Noname manuscript No.
(will be inserted by the editor)

Incremental word-vectors for time-evolving sentiment lexicon induction

Felipe Bravo-Marquez · Arun Khanchandani · Bernhard Pfahringer

Received: date / Accepted: date

Abstract Background A sentiment lexicon is a list of expressions annotated according to affect categories such as positive, negative, anger and fear. Lexicons are widely used in sentiment classification of tweets, especially when labeled messages are scarce. Sentiment lexicons are prone to obsolescence due to: 1) the arrival of new sentiment-conveying expressions such as #trumpwall and #PrayForParis, and 2) temporal changes in sentiment patterns of words (e.g., a scandal associated with an entity).
Methods In this paper we propose a methodology for automatically inducing continuously updated sentiment lexicons from Twitter streams by training incremental word sentiment classifiers from time-evolving distributional word vectors. We experiment with various sketching techniques for efficiently building incremental word-context matrices and study how the lexicon adapts to drastic changes in the sentiment pattern. Change is simulated by randomly picking some words from a testing partition of words and swapping their context with the context of words exhibiting the opposite sentiment.
Results and Conclusions Our experimental results show that our approach allows for successfully tracking of the sentiment of words over time even when drastic change is induced.

Keywords Lexicon induction · incremental word vectors · sentiment analysis

Felipe Bravo-Marquez (Corresponding author)
Department of Computer Science, University of Chile
Millennium Institute for Foundational Research on Data, IMFD-Chile
E-mail: fbravo@dcc.uchile.cl

Arun Khanchandani · Bernhard Pfahringer
Department of Computer Science, University of Waikato
E-mail: {ak293,bernhard}@waikato.ac.nz

1 Introduction

Sentic computing is an interdisciplinary field for the study of sentiment and affect in human language that complements modern statistical and neural techniques for natural language processing (NLP) with other related disciplines such as linguistics, common sense reasoning and affective computing [16]. SenticNet [18] is a lexical resource that broadly reflects the spirit of this field. The core of this resource is a semantic network of more than 200,000 common-sense concepts, represented by both single words and multi-word expressions with rich semantic and affective associations. This resource enables the integration of logical reasoning into modern deep learning models to extract complex affective/sentic patterns from texts [17].

However, static sentic resources such as SenticNet exhibit significant limitations for their application to social media such as Twitter. First, these resources unlikely capture the diversity of informal expressions found in Twitter, including misspelled words, acronyms, or hashtags. Furthermore, with novel events often come novel affect-conveying expressions, such as #NeverAgain or #LetsStopTheFlood. A static resource such as SenticNet can only ignore these potentially highly informative new expressions. Additionally, other existing words or expressions can change their sentiment abruptly, when unexpected events associated with an entity occur suddenly (e.g., a scandal linked to a public figure or company) [4].

This paper expands on the field of sentic computing to a dynamic time-evolving setting, proposing a new methodology for building continuously updated sentic resources from Twitter streams. In particular, we focus on *time-evolving sentiment lexicons*. A sentiment lexicon is a list of words and expressions annotated according to affective categories (e.g., "good" and "love" are examples of positive words; "hate" and "fed up" are example of negative ones). We define a time-evolving sentiment lexicon as a dynamic resource that maps words occurring in a stream of text or a diachronic corpus to sentiment or affective categories. In this work we only consider binary prediction problems: categories limited to only a positive and a negative class. Though as discussed in Section 5, our approach could easily be adapted to work with fine-grained emotions such as the Hourglass of emotions [61]. Every time a new word is observed in the stream (i.e., a word that has not been included in the lexicon up to that moment), this new word will be assigned a sentiment score and added to the lexicon. Moreover, every time an existing word is observed in the stream (i.e., a word already added to the lexicon) its sentiment score can be updated. In our opinion, a sentiment lexicon should be the basic building block of a more complex sentic computing resource, and hence, a method for building lexicons incrementally essentially lays the foundation for a dynamic version of the sentic computing field.

Our proposed methodology allows us to build and evaluate time-evolving sentiment lexicons from Twitter streams based on two resources that are relatively cheap to obtain: 1) a stream of unlabeled tweets that can be freely obtained from the Twitter API, and 2) a static seed lexicon, which is a small group of words annotated by sentiment. In this work we make the assumption that the sentiment of the seed words would not change over time.

The main idea of our proposal is to adapt the static approach for inducing Twitter-specific sentiment lexicons proposed in [10,62] to a time-evolving setting. In those

approaches, words from a corpus of unlabeled tweets are represented as distributional vectors [63] and a word-level sentiment classifier is trained on those vectors [10,62]. The training examples are obtained by labeling the word vectors matching the words of a given seed lexicon. The induction is performed by deploying the resulting classifier on the remaining words. In contrast to a previous approach for building time-evolving lexicons [28], in which independent lexicons are created for different periods of time, our approach is fully incremental and based on stream mining methods.

To adapt the static approach described above to a time-evolving setting, our methodology performs two actions in a stream of continuously arriving tweets: 1) calculate incremental distributional representations of words, and 2) train an incremental word-level sentiment classifier.

Distributional vectors or word embeddings represent lexical items such as words according to the context in which they occur in a corpus of documents. These models infer the meaning of a word from the distribution of the words that surround it. They are based on the distributional hypothesis of meaning, which states that words that occur in similar contexts tend to have similar meanings [30]. There are essentially two main approaches for building word vectors: 1) count-based approaches that explicitly build a word-context matrix of word-word co-occurrence counts [63], and 2) distributed methods that rely on the the internal structure of a neural network trained on an auxiliary predictive task (e.g., predicting the center word from a context window) [51]. Theoretical studies have proven that these two approaches are equivalent [43].

Adapting distributional models to the stream scenario is challenging because satisfactory static models rely on operations that require access to the whole dataset [44] (e.g., extracting the vocabulary from the corpus, discarding infrequent words). Moreover, the neural networks employed in distributed approaches require various passes over the data to properly fit internal parameters of the network. All these operations are computationally prohibitive in a streaming setting. In order to satisfy the stream mining constraints, namely that tweets can only be observed once and cannot be stored in memory, we build our incremental word vectors using count-based approaches. This is because count-based approaches can be more easily adapted to a stream mining framework than neural networks. This might sound counter-intuitive, as neural networks seem to naturally learn incrementally, but in practice they need many incremental passes over the same data to achieve good performance.

Each dimension of our word vectors corresponds to a context, which is usually represented as another word that surrounds the target word in a window. These dimensions have numerical values that represent an association between words. This association is calculated using Positive Point-wise Mutual Information (PPMI) [63, 19], which depends on the joint probabilities of word pairs and the marginal probabilities of single words. Therefore, we need to store word counts and word pair counts in memory. Word vectors are created and updated continuously as new tweets arrive from the stream.

Data sketching methods [6] are stream mining techniques that summarize sufficient information from a stream to calculate approximate statistics with significant gains in time and memory consumption and minor losses in accuracy. We experi-

ment with two sketching techniques for approximating word-context co-occurrence counts: Jenkins Hashing [36] and Brown clusters [15].

The dynamic lexicon is built by training a linear classifier on the word vectors. An incremental classifier can evolve and adapt to new training instances as the stream of data flows. Also, incremental classifiers generally scale much better than a batch learning approach for training and testing on large amounts of data. The linear classifier is trained incrementally using stochastic gradient descent (SGD) [9] minimizing logistic loss. The word vector's dimensions are used to form the attribute space of the classifier and the words from the seed lexicon are used to label the training instances. Each time a word is observed in the stream, its vector is updated. If the word is part of the seed lexicon, then the incremental classifier is trained/updated using the corresponding vector and label, otherwise the word is reclassified and its entry in the dynamic lexicon is updated using this newly predicted sentiment score.

The research hypothesis of this work is that the Twitter signal is informative enough to accurately track the sentiment of words over time without human intervention. We postulate that as a consequence of the distributional hypothesis of meaning, the sentiment of new words (e.g., hashtags) will be captured by its context vector (after having seen the word a certain number of times), and hence will be classified correctly [13]. Furthermore, for words exhibiting sentiment drift (e.g., a company name involved in a scandal), their word vectors will change and be reclassified to their new sentiment at some point.

There is no obvious strategy to create a benchmark dataset in which sentiment drift of words can be clearly observed and evaluated[1]. Therefore, we propose a mechanism for generating synthetic data in which this condition can be manipulated. The synthetic dataset is used to evaluate the lexicon induction process in scenarios of sentiment drift. Our sentiment drift generator works by randomly picking some words from a testing partition of the seed lexicon and swapping their context with the context of words exhibiting the opposite sentiment.

The main contributions of this paper are:

1. A new methodology for training incremental sentiment lexicons from Twitter streams using incremental word vectors and incremental classifiers.
2. A new technique for evaluating incremental lexicons under scenarios of drastic sentiment drift based on creating synthetic sentences.

This article is organized as follows. In Section 2, we provide a review of related work. The proposed methodology for building and evaluating incremental sentiment lexicons is provided in Section 3. In Section 4, we present the experiments we conducted to evaluate the proposed approach and discuss results. The main findings and conclusions are discussed in Section 5.

2 Related Work

This section discusses related work in four subsections. The first part reviews sentiment in a time-evolving setting from a general point of view. The second part is more

[1] We have to keep in mind that we are assuming the training words do not change their sentiment over time.

specific and discusses works that build temporal lexicons as well as works studying the semantic change of words. The third subsection tries to connect the dots between our proposed approach and the sentic computing paradigm. Finally, the fourth subsection provides a discussion on why the proposed method is novel and different from previous approaches.

2.1 Time-evolving Sentiment Models

The first study of sentiment from a stream data mining point of view was [4], where three fast incremental learning algorithms for tweet-level sentiment classification were compared: 1) Multinomial Naive Bayes, 2) Stochastic Gradient Descent (SGD) linear classifer, and 3) the Hoeffding Tree. Tweets were represented using unigram features and automatically annotated using positive and negative emoticons. This annotation approach is usually referred to as distant supervision [26]. The Massive Online Analysis (MOA) framework [7] was used for the experiments using prequential accuracy and the kappa statistic for evaluation. The authors argued that the kappa measure is more suitable for unbalanced streams. The results indicated that on the one hand, Hoeffding trees are not suitable for high-dimensional streams, and on the other, an SGD-trained linear model and Naive Bayes perform comparably for this problem. A very strong assumption made by this model is that sentiment labels are available across the entire stream. The authors explored the variation of the SGD coefficients associated with some words during the training process. The model's coefficients determine the influence of the presence of a word on the prediction of negative and positive classes. The argument was that tracking these variations can be an efficient way of detecting changes in the population's opinion of an entity (e.g., a topic, a person).

Bifet et al. developed MOA-TweetReader in [8] as an extension of the MOA framework. This extension allows users to read tweets in real time, store the frequency of the most frequent terms, detect change in the frequency of words, and perform sentiment analysis in the same way as the aforementioned work.

Apart from using emoticons in a distant supervision fashion, emoticons have also been incorporated into existing lexicons yielding significant improvements in classification accuracy [33].

In [35] the temporal dynamics of tweets associated with online retailers in the UK is analyzed using three types of techniques: 1) sentiment analysis, 2) time series analysis, and 3) topic modeling. Sentiment analysis is used to classify tweets to sentiment categories, time series analysis techniques are performed to explain sentiment variations of these tweets over time, and finally topic models are used to identify the topics that caused these deviations.

One of the main reasons for developing time-evolving sentiment models is the quality deterioration of static sentiment classifiers over time. This problem was empirically studied in [22]. Sentiment classifiers were trained using training and testing data from different time periods. The results indicated a significant decrease in the classification performance as the time difference between the training and the testing data was increased. There are three approaches to this problem in [57]: 1) using a

weighing scheme for continuous vocabulary updating, 2) adding sentiment lexicons to the feature space, and 3) using distributed word vectors. The results show that these approaches can successfully reduce the deterioration of sentiment classification results over time.

2.2 Temporal Lexicons and Semantic Change Detection

A model for creating domain-specific opinion lexicons is proposed in [28], where words from a source corpus of unlabeled text data (not necessarily tweets) are represented by embedding vectors. A square matrix of size $V \times V$ (V is the vocabulary size), is built where each entry (i, j) corresponds to a smoothed PMI score of the word pair w_i,w_j based on co-occurrence counts within fixed-size sliding windows of text. This matrix is projected onto a low-dimensional space using singular value decomposition (SVD) and used for building a graph of word associations. Each word in the graph is connected with its k most similar words according to cosine similarity in the low-dimensional space. The sentiment induction is carried out by propagating the known polarities of a seed lexicon to the remaining nodes in the graph using random walks. The model is used for creating domain-specific sentiment lexicons for different *Reddit* communities. Examples exhibiting different polarities in different communities are the words "soft" and "animal", which are positive in a community dedicated to female perspectives and gender issues, but negative in sports. Conversely, the words "crazy" and "insane" exhibit contradictory polarities in both domains. The same approach was also used for studying the evolution of opinion words by building lexicons from documents from the Corpus of Historical American English[2] written in consecutive decades between 1850 to 2000. The authors found that several words have changed their polarity over time, for instance the word "terrific" has changed from a negative to a positive polarity in the last few decades.

Another approach to creating domain and time-specific lexicons is proposed in [27]. This work argues that manually created lexicons cannot be readily applied to a specific domain or time period because they ignore domain-specific sentiment terms. Their approach consists of extracting relevant terms associated with the target domain or time period from news sources. The sentiment of each term is determined by collecting tweets containing the term and averaging their sentiment. The sentiment of these tweets is obtained using an ensemble of various sentiment analysis systems.

Another problem that is closely related to incremental sentiment lexicon induction is the tracking of temporal changes in word meaning or the study of "semantic shifts". The relationship between these two problems comes from the observation that sentiment can be viewed as a sub-dimension of semantics [14].

A pioneering work tracking semantic change using word embedding models was carried by out Kim et at. [39]. The approach consists of training year-specific neural word vectors from the Google Books Ngram corpus between year 1900 to 2009 and to identify words whose meanings have changed significantly. The word vectors of each year are initialized with the vectors of the previous one.

[2] `http://corpus.byu.edu/coha/`

Hamilton et at. [29] conducted a similar study for quantifying semantic change over time by training word vectors on six historical corpora. The study revealed two statistical laws of semantic evolution. The first one is called the law of conformity and suggests that the rate of semantic change scales with an inverse power-law of word frequency (i.e., frequent words are less likely to change). The second one, called the law of innovation, suggests that words that are more polysemous have higher rates of semantic change regardless of their frequencies.

A problem with comparing word vectors trained on different time periods is that those vectors are not necessarily compatible. This is a consequence of the stochasticity of neural word vectors: neural embedding models are likely to produce different vectors on different runs even if they are trained on the same data. This situation strengthens for models trained on different corpora (e.g., corpora from different periods of time) [42].

Kulkarni et al. [41] suggested aligning embedding models from different periods using linear transformations in order to calculate similarities between words in different periods. However, this operation is computationally prohibitive in a stream mining setting.

There have been some attempts to build incremental word vectors using neural networks [38], [49], but none of these methods has been studied in the context of incremental sentiment lexicons.

The semantic shift detection problem should be evaluated on ground-truth data comprised of human-annotated semantically shifted words. However, such a resource is rarely available, and many evaluations have been conducted on small datasets formed by a handful of examples [42]. This is not sufficient for evaluating large-scale semantic drift models.

The idea of creating synthetic data for evaluation purposes was explored by Rosenfeld and Erk [56]. Their approach creates synthetic words by merging two real words. In the merging process the meaning of the synthetic word shifts from the meaning one of the source words to the other.

We refer the reader to [42] for a comprehensive review of the state of academic research related to diachronic word embeddings and semantic shift detection.

2.3 Sentic Computing

The *sentic computing* paradigm focuses on a semantic-preserving representation of natural language concepts and on sentence structure [16]. Many dimensions of the sentiment analysis problem can be studied through the eyes of this paradigm, including the study of negation [64,32], the study of sentiment with respect to the entities mentioned within a sentence and its corresponding aspects [48,58,47], or multilingual approaches [21,34].

In relation to the process of automatically inducing sentiment lexicons or other lexical resources for sentiment analysis, apart from the works discussed above that use distributional word vectors, there are alternative approaches based on semantic

networks and knowledge-graphs such as WordNet [52] and ConceptNet[3] to be discussed below.

In [24], a supervised classifier is trained using a seed of labeled words that is obtained through expansion based on synonyms and antonyms. For each word, a vector space model is created from the definition or *gloss* provided by the WordNet dictionary. This representation is used to train a word-level classifier that is used for lexicon induction. An equivalent approach was applied later to create SentiWordNet[4] [2,25]. In SentiWordNet, each WordNet *synset* or group of synonyms is assigned to classes *positive*, *negative* and *neutral*, with soft labels in the range $[0,1]$. In [31], various techniques are explored to exploit the semantic relationships of existing lexical resources for automatic sentiment lexicon creation. These techniques include a PageRank-based method and a machine learning approach.

SenticNet[5] is a well-known commonsense lexical resource for sentiment analysis built from concept-level semantic networks. In this resource, multi-word concepts are labeled according to both affective and semantic information. Various techniques have been used in the different versions of SenticNet. The first two versions were based on graph-mining and dimensionality reduction techniques, while most recent versions integrate multiple knowledge sources establishing paths between concepts and make use of context embedding models [18]. The sixth and latest version of SenticNet integrates logical reasoning within modern deep learning architectures via an ensemble of symbolic and subsymbolic AI tools [17]. A salient example of how deep learning architectures can be integrated with commonsense knowledge, is the Sentic LSTM [47], which extends the LSTM cell to include a separate output gate that interpolates token-level memory with concept-level input.

2.4 Discussion

In this section, we reviewed several techniques for sentiment analysis, with particular attention to methods for inducing sentiment lexicons and tracking sentiment change over time. To the best of our knowledge, this is the first work in which an incremental word-sentiment classifier is trained on time-evolving word-vectors for building incremental lexicons. Moreover, this is also the first work in which large-scale synthetic data is created for evaluating the detection of word-level sentiment drift.

3 Proposed Methodology

The sentiment of words on social media sites like Twitter can change dramatically. This fact was highlighted in a study centered on the dynamic nature of words [60]. The study shows the case of the word “ukrop” as an example, which changed its meaning from “dill” to “Ukrainian patriot” during the Russian-Ukrainian crisis, developing a more negative connotation in time.

[3] `http://conceptnet5.media.mit.edu/`

[4] `http://sentiwordnet.isti.cnr.it/`

[5] `http://sentic.net/`

We postulate that in order to efficiently work with these types of changes, a dynamic lexicon must adapt quickly to identify changes in distribution using an incremental approach. This is the basis of our proposed methodology for building time-evolving sentiment lexicons from a stream of tweets, which is described in detail in this section. The proposed process is illustrated in Figure 1, and summarized in the following steps:

1. Connect to a stream of continuously arriving text (e.g., Twitter).
2. Every time a new tweet arrives, a sliding window of W words centered on a target word is shifted across the content.
3. The word vector associated with the target word is updated according to its context. This process implies updating word-context counts and computing PPMI-based associations.
4. If the target word is new, a new vector associated with this word is created.
5. If the target word is contained in the seed lexicon, the incremental classifier is updated/trained using the target word vector and the lexicon's sentiment as the gold label.
6. If the target word is not contained in the lexicon, its sentiment is estimated using the classifier and the dynamic lexicon is updated.

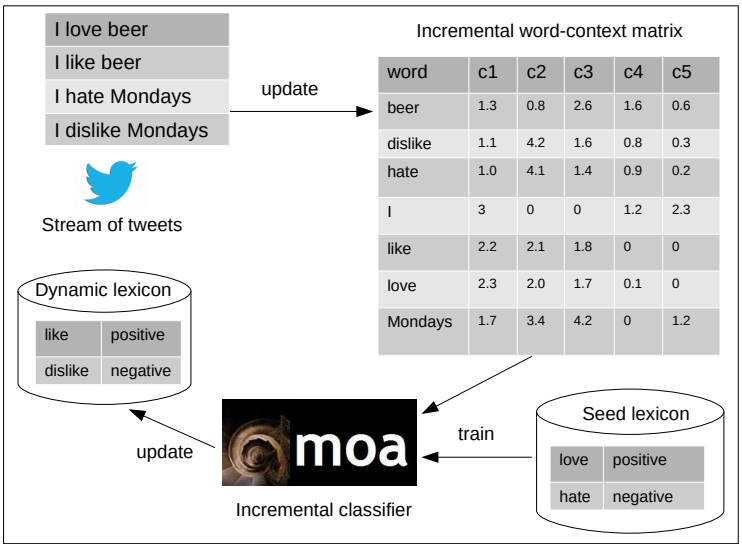


Fig. 1 Dynamic lexicon induction process. The incremental classifier is illustrated with the logo of MOA [7], a stream mining software. The example shows two pairs of words that occur in the same contexts (love, like) and (hate, dislike) receiving the same word vectors. However, only one word in each pair belongs to the seed lexicon. Consequently, the two unknown words are classified with the lexicon's sentiment of their counterparts.

The key parts of the methodology are described in the following subsections. The mechanisms studied to build time-evolving word vectors are described in Section 3.1.

The incremental word-level sentiment classifier is described in Section 3.2. The evaluation framework is described in Section 3.3.

The MOA[7] framework comprises of a large collection of tools for the analysis of online and offline data streams. MOA's Java API was used for implementing the word vectors, the incremental classifier and the various evaluation methods.

3.1 Time-evolving word-vectors

The Distributional Hypothesis [30] states that words occurring in similar contexts tend to have similar meanings. This hypothesis is exploited in our method for inducing time-evolving sentiment lexicons. To this end, words are represented by incremental distributional vectors that are projected onto a sentiment space using an incremental classifier. Distributional vectors [63] are used for representing lexical items such as words according to the context in which they occur in a corpus of documents or tweets. In other words, distributional models infer the meaning of a word from the distribution of the words that surround it.

The most common approach for building distributional vectors of words in a static setting is the word-context matrix [63]. This matrix has dimensionality $V \times C$ (V and C are the number of different words and contexts), and each cell (i, j) is a co-occurrence based association value between a target word w_i and a context c_j calculated from a corpus of documents. Contexts are usually represented by the words surrounding the target word in a window. The window length is a user-specified parameter that is usually between 1 and 8 words on both the left and the right sides of the target word, i.e., the total contexts are usually between 3 and 17 words. Whereas shorter windows are likely to capture syntactic information, longer windows are more useful for representing meaning [37]. The associations between words and contexts can be calculated using different approaches such as: co-occurrence counts, positive point-wise mutual information (PPMI), and the significance values of a paired t-test. According to [37] the most commonly used measure of these three is PPMI. This measure is a filtered version of the traditional PMI measure in which negative values are set to zero.

PMI alone calculates the log of the probability of word-context pairs occurring together over the probability of them being independent:

$$PMI(w,c) = \log_2 \left(\frac{P(w,c)}{P(w)P(c)} \right) = \log_2 \left(\frac{count(w,c) \times D}{count(w) \times count(c)} \right) \quad (1)$$

where D corresponds to the total number of tokens in the corpus. Negative PMI values suggest that the two words co-occur less often than they would by chance. These estimates are unreliable unless the counts are calculated from very large corpora [37]. PPMI corrects this problem by replacing negative values by zero:

$$PPMI(w,c) = \max(0, PMI(w,c)) \quad (2)$$

We next describe how to implement a word-context matrix with PPMI scores incrementally. We identify the following requirements to carry out this implementation:

1. Each tweet can only be processed once.
2. The calculation of counters required for computing PPMI must be incremental: $count(w_i, c_j)$,$count(w_i)$, $count(c_j)$, and D.
3. The model must deal with the fact that the vocabulary V and the context space C are dynamic i.e., unknown beforehand.

Our implementation is based on the Streaming PMI algorithm proposed by Durme et al. [23]. The vocabulary size V, the context space C (which defines the number of columns in our vectors), and the window size W (W words to the left and W words to the right) are the parameters of the model. Consequently, the number of word vectors maintained in memory is restricted to V and the number of contexts words is limited to C. Our dynamic word-context matrix is represented as a sparse matrix implemented with type-specific dictionary data structures from the *fastutil Java Library*[6]. These dictionaries are used to store and maintain all the counters required for computing PPMI. Each tweet in the stream is tokenized, and then a sliding window of $2W$ tokens is shifted across the whole sequence. The window is centered on a target word w and all the surrounding tokens within the window are considered as the context tokens $c_1, \ldots, c_{2W}$[7]. Unseen target words and contexts are dynamically allocated in the sparse word-context matrix upon arrival. For existing words and contexts the corresponding counters are updated accordingly.

We consider three approaches for representing contexts. In the vanilla version of our model (referred to as "No-Hashing"), context tokens are words. Further in this section, we describe other variations in which the context space is reduced using equivalence classes between contexts.

Data: tweets, window size W, vocabulary size V, context size C
Result: PPMI Matrix M
Initialize PPMI Matrix M of size $V \times C$;
$D \leftarrow 0$;
while *tweet in Data Stream* **do**
  tokens $\leftarrow$ tokenize(tweet);
  for ***each*** w *in tokens* **do**
    $D \leftarrow D + 1$;
    $c_1, \ldots, c_{2W} \leftarrow$ getContexts(w,tokens);
    updateContextMatrix($w, c_1, \ldots, c_{2W}$);
    for ***each*** c_j *in* $c_1, \ldots, c_{2W}$ **do**
      $\text{PPMI}(w, c_j) \leftarrow \max\left(0, \log_2\left(\frac{count(w,c_j) \times D}{count(w) \times count(c_j)}\right)\right)$;
    end
  end
end

Algorithm 1: Incremental PPMI word-context matrix building process.

Algorithm 1 describes the process of calculating and storing PPMI values for word-context pairs. The 'getContexts' function returns the column indexes of the $2W$

[6] `http://fastutil.di.unimi.it/`

[7] It is important to remark that the target word w is excluded from the context window $c_1, \ldots, c_{2W}$. For example, for the sentence "I like my nice dog", target word w = "my", and window size $W = 2$, then the context words c_1, c_2, c_3, c_4 ($2W = 4$) would be "I","like","nice","dog".

words surrounding target word w[8]. The 'updateContextMatrix' function increments counters ($count(w,c_j)$,$count(w)$, $count(c_j)$) in the context matrix for the neighboring words in the window of size W. These calculated PPMI values are then used as attributes or features to train the incremental classifier described in Section 3.2.

To keep memory usage low, there is a maximum number of words and contexts that can be stored in memory defined by parameters V and C. There are various approaches to deal with new arriving words and contexts once the maximum number of words and contexts have been allocated. A simple approach is to either ignore them, or to allocate them into special cells for unknown tokens. Alternatively, we can keep in memory the k most frequent words using the Space-Saving algorithm [50]. We next describe the sketching techniques we employ to reduce the number of contexts based on equivalence classes.

The first approach, which is based on the work of [55], is to hash context words into a fixed number of buckets equal to the context size C. Thereby, any possible context word will be mapped to one of the matrix's columns. There is a downside to this: unrelated contexts could be merged into the same bucket affecting the representation power of the corresponding column. For each context word, the corresponding context bucket (or matrix column) c_j is obtained using the Jenkins hash function [36] and the modulo operation ($c_i \leftarrow hash(\text{token})\%C$). We implement this procedure within the 'getContexts' function from Algorithm 1. The Jenkins hash function has been shown to have a low collision rate for short words [55].

The second sketching technique we explore in this paper consists of using pre-trained word clusters. The clusters are trained from a corpus of 56,345,753 tweets[9] using the Brown clustering algorithm [15], which produces hierarchical clusters by maximizing the mutual information of bigrams. These clusters have been shown to be useful for part-of-speech tagging of tweets [53].

The clusters are used analogously to the hashing function from the previous method. We replace the hash values by the cluster IDs of the context words. The context size is equal to the number of clusters, which is 1,000 in this case. Context words that are not associated with any cluster are ignored. It is important to remark that even though the clusters are calculated in a static fashion, our word vectors are dynamic: when the distribution of a word changes it will start co-occurring with other word-clusters and consequently, its vector will change. In the same way, words that are not part of the cluster set will receive their own vector as long as they co-occur with words from the cluster set.

The process described above was implemented as a MOA stream generator. Essentially, the generator converts a stream of tweets into another stream of word vectors that can be fed into any of MOA's incremental classifiers. The three variants of contexts in our method: No-Hashing, Jenkins Hashing, and Brown clusters, are implemented as options in our stream generator.

[8] The method can return less than $2W$ words for out-of-range positions.

[9] CMU TweetNLP - `http://www.cs.cmu.edu/~ark/TweetNLP/`

3.2 Incremental Learning

An incremental classifier is a supervised model that can learn from new training instances without needing to be retrained on the entire training set from scratch. We use incremental learning to train an adaptive word-level classifier that maps our word vectors into a unidimensional sentiment variable. Our training instances are obtained from a seed lexicon formed by positive and negative words. Every time a word from the seed lexicon is observed, the classifier is updated with a new training instance formed by the current vector of the word and the label provided by the lexicon.

Technically speaking, our implementation of word vectors would allow us to use any incremental classifier provided by MOA. However, we choose binary logistic regression because of: 1) its scalability to high-dimensional sparse data, and 2) the probabilistic interpretation of its outcome.

In logistic regression, the classifier's output is transformed using the sigmoid function to the range $[0,1]$, and is interpreted as the conditional probability of the class given the attribute vector $P(y=1|\mathbf{x})$ i.e., the probability of a word being positive given its context vector.

Let $\mathbf{x}$ be a word vector, and y be the ground-truth sentiment of that word according to the seed lexicon (1 if is positive and zero otherwise), then the word's sentiment $\hat{y}$ is estimated with the following linear model:

$$\hat{y} = \sigma(f(\mathbf{x})) = \frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}+b}}$$

The weight vector $\mathbf{w}$ and the bias term b are the parameters of the model. The model's weights determine how strongly a particular context influences the prediction of negative and positive sentiment.

A logistic regression is trained by minimizing the logistic loss function with an L_2 penalty:

$$L_{logistic}(\hat{y},y) = -y\log\hat{y} - (1-y)\log(1-\hat{y}) + \frac{\lambda}{2}||w||^2$$

This loss function is essentially calculating an error rate based on comparisons between model's predictions and expected outcomes. The regularization term prevents the parameters from taking large absolute values, in order to avoid over-fitting. This makes our model more robust to noise: mislabeled words in the lexicon. Our approach assumes that the majority of the words from the lexicon are labeled correctly and unlikely to change its sentiment over time.

As described below, our logistic regression is trained incrementally using stochastic gradient descent (SGD). All the parameters of the model $(\mathbf{w},b)$ are randomly initialized and tuned iteratively with the help of the training words from the lexicon. The classifier makes predictions using the current values of the internal parameters, and computes gradients for these parameters with respect to the loss estimate to move or update the parameters in the opposite direction of the gradient in proportion to a learning rate η.

More specifically, for each training word $(\mathbf{x},y)$ the loss L is calculated with current parameter values and updated with the following rule: $w_i \leftarrow w_i - \eta\frac{\partial L}{w_i}(\hat{y},y)$ (for

all parameters). In our experiments we set the learning rate and the regularization parameters to 0.01 and 0.0001 respectively. These values are the default parameter values of MOA's SGD implementation, which were also used in previous studies [11, 12]. Therefore, our model is capable of predicting the sentiment of a word at any time, and adapt itself to changes in the sentiment pattern.

3.3 Evaluation

In order to evaluate our approach we divide our seed lexicon into two sub-partitions, training and testing, and perform an incremental hold-out evaluation [6]. For each token in the stream, the corresponding word vector gets updated as described in Section 3.1. If the word is part of the training lexicon, then the word vector and the label are used to train/update the incremental classifier. Otherwise, if the word belongs to the test lexicon, a sentiment prediction is obtained by deploying the current classifier on the word vector, which is then compared against the actual class. Algorithm 2 exhibits a summary of this process.

```
Data: tweets, training seed lexicon, testing seed lexicon
while Tweet T in Data Stream do
    for each token t in tokens do
        updateContextCounters(t,T);
        updatePPMIValues(t,T);
        if token ∈ train seed lexicon then
            trainClassifier(PPMI(t),label(t));
        else if token ∈ test seed lexicon then
            updateEvaluator(PPMI(t),label(t));
    end
end
```

Algorithm 2: Training and evaluation process.

We report accuracy and the kappa statistic as evaluation metrics. Kappa is considered as a more sensitive measure for quantifying the predictive performance of streaming classifiers [4,6], because of its robustness against class imbalance. It measures how closely the predictions made by the classifier match with the gold standard labels, while controlling for the instances classified by random chance.

$$kappa = (totalAccuracy - randomAccuracy)/(1 - randomAccuracy) \quad (3)$$

Kappa values range from -infinity to +1. A 0 value would suggest that the predictions are equal to that of a random classifier. Any kappa value less than 0 indicates that the predictions by the classifier are worse than that of a random classifier. A kappa value of 1 means that the class label predictions of the trained classifier are perfectly aligned with the actual labels of the instances.

Another important concern when evaluating stream classifiers is how to aggregate the values of the evaluation metric over time. We consider three aggregation approaches, all of them implemented in MOA, that are described below:

- Simple evaluator: performs basic incremental evaluation by averaging the value of a metric over time.
- Fading Factor evaluator: updates evaluation results using a smoothing factor in which recent examples are weighted higher than the older ones. In our experiments we set the fading factor parameter to 0.999.
- Reseting evaluator: resets evaluation results after a pre-defined number of test instances. In our experiments, we set this value to 10,000.

The parameter values of the above settings correspond to the default MOA values.

3.3.1 Non-stationary Evaluation

The sentiment of the test words in the evaluation framework described above is static. Even if our model can detect a change in sentiment of words that are not part of the test lexicon, this would not be reflected in our evaluation metrics. This poses a limitation on studying the ability of our dynamic lexicon to adapt to a drastic change in the sentiment of words.

As discussed in Section 2 the availability of ground-truth evaluation data for the task of semantic change detection is fairly limited. The most comprehensive dataset created so far for this task was developed for SemEval 2020 Task 1 [59], which consists of four pairs of corpora spanning two different periods of time. Each corpus pair is written in one of the following four languages: German, English, Swedish, and Latin, and the gold data consists of a list of target terms for each language with binary and ordinal labels that indicate the degree of semantic change of the target term between the two periods of time.

However, this dataset is not suitable for our task. First, in our work we focus on sentiment change, not semantic change, and second, we focus on detecting spontaneous change in real time, which is far more complex than detecting change between two discrete time periods.

Unfortunately, creating ground-truth data with real-time word-level sentiment change would require an enormous amount of annotation effort and linguistic knowledge, which is beyond our current capabilities. Consequently, we decided to create synthetic data in which our object of study (i.e., spontaneous sentiment drift) can be controlled and evaluated accordingly.

In our mechanism, we introduce a drastic change in sentiment in a portion of the test lexicon to monitor its effect on the performance of the incremental sentiment classifier. We add synthetic tweets to our stream where a percentage of tokens was replaced with other tokens of opposite sentiment. This is done after an initial learning period in which a predefined number of tweets is processed without inducing change.

For example, in the event of the generator simulating sentiment drift of the word "love" from positive to negative, it will proceed to generate tweets in which the word love occurs in the context of a negative word such as "hate", e.g., the tweet "I *hate* you, I regret that I met you" would be replaced to "I *love* you, I regret that I met you". Since evidence suggests that the sentiment of a word is determined by its context [13], we believe that a good dynamic lexicon should be capable of detecting this change (i.e., classifying the word to its new sentiment class) in a timely manner.

The process is described in more detail below. The drift generator receives a corpus of chronologically ordered tweets and the test lexicon as input. Other parameters of the generator are the fraction of words to be swapped f and the time of the drift d measured by number of tweets. For example, if f is 0.4 and d is $1,000$, after seeing 1000 tweets, 40% of the test words will be paired with words with the opposite sentiment and their polarities will be exchanged. For all subsequent occurrences of those word pairs, they will be swapped with each other in the content of the tweets.

In order to preserve the grammatical correctness of the synthetic sentences, only adjectives from the test lexicon are considered as candidates for replacement[10]. They are identified using the POS tagger from the NLTK library[11]. We prioritize frequent adjectives for replacement to ensure that a sufficient number of inverted occurrences are observed in the remainder of the data stream, and also only consider pairs of words of similar frequency for the exchange to keep the training class distribution roughly balanced across the full stream.

4 Experiments

In this section, we report the experimental results divided into three parts. The first part describes the data, the second part shows experimental results using a fixed testing lexicon, and the third part shows results with induced change.

4.1 Data

Our experiments require a collection of chronologically ordered tweets to represent a stream of tweets. We take a collection of 2 million English tweets from the Edinburgh corpus (ED) [54]. This corpus is a general purpose collection of 97 million tweets in multiple languages collected with the Twitter streaming API[12] between November 11, 2009 and February 1, 2010.

We consider the *AFINN-111* lexicon [1] as the seed lexicon. This resource is formed by 2,477 positive and negative words scored from 5 to -5. It includes slang, obscene words, acronyms and Web jargon. A binary prediction problem was generated by only using the sign of the numeric sentiment scores. We split the seed lexicon randomly into training (70%) and testing (30%) sets for evaluating the supervised learning process.

4.2 Experiments with a fixed test lexicon

In this section we report classification experiments using a fixed test lexicon as described in Algorithm 2 with the data reported above.

[10] An additional reason to focus on adjectives is that they are the most important class of opinion words [45].

[11] `https://www.nltk.org/`

[12] `https://dev.twitter.com/streaming/overview`

For all the experiments, the SGD classifier from the MOA package was used with a learning rate of 0.01. The loss function for the classifier was set to 'Log' loss in order to obtain probability distributions for prediction classes as explained in Section 3.2. These probability values will be used in later experiments when analyzing the change in sentiment of individual words.

The control variables that we manipulate throughout our experiments are:

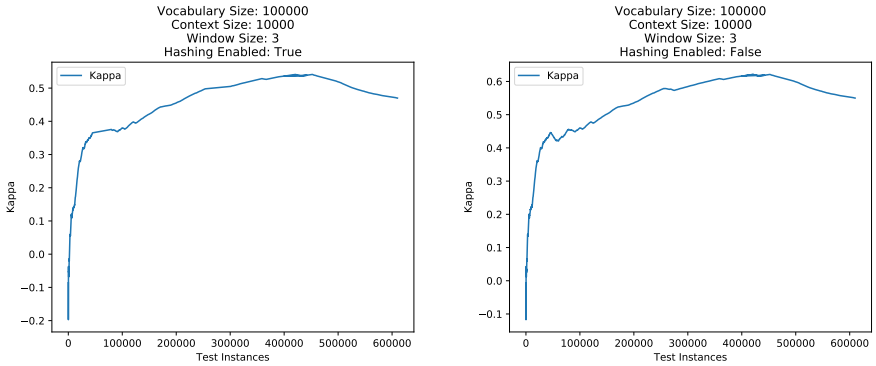
- **Context type**: we experiment with three types of contexts as explained in Section 3.1: No-Hashing, Jenkins Hashing, and Brown clusters.
- **Vocabulary Size** (V): the number of unique word vectors stored in memory. Reducing the size of the vocabulary results in a reduction of processing time. Experiments were performed with vocabulary sizes of 10,000 and 100,000.
- **Context size** (C): represents the number of attributes used to train the classifier. Increasing the context size would also increase the size of the context matrix and the learning time for the classifier. Experiments were performed with various context sizes from 100 to 10,000. When using Brown clusters as the context type, the context size is fixed at 1,000, which is the total number of word clusters.
- **Window length** (W): is the number of neighboring tokens considered as the context of a token for generating the Word Context Matrix. Experiments were conducted using windows of lengths 2, 3 and 4.

Classification performance of the best three configurations for each context type using a simple evaluator (i.e., evaluation metrics are averaged over time) is shown in Table 1.

Vocabulary Size	Context Size	Window Size	Context Type	Final Accuracy	Final Kappa
10,000	500	3	Jenkins Hash	67.19	0.3279
100,000	1,000	4	Jenkins Hash	66.89	0.3229
100,000	10,000	3	Jenkins Hash	**75.80**	**0.4656**
10,000	1,000	4	No-Hashing	71.32	0.3779
10,000	1,000	3	No-Hashing	70.51	0.3527
100,000	10,000	3	No-Hashing	**78.08**	**0.5449**
10,000	1,000	2	Brown Clusters	**72.99**	**0.4336**
10,000	1,000	3	Brown Clusters	67.79	0.3557
10,000	1,000	4	Brown Clusters	68.42	0.3529

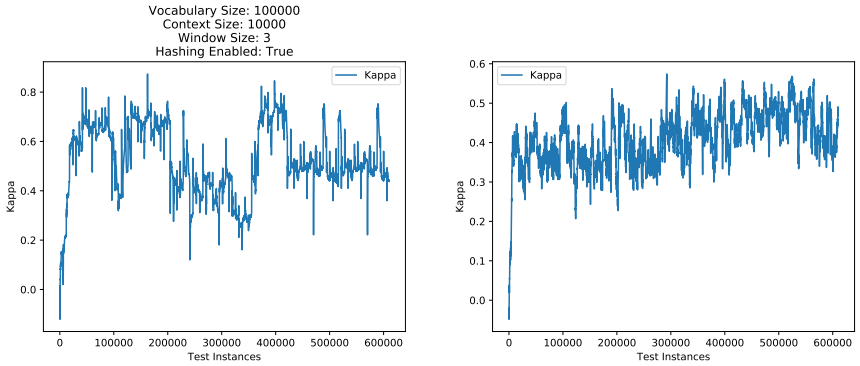
Table 1 Summary of results using a simple evaluator. Best results for each context type are marked in bold.

From the table we can see that increasing the context size yields improvements in the performance of Jenkins Hashing and No-Hashing configurations. This shows that high dimensional contexts are useful for capturing the semantics of words and suitable for SGD training. We believe that the reason No-Hashing outperforms Jenkins Hashings is that the latter tends to merge unrelated contexts into the same cells, which can confuse the classifier. However, Jenkins Hashing's results are competitive and have the advantage of being able to incorporate unlimited new contexts into the vector, unlike No-Hashing, which discards new contexts once all the context columns have been allocated. Brown clusters exhibit similar performance to Jenkins Hashing. A noteworthy property of Brown clusters is that they have achieved their best performance using shorter window ($W = 2$). However, since the approach requires the



1. Jenkins Hashing 2. No Hashing

Fig. 2 Kappa values over time with a simple evaluator.



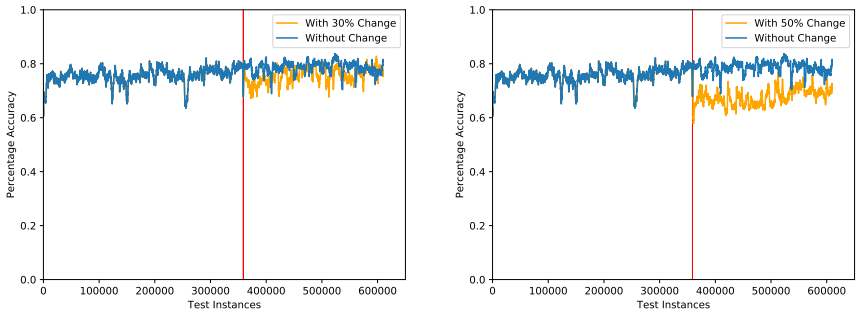
1. Reseting Evaluator 2. Fading Factor Evaluation

Fig. 3 Kappa values over time with a reseting evaluator and a fading factor evaluation.

expensive process of pre-training word clusters in a batch setting, we do not see significant benefits from their use.

When analyzing stream classifiers, it is important to analyze how classification performance develops over time. Figure 2 shows the learning curves of the best Jenkins Hashing and No-Hashing configurations. Because the simple evaluator averages performance over time, it cannot be used to observe short-term fluctuations.

Evaluation schemes that are more suitable for inspecting how performance varies across time are the reseting and the fading factor evaluator described in Section 3.3. Figure 3 shows the learning process of the best configuration using Jenkins Hashing with those schemes.



1. Accuracy for experiment with 30% of Change

2. Accuracy for experiment with 50% of Change

Fig. 4 Accuracy over time using the Fading Factor Evaluator. Experiments performed with parameters - Vocabulary Size - 100,000, Context Size - 10,000, Window Size - 3, Jenkins Hashing - Enabled.

4.3 Experiments with induced change

In this section we follow the non-stationary evaluation framework presented in Section 3.3.1 to study how our model behaves in scenarios of drastic sentiment change. We consider the best configuration using Jenkins hashing (V = 100,000, C = 10,000, and W = 3) for these experiments.

The time of the drift d, that is the point at which the change is introduced, lies in the middle of the training. As for the fraction of words from the test lexicon to swap f, we experiment with values of 0.3 and 0.5.

The graphs in Figure 4 show classification accuracy over time using a fading factor evaluator for the two values of f. The red line on the graphs shows the point at which the change was introduced. After the change, each graph shows classification accuracy with and without change.

We can observe that accuracy decreases after the change is induced in both scenarios. The accuracy of the experiment with 30% of change drops to less than that of the experiment with 50% change (Graph 2 in Figure 4). We also observe that accuracy tends to recover after the drop. This validates the hypothesis that our model is able to adapt to the new sentiment pattern. Another noteworthy result is that when the amount of change is 30%, the model achieves the same level of accuracy as the model without change after a certain number of instances.

In the following experiment we further study the sentiment change of individual words. We visualize and monitor the sentiment of some words over time, which is calculated as the difference between the probabilities of positive and negative classes returned by the incremental logistic regression. This value is referred to as “sentiment score”.

This score ranges from -1 to +1. The closer the sentiment score is to -1 or +1, the more negative or positive the predicted sentiment will be, respectively. This score gives an intuitive way to visualize the evolution of the sentiment. For example, if a word originally has a positive sentiment and over time changes to a negative senti-

ment, the change in sentiment would be detected when the score falls below the 0 mark.

The Table 2 lists the tokens for which the sentiment score is analyzed. It includes 3 randomly paired words that were swapped after a certain number of occurrences.

	Token (a)	**Token (b)**
1.	*dead*	*popular*
2.	*ridiculous*	*advanced*
3.	*nasty*	*lucky*

Table 2 Examples of Tokens Swapped.

The graphs in Figure 5 show the evolution of sentiment score for these tokens with their swapped pair before and after change. The plots report the sentiment evolution in both change and non-change scenarios. The red line illustrates the point at which the token was switched with a token of the opposite sentiment.

In the graphs, it can be seen that the sentiment reverses for words 'ridiculous', 'popular' and 'advanced' with a high confidence (according to the classifier's predictions). The word 'ridiculous' is originally a negative word (according to the seed lexicon), and was replaced with 'advanced', a positive word. After the words were swapped, the classifier takes about 150 occurrences for the word 'ridiculous' and about 60 occurrences for 'advanced' to learn the new sentiment. As for words 'nasty' and 'lucky', it took the classifier about 700 and 600 occurrences to learn the new sentiment. For the word 'dead' (replaced with 'popular'), after the swap more than 1,000 word occurrences were needed for the classifier to consistently label it as a positive token. This could be because it was swapped with a word ('popular') that was constantly misclassified as negative before the change.

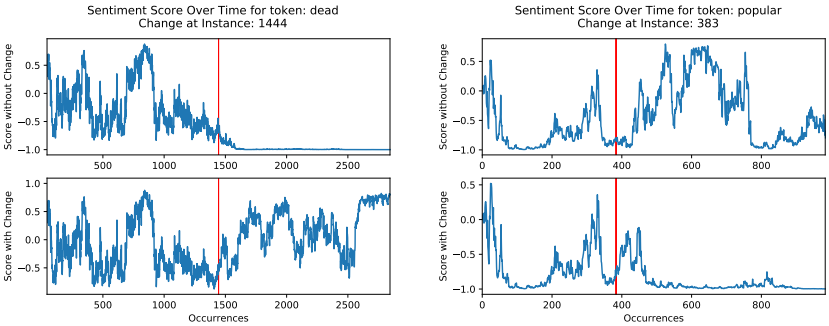
These results indicate that our model can learn the new sentiment of words, but the necessary time to adjust can vary considerably from word to word.

5 Conclusions

Human language has always been subject to continuous change, and the advent of social media has accelerated this process [20]. This phenomenon is particularly relevant to the field of sentic computing, because in a scenario where words can change their sentiment, new words can emerge and others can disappear, existing static sentic resources are limited to ignoring these temporal dynamics of language.

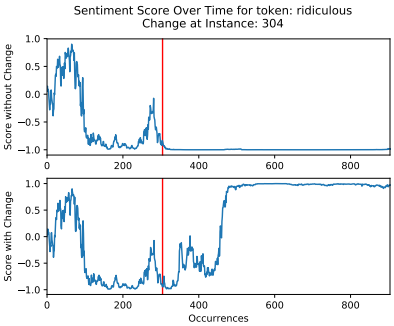
This paper proposed a new methodology to incrementally follow the sentiment dynamics of words using stream mining techniques. Our experimental results validated our research hypothesis, confirming that incremental word vectors and sentiment classifiers can be used together to successfully track the sentiment of words over time. In addition, overall classification accuracy proofed to be robust to abrupt sentiment change of 30% of the testing words.

This approach could be used for online opinion mining of social media streams, and could be useful for monitoring public opinion in relation to various types of

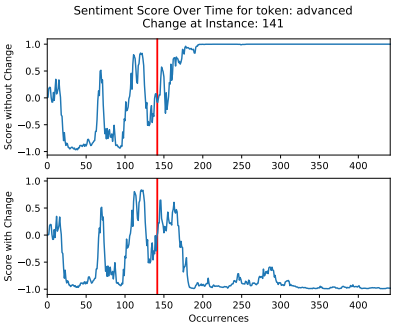


1. (a) dead

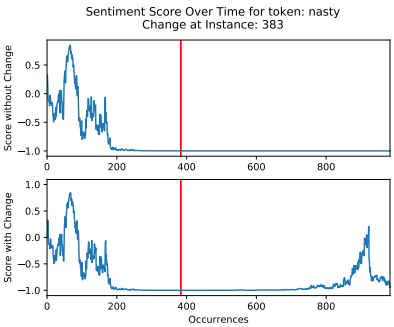
1. (b) popular



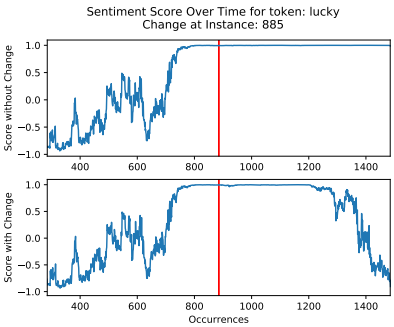
2. (a) ridiculous



2. (b) advanced



3. (a) nasty



3. (b) lucky

Fig. 5 Sentiment changes for some sample words.

events, such as political campaigns, civil unrest, sports competitions, film premieres and natural disasters.

The source code of all the models implemented in this paper are freely released and integrated into MOA [7], a machine learning tool for mining data streams. Our aim is to provide a free tool to track public opinion from tweets that does not depend

on expensive resources (such as labeled tweets) and that can be used by people who do not necessarily have programming skills (e.g., linguists, journalists, sociologists).

However, we strongly recommend being careful with the application of the sentiment assigned by our method to entity words such as people and social groups. Distributional word vectors are likely to inherit all the biases exhibited in the corpora on which they are trained [3]. Therefore, we recommend caution with the interpretation of the learned sentiment values of the entity words in the lexicon, since they can reflect a general perception of the entity in the corpus rather than its underlying sentiment valence.

For future work, we will implement incremental word embeddings models based on neural networks, such as the incremental skip-gram model [38], and incorporate them into our sentiment tracking framework. We will also experiment by adding other sentiment signals into the seed lexicon that were not considered in this work such as hashtags, emoticons and emojis.

It is important to note that our approach is based solely on the distributional hypothesis in which all context words are treated equally. However, several studies in sentiment analysis [46,48,47] state that the sentiment of a sentence is determined by both its aspects and polarity terms. In the future, we plan to pre-process our sentences with an aspect extraction method [48,47] and to treat aspects words as special context attributes.

Another limitation of our approach is that we are not making any distinction between words occurring in affirmative and negated contexts. It is well known that the use of negation terms in a sentence such as "no", "don't", and "never" can significantly affect the sentiment of its scope [40]. We plan on developing two strategies to tackle this problem: 1) use a phrase extractor to add negated multi-word expressions (e.g., not happy) into our vocabulary, 2) treat negation as a special type of context for our word vectors to explicitly distinguish between affirmative and negated contexts. Moreover, we plan to monitor the sentiment scores for tokens using change detection algorithms implemented in MOA such as the ADWIN algorithm [5]. This will trigger an alarm when a significant change is observed.

Other avenues of further research include adding a forgetting mechanism to our word counts to make adaption to change faster, and going beyond binary sentiment classes by using other affective categories such as those included in the Hourglass of emotions [61]: introspection, temper, attitude and sensitivity.

As a final reflection, we hope that our study will open the door to further research in expanding the sentic computing field to a dynamic setting in which language is not static, but in continuous evolution.

Acknowledgements The authors would like to thank former Honors student Tristan Anderson for a preliminary study on incremental sentiment lexicons.

Compliance with Ethical Standards

Funding: This work was funded by ANID FONDECYT grant 11200290, U-Inicia VID Project UI-004/20 and ANID - Millennium Science Initiative Program - Code

ICN17_002.

Conflict of Interest: The authors declare that they have no conflict of interest.

Informed Consent: Informed consent was not required as no human or animals were involved.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Årup Nielsen, F.: A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In: Proceedings of the 1st Workshop on Making Sense of Microposts (#MSM2011), pp. 93–98 (2011)
2. Baccianella, S., Esuli, A., Sebastiani, F.: Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation, pp. 2200–2204. European Language Resources Association (2010)
3. Badilla, P., Bravo-Marquez, F., Prez, J.: WEFE: The word embeddings fairness evaluation framework. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pp. 430–436. International Joint Conferences on Artificial Intelligence Organization (2020). DOI 10.24963/ijcai.2020/60. URL `https://doi.org/10.24963/ijcai.2020/60`
4. Bifet, A., Frank, E.: Sentiment knowledge discovery in twitter streaming data. In: Proceedings of the 13th international conference on Discovery science, pp. 1–15. Springer-Verlag (2010)
5. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining, pp. 443–448. SIAM (2007)
6. Bifet, A., Gavaldà, R., Holmes, G., Pfahringer, B.: Machine Learning for Data Streams: with Practical Examples in MOA. MIT Press (2018)
7. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research **11**(May), 1601–1604 (2010)
8. Bifet, A., Holmes, G., Pfahringer, B., Gavalda, R.: Detecting sentiment change in twitter streaming data. In: T. Diethe, J. Balcazar, J. Shawe-Taylor, C. Tirnauca (eds.) Proceedings of the Second Workshop on Applications of Pattern Analysis, *Proceedings of Machine Learning Research*, vol. 17, pp. 5–11. PMLR, CIEM, Castro Urdiales, Spain (2011). URL `http://proceedings.mlr.press/v17/bifet11a.html`
9. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Y. Lechevallier, G. Saporta (eds.) Proceedings of COMPSTAT'2010, pp. 177–186. Physica-Verlag HD, Heidelberg (2010)
10. Bravo-Marquez, F., Frank, E., Pfahringer, B.: From unlabelled tweets to twitter-specific opinion words. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 743–746 (2015)
11. Bravo-Marquez, F., Frank, E., Pfahringer, B.: Positive, negative, or neutral: Learning an expanded opinion lexicon from emoticon-annotated tweets. In: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, p. 12291235. AAAI Press (2015)
12. Bravo-Marquez, F., Frank, E., Pfahringer, B.: Building a twitter opinion lexicon from automatically-annotated tweets. Knowledge-Based Systems **108**, 65–78 (2016)
13. Bravo-Marquez, F., Frank, E., Pfahringer, B.: From opinion lexicons to sentiment classification of tweets and vice versa: A transfer learning approach. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016, pp. 145–152 (2016)
14. Bravo-Marquez, F., Frank, E., Pfahringer, B.: Transferring sentiment knowledge between words and tweets. Web Intelligence **16**(4), 203–220 (2018)
15. Brown, P.F., Desouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. Computational linguistics **18**(4), 467–479 (1992)
16. Cambria, E., Hussain, A.: Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis. Springer International Publishing, Cham, Switzerland (2015)

17. Cambria, E., Li, Y., Xing, F.Z., Poria, S., Kwok, K.: Senticnet 6: Ensemble application of symbolic and subsymbolic ai for sentiment analysis. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, p. 105114. Association for Computing Machinery, New York, NY, USA (2020). DOI 10.1145/3340531.3412003. URL https://doi.org/10.1145/3340531.3412003
18. Cambria, E., Poria, S., Hazarika, D., Kwok, K.: Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In: S.A. McIlraith, K.Q. Weinberger (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pp. 1795–1802. AAAI Press (2018). URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16839
19. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational linguistics **16**(1), 22–29 (1990)
20. Cunha, E., Magno, G., Comarela, G., Almeida, V., Gonçalves, M.A., Benevenuto, F.: Analyzing the dynamic evolution of hashtags on twitter: a language-based approach. In: Proceedings of the workshop on language in social media (LSM 2011), pp. 58–65 (2011)
21. Denecke, K.: Using sentiwordnet for multilingual sentiment analysis. In: 2008 IEEE 24th International Conference on Data Engineering Workshop, pp. 507–512 (2008). DOI 10.1109/ICDEW.2008.4498370
22. Durant, K.T., Smith, M.D.: The impact of time on the accuracy of sentiment classifiers created from a web log corpus. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, pp. 1340–1346. AAAI Press (2007)
23. Durme, B.V., Lall, A.: Streaming pointwise mutual information. In: Advances in Neural Information Processing Systems, pp. 1892–1900 (2009)
24. Esuli, A., Sebastiani, F.: Determining the semantic orientation of terms through gloss classification. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 617–624. ACM (2005)
25. Esuli, A., Sebastiani, F.: Sentiwordnet: A publicly available lexical resource for opinion mining. In: In Proceedings of the 5th Conference on Language Resources and Evaluation, pp. 417–422. European Language Resources Association (2006)
26. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford **1**(12) (2009)
27. Guimarães, N., Torgo, L., Figueira, Á.: Twitter as a source for time-and domain-dependent sentiment lexicons. In: Social Network Based Big Data Analysis and Applications, pp. 1–19. Springer (2018)
28. Hamilton, W.L., Clark, K., Leskovec, J., Jurafsky, D.: Inducing domain-specific sentiment lexicons from unlabeled corpora. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 595–605. The Association for Computational Linguistics (2016)
29. Hamilton, W.L., Leskovec, J., Jurafsky, D.: Diachronic word embeddings reveal statistical laws of semantic change. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 1489–1501 (2016)
30. Harris, Z.S.: Distributional structure. Word **10**(2-3), 146–162 (1954)
31. Heerschop, B., Hogenboom, A., Frasincar, F.: Sentiment lexicon creation from lexical resources. In: International Conference on Business Information Systems, pp. 185–196. Springer (2011)
32. Heerschop, B., van Iterson, P., Hogenboom, A., Frasincar, F., Kaymak, U.: Analyzing sentiment in a large set of web data while accounting for negation. In: E. Mugellini, P.S. Szczepaniak, M.C. Pettenati, M. Sokhn (eds.) Advances in Intelligent Web Mastering – 3, pp. 195–205. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
33. Hogenboom, A., Bal, D., Frasincar, F., Bal, M., de Jong, F., Kaymak, U.: Exploiting emoticons in sentiment analysis. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC 13, p. 703710. Association for Computing Machinery, New York, NY, USA (2013). DOI 10.1145/2480362.2480498. URL https://doi.org/10.1145/2480362.2480498
34. Hogenboom, A., Heerschop, B., Frasincar, F., Kaymak, U., de Jong, F.: Multi-lingual support for lexicon-based sentiment analysis guided by semantics. Decision support systems **62**, 43–53 (2014)
35. Ibrahim, N.F., Wang, X.: Decoding the sentiment dynamics of online retailing customers: Time series analysis of social media. Computers in Human Behavior **96**, 32–45 (2019)
36. Jenkins, R.: Hash functions. Dr Dobbs Journal **22**(9), 107–+ (1997)

37. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd edn. Prentice Hall, Upper Saddle River, NJ, USA (2008)
38. Kaji, N., Kobayashi, H.: Incremental skip-gram model with negative sampling. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 363–371 (2017)
39. Kim, Y., Chiu, Y.I., Hanaki, K., Hegde, D., Petrov, S.: Temporal analysis of language through neural language models. In: Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science, pp. 61–65. Association for Computational Linguistics, Baltimore, MD, USA (2014). DOI 10.3115/v1/W14-2517. URL https://www.aclweb.org/anthology/W14-2517
40. Kiritchenko, S., Zhu, X., Mohammad, S.M.: Sentiment analysis of short informal texts. Journal of Artificial Intelligence Research **50**, 723–762 (2014)
41. Kulkarni, V., Al-Rfou, R., Perozzi, B., Skiena, S.: Statistically significant detection of linguistic change. In: Proceedings of the 24th International Conference on World Wide Web, WWW '15, pp. 625–635. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2015). DOI 10.1145/2736277.2741627. URL https://doi.org/10.1145/2736277.2741627
42. Kutuzov, A., Øvrelid, L., Szymanski, T., Velldal, E.: Diachronic word embeddings and semantic shifts: a survey. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1384–1397. Association for Computational Linguistics, Santa Fe, New Mexico, USA (2018). URL https://www.aclweb.org/anthology/C18-1117
43. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in neural information processing systems, pp. 2177–2185 (2014)
44. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics **3**, 211–225 (2015)
45. Liu, B.: Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers (2012)
46. Liu, B.: Sentiment analysis and opinion mining. Synthesis lectures on human language technologies **5**(1), 1–167 (2012)
47. Ma, Y., Peng, H., Khan, T., Cambria, E., Hussain, A.: Sentic LSTM: a hybrid network for targeted aspect-based sentiment analysis. Cognitive Computation **10**(4), 639–650 (2018). URL https://doi.org/10.1007/s12559-018-9549-x
48. Marrese-Taylor, E., Velásquez, J.D., Bravo-Marquez, F.: A novel deterministic approach for aspect-based opinion mining in tourism products reviews. Expert Systems with Applications **41**(17), 7764–7775 (2014)
49. May, C., Duh, K., Van Durme, B., Lall, A.: Streaming word embeddings with the space-saving algorithm. arXiv preprint arXiv:1704.07463 (2017)
50. Metwally, A., Agrawal, D., El Abbadi, A.: Efficient computation of frequent and top-k elements in data streams. In: International Conference on Database Theory, pp. 398–412. Springer Berlin Heidelberg (2005)
51. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119. Curran Associates, Inc. (2013)
52. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Wordnet: An on-line lexical database. International Journal of Lexicography **3**, 235–244 (1990)
53. Owoputi, O., OConnor, B., Dyer, C., Gimpel, K., Schneider, N., Smith, N.A.: Improved part-of-speech tagging for online conversational text with word clusters. In: Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies, pp. 380–390 (2013)
54. Petrović, S., Osborne, M., Lavrenko, V.: The edinburgh twitter corpus. In: Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media, pp. 25–26. Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
55. QasemiZadeh, B., Kallmeyer, L., Passban, P.: Sketching word vectors through hashing. CoRR **abs/1705.04253** (2017). URL http://arxiv.org/abs/1705.04253
56. Rosenfeld, A., Erk, K.: Deep neural models of semantic shift. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 474–484. Association for Computational Linguistics, New Orleans, Louisiana (2018). DOI 10.18653/v1/N18-1044. URL https://www.aclweb.org/anthology/N18-1044

57. Rubtsova, Y.: Reducing the deterioration of sentiment analysis results due to the time impact. Information **9**(8) (2018). DOI 10.3390/info9080184. URL https://www.mdpi.com/2078-2489/9/8/184
58. Saeidi, M., Bouchard, G., Liakata, M., Riedel, S.: Sentihood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 1546–1556 (2016)
59. Schlechtweg, D., McGillivray, B., Hengchen, S., Dubossarsky, H., Tahmasebi, N.: SemEval-2020 task 1: Unsupervised lexical semantic change detection. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 1–23. International Committee for Computational Linguistics, Barcelona (online) (2020). URL https://www.aclweb.org/anthology/2020.semeval-1.1
60. Stewart, I., Arendt, D., Bell, E., Volkova, S.: Measuring, predicting and visualizing short-term change in word representation and usage in vkontakte social network. In: Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017., pp. 672–675 (2017)
61. Susanto, Y., Livingstone, A.G., Ng, B.C., Cambria, E.: The hourglass model revisited. IEEE Intelligent Systems **35**(5), 96–102 (2020). DOI 10.1109/MIS.2020.2992799
62. Tang, D., Wei, F., Qin, B., Zhou, M., Liu, T.: Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In: Proceedings of the 25th International Conference on Computational Linguisticss, pp. 172–182. Association for Computational Linguistics (2014)
63. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of Artificial Intelligence Research **37**(1), 141–188 (2010)
64. Wiegand, M., Balahur, A., Roth, B., Klakow, D., Montoyo, A.: A survey on the role of negation in sentiment analysis. In: R. Morante, C. Sporleder (eds.) Proceedings of the Workshop on Negation and Speculation in Natural Language Processing (NeSp-NLP 2010), 10 July 2010, Uppsala, Sweden, pp. 60 – 68. Association for Computational Linguistics, Stroudsburg, PA (2019)