Learning Sentiment-Specific Word Embedding (SSWE) for Twitter Sentiment Classification

Chang Shu

04/02/2020

Introduction

- Sentiment classification
  - Label a text’s sentiment polarity as “positive”, “negative”, and “neutral”

- Why sentiment classification?
  - Commerce
    - Consumer’s attitude towards commercial product and competitors
    - Reveal confidence of the market to predict stock price
  - AI
    - Help AI understand human feelings to build better chatbot
  - Politics
    - public opinion to policy announcements and campaign messages

- Why twitter?
  - Larger
  - Prompt
  - General (compared with blogpost, forum)

How?

- The majority of existing approaches employ machine learning algorithms to build Tweets sentiment classifier.
  - Sentiment Lexicon
  - Feature Learning method

Previous work: sentiment lexicon

- Sentiment lexicons are list of words with associations to positive and negative sentiments *(Mohammad et al.,2013)*
- *Mohammad et al.(2013)* build the top performance Tweets sentiment classification using sentiment lexicons
  - calculate sentiment association score of each term
  - Hand crafted feature: # of positive terms in tweets
- Cons: labor-intensive, need to less depend on extensive feature engineering

Previous work: syntactic based feature learning method

- Word embedding
  - Each word was projected from sparse representation (one-hot) to a dense, low-dimensional and real valued vector
  - C&W model (*Collobert et al., 2011*), Word2Vec (*Mikolov et al., 2013*)

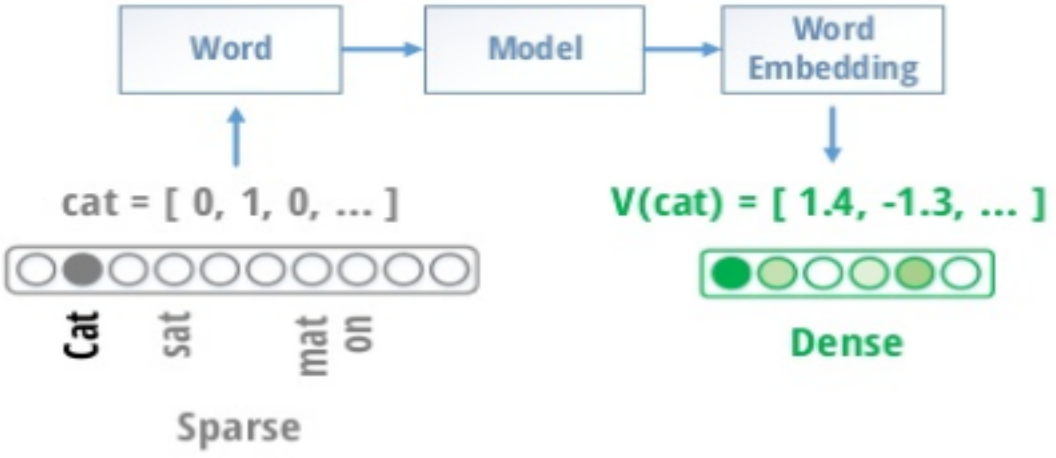
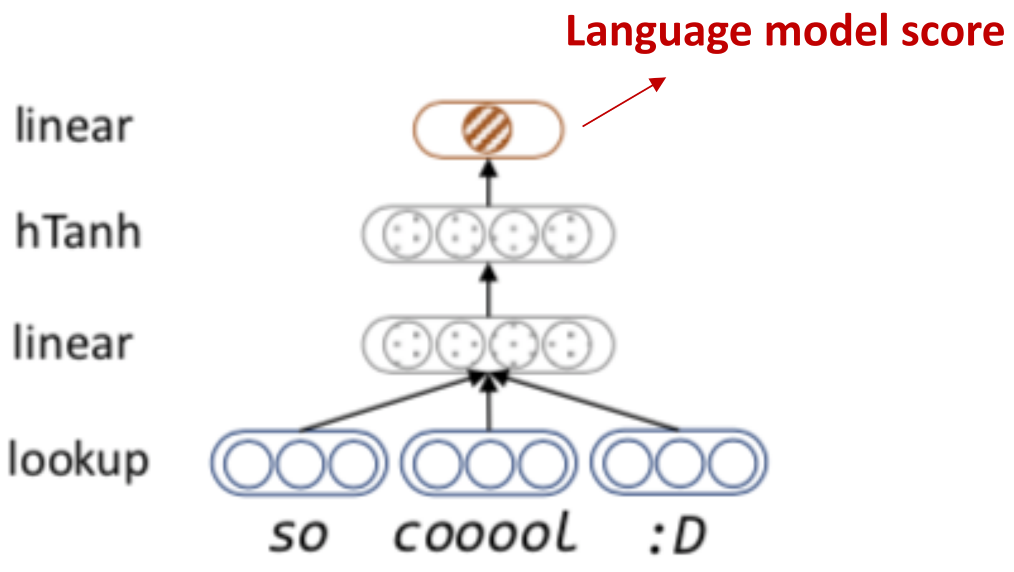


Image from: https://www.slideshare.net/perone/word-embeddings-introduction

C&W Model

Original tweet	cat	chills	on	a	mat
Corrupted tweet	cat	chills	job	a	mat

- Method
  - Replace the center word with a random word and create a corrupted tweet
  - Calculate language model score for each original tweet and its corrupted tweet
- The training objective
  - is that the original n gram is expected to obtain a higher language model score than the corrupted n gram by a margin of 1
- Cons: only model the syntactic context but ignore the sentiment information
  - “good” and “bad” are mapped into close vector
- Calls for combining sentiment context as well



(a) C&W

Algorithms comparison

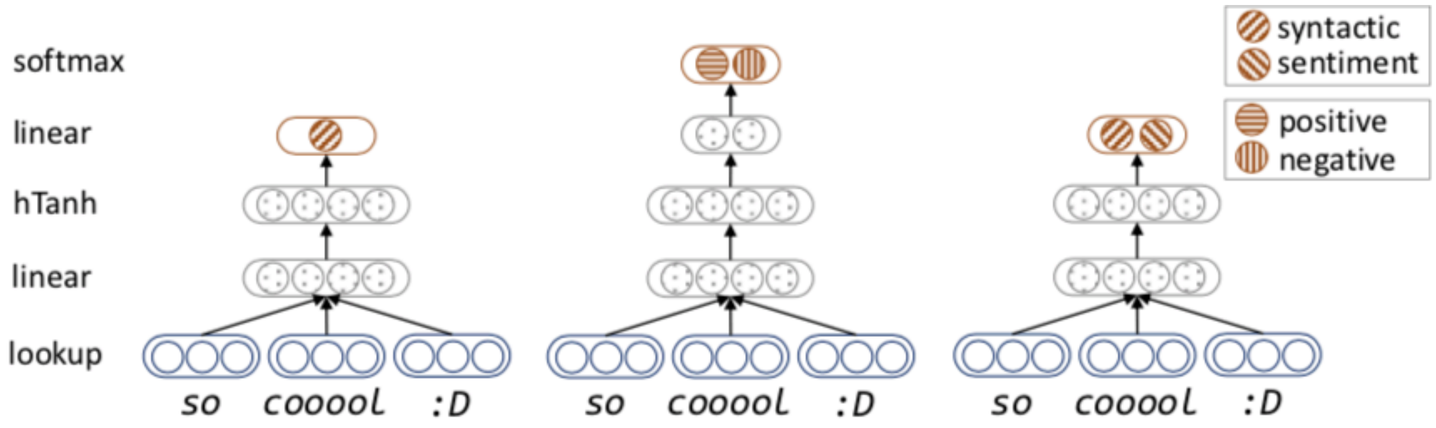
Model	Syntactic	Sentiment
C&W	YES	NO
SSWE(h)	NO	YES
SSWE(r)	NO	YES
SSWE(u)	YES	YES

* Sentiment-Specific Word Embedding (SSWE)

The authors’ approaches, will be introduced in the coming slides

SSWE(h)

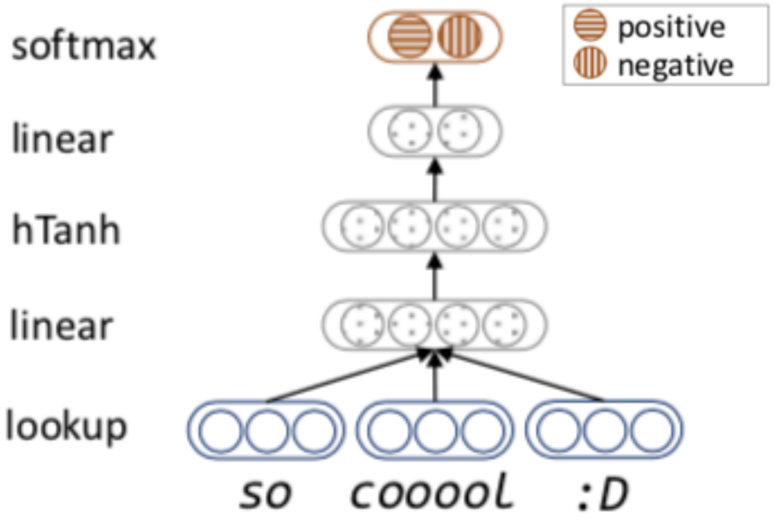
Original tweet	cat	chills	on	a	mat
lookup	[1,0,0,0,0….]	[0,0,0,0,1,….]			



(a) **C&W** (b) **SSWE$_h$** (c) **SSWE$_u$**

SSWE(h)

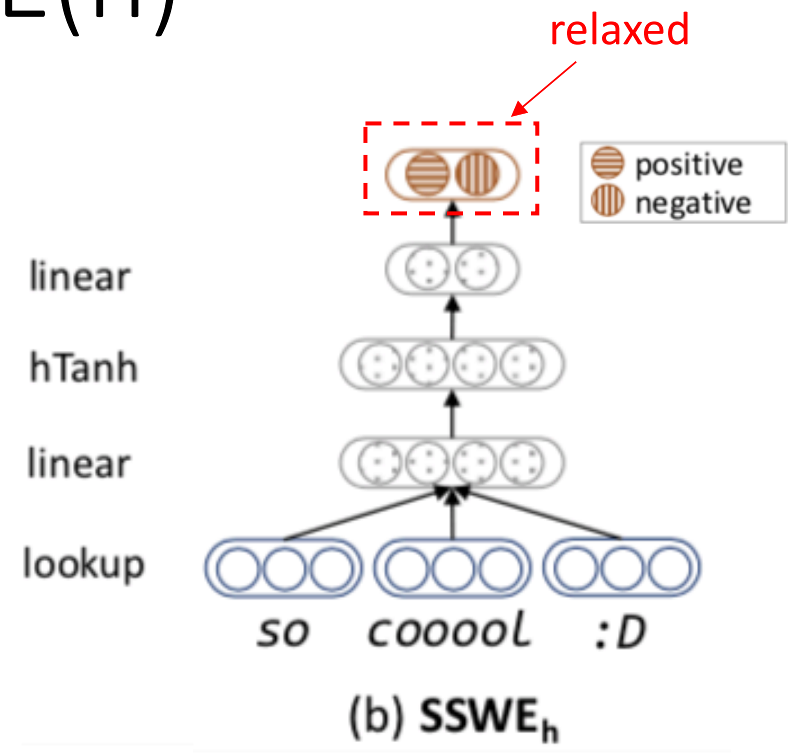
- Learn sentiment specific information
  - Predict sentiment classification for each tweet
  - Ground truth: [1, 0] as positive, [0, 1] as negative
  - Sample output: [0.7, 0.3]
- Process
  1. Lookup corresponding word embeddings
  2. Linear projection to lower dimension
  3. Hyperbolic tangent to constraint the range
  4. Linear projection to # classes (2)
  5. Softmax for prediction



(b) $\mathbf{SSWE_h}$

SSWE(r): relaxed version of SSWE(h)

- Relax the hard constraint of SSWE(h)
  - SSWE(h) define [1,0] as positive sentiment and [0,1] as negative
  - After relax the constraint, SSWE(r)'s output [a,b], if a > b, then positive, and vice versa
    - Borrow the first 4 layer from SSWE(h) and remove softmax layer

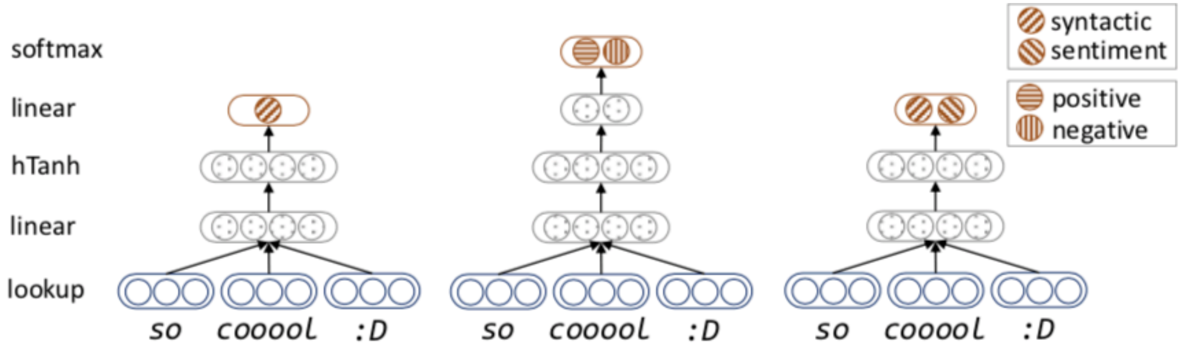


SSWE(u): unified sentiment and syntactic context

Original tweet	cat	chills	on	a	mat
Corrupted tweet	cat	chills	job	a	mat

$$loss_u(t, t^r) = \alpha \cdot loss_{cw}(t, t^r) + (1 - \alpha) \cdot loss_{us}(t, t^r)$$

- *"t"* is the original tweet and *"t^r"* is corrupted tweet
- $loss_{cw}(t, t^r)$ is the syntactic loss
- $loss_{us}(t, t^r)$ is the sentiment loss
- A is the hyper-parameter that weights two parts



(a) C&W (b) $\mathbf{SSWE_h}$ (c) $\mathbf{SSWE_u}$

Experiment

- Word embedding learning dataset
  - 10M Tweets (04/01/2013-04/30/2013), 5M positive and 5M negative, detected by emoticons.
  - *"We use the emoticons selected by Hu et al. (2013). The positive emoticons are :) :) :-) :D =), and the negative emoticons are :(: (:-("*

Positive:
:)
:)
:-)
:D
=)

Negative:
:(
: (
:-(

- Experiment dataset:
  - Twitter sentiment classification benchmark dataset: SemEval2013

	Positive	Negative	Neutral	Total
Train	2,642	994	3,436	7,072
Dev	408	219	493	1,120
Test	1,570	601	1,639	3,810

Result

Embedding	Macro-F1
C&W	74.89
Word2vec	73.21
ReEmb(C&W)	75.87
ReEmb(w2v)	75.21
WVSA	77.04
SSWE_h	81.33
SSWE_r	80.45
SSWE_u	**83.70**

Table: Macro-F1 on positive/negative classification of tweets with different word embeddings.

*Evaluation Metric: Macro-F1

Result

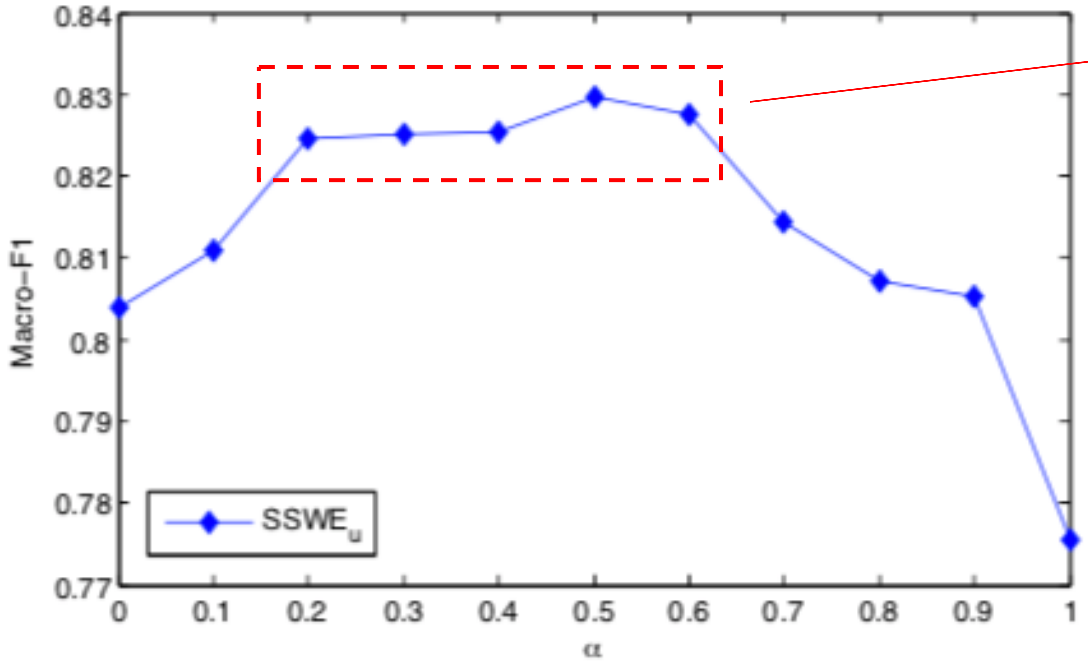


Figure 2: Macro-F1 of SSWE$_u$ on the development set of SemEval 2013 with different α.

Model performance is relatively robust when there is a “good” balance between syntax and sentiment

Weakness

- The embedding training dataset is biased by its twitter crawling criteria: using emoticon to determine sentiment polarity
- Human language expression is more than that (sarcasm):
  - E.g., I love doing my homework :)
  - This could mean: 1) I really love doing homework or 2) Nah, I’m just kidding
- The sarcasm, which usually comes with positive emoticon, will be labeled as positive sentiment

Future work

- Better ways to replace the twitter sentiment labeling method instead of using emoticon:
  - E.g., use online product review, classify sentiment polarity by the stars given

Thank you!

Back up

- Min, Max, Average of “embeddings”

$$z(tw) = [z_{max}(tw), z_{min}(tw), z_{average}(tw)]$$

Back up

- SSWE(u) loss function

$$loss_u(t, t^r) = \alpha \cdot loss_{cw}(t, t^r) + (1 - \alpha) \cdot loss_{us}(t, t^r)$$

$$loss_{cw}(t, t^r) = max(0, 1 - f^{cw}(t) + f^{cw}(t^r))$$

$$\begin{aligned} loss_{us}(t, t^r) = max(0, 1 &- \delta_s(t)\boldsymbol{f}_1^u(t) \\ &+ \delta_s(t)\boldsymbol{f}_1^u(t^r)\ ) \end{aligned}$$

Weakness

- Authors do not provide performance comparison between SSWE(h), SSWE(r) and SSWE(u)
  - Only SSWE(u) performance is compared with other state-of-the-art models
  - If SSWE(h)/SSWE(r) has close/better performance than SSWE(u), then there’s no need to use syntax information