Using Twitter Streams for Opinion Mining: A Case Study on Airport Noise

Iheb Meddeb, Catherine Lavandier, and Dimitris Kotzinos(✉)

ETIS Lab, UMR 8051, CY Cergy Paris University, ENSEA, CNRS,
2 Avenue A. Chauvin, 95000 Pontoise, France
{Iheb.Meddeb,Catherine.Lavandier,Dimitrios.Kotzinos}@u-cergy.fr

Abstract. This paper proposes a classification model for opinion mining around airport noise based on techniques such as event detection and sentiment analysis applied on Twitter posts. Tweets are retrieved using the Twitter API either because of location or content. A dataset of preprocessed, with NLP techniques, tweets is manually annotated and then used to train an SVM (Support Vector Machine) classifier in order to extract the relevant ones from the obtained collections. The extracted tweets from the SVM classifier are fed to a lexicon-based classifier to filter out the false relevant and to increase precision. A lexicon-based sentiment classifier is then applied in order to separate positive, negative and neutral tweets. The sentiment classifier uses emoticons, polarity of words with subjective intensity, intensifiers, negation effect with dynamic scope, contrast effect and SWN to detect the sentiment of tweets in a hierarchical manner. The information present in the classified tweets is used for a statistical survey-like study.

Keywords: Twitter · Opinion mining · Natural language processing · Machine learning · Sentiment analysis · Text mining

1 Introduction

Microblogging has become a very popular communication intermediary these last years, such as Twitter [3], Tumblr [2], etc. Offering a social network service for people, they use it to share daily news and express their opinions or emotions towards several topics, in a completely free manner. In fact, Twitter has reached 336 million active users in the first quarter of 2018, according to Statista [1], and sharing around 500 million tweets per day. These numbers indicate the big amount of information shared and rapidly spread due to Twitter characteristics that enables 280 maximum characters in a post and introduces hashtags and usernames tagging. All of this has encouraged research in the field of data mining and natural language processing (NLP) to exploit microblogging services and especially Twitter. Different works have taken place in this context but aiming at different objectives. In our project, we aim to capture tweets shared by users who live in the area of an airport and discuss about noise problems generated by

G. Flouris et al. (Eds.): ISIP 2019, CCIS 1197, pp. 145–160, 2020.
https://doi.org/10.1007/978-3-030-44900-1_10

both air and road traffic and due (or not) to the presence of the airport and to understand their perception on the quality of life in the area. Heathrow airport is taken as an example to work with as it is one of the busiest airports and located in a highly and densely populated area.

The main goal of this project is to build a customizable platform that collects the stream of relevant tweets generated by users, store them and do the sentiment analysis. This wealth of expressed opinions though comes with a price: not all opinions, posts, discussions are relevant to a specific subject so we need first to be able to extract the relevant posts or discussions. This is not a trivial subject by itself, since the definition of a subject is not exact and the way people express themselves varies greatly. Moreover, the case of Twitter and other microblogging services is more complicated since their limit in the number of characters for each post forces people to express themselves in unique and sometimes difficult to decipher ways. So this led us to create ways to collect data automatically using information retrieval, data mining and machine learning techniques to extract the relevant posts. Additionally, we used sentiment analysis techniques in order to analyze the opinions expressed in tweets and extract the sentiment (positive, negative or neutral) involved. We hope to be able to offer an alternate method to the traditional surveying methods with an automatic and timely way. This faces several challenges such as dealing with trivial tweets, incomplete sentences, misspelling and abbreviation due to strictly short messages. Sentiment classification is a hard challenge that faces contextual meanings of messages such as irony and the use of emotional expressions. Our work can be used to survey opinions on different aspects of people's everyday lives but the Machine Learning (ML) algorithms we use, will need to be retrained in order to achieve reasonable results. So while this is not an out of the box approach, it is a complete effort to support online surveying on non-trivial subjects.

The rest of this paper is organized as follows. In Sect. 2, a study of the state of the art and related work is presented. Section 3 describes the proposed approach and the workflow for extracting sentiments about noise and quality of life from tweets. Experiments and results are shown in Sect. 4. Section 5 is the conclusion of this work and discussion of future perspectives.

2 State of the Art

2.1 Machine Learning Approaches for Sentiment Classification

Related works have mostly used emoticons [12], slangs and acronyms [11], words in text and their respective part-of-speech (POS), which is the grammatical description of word (e.g. noun, verb, adjective, etc.), intensifiers such as all caps and characters repetitions (e.g. happpyyy) [14], punctuation marks, n-grams and negation mark as features of tweets. The sentiment polarity of a tweet is, then, calculated using machine learning approaches or lexicon-based approaches.

According to [17], there are two classifier models, a 2-way and a 3-way sentiment classification. The 2-way model classifies texts into positive or negative and the 3-way model includes a neutral class with the previous ones. [12] showed that

emoticons have a significant indication on the polarity of texts with a 2-way classification and emoticon-trained SVM (Support Vector Machine) [8] and Naive Bayes (NB) [10] classifiers were able to have more than 70% accuracy. However, this method has a poor performance with a 3-way classification. [17] tested the impact of n-grams on the classifier performance. They used NB for classification and showed that using bigrams leads to the best accuracy as it provides a good trade-off between a word meanings (unigram) and capturing sentiment expressions (trigrams). They also revealed that attaching negation words when using n-grams has a high accuracy even with a small training set. [14] used collections of hashtagged tweets and tweets with emoticons to see how useful features are. They took n-grams as baseline feature, and then tried combinations of it with a dictionary of subjective lexicon, POS features such as counting of verbs, nouns, adjectives and microblogging features (e.g. intensifiers, emoticons, slangs and abbreviations). They showed that applying all features together does not lead to the best performance but it depends on the type of features. Tree kernel is also a useful method to represent tweets [4] because polar (positive/negative) and non-polar (counts) features can be easily extracted. They also detect emoticons, negation and exclamation marks, stop and non-English words within the tree kernel. Their study showed that tree kernel combined with sentiment features (e.g. positive/negative words, count and prior polarity of POS, emoticons, etc.) outperforms the base line unigram. It is also important to mention that they took into account the subjective intensity of emoticons (e.g. extremely positive, positive, negative, etc.) but not those of words. Same as [14], they used combination of features to get the most effective ones. Their feature analysis showed that combining the prior polarity words with their POS gives the best performance, contrarily to [14]. This may be explained by the tagger errors and the use of POS in [4] (prior polarity of words by POS) and in [14] (count of POS).

[18] has used a context-based convolutional neural network (CNN) to apply sentiment classification on Twitter corpus with 5 main layers: tweets are represented by word embedding vectors to be passed, then, to the input layer. The convolution layer extracts lexical n-grams information and a max, min and average-pooling layer is used to know how important an n-gram is. They also used as sub network to extract contextualized words form tweets which were represented using tf-idf. A hidden layer is used to concatenate the values from the pooling layers of the main network and the sub network, which leads to the final output layer to get the polarity of tweets. They tested their model on tweets extracted from conversations, tweets sorted by author and tweets sorted by topic. Their study showed that their model gives the best performance on topic-based tweets.

2.2 Lexicon-Based Approaches for Sentiment Analysis

Besides machine learning approaches for sentiment classification, previous works have also used lexicon-based approaches that imply the use of dictionary of subjective words. For this purpose, many dictionaries from previous sentiment analysis already exist and research continues to take advantage of them because the

creation of lexicon datasets is a time consuming task. Other than lexicon dictionaries, sentiment research works on microblogging messages have also used sets of positive and negative emoticons to detect sentiment classes, despite the fact that subjective words can be interpreted differently from one annotator to another. Moreover, even if the contents of the dictionaries (words) can be the same, their polarity might differ. To avoid these problems, [20] indicates the need of having more than one dataset to take into account multiple subjective perspectives of the word and to modify the existing dictionary, when necessary, to satisfy the topic sentiment characteristics or to create a domain specific dictionary using lexicon expansion techniques. [5] proposed a lexicon enhanced sentiment classifier on reviews to improve classification performances. In fact, they calculated the scores of positive and negative emoticons and words. The polarity score of a word is calculated using SentiWordNet classifier (SWNC) and a domain specific classifier (DSC) that takes into account the polarity of domain specific words both existing or unknown in SWNC. They also take into account negation (inverting the polarity score of the word next to the negation word) and modifiers, which are a sort of positive and negative grammatical intensifiers such as very, slightly, less, extremely, etc. They assign an intensity percentage to every modifier that represent its effect on the next word. The score of a sentence in a review is the summation of emoticons, modifiers, DSC and SWNC scores. Then a review is classified as positive, negative or neutral depending on the summation of sentences sentiment scores. Their study shows that DSC and modifiers have the best effect on improving performance and that DSC is used to give a correct classification of the misclassified neutral reviews due to the domain specific words that are nonexistent in SWNC so given a score of 0 (neutral).

2.3 Hybrid Classification Models

[13] also presented an hybrid sentiment classification framework on Twitter data. They used three different classifiers: emoticon classifier (EC), improved polarity classifier (IPC) and SWNC. Contrarily to [5], they detect the polarity of a tweet using a sequential method: After preprocessing tweets, they are passed to EC, which has positive and negative sets of emoticons. Depending on the emoticons in a tweet, EC classifies them into positive or negative. If tweet has a neutral score (i.e. does not have emoticons), it is passed to IPC which has sets of positive and negative words build from multiple existing lexicons datasets. Same to EC, the polarity of a tweet is calculated but this time, depending on words. If it is still neutral, the tweet is passed to SWNC. This algorithm has showed a good performance on classifying tweets especially on reducing the number of neutral tweets. However, they do not take into account the subjective intensity of words, negation nor modifiers.

3 Workflow for Extracting Sentiments from Tweets

Our proposed approach is presented as a workflow, which is divided into four main parts. First, queries are sent to Twitter Streaming API to collect tweets.

As the geographic area of our study is known (Heathrow airport). So we are collecting tweets using a location query to get messages within that area and also using a keywords query to get messages around Heathrow and aircraft noise. Then, messages are preprocessed using NLP techniques such as stop words removal, spelling correction, lemmatization, POS tagging, tokenization, etc. Afterwards, a machine learning algorithm, trained on an annotated dataset, is set up to filter out the irrelevant tweets and get the relevant ones. A domain knowledge classifier, which is lexicon-based, is also used to filter out irrelevant tweets. Relevant tweets are then preprocessed again because the first preprocessing task is only suited for relevance classification and does not satisfy sentiment classifier requirements The sentiment classifier uses sets of positive and negative emoticons, positive and negative lexicon with subjective intensity, and SWN to calculate the sentiment scores of tweets and to classify them into positive, negative or neutral. The use of these three classifiers is done in a hierarchical way by applying weights on their scores to have better performances.

3.1 Gathering Data: Twitter API

Twitter provides an API[1] to allow developers and researchers to access the publicly available user posts. They allow getting real time tweet streams with filtering by keywords, locations, languages, users, etc. the received tweet is represented as a JavaScript object notation (JSON) object that carries a lot of information about the tweet such as creation time, text, user description and location.

Retrieving Tweets with Location Query (TWLQ). Firstly, we define the area around Heathrow airport in which people will be talking about aircraft noise. We use the airport day, evening and night level (L_{den}) noise contours [6] to set the minimum surface of the area. We end up by defining a bounding box of 167 km wide, 73 km long and centered in Heathrow airport. The coordinates of the bounding box are introduced as a filter to Twitter API that is also configured to extract only English language tweets.

Retrieving Tweets with Keywords Query (TWKQ). The previous method gives only tweets having location, which are a small proportion of the overall accessible tweets (i.e. it misses a large number of relevant tweets that do not have a location). Moreover, it returns all tweets within that area so we get tweets talking about everything, which makes it impossible to take a sample with significant number of relevant ones for training. Therefore, we also use keywords queries to extract relevant tweets. We use "Heathrow", "LHR" and "noise" as keywords in a certain way to get tweets that have the words Heathrow and noise or LHR[2] and noise in the text.

[1] https://developer.Twitter.com/en/docs (Accessed on 08/17/2018).
[2] Airport code for London Heathrow.

3.2 Preprocessing Tweets (NLP)

Preprocessing tweets is an essential task for relevance classification and sentiment analysis. After retrieving tweets, URL links, numbers, emoticons and Twitter special words such as RT (denotes retweet) are removed. We keep usernames and hashtags as they can be informative features for relevance classification. Then the text is set to lowercase to ensure homogeneity of the following operations: Tokenization is applied to form a bag of words. Spelling errors within text are reduced by correcting intensified words (e.g. "happyyyy" becomes "happy"). Then, a POS tag is assigned to each word and the stop words are removed. Finally, lemmatization is applied to get a bag of root words that defines a tweet along with its usernames and hashtags. The preprocessed tweets will be used for relevance classification, which extracts relevant texts to be used for sentiment classification. However, this set of tasks is not very effective for sentiment analysis as they represent more the topic by the root words and so, loses the sentiment of sentences. Moreover, doing all the preprocessing in one step is not a desirable solution since the number of relevant tweets is much smaller than the number of irrelevant tweets. So the relevant tweets are preprocessed again, but differently; it starts with extracting emoticons and hashtags from text to be used later, followed by removing URL, usernames and punctuation marks. The symbol "#" is also removed from hashtags and we correct those who are composed by multiple words (e.g. hashtags "#NoisePollution" or "#noise_pollution" become "noise pollution") because words in hashtags can also be involved in the tweet's sentiment. However, the position of hashtags is not taken into account as we add all modified hashtags at the end of the tweet. The text is then set to lower case and tokenized. We use, as in the first step, the same spelling correction on each word but also detecting intensifiers such as character repetition and all caps. Words are then POS tagged and negation marks (e.g. not, 't and no) are detected. In that case, a negative mark is assigned to each of their following words. It is important to know where negation effect stops. In our case, the assignment gets back to normal when a sentence in a tweet ends. In microblogging messages, "," and "-" can also be used to end or start sentences besides normal ones such as points, exclamation and question marks. The negation scope also stops when conjunctions like "and", "or", wh-determiners (e.g. that, which), wh-pronouns (e.g. what, who), wh-adverbs (e.g. where, when) or contrast (e.g. but, however) words are found [9]. We also detect contrast in tweets as they have an effect on determining sentiments. The sets of emoticons, words with their POS and normal/negative effect and intensifiers are passed to the sentiment classifier.

3.3 Relevance Classification

After the first preprocessing part, tweets are set to be in the form of bag of root words and hashtags. We take unigrams, bigrams and hashtags as features and we used tf-idf technique to represent tweets. SVM algorithm is trained on an annotated sample of tweets, which are taken from the retrieved datasets TWLQ and TWKQ. The relevant classified tweets from SVM are introduced

to a lexicon-based classifier. This classifier uses datasets of domain knowledge unigrams, bigrams and related hashtags and usernames, which were created from manually labeled relevant tweets, to calculate a domain knowledge score of each tweet. Then, the lexicon-based algorithm classifies a tweet as relevant when its relevance score is over a threshold ϵ. Else, the tweet is classified as irrelevant. ϵ is user or experimentally defined and is application specific. Figure 1 describes the flowchart of relevance classification. The threshold ϵ is set to be low to have a small impact on missing more tweets that are relevant but an important impact on reducing the number of false relevant. This method helps to filter out false relevant tweets and to have more classification precision as relevance results will affect the results of our sentiment analysis later.

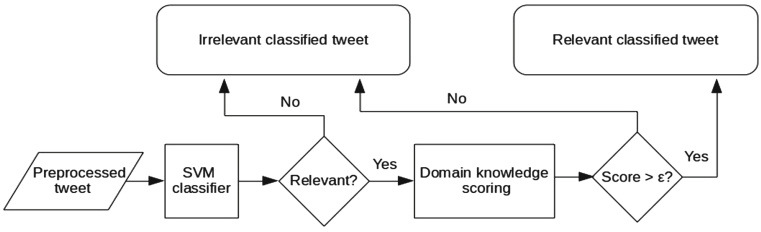

Fig. 1. Relevance classification flowchart

3.4 Sentiment Analysis of Relevant Tweets

After extracting relevant tweets from the stream, the appropriate preprocessing tasks are applied on relevant tweets. The proposed approach classifies them by their positive, negative or neutral sentiment using emoticons (Em), lexicon polarity (LP) of words and SWN. Let RT be the set of relevant tweets rt, W be the set of words w including the preprocessed hashtags and E be the set of emoticons e extracted from a tweet such as:

$$RT = \{rt_1, rt_2, ..., rt_j, ..., rt_n\} \tag{1}$$

$$W = \{w_1, w_2, ..., w_j, ..., w_m\} \tag{2}$$

$$E = \{e_1, e_2, ..., e_j, ..., e_t\} \tag{3}$$

Therefore, a relevant tweet rt is defined by:

$$rt = \{W, E\}, (rt \in RT) \tag{4}$$

Emoticon (Em) Score Calculation. Emoticons are extracted from tweets using regular expressions. We extract emoticons that are represented by punctuation marks or by Unicode. We created two datasets of positive emoticons PE and negative emoticons NE stored in files. The datasets have 64 emoticons

divided into 38 positive and 26 negative. Sentiment scores of emoticons in a tweet rt_j are normalized and scaled between 1 and −1 such as:

$$score_{Em}(rt_j) = \frac{\sum_{i=1}^{t} emscore(e_i)}{t}, (e_i \in E) \wedge (E \in rt_j) \tag{5}$$

And:

$$emscore(e_i) = \begin{cases} 1 & \text{if } (e_i \in PE) \\ -1 & \text{if } (e_i \in NE) \\ 0 & \text{if } (e_i \notin PE) \wedge (e_i \notin NE) \end{cases} \tag{6}$$

Lexicon Polarity (LP) Score Calculation. LP score calculation is based on datasets of positive and negative words. Datasets are created from multiple existing lexicon collections to expand them and to avoid misinterpretation of sentiment of certain words. Lexicon lists from Liu [15], McDonald [16] and MPQA [19] are used to create the dataset. Duplicates and words that do not have the same polarity within all datasets are removed. We have also added, when missing, some of the domain knowledge subjective words from the work in [7] such as deafening, awake, unbearable, etc. And we removed the words "noise" and "noises" because they appear in most of the tweets so they would wrongly affect the sentiment polarity. Table 1 presents the statistics of positive and negative words from each resource and those that are used. Let PW denotes the set of positive words and NW the set of negative words. As subjective intensity of words is defined in [19], we used this intensity in scoring and we set the subjective intensity of additional words from the other dataset to unknown. So the dataset has 3 descriptions of subjective intensity of words: strong subjectivity, weak subjectivity and unknown subjectivity. We also set the domain knowledge subjective words to have strong subjectivity and changed the polarity of some related words to be suited for our topic. For example, we set "low" to have negative polarity as low flying planes cause more noise. Since intensifiers, negation and contrast words are detected. We use other additional sets for the scores. Let ACI be the set of all caps intensifier scores aci and CRI be the set of character repetition intensifier scores cri of each word such as:

$$ACI = \{aci_1, aci_2, ..., aci_j, ..., aci_m\} \tag{7}$$

$$CRI = \{cri_1, cri_2, ..., cri_j, ..., cri_m\} \tag{8}$$

Table 1. Statistics of lexicon datasets

Words	Datasets				
	Bing Liu	Bill McDonald	MPQA	Clashes and duplicates	Final dataset
Positive	2006	347	2719	6548	3251
Negative	4780	2306	4919		7278

If one of these intensifier is detected in a word, its following score will be 1.5 and 1 if it is not. For example the sets ACI and CRI of the tweet "plane noise is LOUD tonight! Respiiiiite #NOIIISE" will be $\{1, 1, 1, 1.5, 1, 1, 1.5\}$ and $\{1, 1, 1, 1, 1, 1.5, 1.5\}$ respectively. Let NEG and CON be the sets of negation and contrast words respectively. As the algorithm detects the negation, contrast and negation stop marks from the preprocessing part, normal, negative or inverse effect is assigned to each word using keywords. The normal sentiment score $swscore_{LP}$ of a word w_j in a tweet rt_j is:

$$swscore_{LP}(w_j) = \begin{cases} 1 \times weight \times aci_j \times cri_j & \text{if } (w_i \in PW) \\ (-1) \times weight \times aci_j \times cri_j & \text{if } (w_i \in NW) \\ 0 & \text{if } (ur_i \notin PW) \wedge (ur_i \notin NW) \end{cases} \tag{9}$$

where $weight$ is the subjective weight of the word. Its multiplication with aci_j and cri_j indicates the impact of the word on the tweet sentiment score and its polarity. When a word has a negative effect due to negation words, its score is multiplied by -1, allowing its opposite effect to be counted rather than its normal effect. So the sentiment score of the word, in this case, will be:

$$swscore_{LP}(w_j) = (-1) \times swscore_{LP}(w_j) \tag{10}$$

This score is valid for all the words following the negation mark until a negation stop word is found or the sentence in a tweet ends. The LP score is calculated in an iterative manner, initializing it to zero and adding each time the score of the word such as:

$$LPscore = LPscore + swscore_{LP}(w_j),\ (w_j \in W) \wedge (W \in rt_j) \tag{11}$$

When a word has an inverse effect, which means it is a contrast word, the following part of the sentence often has an opposite meaning of the first part and it also indicates the overall sentiment toward a subject. So, when a word such as "but" and "however" is found in a tweet, the polarity of the current score is inverted:

$$LPscore = (-1) \times LPscore \tag{12}$$

This allows us to take into account the opposite meaning of sentence before the contrast word. After inverting the polarity, the algorithm continues to add scores of words normally. When another contrast word is found in the same tweet, the polarity will be inverted again. When all the polarity scores of words in a tweet are calculated and added to $LPscore$, it is normalized to ensure the sentiment intensity of a tweet:

$$score_{LP}(rt_j) = \frac{LPscore}{m},\ (rt_j \in RT) \tag{13}$$

SentiWordNet (SWN) Score Calculation. SWN dictionary is used for this purpose. In fact, each word in the dictionary have a positive, a negative and a

neutral score, with a total score of 1. Its scores also depend on its POS tag and so, how it is employed in a text. Each word w_j in a tweet rt_j is introduced, with its POS tag to SWN to get also its synsets, which are words having the same meaning of w_j in a particular POS, to be counted in the word polarity scoring such as:

$$syscore_{SWN}(sy_i) = posscore_{SWN}(sy_i) - negscore_{SWN}(sy_i), (sy_i \in SY_{w_j}) \quad (14)$$

where SY_{w_j} is the set of synsets of the word w_j:

$$SY_{w_j} = \{sy_1, sy_2, ..., sy_j, ..., sy_v\} \quad (15)$$

And $posscore_{SWN}$ and $negscore_{SWN}$ are positive and negative scores of a word in SWN respectively. This enables us to take into account, same as LP scoring, the sentiment intensity of a word, however, the scores are not related to the topic. After calculating the score of each synset, we take their average to get the sentiment score of the word w_j such as:

$$swscore_{SWN}(w_j) = \frac{\sum_{i=1}^{v} syscore_{SWN}(sy_i)}{v}, (sy_i \in SY_{w_j}) \wedge (w_j \in W) \quad (16)$$

And the sentiment score of a tweet rt_j by SWN scoring method is:

$$score_{SWN}(rt_j) = \frac{\sum_{i=1}^{m} swscore_{SWN}(w_i)}{m}, (w_i \in W) \wedge (W \in rt_j) \wedge (rt_j \in RT) \quad (17)$$

Sentiment Score Calculation and Classification. The sentiment analysis approach uses the three scoring methods to determine sentiment polarities of tweets. They are used in a hierarchical way using weightings of scores and priority steps. Figure 2 shows the flowchart of sentiment classification algorithm. Firstly, emoticons and LP scoring algorithms are used to identify the sentiment of a tweet rt_j such as:

$$score_{Em+LP}(rt_j) = wei_i \times score_{Em}(rt_j) + wei_2 \times score_{LP}(rt_j) \quad (18)$$

where wei_1 and wei_2 are the weights assigned to Em and LP respectively. The classifier detect the sentiment of a tweet on the basis of thresholds. Let θ_1 and θ_2 be the respective positive and negative thresholds close to zero. If $score_{Em+LP}$ of a tweet rt_j is higher than θ_1, it is classified as positive and it is classified as negative if $score_{Em+LP}$ is lower than θ_2. Otherwise, if the score is between θ_1 and θ_2, the sentiment class of the tweet is not defined yet and it is fed to SWN scoring algorithm. Let $sclass_{Em+LP}(rt_j)$ be the sentiment class of the tweet on the basis of Em and LP scores such as:

$$sclass_{Em+LP}(rt_j) = \begin{cases} \text{positive} & \text{if } score_{Em+LP}(rt_j) > \theta_1 \\ \text{negative} & \text{if } score_{Em+LP}(rt_j) < \theta_2 \\ score_{SWN}(rt_j) & \text{if } score_{Em+LP}(rt_j) \in [\theta_2, \theta_1] \end{cases} \quad (19)$$

This reduces the number of tweets that are misclassified as neutral. Same as before, the sentiment classification of tweets on the basis of SWN score is done, using thresholds. Let τ_1 and τ_2 be thresholds close to zero. The sentiment class $sclass_{SWN}(rt_j)$ of a tweet rt_j on the basis of SWN is:

$$sclass_{SWN}(rt_j) = \begin{cases} \text{positive} & \text{if } score_{SWN}(rt_j) > \tau_1 \\ \text{negative} & \text{if } score_{SWN}(rt_j) < \tau_2 \\ \text{neutral} & \text{if } score_{SWN}(rt_j) \in [\tau_2, \tau_1] \end{cases} \tag{20}$$

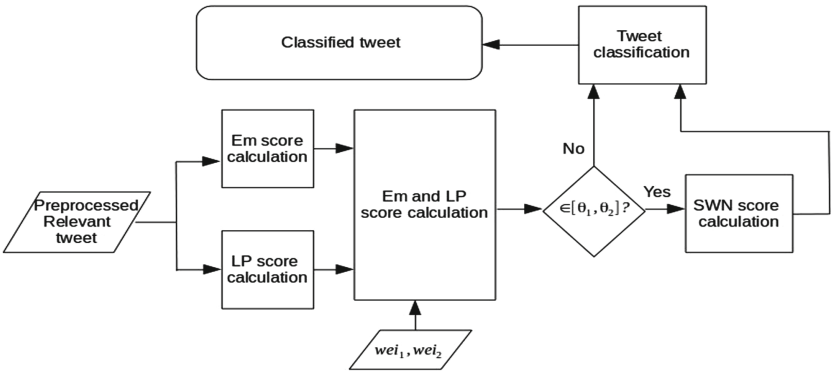

Fig. 2. Sentiment classification flowchart

4 Experimental Results

We have implemented all the workflow described in the previous section in the python programming environment. The confusion matrix, is used to analyze data and metrics such as *precision*, *recall* and $F-measure$ for each class and the *accuracy* are used to evaluate the classifiers performances. The confusion matrix is defined in Table 2. For example, *precision*, *recall* and $F-measure$ of the positive class are defined as follows:

$$precision_{pos} = \frac{T_{pos}}{T_{pos} + F_{pos\_neg} + F_{pos\_neu}} \tag{21}$$

$$recall_{pos} = \frac{T_{pos}}{T_{pos} + F_{neg\_pos} + F_{neu\_pos}} \tag{22}$$

$$F - measure_{pos} = 2 \times \frac{precision_{pos} \times recall_{pos}}{precision_{pos} + recall_{pos}} \tag{23}$$

And the accuracy is:

$$accuracy = \frac{T_{pos} + T_{neg} + T_{neu}}{All\_tweets} \tag{24}$$

Table 2. Confusion matrix

Confusion matrix		Predicted class		
		Positive	Negative	Neutral
Known class	Positive	T_{pos}	F_{neg_pos}	F_{neu_pos}
	Negative	F_{pos_neg}	T_{neg}	F_{neu_neg}
	Neutral	F_{pos_neu}	F_{neg_neu}	T_{neu}

Relevant tweets from the datasets TWLQ and TWKQ are taken to create a new dataset D_1. Tweets were labeled as positive, negative and neutral and were used for testing. Some details of D_1 are given in Table 3. As, generally, nobody likes being affected by aircraft noise and shows happy emotions towards airport noise, we defined a tweet as positive when the user shows a contrary opinion to negative tweets (e.g. "I live 10 min away from Heathrow. Noise is not disturbing, there is no air pollution.") and a tweet as neutral when it does not show a sentiment toward the topic or does not refer to airport noise (e.g. "Daytime aircraft noise was defined as that occurring between 0700 and 2300 h, and that occurring between 2300 and 0700 h was defined as night-time aircraft noise"). The numbers of positive and neutral tweets in the corpus, as showed in Table 3, are very small compared to the number of negative tweets (601 negative and 26 for each positive and neutral). Something expected, as most of the people have negative sentiments toward airport noise.

4.1 Sentiment Classifiers Comparison

We have studied the performance of the proposed classifier (PC) compared to emoticon classifier (EmC), LP classifier (LPC) and SWNC. We tuned thresholds to be suited for each classifier and fed tweets from D_1 to be classified. We set $wei_1 = 0.7$ and $wei_2 = 0.3$ so setting Em to have the priority over LP to classify tweets, when emoticons are present. We also set the weights of words that have strong subjectivity to 1, weights of words that have weak and unknown subjectivity to 0.75. The classifiers' performance is evaluated by calculating their respective confusion matrix and metrics. The results are given in Table 4 and show that our classifier outperforms the other classifiers. In fact, EmC has the worst results, having only 4.90% accuracy. Since this classifier only relies on emoticons to detect tweets sentiments, it classifies all the tweets that haven't emoticons as neutral. Table 3 shows that only 18 tweets in D_1 have emoticons and not all emoticons are recognized as some of them are neutral and others are not included in emoticon lists PE and NE. So the rest of tweets are automatically classified neutral which leads to have a big number of F_{neu} and consequently, a weak $F - measure$ and *accuracy*. However, it's still a good classifier when emoticons are present in a tweet because it captures negative tweets and shows a good *precision*. It misclassifies some of the negative tweets because of irony.

LPC has better results than EmC, with 62.17% *accuracy* and a good *recall* for negative and neutral classes (63.23% and 61.54% respectively) and

Table 3. Statistics of D_1

	Positive tweets	Negative tweets	Neutral tweets	Total	Tweets with emoticons
D_1	26	601	26	653	18

less $recall_{pos}$. Moreover, it has the best $F-measure$ results of positive and neutral tweets among all the classifiers. However, the difference between the numbers of negative tweets and the number of positive and neutral tweets has an effect on the precision of LPC. In fact, the number of the misclassified negative tweets (i.e. F_{pos_neg} and F_{neu_neg}) is higher than the total number of positive and neutral ones, leading to have low precision of positive and neutral classes with 12.50% and 8.89% respectively. The proportions of positive, negative and neutral tweets in the corpus depend on the topic, so it is part of the problem and needs to be taken into account. The 155 misclassified negative tweets as neutral F_{neu_neg} are, principally, due to the missing sentiment words in the lexicon lists PW and NE and so, most of these tweets have a score of 0 and are within the thresholds interval, consequently, they are classified as neutral. On the other hand, the misclassified negative tweets as positive are due to multiple reasons but mainly, the use of contrast in score calculation, which it does not take into account the sentence level in a tweet. SWNC is better than LPC, with 7.20% more *accuracy*. However, $F-measure$ of positive and negative classes are lower than those of LPC resulting to low values of *precision* and *recall* for each one of them. This is due to the decrease of T_{pos} and T_{neu} tweets. Additionally, F_{neu_neg} has decreased and T_{neg} has increased, compared to LPC, which leads to the increase of $recall_{neg}$. PC has the best performance with 77.79% *accuracy*.

Table 4. Experiments and results of EmC, LPC, SWNC and PC on D_1

D_1			Confusion matrix			Metrics			
			Positive	Negative	Neutral	*precision*	*recall*	$F-measure$	*accuracy*
EmC	Thresholds	Positive	0	0	26	0%	0%	-	4.90%
	$\theta 1 = \theta 2 = 0$	Negative	3	6	592	100%	1%	1.98%	
		Neutral	0	0	26	4.04%	100%	7.76%	
LPC	Thresholds	Positive	10	7	9	12.50%	38.46%	18.86%	62.17%
	$\theta 1 = 0.027$	Negative	66	380	155	96.69%	63.23%	76.46%	
	$\theta 2 = -0.001$	Neutral	4	6	16	8.89%	61.54%	15.53%	
SWNC	Thresholds	Positive	4	15	7	4.44%	15.38%	6.90%	69.37%
	$\tau_1 = 0.015$	Negative	79	443	79	95.05%	73.71%	82.64%	
	$\tau_2 = 0.005$	Neutral	7	13	6	6.52%	23.07%	10.17%	
PC	Thresholds	Positive	12	10	4	12.12%	46.15%	19.20%	77.79%
	$\theta_1 = 0.01$, $\theta_2 = -0.001$	Negative	80	491	30	95.34%	81.70%	87.99%	
	$\tau_1 = 0.015$, $\tau_2 = 0$	Neutral	7	14	5	12.82%	19.23%	15.38%	

It also has a good *precision*, *recall* and $F - measure$ compared to the other classifiers. The architecture of PC enabled us to decrease, significantly, the number of tweets classified neutral and so the number of F_{neu_neg}. It firstly, uses LP and Em scores to have a good *precision* results on positive and negative classes. Secondly, F_{neu} is decreased by classifying all the unclassified tweets with SWN score algorithm which also leads to increase T_{neg} and T_{pos}. This method, however, increases F_{pos_neg} and decreases T_{neu} with the worst $recall_{neu}$ of 19.23% but it still keeps a good $F - meausre_{pos}$ and $F - measure_{neu}$ compared to the other classifiers with 19.20% and 15.38% respectively.

5 Conclusions

This paper presents the workflow for a solution for detecting tweets relevant to a specific subject and extract their sentiments. Noise around Heathrow airport is taken as an example to work with. Tweets are retrieved using Twitter API with two methods: the first with location filter (area around Heathrow airport) and the second with keywords filter ("Heathrow", "LHR" and "noise"). Tweets are then preprocessed using a combination of NLP techniques which is suited for relevance classification. Relevant tweets towards airport noise are then extracted from the stream using SVM and a lexicon-based classifier. Relevant tweets are then preprocessed again using other combination of NLP techniques to be suited, this time, for sentiment classification. The sentiment classifier uses emoticons, lexicon polarities with subjective intensity of words, negation effect with dynamic scope, intensified words and also contrast words and SentiWordNet scores in a hierarchical way to detect the sentiments of tweets and classify them into positive, negative or neutral sentiments.

Experimental results showed that the proposed classifier outperforms the emoticon classifier, the subjective lexicon-based classifier and SWN classifier. Moreover, it still has a margin for improvement as it captures significant number of false positive tweets. As perspectives, we suggest to improve the sentiment classifier by expanding the subjective lexicon. The spelling correction needs also to be improved by replacing slang words, correcting different types of misspelling. The polarity inverting feature due to contrast can also be improved by limiting the effect at the sentence level or only count the polarity of the sentence following the contrast word to avoid misclassifications. Sentiment classes can also be divided into normal, strong or weak sentiments. Grammatical intensifiers (e.g. very, more, less, extremely, quite) can also be taken into account in further works. Finally, we plan to apply the same methodology and validate the method followed on a number of different topics, so as to demonstrate its wider applicability to the problem of exploiting social media data in order to extract people's sentiment for a particular topic of interest.

Acknowledgements. This work has been partially supported by the ANIMA project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769627. Website: https://anima-project.eu/.

References

1. Statista. https://www.statista.com/statistics/282087/number-of-monthly-active-Twitter-users/. Accessed 13 Aug 2018
2. Tumblr. https://www.tumblr.com/. Accessed 13 Aug 2018
3. Twitter. https://Twitter.com/. Accessed 13 Aug 2018
4. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R.: Sentiment analysis of twitter data. In: Proceedings of the Workshop on Languages in Social Media, LSM 2011, pp. 30–38. Association for Computational Linguistics, Stroudsburg (2011). http://dl.acm.org/citation.cfm?id=2021109.2021114
5. Asghar, M.Z., Khan, A., Ahmad, S., Qasim, M., Khan, I.A.: Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. PLoS ONE **12**(2), 1–22 (2017). https://doi.org/10.1371/journal.pone.0171649
6. Civil Aviation Authority: Heathrow airport 2016 summer noise contours and noise action plan. Technical report (2017)
7. Barbot, B., Lavandier, C., Cheminée, P.: Linguistic analysis of field surveys carried out around two French airports. Technical report (2007)
8. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
9. Farooq, U., Mansoor, H., Nongaillard, A., Ouzrout, Y., Qadir, M.A.: Negation handling in sentiment analysis at sentence level. JCP **12**(5), 470–478 (2017)
10. Harry, Z.: The optimality of Naive Bayes. In: Proceedings of Florida Artificial Intelligence Research Society Conference (FLAIRS), pp. 562–567. AAAI Press (2004)
11. Hutto, C.J., Gilbert, E.: VADER: a parsimonious rule-based model for sentiment analysis of social media text. In: Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media. The AAAI Press (2014)
12. Jonathon, R.: Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In: ACL the Association for Computer Linguistics, pp. 43–48 (2005)
13. Khan, F.H., Bashir, S., Qamar, U.: TOM: twitter opinion mining framework using hybrid classification scheme. Decis. Support Syst. **57**, 245–257 (2014)
14. Kouloumpis, E., Wilson, T., Moore, J.D.: Twitter sentiment analysis: the good the bad and the omg! In: Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media. The AAAI Press, Barcelona (2011)
15. Liu, B., Hu, M., Cheng, J.: Opinion observer: analyzing and comparing opinions on the web. In: Proceedings of the 14th International World Wide Web conference (WWW-2005). ACM, Chiba (2005)
16. Loughran, T., McDonald, B.: When is a liability not a liability? Textual analysis, dictionaries, and 10-ks. J. Finan. **66**(1), 35–65 (2011). https://EconPapers.repec.org/RePEc:bla:jfinan:v:66:y:2011:i:1:p:35-65
17. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC 2010), Valetta, Malta (2010). http://www.lrec-conf.org/proceedings/lrec2010/pdf/385_Paper.pdf

18. Ren, Y., Zhang, Y., Zhang, M., Ji, D.: Context-sensitive twitter sentiment classification using neural network. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, Phoenix (2016)
19. Theresa, W., Janyce, W., Paul, H.: Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of HLT-EMNLP-2005, pp. 347–354. The Association for Computational Linguistics, Vancouver (2005)
20. Yadollahi, A., Shahraki, A.G., Zaïane, O.R.: Current state of text sentiment analysis from opinion to emotion mining. ACM Comput. Surv. **50**(2), 25:1–25:33 (2017)