Don't Count, Predict! An Automatic Approach to Learning Sentiment Lexicons for Short Text

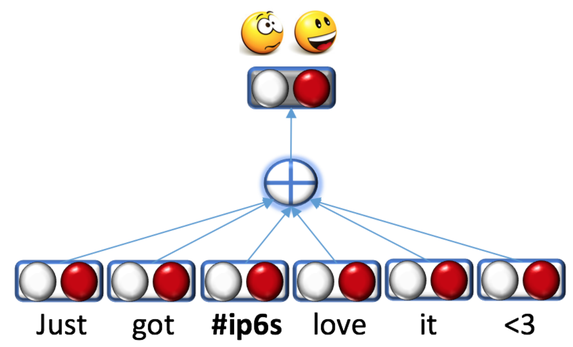
Duy-Tin Vo & **Yue Zhang**
Singapore University of Technology and Design
duytin_vo@mymail.sutd.edu.sg, yue_zhang@sutd.edu.sg

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Introduction

We describe an efficient neural network method to automatically learn open language sentiment lexicons without relying on any manual resources. The method takes inspiration from the NRC method, which gives the best results in SemEval13 by leveraging emoticons in large tweets, using the PMI between words and tweet sentiments to define the sentiment attributes of words. We show that better lexicons can be learned by using them to predict the tweet sentiment labels. By using a very simple neural network, our method is fast and can take advantage of the same data volume as the NRC method. Experiments show that our lexicons give significantly better accuracies on multiple languages compared to the current best methods.

Model



Learning Lexicons

1 Counted-based

Given a word w and a corpus with sentiment labels, the sentiment score $SS(w_i)$ of w_i is computed as:

$$SS(w_i) = \log_2 \frac{freq(w_i, pos) * freq(neg)}{freq(w_i, neg) * freq(pos)}$$

Here $freq(w_i, pos/neg)$ is the number of times the term w_i occurs in positive/negative tweets, $freq(pos/neg)$ is the total number of tokens in positive/negative tweets.

2 Predicted-based

As shown in the above figure, given a tweet $tw = w_1, w_2, ..., w_n$, a simple neural network is used to predict its two-dimensional sentiment label y:

$$w_i = (n, p), n, p \in \mathbf{R}$$

$$h = \sum_i (w_i)$$

$$y = softmax(hW)$$

where W is fixed to the diagonal matrix ($W \in \mathbf{R^{2\times2}}$). The cross-entropy error is employed as the objective function:

$$loss(tw) = -\sum \hat{y}.\log(y)$$

Backpropagation is applied to learn (n, p) for each token. After learning positive and negative scores from labeled data, the sentiment score of w_i is computed as:

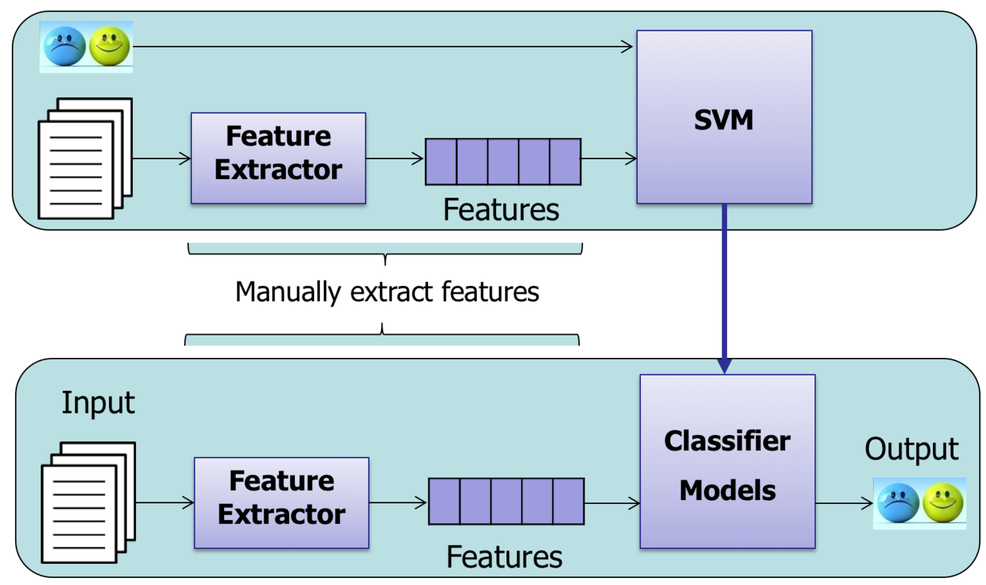
$$SS(w_i) = p_i - n_i$$

Comparing Methods

3 Unsupervised Classification

$$label(tw) = \begin{cases} positive & \text{if } \sum_i (SS(w_i)) \geq 0 \\ negative & \text{if } \sum_i (SS(w_i)) < 0 \end{cases}$$

4 Supervised Classification



1. $count(w_i),\ SS(w_i) \neq 0$;
2. $\sum_{w_i} SS(w_i), SS(w_i) > 0$;
3. $\sum_{w_i} SS(w_i), SS(w_i) < 0$;
4. $SS(w_n)$;
5. $\sum_{w_i} SS(w_i)$;
6. $max_{w_i} SS(w_i)$.

Datasets

English dataset

Type		#pos	#neg	#Tweets
Supervised	train	3009	1187	4196
	dev	483	283	766
	test	1313	490	1803
Unsupervised		4805	1960	6765

Arabic dataset

Labels	Balanced			Unbalanced		
	train	dev	test	train	dev	test
#pos	481	159	159	481	159	159
#neg				1012	336	336
#mix				500	166	166
#obj				4015	1338	1338
#Tweets	1924	636	636	6008	1999	1999

Training and Settings

- Crawl labeled tweets in English (9m tweets) and Arabic (800k tweets) using emoticons (Go et al., 2009);
- Train sentiment score by applying AdaDelta (Zeiler, 2012) for backpropagation over 5 iterations;
- Initialize randomly polarity score $[-0.25, 0.25]$;
- Shuffle mini-batches with size of 50;
- Employ grid search on supervised classification to tune parameters.

Analysis

Words	nnLexicon	NRC
bad	-1.122	-1.295
worse	-1.626	-1.417
worst	-2.256	-1.875
busy	-0.520	-0.003
busier	-0.609	0.106*
busiest	-1.254	-0.712
suitable	0.502	-0.040*
satisfy	0.570	-0.173*
lazy	-0.462	0.224*
scummy	-0.852	0.049*
old wine	0.453	0.552
old meat	-0.172	0.014*
strong memory	0.081	-0.083*
strong snowstorm	-0.554	0.182*

Example sentiment scores, where * denotes incorrect polarity.

Experiments

English results

Lexicons		Unsup			Sup
		P	R	F	Acc
WEKA	ED	61	55.9	55.4	73.8
	STS	66.4	52.5	47.7	73.7
HIT		**75.3**	73.3	74.1	78.5
NRC	Hashtag	70.3	71.4	70.8	77.4
	Emoticon	73.2	74.6	73.8	79.9
nnLexicon		74.4	**77.3**	**75.3**	**81.3**

Arabic results

Lexicons		Balanced	Unbalanced
NRC	Hashtag	31.9	63.4
	Emoticon	31.4	65.3
nnLexicon		**33.3**	**66.5**

Conclusion

We constructed open language sentiment lexicons for short text automatically using an efficient neural network, showing that prediction-based training is better than counting-based training for learning from large tweets with emoticons. In standard evaluations, the method gave better accuracies across multiple languages compared to the state-of-the-art counting-based method. Due to the model's simplicity, the learning phrase is very fast, training a sentiment lexicon over 9 million tweets within 35 minutes per epoch on an Intel® core™ i7-3770 CPU @ 3.40 GHz. We make the Python implementation of our models to learn sentiment lexicons in atbitrary languages and the resulting sentiment lexicons used in this paper available at *https://github.com/duytinvo/acl2016*