Neural Induction of a Lexicon for Fast and Interpretable Stance Classification

Jérémie Clos and Nirmalie Wiratunga

Robert Gordon University
Garthdee Road
Aberdeen, United Kingdom
initial.lastname@rgu.ac.uk

Abstract. Large-scale social media classification faces the following two challenges: algorithms can be hard to adapt to Web-scale data, and the predictions that they provide are difficult for humans to understand. Those two challenges are solved at the cost of some accuracy by lexicon-based classifiers, which offer a white-box approach to text mining by using a trivially interpretable additive model. However current techniques for lexicon-based classification limit themselves to using hand-crafted 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 this work we propose a new approach to learn robust lexicons, using the backpropagation algorithm to ensure generalization power without sacrificing model readability. We evaluate our approach on a stance detection task, on two different datasets, and find that our lexicon outperforms standard lexicon approaches.

1 Introduction

Text classification is one of the main tasks in natural language processing, with applications ranging from web search to opinion mining. For instance, being able to perform this task on large amounts of social media data enables businesses to know in real time how the public perceives them, talks about them, which has repercussions in key areas such as predictions of the stock market prices for a given company.

Large-scale social media classification faces the challenges of scaling algorithms and producing predictions that can be explained and interpreted. Those two challenges are solved at the cost of some accuracy by lexicon-based classifiers, which offer a white-box approach to text mining by using a trivially interpretable additive model, where the probability of an instance I belonging to a class C is nothing more than a weighted sum of the probabilities of each term in I belonging to C. However, current techniques used to create those lexicons fall short in many ways compared to more traditional black-box machine learning models. That difference in performance is easily explained by the fact that, unlike lexicon-based classifiers, those models are trained in a black-box way, with no regard to their interpretability. This paper attempts to conciliate lexicon-based classification and traditional text classification by designing a simple and efficient training procedure that can generate domain-specific lexicons with a high degree of interpretability and a high classification performance.

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. We then detail our evaluation protocol on a stance detection task and on two different datasets. We perform an evaluation against standard lexicons and baselines found in the literature and report that our approach significantly outperforms standard techniques for generating lexicons. Finally, we analyze and discuss our results, before concluding on the next steps of our work.

2 Related works

The literature on text classification is rich in approaches of varying degree of complexity, but there has been scarce research done on approaches which are both interpretable [11] and accurate.

2.1 Lexicon-based classification

Lexicons are early tools adopted by the computational linguistics community to automatically classify text. They can take many forms, the most common of which being either a simple list of terms associated to a certain class of interest, or a $T \times C$ matrix where each pair (t, c) where $t \in T$ is one of the T terms and $c \in C$ is one of the C classes is mapped to a strength of association score $s = l(t, c)$. Several lexicons also contain additional contextual information in order to help their users build more complex models, but they all share the same core architecture, which we formalize as follows:

Definition 1. *A lexicon Lex is a tuple $Lex = \langle L, A, D \rangle$ where:*

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

In this definition, L is a mapping function that assigns an unbounded value to each pair (t, c) where term $t \in T$ and class $c \in C$, A is an aggregation function that aggregates the accumulated scores into one value, and D is a decision function that selects one of these aggregated values.

Concretely, the mapping determines an evidence value for each term using a look-up list, propagates it to the aggregation function which aggregates the set of evidence values from the terms contained within one instance into multiple stacks of evidence (one for each class). Finally, the decision function evaluates each stack of evidence to select the one that is the most likely. Algorithm 1 shows a general form of lexicon-based classification using this formal definition. Using this formal definition we can reformulate previous lexicons using the same format.

Simple lists of terms fit under this definition by having the decision function be $ArgMax$, the aggregation function be a simple sum, and the mapping function be the indicator function where $L(t) = 1$ if t is in the Lexicon and 0 otherwise.

Traditional matrix-shaped lexicons fit under this definition by having the decision function be $ArgMax$, the aggregation function be a simple sum, and the mapping function be a simple look-up in the Lexicon which defaults to 0 if the term is not in the Lexicon.

Other, non lexicon-based approaches can also be described under the same formal framework, such as Naive Bayes classification where the mapping function maps terms to conditional probabilities of that term being present given that class, the aggregation function is a multiplication (or alternatively addition of logarithmic probabilities) where all values are multiplied together and to a prior, and the decision function is an $ArgMax$ function applied to log-ratios of those aggregated class probabilities.

```
Data: Instance $I$
          Lexicon function      $L_c$
Input:    Aggregation function  $\oplus$
          Decision function     $D$
Result: A class label $c$
$C_A \leftarrow 0$;
$C_B \leftarrow 0$;
for Term $t$ in $I$ do
    $C_A \leftarrow C_A \oplus L_A(t)$;
    $C_B \leftarrow C_B \oplus L_B(t)$;
end
return $D(C_A, C_B)$;
```

Algorithm 1: Lexicon-based classification algorithm on a binary problem

2.2 Traditional hand-crafted lexicons

The first lexicons were not obtained using computational means but rather hand-crafted by domain experts. This is due to the computational cost of building a lexicon and the fact that it is only recently that we have had access to the computational resources to parse the amount of data necessary to the generation of useful lexicons. Traditional lexicons were usually either a hand-crafted list of words with a numerical or categorical value associated to each class or a simple list of words that are known to be associated to a class (with no quantification of that association). This comprises sentiment lexicons as well as more complex linguistic patterns such as emotion lexicons, argument lexicons, etc.

Two different aggregation/decision mechanisms appear in the literature using these lexicons. In the case of non-quantified lexicons, a counting of the number of lexicon terms appearing in the text produces an appropriate aggregated value. The class which has the most terms appearing in the text is then chosen as part of the decision function. In the case of quantified lexicons, a sum of the weights of the lexicon terms appearing in the text produces an aggregated weight. The class which has the highest weight is then chosen as part of the decision function.

The strength of these approaches is twofold: firstly in how well they generalize, because they were consciously created by subject domain experts, and secondly in their human-interpretability, because they were formed by human users assigning scores to each term. To this day, hand-crafted lexicons such as the LIWC lexicon [10] are still sold for commercial computational linguistics applications. 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.

2.3 Lexicon induction techniques

In order to attend to the issues inherent to hand-crafted lexicons, research in computational linguistics evolved towards computing lexicon scores from external data sources, rather than being generated by a set of experts with domain knowledge. In this section we will review learning techniques of existing approaches as well as the challenges that they face. Research in lexicon induction outlines multiple families of techniques that can be used in order 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.

Graph propagation based lexicons (GPBL). GPBL learning techniques use a few human-provided seed words for which the class is known, and leverage some external relationship (typically synonyms, antonyms and hypernyms) in a semantic graph such as WordNet [7] to propagate class values along that graph [3]. For example, if the term "agreement" was deemed fully associated to a class, its synonym "accord" would be associated to the same class while its antonym "disagreement" would be associated to its opposite class. 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.

Conditional probability-based lexicons (CPBL). CPBL learning techniques are the baseline against which we evaluate our lexicon induction algorithm. They operate by computing the conditional probability of observing each lexicon entry under each class [1]. The value for each pair (t, c) where term $t \in T$ and class $c \in C$ is computed as indicated in equation 1. The main flaw of this technique is that it overestimates strength of association based on coincidences, which means that it would be easy to build a completely correct dataset to trick the algorithm in learning a lexicon full of spurious association scores.

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

Mutual information based lexicons Mutual information-based lexicon learning techniques attempt to fix the issues of previous approaches by estimating the pointwise mutual information (PMI) between a term and a class [12] using the formula described in equation 2. Mutual information being inherently more robust to coincidences because of its denominator, is chosen as a strength of association measure. Some works [2] have shown that NPMI, a normalized version of the standard PMI metric described in equation 3, slightly improves classification performance. While this approach is sensible to create a general purpose lexicon, it suffers some flaws in the following cases: (1) if none of the terms used in the child post has an argumentative value or is present within the lexicon, no classification is possible, and (2) some terms might end up with an undeserved score because they accidentally appear more frequently within comments of one class. For example if non-argumentative terms such as "Monday" accidentally co-occur too often within one class, they will be misconstrued as being indicative of that class.

$$\mathrm{PMI}(x;y) = \frac{\log(p(x;y))}{p(x)p(y)} \tag{2}$$

$$\mathrm{NPMI}(x;y) = \frac{\frac{\log(p(x;y))}{p(x)p(y)}}{-\log\left[p(x,y)\right]} \tag{3}$$

Hybrid lexicons Recent work [9] has attempted to hybridize handcrafted and automatically generated lexicons, based on the assumption that the coverage of the former would help the specialization of the latter as a fallback option, and that a hybrid lexicon would thus be able to deal with domain-specific and general knowledge. Hybrid lexicons tend to improve classification accuracy as shown in [9] but have the drawback of requiring a handcrafted lexicon and are thus beyond the scope of this work as of now.

3 Building a neural lexicon

Traditional ways of learning a lexicon from a corpus of data either use point estimates of some statistical values, such as pointwise mutual information, or semantic values directly derived from human expertise. However, we can observe in figure 1 that a standard lexicon can be expressed in the form of a computational graph, where the lexicon is described as a composition of functions as seen in equation 4. That graphical form gives us the possibility of using gradient-based learning techniques such as backpropagation in order to learn both the lexicon and the strength of association scores.

$$\mathrm{Class}(i) = \mathrm{ArgMax}_c\left(\sum_{t\in i}\left[s_c(t)\right]\right) \tag{4}$$

Traditional lexicons are thus considered as a specific network topology that do not have sigmoid activation functions but instead a simple aggregation layer with one aggregation unit per class and a output layer of one single unit that transforms the aggregated evidence into the relevant format. The details of the network topology and the training protocol are explained in the following sections.

3.1 The Lexicon network topology

The lexicon follows a specific network topology where each vocabulary input is mapped to one unit, which is linked to as many hidden units as there are classes, which are then aggregated by the following layer into a sum of evidence towards that class. Finally, the last layer uses this sum of evidence to produce a decision which is the output of the classifier. In this section we review each layer of the neural lexicon and their precise function.

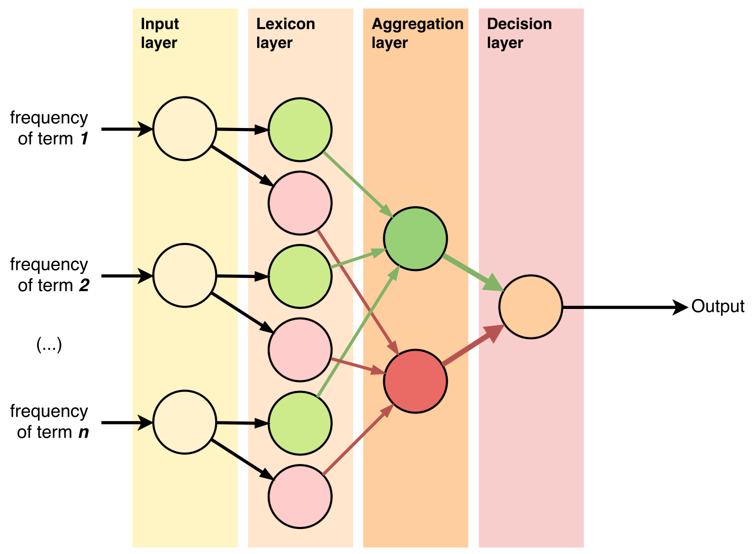


Fig. 1. Lexicon computational graph

The first layer: vocabulary input The first layer is the input layer, which maps a term to its matching unit. The input signal coming to this layer is some measure of frequency of occurrences of each term in the text that is being classified. We can apply a scaling function such as $scaledFrequency = log(1 + frequency)$ in order to smooth out the differences between long and short comments, or just take the raw frequency and communicate it to the next layer.

The lexicon layer The lexicon layer is a function that maps a lexicon entry (linked to terms) to their respective scores, which, because of the additive nature of the next layer, are a numerical amount representing the evidence brought towards a class by the presence of a term. The output of that layer is the score of the term concerned multiplied by the input of the previous layer,.

The aggregation layer The aggregation layer adds up evidence towards a class from a list of units in the previous layer. The most common function in lexicon-based classification is the simple arithmetic sum, which is then fed into the output layer.

The decision layer The decision layer is a function that, given a set of numbers representing the amounts of evidence for each class, produces a classification. A common function used in both lexicon classifiers and Bayesian classifiers is the simple $ArgMax$ function, which selects the class that maximizes a numerical amount. However, because it is impossible to differentiate the $ArgMax$ function, a proxy function is used during the training phase of the algorithm.

3.2 Lexicon network training

In this section we detail the process of training the lexicon network.

Cost function and regularization The backpropagation algorithm relies on a process called reverse-mode differentiation in order to train the network in a computationally efficient way, by updating the weight of local units based on the error partial derivative with respect to those units. This process requires that the network be differentiable in order to compute the error properly. We use a proxy decision function during the training process because the original ArgMax function is not differentiable.

Because the only thing the ArgMax relies on is the proportion of evidence in favor of one class versus another, we can use the cross-entropy error function E:

$$E(C,\hat{C}) = -\sum_{i=1}^{|C|}\left(C_i \times log(\hat{C}_i) + (1 - C_i) \times log(1 - \hat{C}_i)\right) \tag{5}$$

Here, the optimal class distribution is $\left[\hat{C}_1, \hat{C}_2, ..., \hat{C}_n\right]$ and the predicted class distribution is $[C_1, C_2, ..., C_n]$ where each prediction is normalized from the aggregated evidence using the Softmax function $C_i = \frac{\exp a_i}{\sum_{j=0}^{|a|} \exp a_j}$ where a_i is the aggregated weight for a class i. However, optimizing over a direct function of the error with a large amount of free parameters (numbers of classes $\times$ number of lexicon entries) will lead to overfitting on the training data and poor performance on the test data, which emphasizes the need to regularize our training process. We selected L_2 regularization, which is a minimization of the $L_2 norm$ of the parameters, because it is differentiable and minimizes weights without pushing them completely to 0 (unlike L_1 regularization). This property is desirable for learning a lexicon because pushing a weight to 0 would just remove many terms observed only in the training data and thus increase overfitting. The resulting cost function J is shown in equation 6.

$$J(C,\hat{C}) = E(C,\hat{C}) + \lambda * \sqrt{\sum_{j=0}^{m} w_j^2} \tag{6}$$

Here λ corresponds to a regularization parameter, 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, w_i corresponds to the weight of unit i in the lexicon layer.

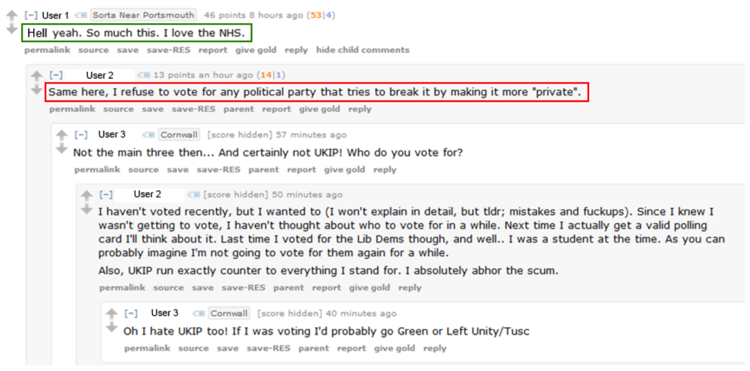
Optimization The backpropagation algorithm trains the network by propagating the error gradient backwards through the computational graph and applying a local update rule based on its value. Using the chain rule, the partial derivative of the error with respect to each lexicon input can be decomposed in a set of simpler partial derivatives. Equations 7 shows the update rule for a lexicon weight w from the error J.

$$w_i = w_i - \gamma \times \frac{\partial J}{\partial w_i} \tag{7}$$

Here J corresponds to the cost (equation 6) of the current iteration over the dataset and γ represents the learning rate, a parameter that we manually set to a very small value. Since our datasets are small, we train our network using the Conjugate Gradient Descent algorithm [6], updating the weights of the network after each iteration over the dataset. It is however important to note that since the goal of the network is only to generate a lexicon, the update rule is only applied to the weight of the edges coming from the lexical layer and all others are held constant.

4 Evaluation

We evaluate our approach on an argument stance classification task, which is a type of text classification specializing on argumentative discourse. Argument stance classification is the classification of textual content coming from an agent, e. g., user comments extracted from a discussion forum, into multiple classes representing the stance of those comments with respect to the comments they are responding to. We study argument stance classification on a binary scale, where neutral responses are removed and only rebutting/disagreeing or supporting/agreeing statements are conserved. Using figure 2 as an example, we can see a debate on the social discussion website Reddit[1] where user 1 (in green) is in complete agreement with the parent comment, while user 2 (in red) is in complete agreement with user 1 while being in complete disagreement with the core topic of the discussion. This difference differentiates stance classification from sentiment analysis and makes it a harder problem.


[–] User 1 Sorta Near Portsmouth 46 points 8 hours ago (53|4)
Hell yeah. So much this. I love the NHS.
permalink source save save-RES report give gold reply hide child comments

[–] User 2 13 points an hour ago (14|1)
Same here, I refuse to vote for any political party that tries to break it by making it more "private".
permalink source save save-RES parent report give gold reply

[–] User 3 Cornwall [score hidden] 57 minutes ago
Not the main three then... And certainly not UKIP! Who do you vote for?
permalink source save save-RES parent report give gold reply

[–] User 2 [score hidden] 50 minutes ago
I haven't voted recently, but I wanted to (I won't explain in detail, but tldr; mistakes and fuckups). Since I knew I wasn't getting to vote, I haven't thought about who to vote for in a while. Next time I actually get a valid polling card I'll think about it. Last time I voted for the Lib Dems though, and well.. I was a student at the time. As you can probably imagine I'm not going to vote for them again for a while.
Also, UKIP run exactly counter to everything I stand for. I absolutely abhor the scum.
permalink source save save-RES parent report give gold reply

[–] User 3 Cornwall [score hidden] 40 minutes ago
Oh I hate UKIP too! If I was voting I'd probably go Green or Left Unity/Tusc
permalink source save save-RES parent report give gold reply

Fig. 2. Local stance classification in context: a debate on Reddit

4.1 Datasets

We performed our experiments on two social media datasets. The first one is the Internet Argument Corpus [13] (further referred to as IAC). It is a subset of a publicly available

[1] http://www.reddit.com

dataset collected on a discussion forum and manually labeled. The second one is the Reddit Noisy-Labeled Corpus (further referred to as RNLC). It was created by collecting data from a discussion forum and automatically labeling it using distant supervision learning [4, 8]. Statistics on the corpora can be found in table 1 and show that the two datasets are similar with the exception that the RNLC was collected on comments which were on average twice as long as the IAC. This has an important impact on lexicon-based methods because of the risk of inserting more noise into the system.

The Internet Argument Corpus (IAC). The IAC [13] is a corpus of forum comments manually labeled by 5 annotators that contain (among other things) degree of agreement/disagreement with their immediate parent comment. A subset of this dataset was used for our experiment, by selecting the comments that ensured disjoint class membership (meaning filtering out comments with an average score close to 0). For instance, a comment such as "*For the same reasons that I do not agree with the first conclusion of your statement, I feel that your second conclusion is correct*" would technically belong to both classes as the user both supports and attacks the previous comment and it would thus be filtered out.

The Reddit Noisy-Labeled Corpus (RNLC). The RNLC is a newly formed corpus of comments extracted from the Reddit[2] website and automatically labeled with a binary class using evidence contained in the comment. A list of explicit expressions such as *"I [positive adverb] agree"* and *"I [positive adverb] disagree"* (and variations) were used to detect strong evidence of a user comment belonging to a class. In the case of the presence of conflicting evidence, i. e., expressions acting as strong evidence towards both classes, the comments were not considered. Otherwise, comments were automatically assigned to their respective class and the corresponding sentences were deleted from the comments in order to avoid an advantage due to class bias. The data is labeled using a noisy labeling approach inspired from distant supervision learning [8] whereby highly discriminative expressions such as *"I agree"* and *"you are wrong"* are used as cues to class labels agreement and disagreement. A minimum comment length was also added as a requirement in order to remove uninformative data points. The complete dataset, spanning posts from a year of crawling, was then randomly subsampled for computational efficiency.

Dataset	IAC	RNLC
Number of comments	8000	100000
Average terms/sentence	39.8	33.7
Average sentences/comment	3.1	12.2
Instances of agreement	4000	50000
Instances of disagreement	4000	50000

Table 1. Descriptive statistics on IAC and RNLC

[2] http://www.reddit.com

4.2 Baselines

We contextualize our approach by comparing it to Naive Bayes (NAIVEBAYES), a strong baseline in text classification that allows the user to manually inspect the parameters of its model in the form of word probabilities, thus allowing predictions to be interpreted and corrected. We will also compare our approach to two existing approaches for lexicon induction from text data: the Conditional Probability-Based Lexicon (CPBLEX) which models each term score as the conditional probability of observing that term in that particular class, and both the Pointwise Mutual Information Lexicon (PMILEX) which models each term score as the pointwise mutual information between that term and that particular class and its variation using normalized pointwise mutual information NPMILEX. Both lexicons then use the classification rule described in equation 8, which classifies a user comment x on the basis of maximizing the sum of associations between each of its terms t and each class c. Lexicon size is always kept to 400 to avoid overfitting after removal of stopwords[3] and non-alphanumerical characters.

$$\text{ClassLabel}(x) = \text{ArgMax}_c \left[\sum_{t \in x} TermScore(t, c) \right] \tag{8}$$

CPBLEX *baseline* We compute this lexicon using the conditional probability of observing each term in each class, as referred in equation 1.

PMILEX *baseline* We compute this lexicon using normalized pointwise mutual information (NPMI, referred in equation 3) as a way to measure strength of association between terms and their class.

5 Results and discussion

We present the results of our experiment in table 2 and 3. Two approaches are tested: LEXICNET$_1$ uses a raw term frequencies as input, while LEXICNET$_2$ uses a logarithmically scaled frequency. A 10 fold cross-validation was done, the accuracy results were averaged over the 10 folds and a 2-tailed paired T test was performed on the folds to compute statistical significance in the difference between, with a 95% confidence threshold (i. e., a p-value < 0.05). In the following tables, the best lexicon approach is highlighted in bold, while the best approach overall is marked.

We can observe that the $Log(tf)$ scaling yields a higher accuracy ($+0.704\%$), which can be explained by the difference in comment length that we can see in table 1. A logarithmic scaling of the input scores make sure that there is no large difference between two user comments based only on their length and as such would have a higher impact where there is great variation in comment length. It is evident from the results that LEXICNET$_2$ performs significantly better than standard lexicons approaches (CPBLEX and PMILEX) as well as simple but traditional machine learning approaches (NAIVEBAYES), which can be explained by the lack of a typical training phase in the latter approaches. Having an optimization phase on the set of training examples allows

[3] using the stopword list from http://www.ranks.nl/stopwords

	Method	Accuracy
Baseline lexicons	CPBLex	0.604
	PMILex	0.647
	NPMILex	0.679
Baseline ML	NaiveBayes	0.655
Approaches	LexicNet$_1$	0.695
	LexicNet$_2$	**0.701**

Table 2. Results on IAC

	Method	Accuracy
Baseline lexicons	CPBLex	0.538
	PMILex	0.577
	NPMILex	0.609
Baseline ML	NaiveBayes	0.590
Approaches	LexicNet$_1$	0.595
	LexicNet$_2$	**0.627**

Table 3. Results on RNLC

our approach to outperform mere point estimates, while keeping the attractive simplicity of their additive model. Statistical significance ($p < 0.05$) was achieved on the two tailed paired T test between LexicNet$_2$ and the rest of the lexicon-based approaches, thus showing that our approach is a significant improvement over traditional techniques for lexicon induction from text.

Finally, a significant gap between the accuracy obtained in the IAC and the RNLC datasets can be observed, due to the noisy nature of the latter leading potentially to wrongly labeled data from the start. Further study will see the use of these larger amounts of unreliable data as a source of background information to enrich an existing manually labeled dataset, but it is beyond the scope of this work.

6 Conclusion and future work

In this work we showed the viability of modeling classification lexicons as generic computational graphs in order to compute the lexical scores in an efficient manner. There has been research done on finding faster alternatives to the more complex models [5] while maximizing performance but none so far focusing on human interpretability of the models that are produced.

Our future works will thus focus on two different aspects that were not presently developed.

Firstly, the first layer of the graph described in this work can be seen as analogous to a one dimensional convolution on text. It then stands to reason that we should be able to learn that layer as well instead of providing an existing list of terms, thus producing an end-to-end neural algorithm for lexicon induction. Learning it from the data would allow us to generalize the lexicon from unigrams to n-grams.

Secondly, our current solution, while competitive with simple learners that are used for their efficiency and interpretability, is not competitive with the more complex algorithms such as deep neural networks or kernel methods. The main reason for this is that we force a fixed structure on the learning of our model so that each word class association can be inspected and changed a posteriori if necessary. However, there are ways to work around that while keeping this simplicity using more complex inference schemes and taking into account for example sequences of terms rather than unordered bag of words, taking inspiration from techniques such as backpropagation-through-time [14] to detect special terms such as valence modifiers or valence shifters.

References

1. Bandhakavi, A., Wiratunga, N., Deepak, P., Massie, S.: Generating a word-emotion lexicon from# emotional tweets. In: Proc of the Third Joint Conference on Lexical and Computational Semantics (* SEM 2014). pp. 12–21 (2014)
2. Clos, J., Wiratunga, N., Massie, S., Cabanac, G.: Shallow techniques for argument mining. In: ECA'15: Proceedings of the 1st European Conference on Argumentation: Argumentation and Reasoned Action. vol. 63, p. 2 (2016)
3. Esuli, A., Sebastiani, F.: Sentiwordnet: A publicly available lexical resource for opinion mining. In: Proceedings of LREC. vol. 6, pp. 417–422. Citeseer (2006)
4. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford 1(12) (2009)
5. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
6. Luenberger, D.G.: Introduction to linear and nonlinear programming, vol. 28. Addison-Wesley Reading, MA (1973)
7. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM 38(11), 39–41 (1995)
8. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2. pp. 1003–1011. Association for Computational Linguistics (2009)
9. Muhammad, A., Wiratunga, N., Lothian, R.: A hybrid sentiment lexicon for social media mining. In: Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on. pp. 461–468. IEEE (2014)
10. Pennebaker, J.W., Francis, M.E., Booth, R.J.: Linguistic inquiry and word count: Liwc 2001. Mahway: Lawrence Erlbaum Associates 71, 2001 (2001)
11. 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)
12. Turney, P.D.: Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th annual meeting on association for computational linguistics. pp. 417–424. Association for Computational Linguistics (2002)
13. 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)
14. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10), 1550–1560 (1990)