smartphones and tablet computers); and
- *allowing for fine-grained analysis*, e.g. expanding the models and performing coarse-grained OLAP analysis that is useful for applications such as ORM and brand monitoring.

We present different use cases to demonstrate these properties. Besides the core sentiment classification part, we also show data collection and storage methodologies typically required in applicative settings to provide a fully-working solution rather than the sentiment analysis in isolation.

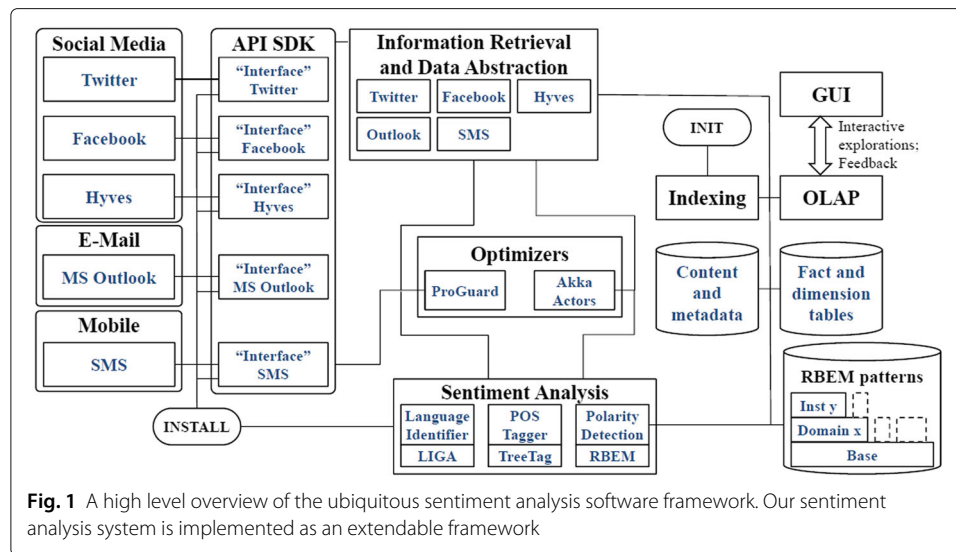
At the core of our framework is the polarity detection algorithm called RBEM. This algorithm is further described in “Polarity detection with RBEM” section. As pre-processing steps we perform language identification and part-of-speech tagging. This three-step approach is our actual working approach for performing sentiment analysis. Around this working pipeline, we construct data collection and data access interfaces that can extract data from different source systems in an independent way. Upon completion of sentiment analysis, the framework allows for persistent storage of the individual results as well as aggregated results. This way, fine-grained analysis can be performed to e.g. expand the models and coarse-grained OLAP analysis can be performed to gain insights, the latter being typically interesting for applications such as ORM and brand monitoring.

The rest of the paper is organized as follows. We introduce our framework for ubiquitous sentiment analysis in “Methods” section. In “Polarity detection with RBEM” section we consider Rule-Based Emission Model (RBEM) heuristics, training and classification procedures and their accuracy on the constructed benchmarks that we make publicly available. Five different use-cases taken from real developed solutions used as commercial services and research prototypes are presented in “Results and discussion” section. We describe related work in “Related Work” section. “Conclusions” section concludes the paper.

Methods

Overall processing pipeline

Our sentiment analysis system is implemented as an extendable framework presented in Fig. 1. This framework allows to crawl and scrape different data sources, mainly focusing on social media, for data that can serve as input for the sentiment analysis process. The analysis of the e-mails is implemented as a subsystem separated from the social media



sources. This is done for pragmatic reasons – people who use mainly e-mail or mainly social media can use one of the plug-ins. For similar reasons, the SMS interface is also decoupled from the other data interfaces. As the information retrieved from various social media, SMS and e-mails differs in format, we use an abstraction layer such that all input looks the same to our sentiment analysis process.

The sentiment analysis itself in our setting consists of three steps but due to the nature of the framework, more steps can be added without jeopardizing the other steps. For example, an emotion classifier – a more elaborate case of sentiment analysis where the emotion of a message is automatically extracted – can be added as an additional step. Each of the separate steps of our framework can also be extended, modified or replaced to suit different purposes. As the output of one step is the input of the other step, there is no dependency between the actual operational properties of the steps but only on the input and the output.

We assume language identification and part-of-speech tagging to be untouched by domain-specificity and hence we do not use domain-specific models or approaches for those two steps (see e.g. [3] for supporting evidence on language identification). For polarity detection however, we think domain-specificity is a necessity. Consider for example using the word ‘horrifying’ in the domain of automotive. A ‘horrifying car’ is a negative sentiment and hence this word indicates it as being such. Considering the domain of (horror) movies however, we find that this word indicates positivity. To this end we use a layered approach in constructing and applying the models for polarity detection. We start with a generic and domain-independent base-model - one per language. This model is the largest model and contains generic polarity information, for example indicating that the word *good* is positive. On top of this base model we define domain-models. Domain-models overrule base-models when intersecting and are much more specific. They can contain indicators like the word *horrifying* for automotive and horror movies in our example. Finally, we also allow for instance-models. These models are tailored towards one specific instance, application or usage of the model. In the setting of ORM, such a model might interpret anything positive about competitors as negative and any-

thing negative about competitors as positive. Since competitors are specific to a given brand or entity and overrule domain- and base-models, we keep this information in the instance-models.

Optimizers are put in place specifically when applying either high-volume sentiment analysis or low-resource (e.g. mobile) sentiment analysis. The entire framework is implemented in the Scala² programming language. The Akka actor framework³ is a framework written in Scala to allow for distributed computing. We use Akka as platform to make our framework work in a distributed setting. Since Scala is implemented on top of Java, compiling any Scala application will incorporate the Scala library itself as well as other often unused dependencies, resulting in relatively large file sizes when applying it to mobile devices. We use ProGuard⁴ to keep the file size to its necessary limit.

Data source management

As shown in Fig. 1, our system contains several kinds of data sources from which we extract information. For data collection, we decouple two different key aspects. For every data source we construct an interface that allows us to communicate with the data source, either through an API (in the case of social media) or through other extraction methodologies such as through plugin (in the case of Outlook) or app (mobile) functionalities. In addition to being able to communicate with the data sources, the nature of each data source requires us to collect data in different manners. Twitter for example, is best suited for streaming data collection whereas Facebook and Hyves only support batched retrieval of data. These differences require us to create a focused information retrieval unit for every data source system.

For social media, we can retrieve information in different ways. We can collect random samples or perform focused queries to find more relevant content. Finding relevant content can typically be done by searching for keywords, user names/ids or geographical bounding boxes. Since the way data should be queried for is application independent, our framework supports all of these query methods, facilitating e.g. ORM using keywords, analysis of personal correspondence (using user names) or demographical segmentation (using geographical bounding boxes).

When we analyze personal correspondence (see also “SentiCorr” and “Mobile SentiCorr” section), we can additionally use Outlook or SMS data. For both these data sources we do not need to query for specific information since a mobile device or Outlook instance is assumed to be connected to a specific individual that we are to analyze data for.

Our core three-step sentiment analysis process is designed to be source-system agnostic. Hence we need a uniform representation of the data we feed into our sentiment analysis process. To achieve this, we place a data abstraction layer over all data sources. In fact, every data source has its own method for abstraction over its data because the representation of the data may differ per source system. Since our sentiment analysis process works on sentence-level, the way we abstract over data is to chunk every information piece into separate sentences. For social media (especially Twitter, limited to 140 characters) and SMS this usually means we require no additional effort. For e-mail however, this means we chunk a single e-mail message into separate sentences and classify sentence-by-sentence instead of the e-mail message as a whole.

OLAP and GUI

Our framework has the ability to forward data to persistent storage. The reason why we do this is two-fold. First, we want to be able to perform OLAP-style analysis on the results of our sentiment analysis. Second, in order to create domain-specific or personal - personal being identical to instance-specific - models, we need to be able to inspect the results and perform feedback into our models. These two separate processes are catered for by our OLAP and GUI functionalities.

Sentiment classification

We consider a three-step approach for designing the automated sentiment analysis. The steps include *language identification*, *part-of-speech tagging* and *polarity detection*, as shown in the Sentiment Analysis module in Fig. 1. To cater for a multilingual approach, we perform language identification using the LIGA [3] algorithm. We use language information as a filter, to remove any messages not written in the language of interest, and as a pre-processing step to be able to apply language-specific models in subsequent steps. For POS-tagging we use the TreeTagger, having publicly available models for different languages [4]. The third and final step is polarity detection for which we use the RBEM algorithm presented in [5]. We provide a detailed description of RBEM and discuss its performance in “Polarity detection with RBEM” section.

This three-step approach is generic in a sense that sentiment analysis can be performed on any data source. But it is also easily extensible, allowing to include specific knowledge of the particular source, e.g. Twitter hashtags.

Polarity detection with RBEM

Pattern groups

The rules used in the RBEM algorithm directly stem from nine different *pattern groups*, defined as follows.

- Positive** patterns are positive when taken out of context. English examples hereof are *good, well done*⁵.
- Negative** patterns are negative when taken out of context, e.g. *bad, terrible*.
- Amplifier** patterns strengthen polarity of n entities to their left and right, either positive or negative, e.g. *very much, a lot*.
- Attenuator** patterns weaken polarity of n entities to their left and right, either positive or negative, e.g. *a little, a tiny bit*.
- Right Flip** patterns flip the polarity of n entities to their right, e.g. *not, no*.
- Left Flip** patterns flip the polarity of n entities to their left, e.g. *but, however*.
- Continuator** patterns continue the emission of polarity, e.g. *and, and also*.
- Stop** patterns interrupt the emission of polarity. Stop patterns usually are punctuation signs such as a dot or an exclamation mark, expressing the general case that polarity does not cross sentence boundaries.
- Neutral** patterns do not have any particular meaning but may eliminate the existence of other patterns in a given context.

The need for positive, negative and negation patterns is evident. The need for continuators and left flips has been indicated in [6]: conjunctive words such as *and* usually connect

adjectives of the same polarity whereas conjunctive words such as *but* usually connect words of opposing polarity. It is easily seen that certain words strengthen or weaken polarity, these are covered by the amplifier and attenuator patterns. The stop patterns are especially useful in determining sentence-based sentiment as these patterns block polarity emission and typically consist of sentence delimiters such as punctuation. The neutral pattern group does not have a specific logic or rule associated with it but is merely there to eliminate the presence of other patterns when a neutral pattern subsumes a pattern of a different pattern group.

Combining these nine pattern groups in an order according to simple rules for setting and removing stops, for emitting positive and negative sentiment, for amplifying and attenuating sentiment and for right- and left-flipping of the sentiment allows us to define an emissive model.

Learning RBEM

Each message m of length n is represented as a list $m = [(w_1, t_1), \dots, (w_n, t_n)]$ of tuples of a word w_i with its respective POS-tag t_i . Upon such a message, patterns can be defined. A pattern is a list of tuples of words and POS-tags represented as m . Patterns belong to a certain pattern group and hence we represent a pattern q as a tuple $q = (g, p)$, where g is the pattern group q belongs to, and p is the list of entities comprising the actual pattern. In general, each element (w'_i, t'_i) of a pattern p consists of a word w'_i which is precisely defined and a POS-tag t'_i which is also precisely defined. As an exception, elements of p may contain wildcards instead. We consider three types of wildcards.

- **Word wildcards** ($_, t'_i$): in this case we only consider t'_i . w'_i can be any arbitrary word.
- **Single-position wildcards** ($_, _)$: in this case a single entity can be any arbitrary combination of a single word and a single POS-tag.
- **Multi-position wildcards** ($(*, *)$): in this case any arbitrary combination of word and POS-tag pairs of any arbitrary length matches the pattern.

Note that word and single-position wildcards can occur at any position in p . But multi-position wildcards can only occur in between two elements that are not multi-position wildcards as co-occurrence of other multi-position wildcards yields another multi-position wildcard.

Our model now simply consists of a set of patterns per pattern group, represented as the set *Model*, containing tuples of groups and patterns; (g, p) . All patterns except for the positive and negative patterns adhere to an action radius \mathcal{E} . We set $\mathcal{E} = 4$ according to the related experimental results with negation patterns reported in [7]. In general it is possible that the optimal choice of \mathcal{E} may vary from pattern to pattern and/or from one language to the other.

Classifying with RBEM

When classifying previously unseen data, we perform two steps. First we collect all patterns in our model that match our sentence. Then, we apply a rule associated with each pattern group - with exception of the neutral group - for each pattern present in our message.

Pattern matching

Each pattern $q = (g, p) \in Model$ is matched against our message $h = [(w_1, t_1), \dots, (w_n, t_n)]$ where $p = [(v_1, s_1), \dots, (v_m, s_m)]$. We consider each tuple (w_i, t_i) and evaluate $(v_1, s_1) =_{match} (w_i, t_i)$ where $=_{match}$ is defined as follows:

$$(v_j, s_j) =_{match} (w_i, t_i) \equiv \begin{cases} \text{true} & (1) \\ \quad \text{if } j > m, \text{ define } end \leftarrow i & \\ \text{false} & (2) \\ \quad \text{if } i > n & \\ \quad v_j = w_i \wedge s_j = t_i \wedge (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) & (3) \\ \quad \text{if } v_i \neq _ \wedge v_i \neq * \wedge j \leq m \wedge j \leq n & \\ \quad s_j = t_i \wedge (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) & (4) \\ \quad \text{if } v_i = _ \wedge s_i \neq _ \wedge j \leq m \wedge j \leq n & \\ \quad (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) & (5) \\ \quad \text{if } v_i = _ \wedge s_i = _ \wedge j \leq m \wedge j \leq n & \\ \quad (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) \vee (v_j, s_j) = & \\ \quad \quad =_{match} (w_{i+1}, t_{i+1}) & (6) \\ \quad \text{if } v_i = * \wedge j \leq m \wedge j \leq n & \end{cases}$$

Note that in the definition of $=_{match}$, cases (4), (5) and (6) correspond to the three different types of wildcards. Moreover, in the evaluation of the first disjunction of (6), $(v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1})$, it must hold that $v_{j+1} \neq * \wedge s_{j+1} \neq *$ due to the restriction we put on the occurrence of multi-position wildcards.

We match all patterns of all groups against every possible element (w_i, t_i) of m . While doing this, we need to keep track of two positions if a pattern matches; the start position of the match in m and the end position of the match in m . The starting position is i whereas the end position is end which is assigned a value in case (1) of $=_{match}$, implying a match between the pattern and the message. We thus get a set of matching patterns containing a start position, an end position and a pattern.

$$matchedPatterns = \{(start, end, (g, [(v_1, s_1), \dots, (v_n, s_n)])) \mid (v_1, s_1) =_{match} (w_{start}, t_{start})\}$$

Elements of $matchedPatterns$ may subsume each other. Subsumption in this sense is defined as follows, where we say that q_1 subsumes q_2 in message m .

$$\exists_{(s_1, e_1, q_1), (s_2, e_2, q_2) \in matchedPatterns} : s_1 \leq s_2 \wedge e_1 \geq e_2 \wedge \neg(s_1 = s_2 \wedge e_1 = e_2) \wedge q_1 \neq q_2$$

All patterns that are subsumed by some other pattern are removed. Note that coinciding patterns, having the same start position as well as the same end position, are not removed but as we deal with sets, such coinciding patterns must be of different pattern groups. Also note that it may be that a pattern containing a wild card may match our sentence multiple times from the same starting position. As the definition of $=_{match}$ dictates, we only find and hence maintain the shortest of such matchings. After removing subsumed patterns, the resulting set $maxPatterns$ only contains maximal patterns and is defined as follows. Note that this is where the neutral pattern group plays a role. Whenever a neutral pattern exists in a context that subsumes any other pattern, the neutral pattern is kept whereas the other pattern is discarded. During the application of rules however, nothing is done with this neutral pattern, explaining the name of this pattern group.

$$\begin{aligned} \text{maxPatterns} &= \{(s, e, q) \mid (s, e, q) \in \text{matchedPatterns} \wedge \\ &\neg(\exists_{(s', e', q') \in \text{matchedPatterns}} : s \leq s' \wedge e' \geq e \wedge \neg(s = s' \wedge e = e') \wedge q \neq q')\} \end{aligned}$$

Rule application

After having collected all maximal patterns, we can apply the heuristic rules for each different pattern group, excluding the neutral pattern group. The rules formally work out the motivation for the presence of each pattern group. The order in which the rules are applied is crucial and so is the role of the action radius \mathcal{E} . We outline each of the rules in the order in which they are to be applied. We assume we are given a message m and a model $(\text{Model}, \mathcal{E})$ on which maxPatterns is defined. Every element $e_i = (w_i, t_i) \in m$ has a certain emission value $em(e_i)$ which initially is set to 0 for all $e_i \in m$.

Rule 1. Setting stops – This rule sets emission boundaries in our message m . It uses all left flip and stop patterns and sets a stop at the starting position of such a pattern. We thus get a set of stops:

$$\text{stops} = \{s \mid (s, f, \text{leftflip}) \in \text{maxPatterns} \vee (s, f, \text{stop}) \in \text{maxPatterns}\}$$

Rule 2. Removing stops – Stops set in the previous step can be removed by continuator patterns. This however, only happens to the left of a continuator pattern. We thus remove all stops that occur closest to the left of a continuator pattern, taking \mathcal{E} into account:

$$\begin{aligned} \text{stops} &= \text{stops} \setminus \{t \mid t \in \text{stops} \wedge \\ &\times (\exists_{(s, f, \text{continuator}) \in \text{maxPatterns}} : t \leq s \wedge s - t < \mathcal{E} \wedge \neg(\exists_{t' \in \text{stops}} : t < t' \leq s))\} \end{aligned}$$

Rule 3. Positive sentiment emission – A positive pattern can emit positive sentiment among elements of m . The strength of the emission decays over distance and hence we need a decaying function. We use e^{-x} as decaying function, where x is the distance between the positive pattern and an element of m . The choice of the formula e^{-x} is just a choice made by the authors and is not proven to be the optimal formula. As center for the emission, we take the floor of the center of the pattern in m , computed by taking the center of start and end position. We also need to take all stops into account. For each positive pattern, we update the emission values $em(e_i)$ as follows:

$$\begin{aligned} \forall_{(s, f, \text{positive}) \in \text{maxPatterns}} : c = \left\lfloor \frac{s+f}{2} \right\rfloor \wedge (\forall_{e_i \in m} : \neg(\exists_{t \in \text{stops}} : c \geq i \Rightarrow i \leq t \leq c \vee i \geq c \\ \Rightarrow c \leq t \leq i) \Leftrightarrow em(e_i) = em(e_i) + e^{-i}) \end{aligned}$$

Rule 4. Negative sentiment emission – Negative patterns are dealt with in the same way positive patterns are. The only difference is that our decaying function is now negative, yielding $-e^{-x}$. The updating of emission values happens in the same manner:

$$\begin{aligned} \forall_{(s, f, \text{negative}) \in \text{maxPatterns}} : c = \left\lfloor \frac{s+f}{2} \right\rfloor \wedge (\forall_{e_i \in m} : \neg(\exists_{t \in \text{stops}} : c \geq i \Rightarrow i \leq t \leq c \vee i \geq c \\ \Rightarrow c \leq t \leq i) \Leftrightarrow em(e_i) = em(e_i) + -e^{-i}) \end{aligned}$$

Rule 5. Amplifying sentiment – Amplifier patterns amplify sentiment emitted either by positive or negative patterns. Similar to the decaying function used for positive and negative patterns, amplification diminishes over distance. Moreover, since entities may already emit sentiment, we use a multiplicative function instead of an additive function. The function we use is $1 + e^{-x}$ where x is the distance. Again this formula is just chosen

by the authors and not proven to be optimal. In contrast to positive and negative patterns, amplifiers adhere to the action radius \mathcal{E} . The emission values are updated as follows:

$$\forall_{(s,f,amplifier) \in \text{maxPatterns}} : c = \left\lfloor \frac{s+f}{2} \right\rfloor \wedge (\forall_{e_i \in m} : (\neg (\exists_{t \in \text{stops}} : c \geq i \Rightarrow i \leq t \leq c \vee i \geq c \Rightarrow c \leq t \leq i) \wedge 0 < |c-i| < \mathcal{E}) \Leftrightarrow em(e_i) = em(e_i) \cdot (1+e^{-i}))$$

Note the $0 < |c-i| < \mathcal{E}$ clause. This constraint dictates that $|c-i|$ is at least 1 in $1 - e^{-|c-i|}$ (which is our $1 + e^{-x}$ function), thus avoiding the case that we multiply by 0 (when we allow $|c-i| = 0$, we get $1 - e^0 = 0$) and hence completely remove emission values.

Rule 6. Attenuating sentiment – Attenuator patterns perform the reverse of amplifier patterns and weaken sentiment. To do so, instead of using $1 + e^{-x}$, we use $1 - e^{-x}$:

$$\forall_{(s,f,amplifier) \in \text{maxPatterns}} : c = \left\lfloor \frac{s+f}{2} \right\rfloor \wedge (\forall_{e_i \in m} : (\neg (\exists_{t \in \text{stops}} : c \geq i \Leftrightarrow i \leq t \leq c \vee i \geq c \Leftrightarrow c \leq t \leq i) \wedge 0 < |c-i| < \mathcal{E}) \Leftrightarrow em(e_i) = em(e_i) \cdot (1-e^{-i}))$$

Rule 7. Right flipping sentiment – Right flip patterns simply flip the emission of sentiment to their right as follows. If there is a stop at the exact center of our right flip, we disregard it:

$$\forall_{(s,f, \text{rightflip}) \in \text{maxPatterns}} : c = \left\lfloor \frac{s+f}{2} \right\rfloor \wedge (\forall_{e_i \in m} : (\neg (\exists_{t \in \text{stops}} : c < t \leq i) \wedge |c-i| < \mathcal{E}) \Leftrightarrow em(e_i) = -em(e_i))$$

Rule 8. Left flipping sentiment – Left flip patterns mirror the effect of right flip patterns:

$$\forall_{(s,f, \text{leftflip}) \in \text{maxPatterns}} : c = \left\lfloor \frac{s+f}{2} \right\rfloor \wedge (\forall_{e_i \in m} : (\neg (\exists_{t \in \text{stops}} : i \leq t < c) \wedge |c-i| < \mathcal{E}) \Leftrightarrow em(e_i) = -em(e_i))$$

Once the above rules have been applied in the order given, every element e_i of m has an emission value $em(e_i)$. The final polarity of the message is defined by the sum of all emission values for all elements of m :

$$\text{polarity} = \sum_{i=1}^n em(e_i)$$

Straightforwardly, we say that m is *positive* (class +) if and only if $\text{polarity} > 0$. Likewise, we say that m is *negative* (class -) if and only if $\text{polarity} < 0$. Whenever $\text{polarity} = 0$, we say that m is *neutral* (class =).

When looking at the rules, it becomes clear that the order is important. Stops need to be set first since the other rules depend on stops. Next positive and negative sentiment need to be defined because amplifying, attenuating and flipping sentiment requires sentiment beforehand. Next the sentiment is amplified and attenuated based on the positive and negative emissions defined before. Finally the flips change the direction of the sentiment.

Benchmarking on dataset with verified labels

The first goal of our experiments is to benchmark the performance of the proposed RBEM comparing it against popular classification approaches for polarity detection. As a second goal, we want to evaluate our RBEM method against the current state-of-the-art Deep Learner. Please note that we do not attempt to show that RBEM is the most accurate method for polarity detection. We conduct experiments to check whether RBEM can achieve comparable results with existing approaches in term of accuracy, while providing

a better utility in different application settings. The training set for our polarity detection algorithm contains messages in multiple languages, multiple sentiments and multiple domains, stemming from social media. For our training set we use two different retrieval approaches - a large volume, unsupervised retrieval approach and a lower volume, supervised approach involving manual annotation. We use both approaches to obtain both large volumed data as well as highly accurate data on which it is more likely to find useful patterns from the polar messages.

The methodology we use to collect large volumes of data covering different sentiments is similar to [8, 9], in which smileys are used as noisy labels for sentiment and news messages as noisy indicators of neutral messages. For positive and negative messages we query Twitter just for 30 minutes searching for content with happy smileys such as :), :-), :D etc. for another 30 minutes with sad smileys such as :(, :-(, :(etc.. For neutral messages, we extract all messages produced by news instances such as the *BBC*, *CNN* (English) or *EenVandaag* (Dutch). We do this again for 30 minutes.

In addition to collecting training data based on smileys as well as to construct our test set, we collected random data from Twitter as well and then manually annotated the messages. We first labeled messages on language and kept only Dutch and English ones. We next labeled each message on its polarity, being either one of positive, negative or neutral. Finally, we extracted numerous RBEM patterns from each message.

The labeling of this additional training set and the test has been performed by multiple annotators, divided into two groups. The first group consisted of three annotators and focused on extracting Dutch messages only and annotating their polarity only, they did not identify RBEM patterns as for testing merely a polarity label is required. The second group consisted of two annotators and focused on extracting English messages only, followed by the same process as for Dutch. We use messages for Dutch in which at least two out of three annotators agreed upon polarity and for English we use messages for which both annotators agreed upon polarity⁶.

The constructed benchmark datasets are made publicly available⁷ for reproducibility of the experimental study and facilitation of further experimentation by other researchers.

The size of the resulting training and test sets is shown in Table 1. The numbers of patterns present in our RBEM model used in benchmarking are shown in Table 2.

We compare RBEM against other popular approaches used for sentiment classification, including Prior Polarity Classifier (PPC), Naive Bayes (NB), AdaBoost (AB) with decision stumps as base classifiers, and Support Vector Machines (SVMs).

We experimented with using four different feature spaces where we use either tokens, POS tags, a combination of both or patterns. We also experimented with using all features or the top 2000, 4000 or 8000 features as ranked by mutual information. The resulting

Table 1 The sizes of the training/test sets

	Training set/test set size	
	English	Dutch
Positive	3657/205	1402/262
Negative	3491/200	1546/200
Neutral	4802/454	2917/595
Total	11950/859	5865/1057

Table 2 The number of patterns present in the English and Dutch RBEMs

Pattern type	English count	Dutch count
Amplifiers	67	53
Attenuators	12	6
Rightflips	39	8
Continuators	10	4
Leftflips	5	2
Negatives	541	364
Positives	308	231
Stops	0	2

accuracies are given in Table 3. Note that instead of exhaustively reporting all possible feature sets, we only report the best performing feature set for each approach.

The precision and especially recall of the RBEM algorithm are much higher than those of other approaches. The RBEM algorithm is thus the most favored approach by the experiments conducted.

To investigate how much we can improve the performance of the RBEM algorithm, we investigate how much more the accuracy increases when we have more patterns in RBEM.

In approximately six hours of dedicated labeling we found 81 additional patterns. A linguist however would most likely do this much quicker. We mainly label more on positive and negative patterns as we expect to gain the most with these pattern types. Moreover, as our Dutch model was relatively small with respect to our English model, we focused on Dutch patterns. Including additional patterns to RBEM resulted in accuracy increase from 72.4 to 74.1 %, indicating a quick win in accuracy. Precision and recall increased to 73.3 and 87.6 % correspondingly.

RBEM vs. deep learning

Deep learning utilizing recursive auto-encoders have been recently shown to achieve the highest accuracy [10, 11] on existing publicly available benchmarks like movie reviews dataset stemming from Rotten Tomatoes [12] and can be considered as the state-of-the-art in polarity classification accuracy.

To compare our RBEM method against the recursive auto-encoder, we reproduced the 10-fold cross-validation results (77.0 % accuracy) reported in [10] for the Rotten Tomatoes dataset, using their implementation⁸ of the recursive auto-encoder.

As a side note, reproducing results was not so trivial as the predictive accuracy of deep learner was dropping significantly (to 57 %) with different choices of parameters and/or number of features used to train the model.

Using the Rotten Tomatoes dataset⁹, we have to perform a lot of manual labeling to incorporate the entire training set into a model. We split the training set into a training

Table 3 The overall accuracy (A), precision (P) and recall (R) of polarity detection

Approach	A	P	R
NB - All Tokens	0.616	0.551	0.393
SVM - All Patterns	0.637	0.646	0.564
RBEM	0.724	0.719	0.862
PPC - Using SentiWordNet	0.681	0.737	0.498
AB - POS-tags - 50 stumps	0.723	0.729	0.691

set of $\frac{2}{3}$ of the entire dataset and a test set. We incrementally process data in batches of 20 messages each and extract all patterns we can find from a batch and next evaluate accuracy on our test set. We do this 62 times and observed a smooth linear increase of the RBEM predictive accuracy from 30 to 50 %. This contrasts our expectation since typically a lot of increase is found in the first few iterations and then little increase is obtained in subsequent iterations. We think the linearity is best explained by the high diversity of the dataset.

To compare the RBEM model trained on the 620 messages against that of the recursive auto-encoder, we train the auto-encoder on the same 620 messages and evaluate both models on the remaining rotten tomatoes data. Recursive Auto-Encoder and RBEM achieve similar accuracies of 59.8 % and 58.2 % correspondingly. This is far from the 77.0 % accuracy achieved with a larger labeled training set. Nevertheless, the point we make here is that in the unfavorable settings for RBEM we can still reach comparable results with the current state of the art.

We also applied the same recursive auto-encoder to the benchmark dataset described in “Benchmarking on dataset with verified labels” section.

We train a deep learner using the exact same parameters and setup as in the reported Rotten Tomatoes experiment. Since our datasets contain both English and Dutch sets, we train a separate model for each language and test them on the corresponding test sets. The resulting accuracies of the recursive auto-encoder were 40.0 % for English 62.2 % for Dutch tweets.

Observe that these accuracies are significantly lower than RBEM’s accuracies on the same dataset (Table 3) and other much simpler approaches.

This might be possible to explain by the nature of the data, originating from Twitter, which could be harder to classify due to improper use of language and grammar, presence of class imbalance and realistic setting of including positive, negative and neutral messages.

We also believe that the fine-tuning of the model parameters could give better performance, although we could not achieve this easily.

We would like to emphasize once again that with the benchmarking attempts we performed, we did not aim to show a superiority of RBEM over popular classification techniques or a state-of-the-art deep learner for sentiment classification. We illustrated that RBEM provides reasonable and comparable results to other existing well-performing approaches for polarity detection in terms of accuracy.

Results and discussion

We discuss five case studies coming from a real-life application of (ubiquitous) sentiment analysis. With these case studies we demonstrate the strengths of the proposed framework and suitability of the RBEM approach for sentiment analysis. The first case study in “TV shows buzz monitoring” section is about monitoring the discussions of TV shows on social media. It demonstrates the ease of portability of the sentiment analysis related data processing pipeline to a new domain.

The second case study we present (“Emotion tracker” section) is called Emotion Tracker or Emotiepeiler in Dutch, which is an existing showcase website driven by the framework we presented in “Methods” section. With this case study we demonstrate that we successfully reuse the framework as-is for developing a new application. Besides, the modularity

of the framework allowed to extend the application with additional functionality without introducing any changes to the core.

The third case study that we present is called SentiCorr (“SentiCorr” section) that concerns sentiment analysis of personal correspondence. It mainly focuses on local application of our framework with an opportunity to tailor the actual models to a specific individual’s needs.

The fourth case study (“Mobile SentiCorr” section), similar to SentiCorr but used on mobile devices, that introduces new challenges and opportunities. With this case study we intend to demonstrate the portability of our framework to resource-constrained environments.

We conclude by presenting another case study (“Extending RBEM for Emotion classification” section) that illustrated how RBEM can be extended to handle emotion classification based on Plutchik’s wheel of emotions [13].

TV shows buzz monitoring

Use of language is highly dependent upon the domain in which it is being used. As such, it is expected that a generically trained model does not perform as well as it should on a specific domain and that domain-specific models do not port well to other domains. In these experiments we show that regardless of whether a concrete instance of RBEM is domain-agnostic or not, its models are easily adapted to new domains for which no previously annotated data were available.

We illustrate the adaptability of the RBEM algorithm through a real-life use case in which it has been applied to a highly specific domain, being the domain of media and particularly television. We do this by taking the generic base model, its characteristics being given in Table 2, and adapting it to fit the television domain. This process involves human interaction, but we show that the adaptation requires little effort of a domain expert. This is in fierce contrast to general-purpose state-of-the-art classification techniques used for sentiment classification, including e.g. SVMs, supervised sequence embedding [14] or deep learning neural networks [15] with which adaptation of models is a nontrivial labor-intensive process requiring a deep understanding of machine learning.

Experiment setup

To demonstrate the ease of portability to a new domain, we applied the generic model constructed in “Benchmarking on dataset with verified labels” section to the television domain in two different use cases. For convenience we name them *Experiment 1* and *Experiment 2*. The use cases arose from two real-life scenarios in which two different and non-related Dutch television broadcasters wanted to use our approach for sentiment analysis on social media with respect to specific television shows, news bulletins or movies being broadcasted¹⁰.

Even though language use may be different in the television domain, it is expected that language n-gram characteristics as used by the LIGA algorithm hardly change. This expectation is supported by the experiments conducted in [3] where it is shown that LIGA generalizes well across domains.

Both experiments were conducted in the same manner and both applied solely to Dutch messages originating from Twitter.¹¹ For both experiments we initially collected data starting from 30 minutes before and 2 hours after a television broadcast exactly once. The

data was collected from Twitter by searching for the keywords given in Table 4. For each keyword, the amount of messages extracted is also mentioned.

Each of the Dutch messages (as classified by LIGA) is classified with Dutch RBEM as being *positive*, *neutral* or *negative*. These messages along with their polarity labels have been handed over to domain experts for judgement with the purpose of identifying common or drastic patterns that are often misunderstood by the generic RBEM base model. The domain experts were asked to return messages that they identified as being misclassified by RBEM, give their judgement on what the correct label would be and if possible, give a brief argument.

We investigated the received feedback to identify common patterns and drastic misinterpretations by the RBEM algorithm. These common and drastic patterns directly lead to modifications of our base model and can either entail removal of present patterns, addition of new patterns or both.

Generic model refinement

For *Experiment 1*, the domain experts returned 90 messages in total across all given keywords that were misclassified according to domain experts. For *Experiment 2* only 8 messages were returned. The great difference in the number of messages returned is not investigated but is most likely to due variance in commitment by the different domain experts.

Table 5 shows the patterns extracted from the resulting messages that corrected the greatest amount of misclassified messages. Note that in these experiments, we did not verify whether these corrections introduce new errors in messages that were not in the set of messages returned by domain experts.

From *Experiment 1*, we found that the pattern [(*te*, *partte*), (*adj*)] (in English: *too ...*), which is a negative pattern in the generic base model, expressing that having too much of something is often bad, is not always used to express something negative. Removing this pattern mainly resulted in messages classified as negative before being classified as neutral or even positive after. The words *jammer* (in English: *pity*) and *huilen* (in English: *crying*) are generically associated with negative polarity and hence existed as such in our generic model. In the television domain however, the Dutch word for pity is often used to

Table 4 The keywords used in the portability experiments and the number of resulting messages. Note that for *Experiment 2*, a single message may be included for multiple keywords

Description	Keyword	# Messages
<i>Experiment 1</i>		
TV show <i>Babyboom</i>	#babyboom	226
Talent shows	#bestezangers	199
TV series <i>Hitch</i>	#hitch	305
Real-life show	#htmd	969
TV series <i>House</i>	#house	199
Game show ¹²	#ihvh	772
<i>Experiment 2</i>		
Soap	goede tijden	2465
Soap	goedetijden	432
Soap	gst	4013
Venue of soap	meerdijk	232

Table 5 Best scoring patterns found in both experiments

Pattern	G	O	#C
<i>Experiment 1</i>			
[(<i>huilen</i> , <i>verbpressg</i>)]	-	-	7
[(<i>te</i> , <i>partte</i>), (<i>adj</i>)]	-	-	3
[(<i>jammer</i> , <i>verbpressg</i>)]	-	-	2
[(<i>jammer</i> , <i>verbpressg</i>), (*, *)], (<i>afgelopen</i> , <i>verbpapa</i>)]	+	+	1
<i>Experiment 2</i>			
[(<i>goede</i> , <i>adj</i>), (<i>tijden</i> , <i>nounpl</i>)]	=	+	2
[(<i>slechte</i> , <i>adj</i>), (<i>tijden</i> , <i>nounpl</i>)]	=	+	2
[(<i>stotterd</i> , <i>nounsg</i>)]	-	+	1

G - pattern group (positive +, negative - and neutral =), O - operation (add +, remove -) and #C - number of corrections

indicate that it is a pity a show is over and hence is positive instead of negative. Similarly, expressing an emotional act of crying often indicates a television broadcasting has high impact and hence is a positive pattern.

From *Experiment 2*, the main correction was a straightforward one. The television show *goede tijden*, *slechte tijden* contains the words *goede* and *slechte*, indicating positive and negative sentiment when no context is given. When talking about *goede tijden* in the context of this specific television show however, it is obvious that this is the name of the show and hence bears no emotional value. To this end, adding *goede tijden* (and likewise, *slechte tijden*) as a neutral pattern ensures that when this bigger context is given, the positive pattern containing just the word *goede* is subsumed by this newly introduced neutral pattern and hence eliminated.

After incorporating new patterns based on the feedback by domain experts, we reduced the number of misclassifications from 90 to 32 in *Experiment 1* and from 8 to 1 in *Experiment 2*.

This case study shows that the *transparency* provided by our framework can help us in tracing classification errors and correct them, making it more reliable and trustworthy.

Emotion tracker

The Emotion Tracker or Emotiepeiler¹³ in Dutch is a Dutch showcase website where sentiment around different topics or brands is collected, analyzed and presented in a visual way. The input for the system originates from Twitter only and is collected by searching for specific keywords.

The main view of the Emotion Tracker application is shown in Fig. 2. Different topics, brands, individuals and other entities are presented without any additional analysis information. Every entity can be clicked upon to show a more detailed view where actual analysis results are shown. Figure 3 shows this detailed view, consisting of three concrete sections. The left section shows the 10 latest tweets on the topic of interest. A green background indicates a positive label from our sentiment analysis, a red background indicates a negative label and a white background indicates neutrality. These tweets are shown in real-time. The upper-right section shows a gauge that summarizes the current sentiment level on a scale from 1.0 to 10.0, where a value of 5.0 implies a completely neutral value. This gauge is reset every hour, on the hour. The lower-left section shows an aggregated trend-line that summarizes the sentiment scores per hour for the last 48 hours. The tweet

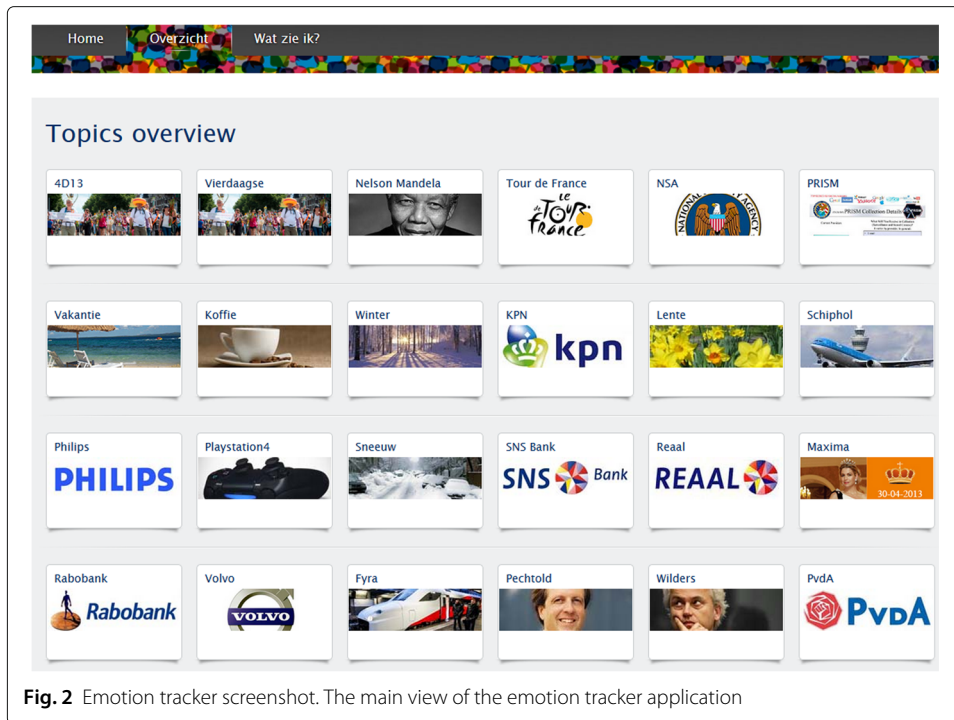


Fig. 2 Emotion tracker screenshot. The main view of the emotion tracker application

list and the gauge are real-time indicators, whenever a tweet is being posted that contains the keyword of interest, it will show up with a delay of 10 milliseconds.

Emotion Tracker was developed directly based on the described framework. The main benefit of doing this is that we could reuse the framework as-is; the only effort introduced was to create the actual website based on the selected functionality provided by the core

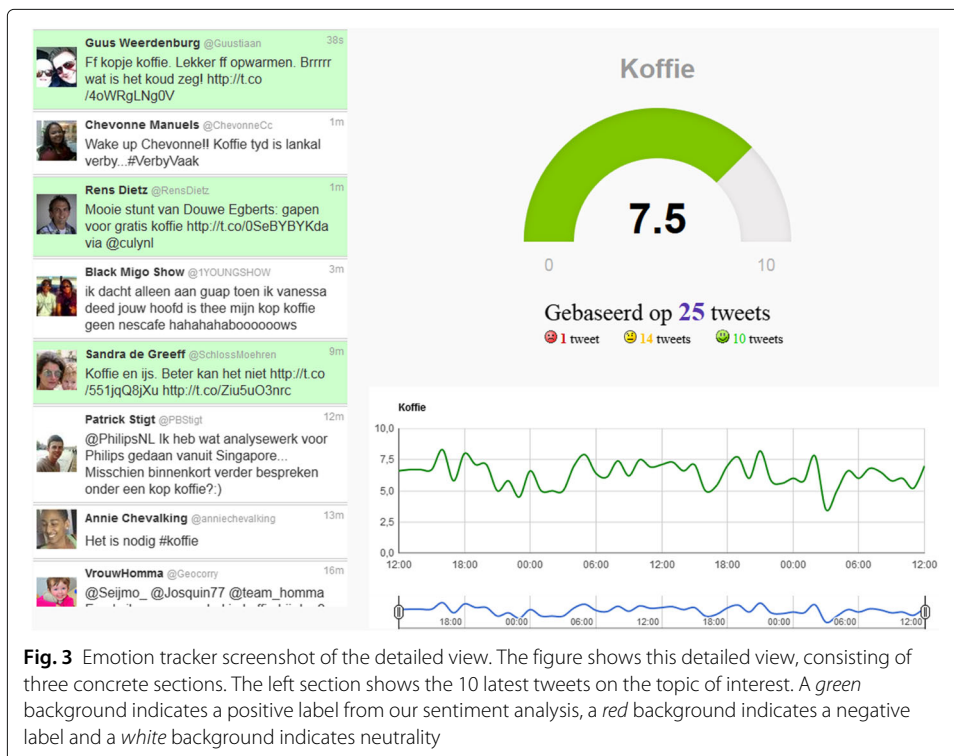


Fig. 3 Emotion tracker screenshot of the detailed view. The figure shows this detailed view, consisting of three concrete sections. The left section shows the 10 latest tweets on the topic of interest. A *green* background indicates a positive label from our sentiment analysis, a *red* background indicates a negative label and a *white* background indicates neutrality

modules. Note that we only used the generic base models since the topics and brands presented on the website cross domains.

Emotion Tracker only focuses on Twitter messages in Dutch. We disabled the other data sources blocks present in our framework. The first step of the sentiment analysis process, language identification (using LIGA), served as a filter. At first we had 6 different languages incorporated into our LIGA models and used a threshold heuristic to filter out any message not written in any of those 6 languages. It turned out that for certain specific keywords, a lot of noise was introduced because this keyword existed as a function or highly often used word in another language similar to Dutch. An example hereof is *KPN*¹⁴, yielding a lot of Tagalog¹⁵ messages returned by Twitter. It turned out that Tagalog oftentimes resembles Dutch and hence Emotion Tracker also listed a lot of Tagalog messages on *KPN*. The LIGA algorithm is strong at disambiguating between messages written in languages it has knowledge of. We therefore remedied our Tagalog issue by incorporating Tagalog in our LIGA models. We did this for other languages as well, yielding a LIGA model of 14 known languages in total. Through this, we show the *adaptability* of our framework.

The name Emotion Tracker hints on actual emotions being tracked. Though polarity detection can be seen as a superficial form of emotion analysis, no actual emotions are involved. We are currently expanding Emotion Tracker by adding emotion classification after our polarity detection step. Due to the nature of our framework, this implies adding a single building block. This does not affect any of the existing functionalities of our framework and hence we can fully focus on developing the emotion classification in isolation. This shows the *extendability* advantage given by our framework.

SentiCorr

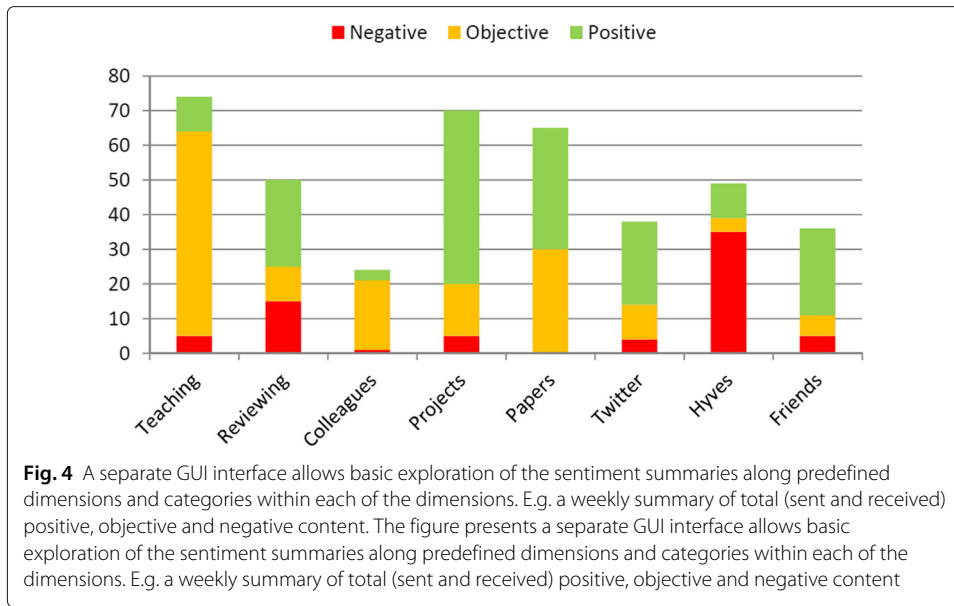
SentiCorr is the system for automated sentiment analysis on multilingual user generated content from various social media and e-mails [16], applied to personal correspondence. One of the main goals of the system is to make people aware how much positive and negative content they read and write. The output is summarized into a database allowing for basic OLAP style exploration of the data across basic dimensions including time, correspondents, read/write and alike.

Figure 4 illustrates one of the ways the information is presented to the user in an integrated view. Social media content and the summary views on the quantities of positive and negative content read or written over different periods of time are presented via separate interfaces providing an OLAP-style exploration of the data along the predefined dimension and allowing to zoom in to the level of the individual messages and zoom out to the grand total summary of the sentiments.

We also developed an MS Outlook plug-in that analyses the correspondence and highlights positive and negative sentences (Fig. 5). Using a locally installed GUI¹⁶, end-users of the SentiCorr system are able to construct a personal instance-model on top of the generic base-model of the RBEM algorithm by providing relevance feedback.

The current prototype can be used as is for monitoring the statistics collected from the sentiment analysis of the various texts and their further exploration. At the moment we have full support for English and Dutch languages in the developed three-step sentiment analysis process.

Besides the stand-alone use of the developed system, we have integrated SentiCorr functionality in the so-called Stress Analytics system (early prototype described in [17])



by relating the identified sentiments and their summaries with other events potentially related to the occurrences of stress. Information on such related events is extracted from stress measuring devices (detecting arousal based on heart rate, voiced speech or galvanic skin response measurements [18]), voice analysis, calendar data or working agendas, and analysis of facial expressions captured with a videocamera.

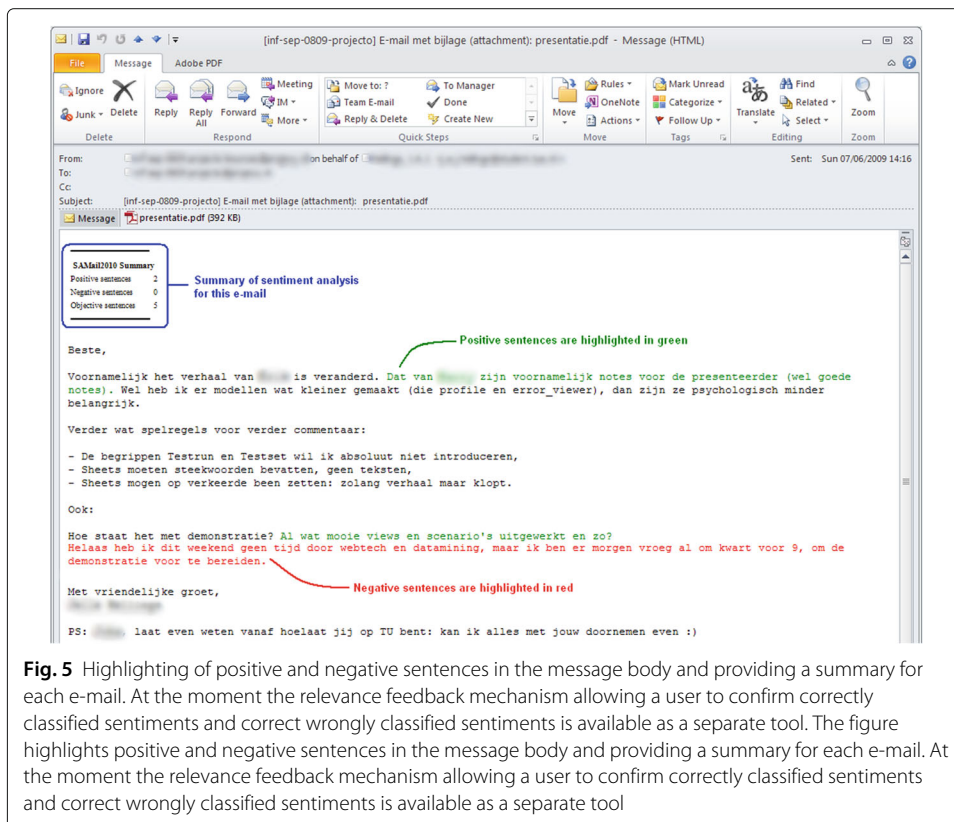


Fig. 5 Highlighting of positive and negative sentences in the message body and providing a summary for each e-mail. At the moment the relevance feedback mechanism allowing a user to confirm correctly classified sentiments and correct wrongly classified sentiments is available as a separate tool. The figure highlights positive and negative sentences in the message body and providing a summary for each e-mail. At the moment the relevance feedback mechanism allowing a user to confirm correctly classified sentiments and correct wrongly classified sentiments is available as a separate tool

This case study shows the *adaptability* of our framework to the application area of sentiment analysis of personal correspondence. A different solution based on the same framework has been developed for smartphone setting that we describe next.

Mobile SentiCorr

Mobile data consumption has been in continuous increase since the advent of the smartphone technology. Moreover, tablet computers have played an important role in shifting a big proportion of daily data consumption to the mobile handheld devices. Taking this into consideration, we found that it is an important development to have a mobile version of *SentiCorr*. With this use case we illustrate the peculiarities of performing sentiment analysis of personal communication on different platforms. Ubiquity of the application allows users to be aware of what the sentiment of the communication they receive either through personal messages or social media items *anytime, anywhere*.

Mobile SentiCorr introduces an application for an *Android* mobile phone that performs analysis on the general sentiment of texts received on the mobile devices such that a label of “objective”, “positive” or “negative” is given to the texts and the information is displayed in a timeline scenario so the user has an overall view of the general sentiment of the texts they are sending or receiving on their mobile device (Fig. 6).

We show that the major obstacles when applying sentiment analysis steps in a mobile scenario are the models that are used in the POS tagging. We experiment with the POS taggers and various models to determine the parameters under which to utilise different taggers and with which tagger models. As the selection of the POS tagger affects the results of the final sentiment analysis, the mobile application is tested on the same data as the original sentiment analysis system and compared to the original results to determine the accuracy of the analysis with respect to the *SentiCorr* system.

During development of the prototype, it was discovered that the original *SentiCorr* POS tagger - TreeTagger - could not be used on the Android mobile platform as it was only available as an executable compiled for the Microsoft Windows or Linux operating systems. The Android platform is based on a Linux operating system that utilises an ARM (Advanced RISC Machine) processor, so applications need to be cross-compiled to operate on the ARM processor. This can only be achieved with the source code which is not available for TreeTagger. Because of this, other POS taggers were investigated which were written in Java; two were selected: OpenNLP and the Stanford POS tagger. The same RBEM method for polarity detection was used.

One of the main concerns was the amount of space required to store the POS tagger model and the polarity detection model. The largest model was the POS tagger model and this also took the longest amount of time to load. Loading these models contributed to how the software was architected. In contrast to *SentiCorr*, we load our POS and RBEM models for a single language only, at start-up time. However, provisions have been made within the software such that it can be extended to include language selection in the future. The architecture of the *Mobile SentiCorr* system is compliant with the general framework shown in Fig. 1.

The *Mobile SentiCorr* application was installed on three mobile phones with various specifications. The aim of varying the mobile phones is to conduct stress testing to reveal the minimum configuration of computational power that is able to run our system. The

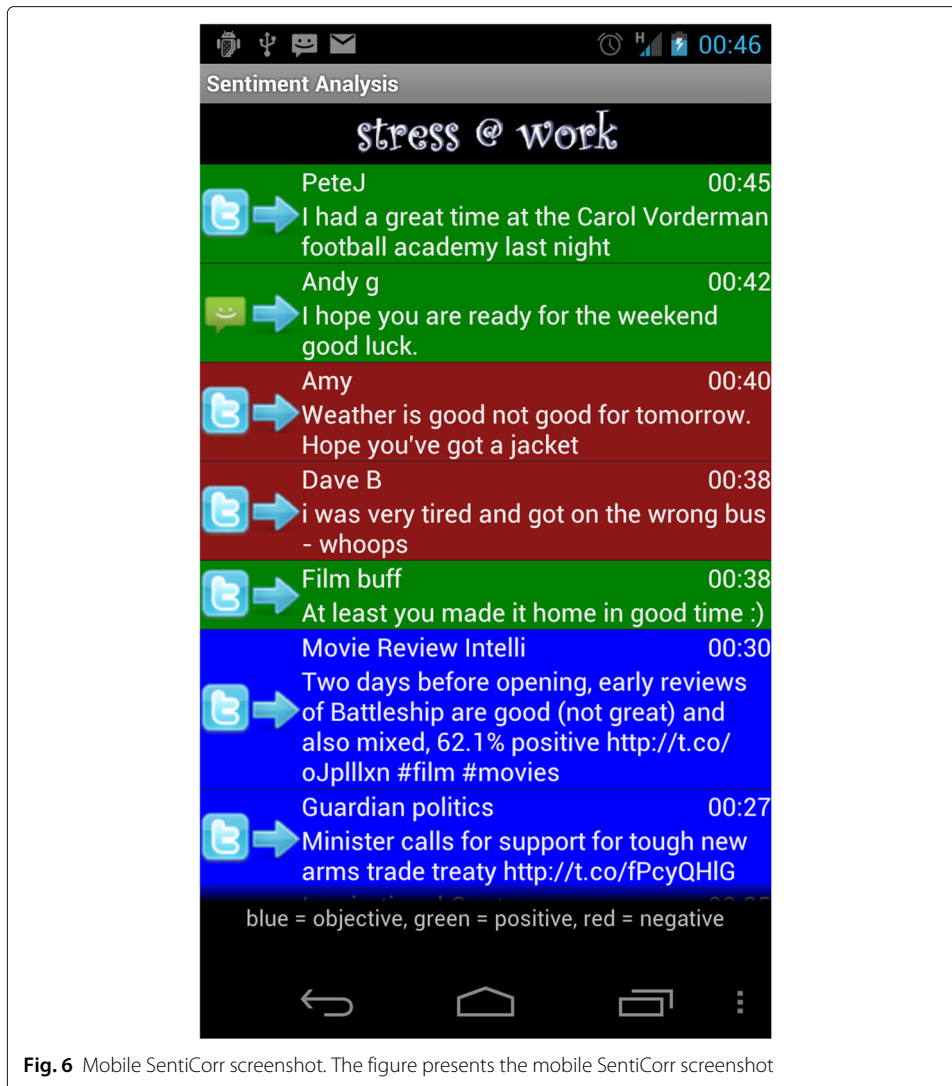


Fig. 6 Mobile SentiCorr screenshot. The figure presents the mobile SentiCorr screenshot

load time of the POS tagger models was the major factor in the duration of the execution time of the application.

As the sentiment analysis code is implemented as a standard *Java* library, we were able to calculate the accuracy of the sentiment analysis for each model on a desktop computer where the accuracy is taken as the number of correctly tagged tokens divided by the total number of tokens and represented as a percentage. The sentiment was analyzed on part of the original data used in the evaluation of *SentiCorr* and is based on 60 texts and utilizes the POS tags from the original data and uses this as the gauge for accuracy. These 60 texts were obtained from Twitter by scraping all public data and manually labelling 20 negative, 20 objective and 20 positive tweets, according to the tweet's text.

The accuracy of the models ranges from 83 to 90 % across the Stanford and the Open NLP POS tagger models, with the most accurate being the *Stanford English left3 words* and the *Stanford wsj-0-18left3words* which also had a low load time but only ran on phones running Android 3.0 and later versions. In most cases, it took nearly ten times as

long to load the Stanford models as it did with the Open NLP model, for a 2 % increase in accuracy.

Mobile SentiCorr gives a clear evidence of the *extendability*, *adaptability* and *portability* properties of our framework. The framework has been extended to suit the application needs of running in a relatively resource-constrained environment. Flexibility of choosing the POS tagger according to availability of resources is a unique extension addressing the need of this specific case study. Also adaptability is evident as the new user interface better suits the mobile environment, while the generic sentiment analysis process is still utilised. Portability is probably the most important lesson we learnt. We have been able to show that tailoring the framework to this environment can make the application portable, while maintaining the core components of the framework.

Extending RBEM for Emotion classification

This case study illustrates how we can extend our sentiment analysis beyond the typical granularity of polarity and instead considered eight basic emotions of Plutchik's wheel of emotions model [13]. Additionally, Plutchik defines eight *human feelings* that are derivatives of combinations of two basic emotions. This in fact means that with modeling only four axes, we can get a total of sixteen dimensions of emotions and feelings. In this model each of these eight basic emotions (*joy*, *sadness*, *trust*, *disgust*, *fear*, *anger*, *surprise*, *anticipation*) are opposites of one of the other basic emotions. This means that we can in fact measure four axes where opposite emotions exist on the two extremes of a single axis.

Crucial to the RBEM algorithm is that positivity and negativity are opposites of each other and hence allow for example negations to simply invert the emission. This specific characteristic of the algorithm makes it work well with Plutchik's model since the emotions defined in that model are also opposites of each other.

We extended the RBEM algorithm to RBEM-Emo [19] to perform the same type of rules but now – instead of having one axis to measure; positive on one end of the extreme and negative on the other extreme – we have four different axes, together yielding eight different emotions being measured. For our RBEM-Emo algorithm, we replace positive and negative pattern groups with eight new pattern groups, one for each basic emotion of Plutchik's model.

As an illustrative example, consider the sentence *I thought I would like the new XYZ phone, but now that I have it, it is a huge disappointment, it makes me angry*. Suppose also that we have the following patterns (Part-of-Speech tags left out for simplicity): (*I * like*, *Anticipation*), (*but*, *Leftflip*), (*huge*, *Amplifier*), (*disappointment*, *Sadness*), (*angry*, *Anger*). The algorithm would first assign the emotion scores to all parts of the sentence where patterns are found. This would yield the first part emitting negatively on *SurpriseAnticipation* dimension, the third phrase emitting negatively on *JoySadness* and the last phrase emitting negatively on *FearAnger*. Next, the scores on pattern indicated by the word *huge* will amplify the emissions on all axes, with the biggest effect on *JoySadness*. Finally, the leftflip indicated by *but* will convert all negative emissions on its left – influencing *SurpriseAnticipation* mainly – to its opposite direction, yielding emissions on *Surprise*. The final outcome will hence be that – ordered by decreasing strength – *Sadness*, *Anger* and *Surprise* are present.

Experiment setup

We compared RBEM-Emo against SVMs (LibShortText [20]), regression and the recursive auto-encoder of [21]¹⁷ on two benchmarks: the *Affect Dataset* [22]¹⁸ consisting of snippets of text obtained from books written by three different authors, and the *Twitter dataset* which we collected and annotated for this study¹⁹.

From the Affect dataset we used only those messages for which both annotators agree upon emotion. Moreover, since 85 % of all sentences in the dataset are neutral, and many general purpose classification techniques suffer from class imbalance, we experimented with two different datasets, one where neutral sentences are removed and only emotion-bearing sentences are maintained and one where neutral messages are included. For evaluation purposes, we use roughly $\frac{2}{3}$ of the data for training and $\frac{1}{3}$ for testing. The resulting sizes of the training sets are 7527 and 1084 instances depending on the in- or exclusion of the neutral class, and for test sets – 3590 and 488 instances correspondingly.

For the Twitter Dataset we collected tweets in three different languages: *English*, *Dutch* and *German*. We had at least two independent annotators to annotate each of these messages using a dedicated Web-based annotation tool. In case of disagreement, we use the prevailing emotion label given by the annotators as actual label for a message. If there is no agreement on the prevailing emotion label, the message was discarded. In addition, the annotators were asked to identify patterns in these messages such that we can later on construct the RBEM-Emo model from them. The resulting training/test set sizes are 289/113 for Dutch, 235/113 for English and 225/109 for German.

To ensure we have the right setup of the auto-encoder, we reproduced the polarity detection experiments on the rotten tomatoes dataset as done in [21] and obtained an accuracy of 77.0 %. This is in line with the results presented in [21], illustrating our setup is valid. When we apply our RBEM-Emo classifier, we get four scores for each axis in Plutchik's model, summing up to eight emotions. Finally, we assign a single label corresponding to the highest of all eight emotion scores.

Results

The accuracies of the best performing general purpose classification techniques on the Affect Dataset are compared to those of RBEM-Emo in Table 6. The majority class classification accuracy is given as a baseline. We report accuracies both for the case when neutral messages are kept in our dataset and when they are filtered out. We do this since the neutral messages compose 85 % of the entire original dataset and it is expected that generic classification techniques will suffer from class imbalance and learn biases towards this data rather than find actual emotions. This is reflected in the accuracies of the SVM and regression classifiers which are marginally higher than the majority class

Table 6 Accuracies on the Affect dataset

Method	Acc. w/ Ntl	Acc. no Ntl
Majority	84.4 %	37.7 %
SVM, W.C.	86.2 %	61.3 %
SVM, TF-IDF	86.2 %	65.0 %
Regr., W.C.	85.8 %	59.5 %
Regr., TF-IDF	85.5 %	63.4 %
RAE	84.4 %	60.4 %
RBEM-Emo	88.4 %	67.1 %

baseline. Surprisingly, the recursive auto-encoder (RAE) that is currently claimed to be the state-of-the-art technique for emotion classification performs worse than several simpler classifiers and in fact is as good as a majority class classifier. One possible reason for this might be that the size of our dataset is relatively small. RBEM-Emo classifier being a tailor approach to deduce emotional patterns outperforms the other classifiers.

In the second column of Table 6, we report the accuracies when all messages belonging to the neutral class are removed, yielding a more class-balanced dataset. Here we see much better improvements over the majority class baseline for SVM and regression and now also for the recursive auto-encoder. Using TF-IDF scores for features is favored over using just word counts. The RBEM-Emo method however, still outperforms the other classifiers.

Table 7 lists the accuracies obtained per language on our own Twitter corpus. For each classifier, we report the accuracy on each language (being Dutch, English and German) and report a total accuracy which is the average accuracy over all messages in all three languages. A generic result over all classifiers is that the accuracies on English data seem to be the lowest, implying most ambiguity within this language. Remarkable is that the recursive auto-encoder performs worse than SVM and regression models and yields no benefit over the majority class guess. Again, this could be due to the small size of the corpus or difficulty in finding the most suitable model parameters. There is no clear evidence on whether TF-IDF scores or word counts work better for this dataset. However, the RBEM-Emo classifiers yields the highest accuracy for each of three languages.

Related Work

Polarity detection

Polarity detection has been studied in different communities and in different application domains. The polarity of adjectives was studied in [6] with the use of different conjunctive words. A comprehensive overview of the performance of different machine learning approaches on polarity detection were presented in [23–25]. Typically, polarity detection is solved using supervised learning methods but more recently attention is being paid to unsupervised approaches [26].

Some of the recent works suggests the so-called sentic computing for utilizing common sense knowledge in sentiment analysis. A notable example is [27], in which a two-level affective common sense reasoning framework is proposed to mimic the integration of conscious and unconscious reasoning for sentiment analysis using data mining techniques. Recent related work can be found in [28–30].

Other works are those of [7, 31–33]. In these related works, the authors start from bootstrapping methods to label subjective patterns. In their latest work, both subjectivity and polarity detection is performed and evaluated using these patterns along with high precision rules defined in their earlier works.

Table 7 Accuracies on the Twitter dataset

Language	Majority	SVM W.C.	SVM TF-IDF	Regr W.C.	Regr TF-IDF	RAE	RBEM-Emo
nl	50.4	53.1	54.9	53.1	53.1	53.1	56.7
en	42.5	46.0	42.5	45.1	42.5	31.0	47.2
de	34.9	46.8	47.7	40.4	46.8	44.0	53.2
<i>all</i>	42.7	48.7	48.4	46.3	47.5	42.7	52.4

More recently attention is being paid to sentiment analysis on social media. Sentiment analysis on Twitter is researched by [8, 34] who use similar methodologies to construct corpora and analyze Twitter messages to determine their polarity. O'Connor et al. [35] use opinion mining on Twitter to poll the presidential election of the United States in 2008 and show how using Twitter opinion time series can be used to predict future sentiment.

Related research on polarity detection often focuses on using deep learners, modeling deep linguistic traits. Though such deep learners - for example neural networks or conditional random fields - might be able to capture hidden linguistic traits that are useful for analysis, understanding the logic underlying resulting models is cumbersome. In addition, such learners are resource and time intensive and hence do not fit well with applications where scalability is important.

Platforms for sentiment analysis

To the best of our knowledge, academic work on actual sentiment analysis platforms are scarce. Work on sentic computing²⁰ provides paradigms for sentiment analysis that can be used as reference for building a platform, but this is the only such resource we are aware of.

In contrast to academic platforms, many commercial platforms for sentiment analysis do exist. These platforms typically arise in specific application areas such as webcare and the broader field of online reputation monitoring. Example vendors of such commercial platforms are DataSift, Radian6 and Coosto²¹. The focus of these platforms is more on specific applications where sentiment analysis is one of the ingredients used to fulfill needs but since it is not a crucial component of the platform, it usually does not receive the attention it should and is treated as a black-box solution.

Application of sentiment analysis

Given that core sentiment analysis methodologies are improving and maturing, much research has been put into the application of sentiment analysis over recent years. Typical recent application areas review prediction of metrics using sentiment analysis on social media.

An example of using sentiment analysis in a predictive application can be found in [36], who studied using sentiment analysis for stock prediction. A commercial approach of using sentiment analysis for stock prediction can be found in SNTMNT²² who use sentiment analysis models and approaches tailored towards the financial market and its jargon.

Another application using sentiment analysis for prediction is found in election forecasting. In both [37, 38], the authors studied if and how sentiment analysis results can be used to predict outcomes of elections. They found that generally such predictions are competitive with panel-based predictions but typically are ahead by days.

More traditionally, sentiment analysis has been heavily studied on reviews, especially movie reviews given that benchmark and evaluation resources in this domain are relatively easy to obtain. Example works on applying sentiment analysis to movie reviews include [39–41].

Though many applications of sentiment analysis have been studied, related work typically focuses on providing a methodology to only one or two specific application areas at

once, not focusing on generic application of the proposed method. Little work is done on generically applicable and flexible sentiment analysis platforms that port well to any given application area.

Conclusions

Previous academic work in the area of sentiment analysis traditionally focused on benchmarking performance of sentiment classification techniques, typically for restricted application settings, e.g. assuming that all messages are written in one language only, usually English as the resources for English are best available and assuming that labeled data can be collected for the domain of interest in sufficient volumes. Such restrictions limit the direct utility of the developed sentiment analysis techniques.

In this paper we introduced a generic framework for ubiquitous sentiment analysis. We presented five different case studies that show its utility. They are witnesses of the ease with which we can develop solutions based on our generic framework for scenarios of different nature.

In this paper the main focus was on the data processing pipeline facilitating ubiquitous sentiment analysis.

In our ongoing work we elaborate on aspects of user interaction with the results of the sentiment analysis. One of the interesting approaches is to enrich and revise the patterns used by RBEM. Another and related important aspect is automated monitoring and continuous improvement of the polarity detection performance. By users we mean not only marketers or sentiment analysts. If the sentiment analysis is used for personal use (rather than for marketing needs) as discussed in the SentiCorr and Mobile SentiCorr use cases, we can expect that lot of input for RBEM will come through a simple relevance feedback mechanism.

Endnotes

¹ As the field is still emerging it is hard to find a white paper summarizing objectively the current state-of-the-art in industry. However, our opinion seems to be shared among the leading industry experts in sentiment analysis, see e.g. <http://www.socialmediaexplorer.com/social-media-monitoring/sentiment-analysis/>

² See <http://www.scala-lang.org/> for more information

³ See <http://www.akka.io/> for more information

⁴ See <http://proguard.sourceforge.net/> for more information

⁵ Note that patterns can consist of any combination of words and POS-tags.

⁶ For the Dutch dataset, the agreement amongst all three annotators is a mere 55 %. The agreement between two out of three annotators varies from 65 % up to 71 %. The agreement on the English dataset is 72.1 %.

⁷ Accessible at <http://www.win.tue.nl/~mpechen/projects/smm/#Datasets>

⁸ We used the Java version mentioned on <http://www.socher.org/index.php/Main/Semi-SupervisedRecursiveAutoencodersForPredictingSentimentDistributions>

⁹ The Rotten Tomatoes dataset only contains positive and negative instances. To remedy for this, whenever RBEM labeled a message as neutral, we assigned it negative class as the most misclassified class. Note also that the consistency of the labels in this benchmark has not been studied, and the disagreement of human annotators can be very high.

¹⁰ Soap *Goede tijden, slechte tijden*, Talent shows *De beste zangers van Nederland*, Real-life show *Hotter than my daughter*, Game show *Ik hou van Holland*, other shorter names are provided in Table 4.

¹¹ We intentionally focus on demonstrating domain portability rather than multilingual or source-agnostic aspects; hence the homogeneity of our input data with respect to these two aspects.

¹² *Ik hou van Holland*

¹³ The application is publicly available at <http://www.emotiepeiler.nl/>

¹⁴ KPN is a Dutch telecom company

¹⁵ Tagalog is one of the main languages spoken in the Philippines

¹⁶ Since we are dealing with personal correspondence, a locally installed GUI is a necessity to reserve privacy

¹⁷ implementation available at <https://github.com/sancha/jrae>

¹⁸ <http://lrc.cornell.edu/swedish/dataset/affectdata/>

¹⁹ <http://www.win.tue.nl/~mpechen/projects/smm/>

²⁰ For references and publicly available information on this paradigm, we refer to <http://www.sentic.net>

²¹ See <http://datasift.com>, <http://salesforcemarketingcloud.com>, <http://coosto.nl>

²² See <http://www.sntmnt.com/>

Acknowledgements

The authors acknowledge the work of Lorraine Chambers in engineering the “mobile SentiCorr” system.

Funding

Not applicable.

Availability of data and materials

There is a patent protecting a major part of the software, and thus cannot be made available online. Datasets used in the paper are diverse and collected over a number of years. Some of the datasets will be made available in the near future, to allow reproducibility of the results.

Authors' contributions

ET is the co-designer and software developer of the SentiCorr system, MP is the co-designer of the SentiCorr system, and the academic advisor for ET, MMG is the co-designer of the mobile SentiCorr system. All the three authors have contributed to the write-up of this paper. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Author details

¹Department Computer Science, TU Eindhoven, Eindhoven, The Netherlands. ²School of Computing and Digital Technology, Birmingham City University, Curzon Street, Birmingham, UK.

Received: 21 May 2016 Accepted: 21 October 2016

Published online: 01 February 2017

References

1. Ravi K, Ravi V. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowl-Based Syst.* 2015;89:14–46.
2. Gaber MM, Cocea M, Wiratunga N, Goker A, Vol. 602. *Advances in Social Media Analysis*: Springer; 2015.
3. Tromp E, Pechenizkiy M. Graph-based n-gram language identification on short texts. In: *Proceedings of the 20th Machine Learning Conference of Belgium and The Netherlands*; 2011. p. 27–34.
4. Schmid H. Probabilistic part-of-speech tagging using decision trees. In: *International Conference on New Methods in Language Processing*; 1994.

5. Tromp E, Pechenizkiy M. Rbm: A rule based approach to polarity detection. In: Proceedings of the Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM@KDD2013). ACM; 2013.
6. Hatzivassiloglou V, McKeown K. Predicting the semantic orientation of adjectives. In: Proceedings of the ACL; 1997. p. 174–81.
7. Wilson T, Wiebe J, Hoffmann P. Recognizing contextual polarity in phrase-level sentiment analysis. In: HLT '05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing; 2005. p. 347–54.
8. Go A, Huang L, Bhayani R. Twitter Sentiment Analysis using Distant Supervision: Tech Rep. Stanford University.
9. Read J. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In: ACLstudent'05 Proceedings of the ACL Student Research Workshop; 2005. p. 43–8.
10. Socher R, Pennington J, Huang EH, Ng AY, Manning CD. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2011.
11. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, Potts C. Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Stroudsburg: Association for Computational Linguistics; 2013.
12. Pang B, Lee L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the ACL; 2005.
13. Plutchik R. A general psychoevolutionary theory of emotion. New York: Academic press; 1980, pp. 3–33.
14. Bespalov D, Qi Y, Bai B, Shokoufandeh A. Sentiment classification with supervised sequence encoder. In: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), vol. LNCS 7523. Springer; 2012. p. 159–74.
15. Glorot X, Bordes A, Bengio Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Proceedings of the 28th International Conference on Machine Learning, ICML 2011; 2011. p. 513–20.
16. Tromp E, Pechenizkiy M. Senticorr: Multilingual sentiment analysis of personal correspondence. In: Proceedings of IEEE ICDM 2011 Workshops. IEEE; 2011. p. 470–9. doi:10.1109/ICDMW.2011.152.
17. Bakker J, Holenderski L, Kocielnik R, Pechenizkiy M, Sidorova N. Stess@work: From measuring stress to its understanding, prediction and handling with personalized coaching. In: Proceedings of ACM SIGHIT International Health Informatics Symposium (IHI 2012). ACM Press; 2012. p. 673–8. doi:10.1145/2110363.2110439.
18. Bakker J, Pechenizkiy M, Sidorova N. What's your current stress level? detection of stress patterns from gsr sensor data. In: Proceedings of ICDM Workshops. 2nd HACDAIS Workshop @ ICDM 2011 (HACDAIS 2011); 2011. p. 573–80. doi:10.1109/ICDMW.2011.178.
19. Tromp E, Pechenizkiy M. Rule-based emotion detection on social media: Putting tweets on plutchik's wheel. CoRR. 2014. abs/1412.4682.
20. Yu H, Ho C, Juan Y, Lin C. LibShortText: A Library for Short-text Classification and Analysis. 2013. <http://www.csie.ntu.edu.tw/~cjlin/libshorttext/>.
21. Socher R, Pennington J, Huang EH, Ng AY, Manning CD. Semi-supervised recursive autoencoders for predicting sentiment distributions. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11). Stroudsburg: Association for Computational Linguistics; 2011. p. 151–61. <http://dl.acm.org/citation.cfm?id=2145432.2145450>.
22. Alm ECO. Affect in Text and Speech, PhD thesis. 2008.
23. Pang B, Lee L, Vaithyanathan S. Thumbs up? sentiment classification using machine learning techniques. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP'02). Association for Computational Linguistics; 2002. p. 79–86.
24. Pang B, Lee L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the ACL; 2004. p. 271–8.
25. Pang B, Lee L. Opinion mining and sentiment analysis. In: Foundations and Trends in Information Retrieval; 2008.
26. Paltoglou G, Thelwall M. Twitter, myspace, digg: Unsupervised sentiment analysis in social media, vol. 3. New York: ACM; 2012. p. 66–19. doi:10.1145/2337542.2337551, <http://doi.acm.org/10.1145/2337542.2337551>.
27. Cambria E, Olsher D, Kwok K. Sentic activation: A two-level affective common sense reasoning framework. In: Proceedings of AAAI; 2012. p. 186–92.
28. Cambria E, Song Y, Wang H, Howard N. Semantic multi-dimensional scaling for open-domain sentiment analysis. *Intell Syst IEEE*. 2013;(99):44–51. doi:10.1109/MIS.2012.118.
29. Cambria E, White B, Durrani T, Howard N. Computational intelligence for natural language processing. *IEEE Comput Intell Mag*. 2014;9(1):19–63.
30. Poria S, Cambria E, Winterstein G, Huang GB. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowl-Based Syst*. 2014;69:45–63.
31. Riloff E, Wiebe J, Wilson T. Learning subjective nouns using extraction pattern bootstrapping. In: Proceedings of the 7th Conference on Natural Language Learning; 2003. p. 25–32.
32. Wiebe J, Wilson T, Cardie C. Annotating expressions of opinions and emotions in language. language resources and evaluation. In: Language Resources and Evaluation (formerly Computers and the Humanities). Association for Computational Linguistics; 2005. p. 347–54. doi:10.3115/1220575.1220619, <http://dx.doi.org/10.3115/1220575.1220619>.
33. Wiebe J, Micalcea R. Word sense and subjectivity. In: Proceedings of ACL'06; 2006. p. 1065–72.
34. Pak A, Paroubek P. Twitter as a corpus for sentiment analysis and opinion mining. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10); 2010. p. 1320–1326.
35. O'Connor B, Balasubramanyan R, Routledge BR, Smith NA. From tweets to polls: Linking text sentiment to public opinion time series. In: Proceedings of the International AAAI Conference on Weblogs and Social Media; 2010. p. 122–9.
36. Bollen J, Mao H, Zeng XJ. Twitter mood predicts the stock market. *CoRR*. 2010. abs/1010.3003.

37. Tumasjan A, Sprenger TO, Sandner PG, Welpel IM. Predicting elections with twitter: What 140 characters reveal about political sentiment. In: Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media. p. 178–85.
38. Balasubramanyan R, Routledge BR, Smith NA. From Tweets to Polls : Linking Text Sentiment to Public Opinion Time Series. 2010.
39. Turney PD. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. ACL '02. Stroudsburg: Association for Computational Linguistics; 2002. p. 417–24. doi:10.3115/1073083.1073153, <http://dx.doi.org/10.3115/1073083.1073153>.
40. Kennedy A, Inkpen D. Sentiment classification of movie reviews using contextual valence shifters. *Comput Intell.* 2006;22:2006.
41. Zhuang L, Jing F, Zhu XY. Movie review mining and summarization. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management. CIKM '06. New York: ACM; 2006. p. 43–50. doi:10.1145/1183614.1183625, <http://doi.acm.org/10.1145/1183614.1183625>.

Submit your next manuscript to BioMed Central
and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at
www.biomedcentral.com/submit

