Sentiment-specific word representations from tweets

Rama Dedeepya(201264161),Jitta Divya Sai(201225167)

Abstract:

A model to learn the word embedding for Twitter sentiment classification is developed. Most existing algorithms for learning continuous word representations typically only model the syntactic context of words but ignore the sentiment of text. As a result, words with opposite polarity, such as good and bad, are mapped into close vectors.We learn the sentiment specific word embedding (SSWE) by encoding the sentiment information in the continuous representation of words.

Introduction:

The objective is to classify the sentiment polarity of a tweet as positive,negative or neutral. We apply SSWE as features in a supervised learning framework for Twitter sentiment classification,which identifies the sentiment polarity of tweets.Instead of hand-crafting features, we incorporate the continuous representation of words and phrases as the feature of a tweet. Word embeddings provide a low dimensional dense vector space representation for words, where values in each dimension may represent syntactic or semantic properties.The sentiment classifier is built from tweets with manually annotated sentiment polarity.SSWE is evaluated by incorporating it into a supervised learning framework for Twitter sentiment classification

Problem: Learning sentiment-specific word representations from tweets

Dataset:

we were given a corpus, which contained 2,30,58,829 tweets and their corresponding polarity.

Method:

Data-Preprocessing:

Firstly, we tokenized the entire data and found that it was still not clear.

Secondly,except for the emoticons, we removed all the punctuations like,#,!,@ etc.Then all multiple spaces were removed.

Input-Format:

we used a window-size of 2 for the context , that is a positive-pair and a negative pair.So, each tweet was slided by a window of 3 and the corresponding label from the tweet was extracted.

This file is named as:

final_50kfilinput.txt

The Models:

We built two models,one is for the syntactic word embedding and one for the sentiment specific word embedding.We used a skip-gram model to obtain the word embeddings. Our word embedding size was 10.This means for a vocabulary of V, each word in it is represented as an embedding of size 10.This is learnt through these models. We used a Hardtanh non-linearity and a soft-max over it. The wordlookup and contextlookup is cloned,that is they are shared by the two models.Hence, we get the word embedding which is a union of both the models, as it takes the sentiment polarity information also.

Approach:

As the resources were not available,we couldn't train the neural network on the entire dataset(2,30,58,829 tweets).we took a part of the dataset which contains 50,000 tweets.The unique tokens in that are 40,640.This means,at the end we have word embedding for these 40,640 words.We used askip-gram model for both syntactic and sentiment specific learning.The input for synatctic neural network is a word which outputs its context(2 words).
For, sentiment specific model,the input is the word and the context (that is n-gram)and the output is the polarity.
To avoid memory errors,we have batch processed the input,taking the batch size as 50 at a time.
The code for this is:

skip_pol_batch.txt

The we have obtained a weight matrix of the word embeddings,which of the size 40,640X10.This embedding is in **cls_feat50k.txt**

Sentiment-specific Classifier:

We used both SVM classifier and KNN choosing k=5.

Problem encountered: As we haven't used the entire **4GB** data.we did not get the word-embedding for all the words in the vocabulary.So,in a tweet , if there is no word embedding for a particular word, we skipped the word .

The tweet is a sum of word-embedding of all its words.Then the vectors are normalized.So,as we skip through a few words,the final feature vector for a tweet might not be having information about a few words.These feature vectors were fed into both the classifiers.

From the training data, we removed those with label “0”(neutral sentiment tweets).We, did this as the 50,000 word _embedding corpus did not have a

tweet with 0 polarity.SO,this is a 2-class classification.
We first did a validation taking 80% of the training data was used for training and 20% for testing.Even in the test-data we removed the neutral tweets and measured the accuracy.

Results:

Sno	Method	Accuracy
1	SVM+unigram (size of word embedding : 10)	68%
2	KNN(k=5) + unigram (size of word embedding : 10)	65%
3	SVM+unigram (size of word embedding : 5)	62%
4	KNN(k=5) + unigram (size of word embedding : 10)	59%
5	SVM+unigram (size of word embedding : 20)	60%
6	KNN(k=5) + unigram (size of word embedding : 20)	56%
7	KNN(k=3) + unigram (size of word embedding : 10)	61%
8	KNN(k=3) + unigram (size of word embedding : 5)	57%
9	KNN(k=3) + unigram (size of word embedding : 20)	63%

Reference:

http://anthology.aclweb.org/P/P14/P14-1146.pdf

http://ceur-ws.org/Vol-1404/paper_21.pdf