Lexicon Induction for Interpretable Text Classification

Jérémie Clos and Nirmalie Wiratunga

Robert Gordon University
Garthdee Road
Aberdeen, United Kingdom
{j.clos,n.wiratunga,s.massie}@rgu.ac.uk

Abstract. The automated classification of text documents is an active research challenge in document-oriented information systems, helping users browse massive amounts of data, detecting likely authors of unsigned work, or analyzing large corpora along predefined dimensions of interest such as sentiment or emotion. Existing approaches to text classification tend toward building black-box algorithms, offering accurate classification at the price of not understanding the rationale behind each algorithmic prediction. Lexicon-based classifiers offer an alternative to black-box classifiers by modeling the classification problem with a trivially interpretable classifier. However, current techniques for lexicon-based document classification limit themselves to using either handcrafted lexicons, which suffer from human bias and are difficult to extend, or automatically generated lexicons, which are induced using point-estimates of some predefined probabilistic measure in the corpus of interest. This paper proposes LEXICNET, an alternative way of generating high accuracy classification lexicons offering an optimal generalization power without sacrificing model interpretability. We evaluate our approach on two tasks: stance detection and sentiment classification. We find that our lexicon outperforms baseline lexicon induction approaches as well as several standard text classifiers.

Keywords: text classification, lexicon induction, sentiment analysis, stance classification

1 Introduction

Text classification is a core task in natural language processing, with applications ranging from web search to author detection. For example, support vector machines [11], a common and extremely powerful classification algorithm [10] have helped improve document navigation tasks by categorizing web search results [4], analyzed corpora to identify anonymous authors [7], and are used to identify spam e-mails [8] at large scale. However, supervised classification algorithms suffer from not providing predictions that can be explained. Understanding the reason behind a classification allows us to establish trust in further predictions, which can have far-reaching consequences in algorithms deployed in production systems such as search engines and document categorization pipelines. Lexicons attend to this need by offering a white-box approach to text mining. They do so by using an additive model, where the probability of an instance belonging to a class is a weighted sum of the probabilities of each term belonging to that class.

A lexicon prediction can thus be interpreted by observing the terms that are contained in the instance and the terms which have contributed the most to the prediction, and it is possible for a human agent to modify the model in order to correct a mistake without restarting the learning process entirely. Figure 1 illustrates the explanation step with an example.

However, current techniques used to build those lexicons are lacking in many respects compared to standard supervised text classifiers. This paper attempts to conciliate lexicon-based classification and traditional classification models by defining a simple and effective training procedure that can generate lexicons with a classification accuracy that is competitive with modern classification algorithms. Firstly, they use point estimates of text statistics (such as raw co-occurrence or mutual information) in order to build a lexicon that is susceptible to overfitting. Secondly, they perform significantly worse than black-box models.

Example 1. In a binary sentiment classification setting, for a given sentence "*I love horror books*", a lexicon $\mathcal{L}$ referred on the figure, the lexicon could find an aggregated score of $f(love) \times 1.0 + f(horror) \times 0.3 + f(books) \times 0.5 = 1.8$ for the *Positive* class, and $f(love) \times 0 + f(horror) \times 0.7 + f(books) \times 0.5 = 1.2$ for the *Negative* class, where f is a function measuring some notion of local term frequency. The decision function $\mathcal{D}$ would then return the class with the maximum value, i. e., *Positive*. A human reader can read the sentence and identify that the term "*love*" is responsible for tipping the classification towards the *Positive* class.

Example Lexicon		
Term	*Positive*	*Negative*
love	1.0	0.0
horror	0.3	0.7
books	0.5	0.5

Fig. 1: Classification and explanation with a sentiment lexicon

We first formalize the concept of lexicons and explore the state of the art in the domain of lexicon-based classification. We then detail our contribution, formalizing lexicon-based classification as a form of computational graph and inducing optimal weights using a regularized objective function. We then detail our evaluation protocol on two classification tasks: stance detection and sentiment classification. We perform an evaluation against standard lexicons and baselines found in the literature and report that our approach significantly outperforms standard text classification techniques. Finally, we analyze and discuss our results, before exploring the next steps of our work.

2 Related works

Despite its widespread use in real-world applications, text classification heavily relies on black-box models offering little if any explanation on their predictions [21]. Lexicon-based classifiers overcome this limitation by constraining the classification to a simple model: each term/class pair is linked to a score, and a new instance gets assigned a score for each class corresponding to a sum of those scores weighted by the frequency of the corresponding term.

, those scores get weighted according to the frequency of that term in the instance and then added together, and finally the class with the highest total score for a given instance is chosen as the prediction. Such a classification model offers the flexibility of transparency: each prediction can be explained trivially by analyzing the terms that were present in the text, and any domain expert could revise the model manually with a simple text editing software. This transparency however comes at the cost of some classification accuracy, due to the simplistic nature of its inference scheme.

2.1 Lexicon-based classification

Lexicons are linguistic tools for the automated analysis of text. Their most notorious uses are classification and feature extraction [5, 2]. They can take many forms, the most common of which is a simple list of terms associated to a certain class of interest. Classification is done by counting the number of terms belonging to each list in a given unlabeled instance, and returning the class associated to the list with the most occurrences. Optionally, the terms can be weighted according to their strength of association with a given class. Some lexicons also contain additional contextual information in order to help their users build more complex models [17], but they all share the same architecture:

Definition 1 (formal classification lexicon). *A classification lexicon Lex is a tuple $Lex = \langle \mathcal{L}, \mathcal{A}, \mathcal{D} \rangle$ where:*

$$\begin{aligned} \mathcal{L} &: T \times C \mapsto \mathbb{R} \\ \mathcal{A} &: \mathbb{R}^n \mapsto \mathbb{R} \\ \mathcal{D} &: \mathbb{R}^n \mapsto \mathbb{R} \end{aligned}$$

For a given dictionary of terms T and set of classes of interest C, $\mathcal{L}$ is a mapping function that assigns an unbounded value to each pair (t, c) where term $t \in T$ and class $c \in C$. The function $\mathcal{A}$ is an aggregation function that accumulates scores and returns one value, and $\mathcal{D}$ is a decision function that selects and returns a single one of these aggregated values. Concretely, the mapping determines an evidence score for each term using a look-up list (the lexicon), propagates it to the aggregation function which aggregates the evidence into one cumulative score per class. Finally, the decision function evaluates each score to select the one that is the most likely. Figure 1 provides an example of the classification process.

We therefore define a core challenge in lexicon-based classification: the lexicon induction problem. The next section reviews techniques traditionally used to solve the lexicon induction problem.

Definition 2 (lexicon induction problem). *The **lexicon induction problem** is the estimation, given aggregation function $\mathcal{A}$ and decision function $\mathcal{D}$, of the optimal function $\mathcal{L}$ so that the resulting lexicon $Lex = \langle \mathcal{L}, \mathcal{A}, \mathcal{D} \rangle$ minimizes its classification errors on unseen data.*

2.2 Lexicon induction techniques

Research in lexicon induction outlines multiple families of techniques that can be used to produce a computational lexicon. Those techniques are either built on an extensive lexical resource such as an ontology, or on an estimation of strength of association between each term and a class in a reference corpus. Research has shown that merging multiple lexicons produces a reliable feature extractor to augment an existing classifier [27], but using those lexicons for direct classification was not explored.

Traditional hand-crafted lexicons (THCL) Due to the computational cost of building a lexicon from text, early lexicons were hand-crafted by domain experts [24] and while higher performance in automated classification tasks has been shown using modern techniques, there still exist handcrafted lexicons in use to this day such as the Linguistic Inquiry and Word Count lexicon [20]. The strengths of these approaches are that they generalize well and are highly interpretable due to their human (and not algorithmic) origin. Conversely their weakness are that they tend to be small due to the human labor involved in generating them, and less effective than other methods due to their focus on human interpretability. However they can provide a commonsense knowledge back-up in hybrid lexicons [16] with some degree of success.

Ontology-based lexicons (OBL) OBL learning techniques use a few human-provided seed words for which the class is known, and leverage some external relationship (typically synonymy, antonymy and hypernymy) in a semantic graph such as WordNet [15] to propagate class values along that graph [9]. Because this family of techniques is extremely foreign to the one we are proposing, we do not evaluate against it and only refer to it for the sake of exhaustiveness.

Corpus statistic-based lexicons (CSBL) CSBL learning techniques use a labeled corpus of interest in order to learn a domain-specific lexicon. The two main statistics used for this purpose are the conditional probability (equation 1) of observing a term given a class, and the pointwise mutual information (PMI, equation 2) between the observation of a term and the observation of a class. These approaches are flawed in that they can overemphasize spurious correlations between terms and classes. For example, if a non-class specific term such as "Monday" accidentally co-occurs too often within one class, it will be misconstrued as being indicative of that class, and the lexicon will overfit. Bandhakavi et al. [1] describe a method for building conditional probability-based lexicons and Turney [25] an approach using PMI and an external search engine to compute lexicon scores. Other works [6] have shown some improvement using the normalized PMI measure (NPMI, equation 3) on a stance classification task.

$$P(t;c) = \frac{p(t|c)}{\sum_{i=0}^{|C|} p(t|c_i)} \tag{1}$$

$$PMI(t;c) = \frac{\log(p(t;c))}{p(t)p(c)} \tag{2}$$

$$NPMI(t;c) = \frac{\frac{\log(p(t;c))}{p(t)p(c)}}{-\log\left[p(t;c)\right]} \tag{3}$$

3 Lexicon induction by backpropagation

CSBL learning techniques traditionally use point estimates of some statistical values on a corpus. Assuming $F^{n \times 1}$ is a $n \times 1$ matrix containing the frequencies of each of the n terms in instance x and $W^{c \times n}$ is a $c \times n$ matrix indexed by a class c and containing an association score for each term-class pair, we define the classification step of a lexicon in the following way:

$$Prediction(x) = ArgMax_c\,(W_c \cdot F) \tag{4}$$

We can observe that a standard lexicon is a computational graph, i. e., a composition of functions, as shown in figure 2 illustrating the network topology of a binary classification lexicon. This allows us to use gradient-based learning techniques such as backpropagation in order to solve the lexicon induction problem. The details of the network topology and the training protocol are explained in the following sections.

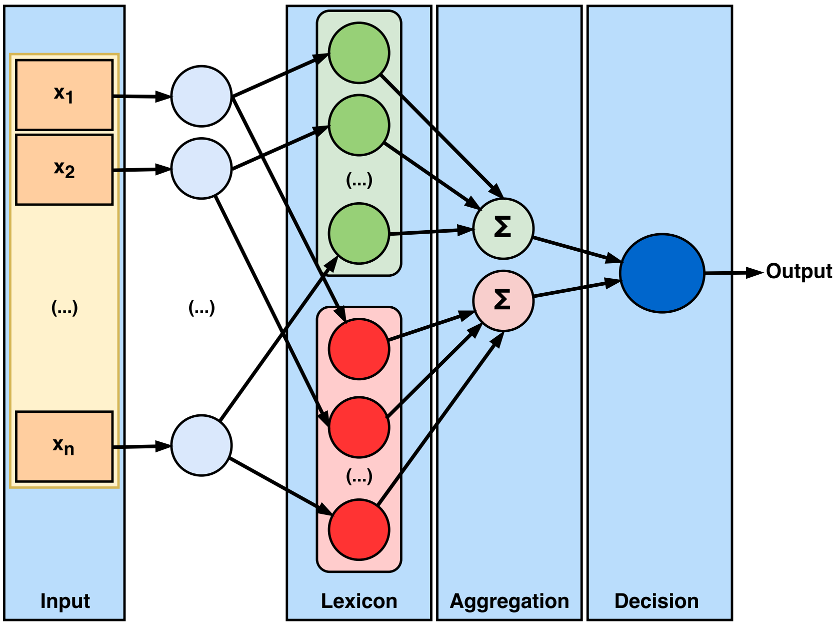

Fig. 2: The LEXICNET network topology

3.1 The Lexicon network topology

The lexicon network topology corresponds to a shallow network with linear units (the lexicon layer), where one regressor is trained per class and the output of each regressor (the aggregation layer in figure 2) is fed into a SoftMax normalization layer (the decision layer in 2) so as to produce a probability distribution as a final output, which is necessary to backpropagate the error gradient to find the optimal lexicon weights. In this section we review each layer of the neural lexicon and their function.

The vocabulary input layer The input layer feeds term frequencies into the network. The output of this layer is a $n \times 1$ matrix F (see equation 4) where n is the number of terms in the lexicon. The inputs can be logarithmically scaled to smooth out the differences in input length using the $\mathrm{ScaledFrequency}$ function where $\mathrm{RawFrequency}$ corresponds to the number of times a term has appeared in the current input. More complex scaling functions are typically applied in text classification

$$\mathrm{ScaledFrequency}(t) = Log(1 + \mathrm{RawFrequency}(t)) \tag{5}$$

The lexicon layer The lexicon layer maps a term to its respective class-dependent scores. This layer is represented by a $c \times n$ matrix W (see equation 4) where n is the number of terms in the lexicon and c the number of classes.

The aggregation layer The aggregation layer adds up evidence towards a class from a list of units in the previous layer by performing an inner product between $n \times 1$ matrix F and $c \times n$ matrix W. The output of this layer is a $1 \times c$ row vector O containing the aggregated scores for each of the classes.

The decision layer The decision layer transforms the row vector O into a probability distribution using the SoftMax function [3] and returns it as the output of the network. At testing time, the decision layer returns the $ArgMax$ of the probability distribution in the output in order to compute the accuracy of the current model. At training time, it returns only the probability distribution since the $ArgMax$ function is not differentiable, which is a required feature for the backpropagation algorithm.

3.2 Lexicon network training

We train the LEXICNET network using an Elastic Net-regularized average cross-entropy error function. In this section we detail our training procedure and cost function.

Cost function and regularization The backpropagation algorithm relies on reverse-mode differentiation in order to train the network in a computationally efficient way, by updating the weights of the units based on the error gradient with respect to those weights. We transform the label of each instance into a probability distribution vector of length n where $Y_i = 1$ for the relevant class and $Y_i = 0$ otherwise in order to use the average cross-entropy cost function (function E detailed in equation 6).

$$E(Y, \hat{Y}) = -\frac{1}{m} \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} \Big(Y_{i,j} \times log(\hat{Y}_{i,j}) + (1 - Y_{i,j}) \times log(1 - \hat{Y}_{i,j}) \Big) \tag{6}$$

Given a set of predictions $\hat{Y}$ and their corresponding ground truth labels Y, the average cross-entropy function iterates over each pair $(y \in Y; \hat{y} \in \hat{Y})$ where both y and $\hat{y}$ are probability distributions over n classes and computes their cross-entropy, which is then averaged over the m instances. However, optimizing over a direct function of the error with a large amount of free parameters (number of classes × number of lexicon entries) will lead to overfitting on the training data and poor performance on the test data, which indicates the need to regularize our training process. To counter that effect a regularization term is added to the optimization process using Elastic Net regularization [30] which has been shown to work on neural networks architectures [13].

The cost function J resulting is shown in equation 7.

$$J(\theta, X, Y) = E(Y, h(X)) + \lambda * \left(\alpha * \sum_{j=0}^{m} |w_j| + (1-\alpha) * \sqrt{\sum_{j=0}^{m} w_j^2} \right) \tag{7}$$

Here we can observe the presence of two different regularization parameters: λ corresponds to a regularization weight, which modulates the importance that we are putting on obtaining a generalizable lexicon against having a low error in the training set and is selected empirically, α corresponds to the elastic weight which weights the importance put on minimizing respectively the $L_1-\mathrm{norm}$ or the $L_2-\mathrm{norm}$ of the lexicon weights, w_i corresponds to the weight of unit i in the lexicon layer.

Optimization We train our network using Backpropagation and the full batch Gradient Descent algorithm with Nesterov momentum [18] as described in equation 8, where μ is the velocity scaling parameter and γ is the learning rate. Nesterov momentum works by updating each weight in two steps: firstly using a scaled version of their previous update (conditioned by a fixed velocity parameter), followed by a course correction step using the error gradient after the first update, mimicking the effect of momentum in physical objects.

$$\begin{aligned} w_i &\leftarrow w_i - \mu \times u_i \\ u_i &\leftarrow \frac{\partial J}{\partial w_i} \\ w_i &\leftarrow w_i - \gamma \times u_i \end{aligned} \tag{8}$$

4 Experiments

We evaluated our approach using a 10-fold cross-validation [23] on multiple tasks. We used standard TF-IDF term weighting [22] for the supervised classifiers because of it is shown to be competitive with more complex weighting methods for text classification where there is little to no term filtering [29]. We measured the performance of our algorithm against standard supervised classifier baselines and lexicon induction techniques using the accuracy performance metric (percentage of test instances correctly classified). The rest of this section details the datasets and baseline algorithms used in our experiments.

4.1 Datasets

We performed our evaluation on two tasks, containing a total of 4 datasets for stance classification and sentiment analysis. Class statistics for each dataset can be found in table 1. We describe the tasks and datasets associated in the rest of this section:

- **Stance detection** is the study of local stance of a document with respect to a topic or another stance. For example, if the topic of discussion is "death penalty" and a document d_1 is for the death penalty, then a document d_2 that is against the death penalty is said to be in disagreement with document d_1, while a document d_3 that is also for the death penalty is said to be in agreement with document d_1. Stance classification is the classification of unseen documents with respect to a topic or an existing stance. In this work, we consider a reduced version of the stance classification task with an unobserved topic, which means that the classifiers do not have any contextual information.
  - **The IAC dataset** is a subset of the Internet Argument Corpus [26] containing forum comments crawled from 4FORUMS[1] on different topics: e. g., politics, . . . and labeled on a scale from -5 to 5. A subset of comments that ensured disjoint class membership (with an average score far from 0) and containing more than 3 words was binned into 2 classes (agreement and disagreement) and used for our experiments.
  - **The CD dataset** is a dataset collected from the CREATEDEBATE forum[2] dedicated to social argumentation on political and religious topics and labeled using 2 classes (agreement and disagreement).
- **Sentiment classification** is the study of the sentiment (positive or negative) contained within a piece of text. While many datasets propose finer-grained sentiment classes (positive, negative, and neutral or numerical gradation of sentiment) we chose to use a binary sentiment classification task as the object of our study.
  - **The AYI dataset** was collected from Amazon[3], Yelp[4] and IMDB[5] and was built from individual sentences from product, location and movie reviews (respectively), labeled with a binary positive/negative judgment.
  - **The AMZ dataset** was collected from Amazon user reviews and labeled with a binary positive/negative judgment. The dataset is provided with preprocessed unigrams and bigrams. Only the unigrams are used for our experiments, so as to be more similar to the other tasks and datasets.

[1] `http://www.4forums.com`
[2] `http://www.createdebate.com`
[3] `http://www.amazon.com`
[4] `http://www.yelp.com`
[5] `http://www.imdb.com`

	Dataset			
	Stance		Sentiment	
	IAC	CD	AYI	AMZ
Number of instances	3,910	4,902	3,000	8,000
Frequency of the *agreement/positive* class	1,955	2,912	1,500	4,000
Frequency of the *disagreement/negative* class	1,955	1,989	1,500	4,000
Minimum instance length (in words)	10	10	43	10
Maximum instance length (in words)	3245	6,685	12,220	79
Average instance length (in words)	68.41	74.10	588.84	13.55

Table 1: Class statistics of datasets IAC, CD, AYI and AMZ.

4.2 Baselines

We used two families of baselines as comparison points with our approach:

Lexicons: Two lexicons used as a baseline are the CPBLEX and the PMILEX, which are standard methods for building lexicons for other purposes, such as sentiment lexicons [12]. Section 2.2 on corpus-based lexicons details their implementation. These algorithms were provided tf-idf normalized raw frequencies, which has been shown to be a competitive term weighting scheme for text classification [14] ;

Standard classifiers: SVM (with a RBF kernel), which has been shown to perform well in stance detection tasks by Yin et al. [28] and is a regular top performer in general classification tasks [10], and NAIVEBAYES and DECISIONTREE which are two popular baselines for text classification. Parameters for the classifiers were taken from the default recommendations of the SCIKIT-LEARN[6] [19] library. Tf-idf was also used here due to the disparity in document length shown in table 1.

5 Results and discussion

Table 2 shows that LEXICNET outperforms all the baselines by a statistically significant margin (on a one-tailed paired T-test, with $p < 0.05$) except for the AYI dataset where decision trees are the top performing algorithm and the AMZ dataset where the performance improvement comes close to statistical significance but does not reach it. The parameter values for LEXICNET were determined empirically on a hold-out dataset, leading to the choice of a lexicon size of 400 words (the 400 most frequent words in the corpus) and a regularization coefficient (λ) of 0.5, an elastic net weight (α) of 0.4 and a momentum velocity coefficient of 0.7.

[6] http://scikit-learn.org

	Method	Accuracy			
		Stance		Sentiment	
		IAC	CD	AYI	AMZ
Baseline lexicons	CPBLex	0.524	0.441	0.528	0.519
	PMILex	0.557	0.529	0.571	0.554
Baseline classifiers	NaiveBayes	0.536	0.474	0.665	0.637
	SVM	0.589	0.594	0.689	0.671
	DecisionTree	0.582	0.573	**0.742***	0.707
Approach	**LexicNet**	**0.647***	**0.642***	0.729	**0.718**

Table 2: Experimental results

We note from manual examination of the results on stance classification tasks that a large proportion of the classification errors made by LexicNet was made on documents where stance was heavily implied and there is no term used that is heavily indicative of the class, e. g., "*Thus, an important point is raised*" (agreement). While algorithms like SVM manage to draw on their complexity to target higher-order (and sometimes even coincidental) relationships between sets of words and classes, LexicNet lacks the expressiveness to do so. On sentiment analysis tasks, we note that a likely explanation of the poor performance on the AYI dataset is due to the multiple sources of data: Amazon, Yelp and IMDB. Such diversity would tend to cause a large standard deviation of input length, something that a lexicon-based classifier is more sensitive to than a decision tree. We conduct another experiment in order to confirm this hypothesis, and display the results in table 3.

We can see from table 3 that when separating the AYI dataset in the three domains a performance improvement can be observed in LexicNet. However the improvement does not completely bridge the gap with DecisionTree, meaning that other factors at play influence the algorithm and will be studied in further work.

6 Conclusion and future work

In this work we showed the viability of using elastic net-regularized backpropagation to learn effective lexicon weights, producing a lexicon that is competitive with standard classifiers and significantly outperforms baseline lexicon learning techniques in several datasets.

However we identified two major flaws in the LexicNet which will be subject to future work: (1) its lack of expressiveness to capture higher-order relationships between sets of terms and classes, as well as (2) its deficiencies when dealing with inputs of varying length, due to the way it models the classification process as a weighted sum of term-class association strength, which means that a term that only appears in very long instances will be assigned a lower weight than one appearing only in short texts.

		Accuracy		
	Method	Amazon	Yelp	IMDB
Baseline lexicons	CPBLex	0.545	0.511	0.562
	PMILex	0.557	0.593	0.612
Baseline classifiers	NaiveBayes	0.691	0.704	0.728
	SVM	0.677	0.692	0.686
	DecisionTree	**0.755***	**0.704**	0.740
Approach	**LexicNet**	0.738	0.691	**0.742**

Table 3: Experimental results after separating domains in the AYI dataset

Our future work will focus on improving the expressiveness of LexicNet to counter those issues while keeping it in the form of a human-readable lexicon. Our objective is to do so by incorporating lexical context such as modifier words (e. g., adverbs) and topical context (words with multiple meanings with different class values) in the network topology, thus improving accuracy without hurting interpretability.

References

1. Bandhakavi, A., Wiratunga, N., Deepak, P., Massie, S.: Generating a word-emotion lexicon from# emotional tweets. In: Proceedings of the 3rd Joint Conference on Lexical and Computational Semantics (* SEM 2014) (2014)
2. Bandhakavi, A., Wiratunga, N., Deepak, P., Massie, S.: Lexicon based feature extraction for emotion text classification. Pattern Recognition Letters (2016)
3. Bishop, C.M.: Pattern recognition. Machine Learning 128, 1–58 (2006)
4. Chen, H., Dumais, S.: Bringing order to the web: Automatically categorizing search results. In: Proceedings of the SIGCHI conference on Human Factors in Computing Systems. pp. 145–152. ACM (2000)
5. Clos, J., Bandhakavi, A., Wiratunga, N., Cabanac, G.: Predicting emotional reaction in social networks. In: European Conference on Information Retrieval. pp. 527–533. Springer (2017)
6. Clos, J., Wiratunga, N., Massie, S., Cabanac, G.: Shallow techniques for argument mining. In: ECA'15: Proceedings of the ECA. vol. 63, p. 2 (2016)
7. Diederich, J., Kindermann, J., Leopold, E., Paass, G.: Authorship attribution with support vector machines. Applied intelligence 19(1), 109–123 (2003)
8. Drucker, H., Wu, D., Vapnik, V.N.: Support vector machines for spam categorization. IEEE Transactions on Neural networks 10(5), 1048–1054 (1999)
9. Esuli, A., Sebastiani, F.: Sentiwordnet: A publicly available lexical resource for opinion mining. In: Proceedings of LREC. vol. 6, pp. 417–422. Citeseer (2006)
10. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems. Journal of Machine Learning Research 15(1), 3133–3181 (2014)
11. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intelligent Systems and their Applications 13(4), 18–28 (1998)

12. Jurafsky, D., Martin, J.H.: Lexicons for Sentiment Extraction, chap. 18. Pearson (2016)
13. Kang, G., Li, J., Tao, D.: Shakeout: a new regularized deep neural network training scheme. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 1751–1757. AAAI Press (2016)
14. Lan, M., Tan, C.L., Low, H.B., Sung, S.Y.: A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In: Special interest tracks and posters of the 14th international conference on World Wide Web. pp. 1032–1033. ACM (2005)
15. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM 38(11), 39–41 (1995)
16. Muhammad, A., Wiratunga, N., Lothian, R.: A hybrid sentiment lexicon for social media mining. In: Tools with AI (ICTAI), IEEE 26th International Conf. on. pp. 461–468 (2014)
17. Muhammad, A., Wiratunga, N., Lothian, R.: Contextual sentiment analysis for social media genres. Knowledge-Based Systems 108, 92–101 (2016)
18. Nesterov, Y.: Gradient methods for minimizing composite objective function (2007)
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research 12(Oct), 2825–2830 (2011)
20. Pennebaker, J.W., Francis, M.E., Booth, R.J.: Linguistic inquiry and word count: Liwc 2001. Mahway: Lawrence Erlbaum Associates 71, 2001 (2001)
21. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144. ACM (2016)
22. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management 24(5), 513–523 (1988)
23. Sammut, C., Webb, G.I. (eds.): Cross-Validation, pp. 249–249. Springer US, Boston, MA (2010)
24. Stone, P.J., Dunphy, D.C., Smith, M.S.: The general inquirer: A computer approach to content analysis. MIT press (1966)
25. Turney, P.D.: Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th annual meeting on ACL. ACL (2002)
26. Walker, M.A., Tree, J.E.F., Anand, P., Abbott, R., King, J.: A corpus for research on deliberation and debate. In: LREC. pp. 812–817 (2012)
27. Wang, L., Cardie, C.: Improving agreement and disagreement identification in online discussions with a socially-tuned sentiment lexicon. ACL 2014 p. 97 (2014)
28. Yin, J., Thomas, P., Narang, N., Paris, C.: Unifying local and global agreement and disagreement classification in online debates. In: Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis. pp. 61–69. ACL (2012)
29. Zhang, W., Yoshida, T., Tang, X.: A comparative study of tf* idf, lsi and multi-words for text classification. Expert Systems with Applications 38(3), 2758–2765 (2011)
30. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(2), 301–320 (2005)