RBEM: A Rule Based Approach to Polarity Detection

Erik Tromp
Adversitement B.V.
Uden, the Netherlands
erik.tromp@adversitement.com

Mykola Pechenizkiy
Department of Computer Science
TU Eindhoven, the Netherlands
m.pechenizkiy@tue.nl

ABSTRACT

We propose the Rule-Based Emission Model (RBEM) algorithm for polarity detection. RBEM uses several kinds of heuristic rules to create an emissive model on polarity patterns. We extensively experiment with our approach on English and Dutch messages extracted from Twitter. Thus we also illustrate that RBEM can be used in multilingual settings and is applicable to social media characterized by use of not always regular language constructs. We demonstrate that designing such an algorithm instead of applying the state-of-the art general purpose classification techniques is a reasonable choice for the automated sentiment classification in practice. Using RBEM we can design a competitive multilingual sentiment classification system showing promising accuracy results of 78.8% on the considered datasets. We provide some further evidence that RBEM-based systems are easy to debug, improve over time and adapt to new application domains.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology

General Terms

Design, Algorithms, Performance

Keywords

rule-based polarity detection, emission model, multi-lingual sentiment analysis

1. INTRODUCTION

Sentiment analysis can be performed at different levels of granularity; the document level [19, 11], word level [7] or the sentence or phrase level [17], and with different levels of detail; determining the *polarity* of a message or the emotion expressed [14]. We perform sentiment analysis on social media in which a single message typically consists of one or two sentences. Supported by this observation, the type of granularity we study is the sentence level. We mainly focus on the polarity detection.

Thus, the problem we investigate is how to learn a functional mapping $p = f(m)$ such that given a new message m as input we can output a polarity indication $p \in \{-, +, =\}$ (being negative, positive and neutral respectively) with a high accuracy.

We propose the *Rule-Based Emission Model* (RBEM) algorithm for polarity detection and study it in the context of (multilingual) sentiment analysis on social media. The name of the algorithm indicates the concepts it carries as it uses rules to define an emissive model. Each entity in a message can emit positive or negative sentiment. The rules to determine the polarity of a text are defined on nine different types of patterns.

To show the performance of RBEM in real settings, 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 Figure 1. For language identification we employ LIGA [20].

Figure 1: Multilingual sentiment analysis.

For POS-tagging we use publicly available models for different languages [18].

This 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 domain specific knowledge of the particular source, e.g. Twitter hashtags.

We extensively experiment with RBEM for polarity detection as an isolated task and as part of the three-step approach to multi-lingual sentiment classification, showing the utility of RBEM and each preceding step by quantifying the importance of having accurate models in the processing pipeline.

Using RBEM we can design a competitive multilingual sentiment classification system showing promising accuracy results of 78.8% on the considered datasets.

We focus on highlighting the peculiarities of sentiment classification on social media data and argue that designing a focused ruled-based approach instead of applying a state-of-the art general purpose classification techniques is a reasonable choice for this application. Besides benchmark-

WISDOM'13, August 11, 2013, Chicago, USA.

ing, we conduct a case study illustrating the practical utility of RBEM and possibility to continuously improve the performance of polarity detection based on user feedback.

The rest of the paper is organized as follows. We introduce RBEM heuristics, training and classification procedures in Section 2. In Section 3 we summarize the results of the different lines of the experimental evaluation. Section 4 concludes.

2. POLARITY DETECTION WITH RBEM

The rules used in the RBEM algorithm directly stem from nine different *pattern groups*, defined as follows.[1]

- **Positive** patterns are positive when taken out of context. English examples hereof are *good, well done.*
- **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 [7]: 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 using some simple rules allows us to define an emissive model. We next describe how a model is constructed and then define how to classify previously unseen data.

[1]Note that the examples only list words but a pattern can consist of any combination of words and POS-tags. This concept is further explained when we describe how to learn a model.

2.1 Learning RBEM

Each message m of length n is represented as a list $m = [(w_1, t_1), .., (w_n, t_n)]$ 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 [24]. 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.

2.2 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 $q = (g, p) \in \mathit{Model}$ is matched against our message $h = [(w_1, t_1), .., (w_n, t_n)]$ where $p = [(v_1, s_1), .., (v_m, s_m)]$. 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:

$$(v_j, s_j) =_{match} (w_i, t_i) \equiv$$

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

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.

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

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.

$$maxPatterns = \{(s, e, q) | (s, e, q) \in matchedPatterns \wedge \neg(\exists_{(s', e', q') \in matchedPatterns} : s \leq s' \wedge e' \geq e \wedge \neg(s = s' \wedge e = e') \wedge q \neq q')\}$$

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 $(Model, \mathcal{E})$ 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:

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

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:

$$stops = stops \setminus \{t | t \in stops \wedge (\exists_{(s,f,continuator) \in maxPatterns} : t \leq s \wedge s - t < \mathcal{E} \wedge \neg(\exists_{t' \in stops} : t < t' \leq s))\}$$

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:

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

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:

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

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:

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

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 $1-e^{-x}$:

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

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:

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

Rule 8. Left Flipping Sentiment – Left flip patterns mirror the effect of right flip patterns:

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

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:

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

Straightforwardly, we say that m is *positive* (class +) if and only if $polarity > 0$. Likewise, we say that m is *negative* (class −) if and only if $polarity < 0$. Whenever $polarity = 0$, 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.

2.3 Related work

Polarity detection has been studied in different communities and in different application domains. The polarity of adjectives was studied in [7] with the use of different conjunctive words. A comprehensive overview of the performance of different machine leaning approaches on polarity detection were presented in [13, 11, 12]. Typically, polarity detection is solved using supervised learning methods but more recently attention is being paid to unsupervised approaches [10].

Some of the recent works adopt a concept-level approach to sentiment analysis [4], which leverages on common sense knowledge for deconstructing natural language text into sentiments. A notable example is [3], 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.

Other works are those of [17, 24, 23, 22]. 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.

The idea of using patterns arises from [24] who label subjective expressions (patterns) in their training data. Nevertheless, in their experiments they limit themselves to matching against single-word expressions. The use of rules stems from a different domain. The Brill tagger [2] for POS-tagging uses rules. We borrow this ideology and apply it to polarity detection. The emission aspect of our RBEM algorithm is related to smoothing which is often applied in different machine learning settings. RBEM has also close resemblance to [16] where different rules and patterns are defined on the top of a full linguistic parser output.

More recently attention is being paid to sentiment analysis on social media. Sentiment analysis on Twitter is researched by [6, 9] who use similar methodologies to construct corpora and analyze Twitter messages to determine their polarity. [8] 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.

3. EXPERIMENTAL EVALUATION

The goal of our experimental study is three-fold: 1) to benchmark the performance of the proposed RBEM comparing it against popular classification approaches for polarity detection, 2) to study the multilingual settings and the effect of language identification, and 3) to study the portability of RBEM to different application domains.

3.1 Datasets

The training set for our polarity detection algorithm contains messages in multiple languages, multiple sentiments and multiple domains, stemming from social media. The methodology we use to collect data covering different sentiments is similar to [6, 15], 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. From this mixture of polar and non-polar messages we extract patterns for RBEM by manual labeling using a custom web interface that allowed to do this in a quick manner.

For polarity detection we train language specific models. Moreover, the models use POS tags as features known beforehand. LIGA is used to filter out those messages that are neither in Dutch nor in English when querying Twitter with smileys. (LIGA was trained on the other benchmark we created earlier [20]). All the resulting messages are processed by the POS-tagger to form the extended representation containing the parts of speech features.

To construct the 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 polar-

Table 1: The sizes of the training/test sets.

Training set/test set size		
	English	Dutch
Positive	3614/205	1202/262
Negative	3458/200	1504/200
Neutral	4706/454	2099/595
Total	11778/859	4805/1057

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

ity, being either one of positive, negative or neutral. Finally, we extracted numerous RBEM patterns from each message.

The labeling of 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 polarity[2].

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

For studying RBEM portability we conducted a case study with additional datasets which we discuss in the corresponding subsection.

3.2 RBEM Accuracy

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 accuracies are given in Table 4.

Even though the accuracy of the SVM approach is close to that of the Naive Bayes approach, the SVM has much higher recall whereas the precisions are also comparable. An SVM approach using all features and patterns performs best among the experimented environments, which was also the case for subjectivity detection. Thus SVM outperforms the Naive Bayes on this dataset.

Table 4 only lists using POS-tags for AdaBoost as we found this to be the best feature set for this approach. Using 50 weak learners yields an accuracy of 72.3%, the best accuracy among the settings and features we experimented with for AdaBoost.

The performance of the Prior Polarity Classifiers (Prior) is better than that of SVM and Naive Bayes and close to the performance of the AdaBoost approach. The SentiWordNet variant (SWN) is the most extensive and performs better in terms of precision and overall accuracy.

The left half of the mid section of Table 4 shows the performance of RBEM. Even when we count the misclassification of polar messages we miss out due to insufficient labeling, the accuracy is already comparable to the highest accuracy for AdaBoost and higher than that of the prior polarity classifiers. When we disregard polar messages for which no patterns are present in our model, we obtain a much higher accuracy of 83.9%.

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 increase 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. The scores for our metrics we then get for the RBEM algorithm are shown in Table 3. The percentage of polar messages that we do not manage to find due to insufficient labeling drops from 12.9% to 9.4%. The accuracy increases 72.4% to 74.1% when taking all messages into account. When we leave out the messages we cannot find due to insufficient labeling, our accuracy increases from 83.9% to 84.2%, indicating that the newly labeled patterns not only allow us to classify those messages we could not classify previously but also help correct the classification of messages that we misclassified previously.

Table 3: The performance of RBEM with 81 more patterns.

With Missed	A	0.741
	P	0.733
	R	0.876
Without Missed	A	0.842
	P	0.832
	R	0.955
Missed %		9.4%

3.3 Three-step Process Evaluation

Since we study multilingual settings, we also want to measure the impact of language identification (accuracy) on po-

[2]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%.

Table 4: The overall accuracy (A), precision (P) and recall (R) of algorithms for polarity detection. The instances per algorithm having the highest F-measure are shown in bold. The 'Missed %' row lists the fraction of subjective texts not found as such due to insufficient labeling.

Naive Bayes	All	MI 2000	MI 4000	MI 8000
Tokens	**A 0.616** **P 0.551** **R 0.393**	A 0.504 P 0.459 R 0.297	A 0.506 P 0.460 R 0.301	A 0.511 P 0.467 R 0.313
Tags	A 0.585 P 0.543 R 0.386	A 0.491 P 0.439 R 0.281	A 0.493 P 0.440 R 0.287	A 0.482 P 0.427 R 0.259
Mixture	A 0.589 P 0.546 R 0.387	A 0.495 P 0.443 R 0.286	A 0.494 P 0.441 R 0.287	A 0.502 P 0.453 R 0.292
Patterns	A 0.545 P 0.497 R 0.338	A 0.502 P 0.419 R 0.313	A 0.489 P 0.447 R 0.302	A 0.510 P 0.428 R 0.326

SVM	All	MI 2000	MI 4000	MI 8000
Tokens	A 0.656 P 0.795 R 0.431	A 0.504 P 0.500 R 0.013	A 0.507 P 0.688 R 0.023	A 0.420 P 0.675 R 0.478
Tags	A 0.451 P 0.559 R 0.532	A 0.551 P 0.549 R 0.490	A 0.531 P 0.523 R 0.473	A 0.544 P 0.533 R 0.452
Mixture	A 0.654 P 0.699 R 0.486	A 0.540 P 0.542 R 0.496	A 0.547 P 0.557 R 0.499	A 0.564 P 0.555 R 0.464
Patterns	**A 0.637** **P 0.646** **R 0.564**	A 0.500 P 0.500 R 0.131	A 0.503 P 0.542 R 0.261	A 0.514 P 0.575 R 0.279

	RBEM	Prior, Tok	Prior, Tok+Tags	Prior, SWN
With missed	A 0.724 P 0.719 R 0.862	A 0.602 P 0.611 R 0.583	A 0.552 P 0.577 R 0.550	A 0.681 P 0.737 R 0.498
Without missed	**A 0.839** **P 0.828** **R 0.953**	**A 0.769** **P 0.785** **R 0.747**	A 0.778 P 0.831 R 0.664	A 0.687 P 0.741 R 0.712
Missed %	12.9%	21.4%	28.7%	0.9%

AdaBoost	25 models	50 models	75 models	100 models	250 models
Tokens	A 0.664 P 0.667 R 0.638	A 0.691 P 0.707 R 0.654	A 0.685 P 0.692 R 0.649	A 0.678 P 0.686 R 0.644	A 0.698 P 0.706 R 0.658
Tags	A 0.699 P 0.704 R 0.668	**A 0.723** **P 0.729** **R 0.691**	A 0.715 P 0.722 R 0.685	A 0.703 P 0.709 R 0.677	A 0.691 P 0.697 R 0.666

Table 5: The correctness scores after each step (vertical) for leaving out each possible step (horizontal).

	No LI	No POS	**All included**
Acc. of LI	–	0.971	0.971
Acc. of Pol	0.782	0.795	0.839
Acc. of Complete	0.782	0.778	**0.788**

larity detection.

- **Leaving out Language Identification.** When we do not know the language of a message, we can no longer use language-specific models and hence need to apply more generic models. We thus need to combine the language-specific models into one. For POS-tagging we simply apply all models that are present and use the model showing the highest probabilities of being correct. For polarity detection we apply both the English as well as the Dutch model on the message, sum up the scores for both languages and assign the resulting class.

- **Leaving out POS-tagging.** Polarity detection uses POS-tags as features. When we do not have these tags we need to resort to using tokens only. For polarity detection with RBEM, we use our patterns without regarding the POS-tags.

Table 5 shows the accuracies on the test set for all different scenarios. The columns list the steps left out whereas the rows list the accuracies after each step. The accuracy for a single step is computed by dividing the number of messages correctly classified by that step by the number of messages correctly classified up to that step. The last row indicates the proportion of messages that are polar but which our polarity detection step could not classify as such due to insufficiently labeled data. The *complete* row lists the accuracies on the entire test set, computed by dividing the number of messages correctly classified by all steps by the corpus' size. As POS-tagging is merely used to expand our feature space, we do not evaluate the accuracy after performing this step.

As we compare Table 5 column by column, we observe that the highest accuracy is obtained when all three steps are included. This indicates the importance of each of them.

3.4 RBEM Domain Portability

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. 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 [1] or deep learning neural networks [5] 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 Section 3.1 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 broadcasted[3].

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 [20] 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.[4] 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 6. 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 aid 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 7 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

[3]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 6.

[4]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.

Table 6: 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
Experiment 1		
TV show *Babyboom*	#babyboom	226
Talent shows	#bestezangers	199
TV series *Hitch*	#hitch	305
Real-life show	#htmd	969
TV series *House*	#house	199
Game show [5]	#ihvh	772
Experiment 2		
Soap	goede tijden	2465
Soap	goedetijden	432
Soap	gtst	4013
Venue of soap	meerdijk	232

Table 7: Best scoring patterns found in both experiments. G - pattern group (positive +, negative − and neutral =), O - operation (add +, remove -) and #C - number of corrections.

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

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 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*.

4. CONCLUSIONS

Previous work in the area of sentiment analysis traditionally focused on benchmarking performance of sentiment classification techniques, typically on one language only, usually English as the resources for English are best available. In this paper we introduced a new rule-based approach for polarity detection and investigated its competitive advantages.

The RBEM algorithm provides a solid foundation that is easily extended. Adding more patterns and rules that further increase its accuracy is straightforward when the relation with the other rules is analyzed. Enriching the model via the relevance feedback from the user is also feasible and automating this process is one of the directions of our further work. We will explore methods to find patterns in an automated fashion rather than through a manual labeling process.

Even though many of existing approaches for sentiment analysis can be extended to support multiple languages, this is not a trivial task and typically not included in the studies themselves. We demonstrated the potential of RBEM to be used in a multilingual solution to sentiment analysis by taking both English and Dutch into account.

We targeted multilingual short texts typically present in social media. However, our approach is applicable to sentiment classification in other settings as well.

For our experimental study we constructed two datasets. The training set was constructed by querying Twitter with smileys and scraping news accounts. This yields messages with noisy labels rather than accurate labels. Moreover, this training data is biased as it only contains messages in which smileys are originally present. Constructing more accurate and more representative training data is expected to increase the accuracy of sentiment classification. Additionally, data can be collected for each social medium separately, allowing for more specific, social medium-tailored, models.

In our experimental study we showed the importance of each step, justifying our three-step approach rather than a more generic approach comprising fewer steps. If the sentiment analysis on social media is used for personal use (as envisioned e.g. in [21]) rather than for marketing we can expect that lot of input for RBEM will come through the simple relevance feedback mechanism.

5. REFERENCES

[1] D. Bespalov, Y. Qi, B. Bai, and A. Shokoufandeh. 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)*, volume LNCS 7523, pages 159–174. Springer, 2012.

[2] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied*

natural language processing (ANCL'92), pages 152–155. Association for Computational Linguistics, 1992.

[3] E. Cambria, D. Olsher, and K. Kwok. Sentic activation: A two-level affective common sense reasoning framework. In *Proceedings of AAAI*, pages 186–192, 2012.

[4] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New avenues in opinion mining and sentiment analysis. *Intelligent Systems, IEEE*, 28(2):15–21, 2013.

[5] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 513–520, 2011.

[6] A. Go, L. Huang, and R. Bhayani. Twitter sentiment analysis using distant supervision.

[7] V. Hatzivassiloglou and K. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the ACL*, pages 174–181, 1997.

[8] 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.

[9] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, pages 1320–1326, 2010.

[10] G. Paltoglou and M. Thelwall. Twitter, myspace, digg: Unsupervised sentiment analysis in social media. volume 3, pages 66:1–66:19, New York, NY, USA, 2012. ACM.

[11] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278, 2004.

[12] B. Pang and L. Lee. Opinion mining and sentiment analysis. In *Foundations and Trends in Information Retrieval*, 2008.

[13] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of Conference on Empirical methods in natural language processing (EMNLP'02)*, pages 79–86. Association for Computational Linguistics, 2002.

[14] D. Potena and C. Diamantini. Mining opinions on the basis of their affectivity. In *2010 International Symposium on Collaborative Technologies and Systems (CTS)*, pages 245–254, 2010.

[15] J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *ACLstudent'05 Proceedings of the ACL Student Research Workshop*, pages 43–48, 2005.

[16] R. Remus and C. Hänig. *Towards Well-grounded Phrase-level Polarity Analysis*, pages 380–392. Springer, 2011.

[17] E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 25–32, 2003.

[18] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, 1994.

[19] V. Sindhwani and P. Melville. Document-word co-regularization for semi-supervised sentiment analysis. In *Eighth IEEE International Conference on Data Mining (ICDM'08)*, pages 1025–1030, 2008.

[20] E. Tromp and M. Pechenizkiy. Graph-based n-gram language identification on short texts. In *Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands*, pages 27–34, 2011.

[21] E. Tromp and M. Pechenizkiy. Senticorr: Multilingual sentiment analysis of personal correspondence. In *Proceedings of IEEE ICDM 2011 Workshops*, pages 470–479. IEEE, 2011.

[22] J. Wiebe and R. Micalcea. Word sense and subjectivity. In *Proceedings of ACL'06*, page 1065–1072, 2006.

[23] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. language resources and evaluation. In *Language Resources and Evaluation (formerly Computers and the Humanities)*, pages 347–354. Association for Computational Linguistics, 2005.

[24] T. Wilson, J. Wiebe, and P. Hoffmann. 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*, pages 347–354, 2005.