Towards the Exploitation of Statistical Language Models for Sentiment Analysis of Twitter Posts

Sukriti Bhattacharya[1](✉) and Prasun Banerjee[2]

[1] University College London, London, UK
s.bhattacharya@cs.ucl.ac.uk
[2] University of Calcutta, Kolkata, India
banerjee.prasun@gmail.com

Abstract. In this paper, we investigate the utility of linguistic features for detecting the sentiment of twitter messages. The sentiment is defined to be a personal positive or negative feelings. We built n-gram language models over zoos of positive and negative tweets. We assert the polarity of a given tweet by observing the perplexity with the positive or negative language model. The given tweet is considered to be close to the language model that assigns lower perplexity.

Keywords: Sentiment analysis · N-gram Language Model · Perplexity

1 Introduction

Sentiment Analysis is one of the interesting applications of text analytics. Although the term is often associated with sentiment classification of documents, broadly speaking it refers to the use of text analytics approaches applied to the set of problems related to identifying and extracting subjective material in text sources. In the past few years, Online Social Networks (OSNs) have been around for helping people to have virtual communities and communications without time and geographical restrictions. The data in OSN contains opinions, emotions and views of people from all corners of the globe. The exponential growth of data in social networks has resulted in a goldmine in form of wealth of data which can be subjected to mining for extracting business intelligence. Sentiment analysis is one of the open research avenues that have high-utility to exploit social media for the growth of real world enterprises. For instance Twitter tweets, when carefully analysed, can provide valuable insights that can pave way for contextual advertising. Spurred by that growth, companies and media organisations are increasingly seeking ways to mine Twitter for information about what people think and feel about their products and services. tweetfeel[1], and Social Mention[2] are just a few who advertise Twitter sentiment analysis as one of their services.

[1] http://twitdom.com/tweetfeel/.
[2] http://www.socialmention.com/.

K. Saeed et al. (Eds.): CISIM 2017, LNCS 10244, pp. 253–263, 2017.
DOI: 10.1007/978-3-319-59105-6_22

In this paper, we describe a language model based approach for classifying tweets into positive and negative sentiment. To overcome the 140-character text limitation, most of the useful information is obfuscated by abbreviations and embedded in unstructured tweet text. Therefore, we choose *n-gram* language model as it takes no advantage of the fact that what is being modelled is language, it may be a sequence of arbitrary symbols, with no deep structure, intention or thought behind them. Our method is a fairly simple variant of scoring by perplexity according to an in-domain language model. The perplexity of a language model with respect to a sample of text, $\mathtt{t}$, is the reciprocal of the geometric average of the probabilities of the predictions in $\mathtt{t}$. The language model with the smaller perplexity will be the one that assigns the larger probability to $\mathtt{t}$. Based on this hypothesis we built two language models on positive and negative tweet samples using KenLM[3] [24] natural language toolkit. We are interested in the fact how well a language model predicts the unseen tweet by calculating the perplexity. Perplexity is subject to sampling error. To make fine distinctions between language models the perplexity should be measured with respect to a large sample. The preliminary experiments, carried out on tweets from Stanford University twitter corpus [27] (containing 800,000 positive and 605,087 negative tweets, respectively), show that the perplexity of a tweet, given two Language Models calculated over positive and negative sentiment of tweets, respectively could efficiently capture the sentiment of that tweet with relatively good accuracy (78%) and low false positive rate (0.19).

The paper is organised as follows, in Sect. 2 we give a brief overview of statistical language modeling focusing on n-gram language model. In the next section we list some related work. We introduce the proposed methodology in Sect. 4. Section 5 describes the evaluation of our proposed methodology by providing the experimental set up and results. We conclude with future work in Sect. 6.

2 Statistical Language Model

Language modeling is the attempt to characterize, capture and exploit regularities in natural language. Natural language is an immensely complicated phenomenon. In addition, any medium for natural language is subject to noise, distortion and loss. The need to model language arises out of this uncertainty. Natural language can be viewed as a stochastic process. Every sentence, document, or other contextual unit of text is treated as a random variable with some probability distribution. Let $\mathcal{L} = w_1^n \overset{def}{=} w_1, w_2, \ldots, w_n$, where w_i's are the words that make up the hypothesis. This may be decomposed into a series of multiplications of conditional probabilities (chain rule), as follows:

$$P(\mathcal{L}) = \prod_{i=1}^{n} P(w_i \mid w_1^{i-1}) \tag{1}$$

[3] http://kheafield.com/code/kenlm/.

$P(w_i \mid w_1^{i-1})$ is often written as $P(w \mid h)$, where $h \overset{def}{=} w_1^{i-1}$ is called the history. The event space (h, w) is very large, and no reasonable amount of data would be sufficient to span it. It is therefore useful to introduce a mapping, $\phi(.)$ that divides the set of possible histories into K *equivalence classes*. Different language modelling techniques arise from applying different equivalence mappings.

From the information theoretic point of view, language is considered as an information source, a high-order Markov chain $\mathcal{L}$ [1], which emits a sequence of symbols w_i from a finite alphabet often called the vocabulary $\mathcal{V}$. The distribution of the next symbol is highly dependent on the identity of the previous ones. $\mathcal{L}$ has a certain inherent entropy $\mathcal{H}$, the amount of non-redundant information conveyed per word, on average, by $\mathcal{L}$. Language modelling in its modern incarnation has its roots in the pioneering work of Claude Shannon. According to Shannon's theorem [2], any encoding of $\mathcal{L}$ must use at least $\mathcal{H}$ bits per word, on average. Under this view, the goal of statistical language modeling is to identify and exploit sources of information in the language stream, so as to bring the perceived entropy down, as close as possible to its true value.

2.1 N-gram Approach

As described in Sect. 2, when estimating the probability of the occurrence of a word, w_i, given a history, h_i, it is helpful to define a mapping $\phi(\cdot)$ that divides the set of possible histories into K equivalence classes. The most frequently used approach is to distinguish equivalence classes by the most recent $N - 1$ words in the history. This implies that each word depends not on the entire history of words that come before it, but on the previous $N - 1$ words only. Therefore, the probability of a word occurring is estimated according to the following conditional distribution:

$$P(w_i \mid h_i) \simeq P(w_i \mid \phi(h_i)) = P(w_i \mid w_{i-N+1}^{i-1}) \tag{2}$$

This is known as an N-gram model. By combining the N-gram probabilities of each word in the sentence, using the product rule of conditional probabilities, we can estimate the probability of the sentence:

$$P(w_1^n) \simeq \prod_{i=1}^{n} P(w_i \mid w_{i-N+1}^{i-1}) \tag{3}$$

The choice of N is based on a trade-off between detail and reliability, and will be dependent on the quantity of training data available. A bigram ($N = 2$) language model will have larger equivalence classes, and hence fewer parameters than a trigram ($N = 3$) model. Parameters in bigram model will therefore be more reliably estimated, while a trigram model will be more precise, and will therefore be a more accurate model, provided there is sufficient training data. For the quantities of language model training data typically available at present, trigram models strike the best balance between precision and robustness, although interest is growing in moving to 4-gram models and beyond [3].

2.2 Smoothing

The key difficulty with using n-gram language models is that of data sparsity. Data sparsity makes the model inaccurate for sequences of words that appear rarely. One can never have enough training data to estimate all of the model's parameters reliably. The parameters of this model, i.e. the conditional probabilities, are estimated using the maximum likelihood criterion on a collection of training text data. The maximum likelihood estimate (MLE) for an N-gram is calculated as the count (i.e. number of times a certain N-gram was encountered in the training corpus) of a current N-gram normalised by the number of all N-grams, sharing the same N-1 history. As the sum of all n-grams sharing the same N-1 history is actually a count of the history itself, the likelihood estimation finally gets the realisation as shown in Eq. 4, where $C(\cdot)$ is an N-gram count and n is the order of an N-gram.

$$P(w_i \mid w_{i-N+1}^{i-1}) = \frac{C(w_{i-N+1}^{i})}{\sum_j C(w_{i-N+1}^{j})} = \frac{C(w_{i-N+1}^{i})}{\sum_j C(w_{i-N+1}^{i-1})} \tag{4}$$

The maximum likelihood solution is problematic as it assigns unseen N-grams a probability of 0 and does not generalize well. The problem of unseen n-grams becomes more severe when moving towards higher order n-gram models. While these models enable us to capture more context, their estimated conditional probabilities becomes more sparse and less robust. To address this problem, smoothing is used, where the probabilities of observed n-grams are discounted and the discounted probability mass is re-distributed to assign non-zero probabilities to unseen N-grams. The most widely-used smoothing algorithms for such models are, Good-Turing smoothing [4], interpolation smoothing [5,6], Katz smoothing [7] and Kneser-Ney [8] smoothing. An extensive survey of different smoothing techniques are given by Goodman [9].

2.3 Performance Measure for Language Models

When a language model is viewed as an information source, the entropy defines the best level a perfect language model which takes account of all possible linguistic information can reach. Entropy of a discrete distribution P over the event space X is defined as:

$$\mathcal{H}[P] = -\sum_{x \in X} P(x)\ log_2\ P(x) \tag{5}$$

The entropy of a distribution P measures how many bits you need on average to encode data from P with the code that is optimal for P. Entropy is the average number of 0/1 questions needed to describe an outcome from $P(x)$. If we are interested in the fact how well a language model predicts the unseen data, this should be measured by calculating cross-entropy [25] of the distribution function of $P_T(x)$ of the text, with regard of the probability function $P_M(x)$ of the model.

That is, cross-entropy is the entropy of some data T as estimated by a model M is defined as:

$$\times\mathcal{H}[T \mid\mid M] = -\sum_{x \in X} P_T(x)\ log_2\ P_M(x) \tag{6}$$

Intuitively speaking, cross entropy is the entropy of T as "perceived" by the model M. Therefore, a good model is one which has a low entropy, and hence assigns a high probability to the test text. Often, the *perplexity* [26] of the text with regard to the model is reported. It is defined as:

$$\mathsf{PP}_M(T) = 2^{\mathcal{H}(T \mid\mid M)} = P(w_1^n)^{-1/n} \tag{7}$$

There is no clear justification in the literature for the use of perplexity over cross-entropy, however perplexity has the appealing quality of describing intuitively how good a language model. Intuitively, perplexity can be roughly interpreted as the "branching factor", or the geometric average number of choices the language model has when predicting the next word. Hence, low perplexity is desirable.

3 Related Work

Wilson et al. [10] focused on phrase-level sentiment analysis to know whether a given expression is polar or neutral. The approach was to find contextual polarity for neutral, positive and negative values. Esuli and Sebastiani [11] exploited the SentiWordNet which is an existing lexical resource available for opinion mining (OM). A synset in this dictionary is associated with three numerical scores namely Pos(s), Neg(s) and Obj(s). Narayanan et al. [12] used conditional sentences, for instance "if your Nokia phone is not good, buy Samsung phone" for sentiment analysis. Authors in [16] explored contextual advertising based on the results of sentiment analysis. They used sentiment detection for identifying contexts and do advertising in a web based applications. This is known as Sentiment-Oriented Contextual Advertising (SOCA). This framework combines both contextual advertising and sentiment analysis in order to achieve best way of using advertisements in web based applications. In [13] authors focused on classification of sentimental polarities. They compared both blog and review data for empirical study. They performed Information Retrieval (IR) based topic analysis. Polarity classification is used for sentiment analysis. A lexicon-based method is proposed for sentiment analysis in [14]. They used a calculator known as Semantic Orientation CALculator (SO-CAL) by using a dictionary of words. The dictionary contains words and corresponding SO values. Wu and Ren [15] state that social networking web sites are producing huge amount of data that can provide a wealth of information pertaining to user behaviour. They designed models for sentimental analysis of Twitter posts. Especially they focused on users influenced and influencing probabilities. They classified the probabilities into negative and positive influencing probabilities. Zhang et al. [17] focused on sentiment analysis for stock trend forecasting. Bayesian model is used for classification. Authors in [18] proposed construction of dictionary for sentiment analysis using semi-supervised learning approach. Stock market news dataset is used for sentiment analysis. Guptha and Shalini [19] used blogs and forums in order to identify Indian railway performance and perform sentiment analysis on it for improving the performance of Indian railways. A polarity dictionary is produced in order to have classification of tweets in order to find whether a sentiment is positive or negative or neutral. Kontopoulos et al. [20] used Onto-Gen for visualising ontology for sentiment analysis based on the tweets collected from Twitter. They stated that ontology is machine-readable conceptualisation of shared content. [21] explored real time event detection by analysing Twitter posts. Especially their methodology throws light into real-time event detection pertaining to earth-quake reporting. Authors in [22] employed machine learning techniques for sentiment classification. They used Naive Bayes and Support Vector Machine (SVM) for the classification. Both are the supervised learning methods. [23] proposed a methodology based on ontology. It is named as Ontology-based Sentiment Analysis Process for Social Media Context (OSAPS).

4 Methodology

The proposed framework shown in Fig. 1 mainly consists of 3 phases,

1. Preprocessing.
2. Training.
3. Classification.

4.1 Preprocessing

Tweets contain colloquial dialect, or informal languages. In this phase the system executes a series of commands to clean text, removes punctuation, special characters, embedded HTTP links, extra spaces, digits, tag Patterns, ‘@’ patterns (A de facto standard to include the @ symbol before the username) and the emoji patterns. It also converts the upper case letters to lower case.

4.2 Training

The cleaned tweets are then divided into two separate zoos based on their polarity - positive and negative feelings, respectively. The KenLM language modeling toolkit was applied to train and binarise the clean tweets. We have used KenLM to model two different language models based on the sentiment polarity (positive and negative, respectively) using a preferably large set of training tweets comprised of different values of n-grams ($n = 2$ to 6).

KenLM [24] developed by Kenneth Heafield, implements two data structures that allow very efficient language queries, a fast probing hash table, and a more compact but slower trie. The KenLM library packs use the Kneser-Ney smoothing method. Knesser-Ney Smoothing is helpful for generalisation and for testing when the test data has words not already present in the vocabulary.

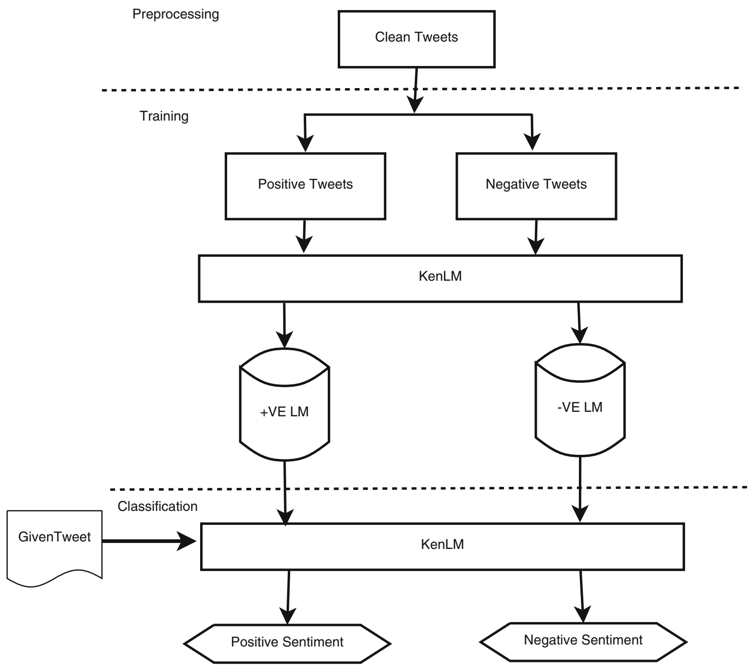


Fig. 1. Sentiment analysis framework

4.3 Classification

Given the parameters in Table 1, the system asks the question "*In which zoo is* t *more likely to belong?*". Using perplexity measure the classifier answers the natural question "*Considering our language models as tweet sentiment analyser, which is more likely to generate our suspect tweet* t*?*".

Table 1. Parameters of our classifier

t	given tweet to classify
$\mathsf{q_p}$	n-gram model built from zoo of positive tweets
$\mathsf{q_n}$	n-gram model built from zoo of negative tweets

Therefore, the sentiment of a given tweet t will be determined as positive (or negative) by the perplexity measure (Eq. 7) based classifier, $\mathsf{PP}(\mathsf{t}, \mathsf{q_p}, \mathsf{q_n})$ defined as follows:

$$\mathsf{PP}(\mathsf{t}, \mathsf{q_p}, \mathsf{q_n}) = \begin{cases} \text{Positive} & \text{if } \mathsf{PP}_{\mathsf{q_p}}(\mathsf{t}) < \mathsf{PP}_{\mathsf{q_n}}(\mathsf{t}) \\ \text{Negative} & \text{if } \mathsf{PP}_{\mathsf{q_n}}(\mathsf{t}) < \mathsf{PP}_{\mathsf{q_p}}(\mathsf{t}) \end{cases} \tag{8}$$

5 Evaluation

In this section we provide the detailed results of the experiments carried out using the proposed framework.

5.1 Experimental Setup

We evaluated our classifier against 1405087 tweets from Stanford University Corpus[4] [27]. The tweets are labeled positive or negative according to the emoticon. For example, emoticons such as ':)' are considered positive labels of the tweets and emoticons such as ':(' are used as negative labels. In this experiment we considered only positive and negative tweets and we ignored all neutral tweets, the exact numbers are shown in Table 2.

Table 2. Corpus description

	+ve Tweets	−ve Tweets
Train set	533334	403392
Test set	266666	201695
Total	800,000	605087

In this experiment we used two-third of the original dataset to train the language model and the remaining one-third to test (Table 2). We train two different language models for both the positive and negative training corpus. The only parameter is the size of the n-gram, i.e. the value of n. We chose $n = 2$ to 6 for each corpus to strike a balance between memory use and providing sufficient information to produce good results.

KenLM tool was used to generate the language models. KenLM uses a smoothing method called modified Kneser-Ney. It is a very memory and time efficient implementation of Kneaser-Ney smoothing and officially distributed with

Table 3. Time and memory consumption for +ve language model construction

n	Size (MB)	Time (≈ mins)
2	83.5	2
3	282.4	3.5
4	583.8	8
5	947.7	12.5
6	1325.0	20

[4] http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip.

Table 4. Classifier accuracy and false positive

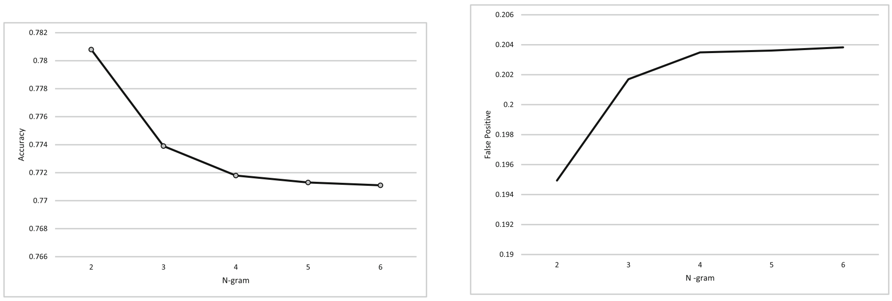
n	Accuracy	False positive
2	0.7808	0.195
3	0.7739	0.202
4	0.7718	0.203
5	0.7713	0.204
6	0.7711	0.204

Moses[5]. For ARPA file format the size of the languages models and the computation time is shown in Table 3. The time consumption depends on two factors, the order of n-gram language model and the size of the corpus in terms of individual file size. On the other hand the memory consumption does not depend of the corpus size or individual file size in the corpus. It only depends on the order of the language model.

5.2 Results and Discussion

In a pilot experiment with gram size varies from 2 to 6 and on the dataset presented in Table 2, we obtained results shown in Table 4. The highest accuracy of 78% is reached for bi-gram language model. It is interesting to see that the accuracies calculated with rest of the gram values (n = 3 to 6) are almost similar to each other, as shown in Fig. 2(a). The false positive rates shown in Fig. 2(b) is not satisfactory either.

Although, we achieved the best false positive rate (0.19) for bi-gram language model but it does not show any significant improvement comparing to other gram values. We assume that this is due to the fact that during preprocessing



(a) Accuracy

(b) False Positive

Fig. 2. Classifier accuracy and false positive

5 http://www.statmt.org/moses/?n=FactoredTraining.BuildingLanguageModel.

of tweets (Sect. 4.1) we removed some important emojis and special characters. We believe, both the accuracy and false positive would be far better if we were to carry out different information theoretic measures to clean the tweets during preprocessing phase.

6 Conclusion

In this paper we have explored the utility of language models and perplexity, a measure to determine the coverage of a language model given a text, for analysing the sentiment of a tweet with a reference corpus. The current finding shows the sentiment analysis can be achieved with the use of language models, especially in the field of social customer relationship managements (CRM) where customer feedbacks on several aspects can be categorised as positive and negative sentiments. We have provided empirical evidence that the training set i.e., the positive and negative tweet samples used to construct a language model has a large influence on the resulting accuracy. Sentiment analysis over twitter data faces several new challenges due to typical short length and irregular structure. Although, finding new methods for such analysis is the main challenge, but the performance of such analysis depends on identifying new sets of features to add to the trained model for sentiment identification, that includes hashtags, emoticons and the presence of intensifiers such as all-capital letters and character repetitions etc. A major determinant of the future direction of this research may lie with a clear definition of such feature identification, at the same time we will explore even richer linguistic analysis, for example, parsing, semantic analysis and topic modeling. Finally, we hope that the methodology and the results presented in this paper will provide a benchmark for future proposals and research efforts.

References

1. Abramson, N.: Information Theory and Coding. McGraw-Hill, New York (1963)
2. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423 (Part I), 623–656 (Part II) (1948)
3. Placeway, P., Schwartz, R., Fung, P., Nguyen, L.: The estimation of powerful language models from small and large corpora. In: ICASSP 1993, pp. 33–36 (1993)
4. Good, I.J.: The population frequencies of species and the estimation of population parameters. Biometrika **40**(3–4), 237–264 (1953)
5. Mori, S., Nishimura, M., Itoh, N.: Word clustering for a word bi-gram model. In: The 5th International Conference on Spoken Language Processing, Incorporating The 7th Australian International Speech Science and Technology Conference, Sydney Convention Centre, Sydney, Australia, 30th November–4th December 1998
6. Furui, S., Ichiba, T., Shinozaki, T., Whittaker, E.W.D., Iwano, K.: Cluster-based modeling for ubiquitous speech recognition. In: 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, pp. 2865–2868, 4–8 September 2005
7. Katz, S.: Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans. Acoust. Speech Signal Process. **35**(3), 400–401 (1987)

8. Kneser, R., Ney, H.: Improved backing-off for M-gram language modeling. In: 1995 International Conference on Acoustics, Speech, and Signal Processing, Detroit, Michigan, USA, pp. 181–184, 8–12 May 1995
9. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. Comput. Speech Lang. **13**(4), 359–393 (1999)
10. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing contextual polarity in phrase-level sentiment analysis, pp. 347–354. Association for Computational Linguistics (2005)
11. Esuli, A., Sebastiani, F.: SENTIWORDNET: a publicly available lexical resource for opinion mining, pp. 1–6. ACM (2005)
12. Narayanan, R., Liu, B., Choudhary, A.: Sentiment analysis of conditional sentences. In: Conference on Empirical Methods in Natural Language Processing, pp. 180–189 (2009)
13. Liu, F., Wang, D., Li, B., Liu, B.: Improving blog polarity classification via topic analysis and adaptive methods. In: The 2010 Annual Conference of the North American Chapter of the ACL, pp. 1–4 (2010)
14. Taboada, M., Brooke, J., Tofiloski, M., Voll, K., Stede, M.: Lexicon-based methods for sentiment analysis, vol. 37, pp. 1–42. Association for Computational Linguistics (2011)
15. Wu, Y., Ren, F.: Learning sentimental influence in twitter. In: International Conference on Future Computer Sciences and Application, pp. 1–4 (2011)
16. Fan, T.K., Chang, C.H.: Sentiment-oriented contextual advertising. Knowl. Inf. Syst. **23**, 321–344 (2010)
17. Zhang, K., Li, L., Li, P., Teng, W.: Stock trend forecasting method based on sentiment analysis and system similarity model. In: The 6th International Forum on Strategic Technology, pp. 1–5 (2011)
18. Mizumoto, K., Yanagimoto, H., Yoshioka, M.: Sentiment analysis of stock market news with semi-supervised learning. In: IEEE/ACIS 11th International Conference on Computer and Information Science, pp. 1–4 (2012)
19. Gupta, A.R., Shalini, L.: Improvisation of experience of Indian railways using sentimental analysis. Int. J. Comput. Appl. **66**, 16–18 (2013)
20. Kontopoulos, E., Berberidis, C., Dergiades, T., Bassiliades, N.: Ontology-based sentiment analysis of twitter posts. Expert Syst. Appl. **40**(10), 4065–4074 (2013). Elsevier
21. Sakaki, T., Okazaki, M., Matsuo, Y.: Tweet analysis for real-time event detection and earthquake reporting system development. IEEE Trans. Knowl. Data Eng. **25**(4), 1–13 (2013)
22. Wawre, S.V., Deshmukh, S.N.: Sentiment classification using machine learning techniques. Int. J. Sci. Res. **5**(4), 1–3 (2016)
23. Thakor, P., Sasi, S.: Ontology-based sentiment analysis process for social media content. Procedia Comput. Sci. **53**, 199–207 (2015). Elsevier
24. Heafield, K.: KenLM: Faster and smaller language model queries. In: Sixth Workshop on Statistical Machine Translation (2011)
25. Rosenfeld, R.: A maximum entropy approach to adaptive statistical language modeling. Comput. Speech Lang. **10**, 187–228 (1996)
26. Jelinek, F., Mercer, R.L., Bahl, L.R., Baker, J.K.: Perplexity-a measure of the difficulty of speech recognition tasks. J. Acoust. Soc. Am. **62**, S63 (1977)
27. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. In Final Projects from CS224N for Spring 2008/2009 at the Stanford Natural Language Processing Group (2009)