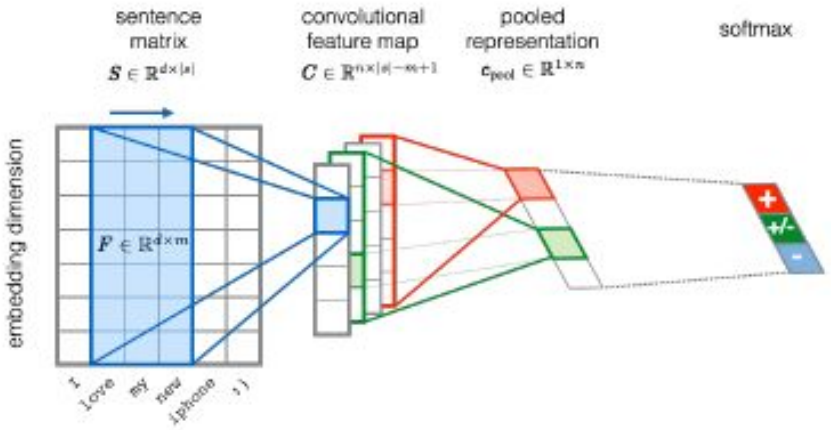
Twitter Sentiment Analysis using Deep Learning Models

Jiaying Guo
Ang Li
Yuan Li
Yujie Sun
Shiping Yi

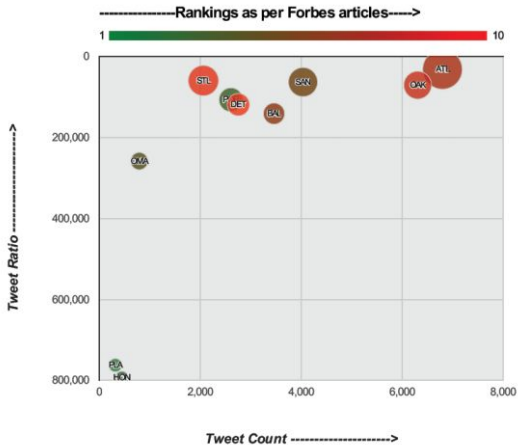
Illustrative example



Severyn, A., & Moschitti, A. (2015). *Twitter Sentiment Analysis with Deep Convolutional Neural Networks.*

Applications of Sentiment Analysis

There have been many breakthroughs with Natural language processing techniques that have paved the way for commercial advertising, social media monitoring of elections, as well as detecting criminal activity from user tweets.



https://www.brandwatch.com/blog/understanding-sentiment-analysis/

Bolla, Raja Ashok (2014). *Crime Pattern Detection using Online Social Media*

Problem Definition

Achieve the state-of-the-art performance using deep learning methods (Convolutional Neural Networks specifically) to train a set of pre-processed dataset obtained from user tweets.

Extensions:

- Feed manually-annotated user tweets related to Covid-19 into the trained final model and derive accuracy results
- Use other word-embedding features to train the network and compare performance
- Vary model architectures to incorporate LSTM into the baseline model

Past Approaches

- Naive Bayes
- Logistic Regression
- CNN with pre-trained word embeddings
- CNN with POS tags as additional input features
- LSTM
- Combination of CNN and LSTM
- CNN and GRNN

Data

- Severyn, A et. al (2015) uses the dataset from Semeval-2015 Twitter Sentiment Analysis Challenge.
- T4SA Dataset - Twitter for Sentiment Analysis Dataset used by Vadicamo, L. et. al (2017)
  - Containing 1,179,957 selected tweets with multi-class probabilities (NEG, NEU, POS) predicted by tandem LSTM-SVM architecture
  - Preprocess:
    - Merge the text file and prediction result file into a file containing text and class probabilities, **-1: negative, 0: neutral, 1: positive**
    - Regular expression matching: convert all urls to “URL” string; convert all “@username” to “AT_USER” string; “#topic” to “topic” string
    - Split the dataset into train, dev, test with size 884,967 58,998 23,992

Glimpse of the Data

- Original Data
  - containing columns: 1) id 2) text 3) TWID 4) NEG 5) NEU 6) POS
  - 768097631864102912,RT@2pmthailfans:[Pic] Nichkhun from krjeong86’sIG https://t.co/5gcAcu9by7,768097631864102912,0.014643660775799998,0.9265568133679999,0.0587995258562
- Processed Data
  - containing columns: 1) label 2) text
  - 0, AT_USER:[Pic] Nichkhun from krjeong86’sIG URL

Random Baseline Performance

- We used a random baseline that predicts positive for a sentence by randomly giving a label.
- T4SA dataset was split into train, test, and dev by 884,967: 58,998: 23,992

Dataset	Precision	Recall	F1 score
Train	0.3341	0.3342	0.3150
Validation	0.3297	0.3289	0.3102

Simple Baseline Performance

- We used a simple baseline that predicts positive for a sentence based on the number of positive words versus negative words
- T4SA dataset was split into train, test, and dev by 884,967: 58,998: 23,992

Dataset	Precision	Recall	F1 score
Train	0.7340	0.6347	0.6641
Validation	0.7354	0.6338	0.6635

Published Baseline Model Architecture

- Weight initialization
  - Word2vec model (300d) to learn embeddings on unlabeled tweet corpus
  - Distant supervision with NN to refine embeddings
- Layers
  - One layer of Convolutional Layer, number of filters = 300, filter size = 5
  - One layer of max pooling
  - One layer of RELU
  - One layer of fully-connected softmax
- Optimization
  - Stochastic Gradient Descent (Adadelta)
- Regularization
  - L2-normalization, dropout = 0.5

Experimental Results

After training for 5 epochs, our results on training set, validation set and test set are as follows:

Dataset	Precision	Recall	F1 score
Train	88.55	87.74	88.12
Validation	93.96	93.04	91.04
Test	93.72	90.60	92.00

Comparing with Published Results

Although we use different datasets for the model without the 3-step pretraining, our final results are slightly better than the paper:

Model	**Severyn, A et. al.**	**Our model**
F1 Score	84.79	92.00

Model Performance Analysis

Our model's results aren't directly comparable to the published baseline model, but its good performance is probably mainly due to the size of our dataset:

Dataset Size	Train	Dev	Test
T4SA	884,967	58,998	23,992
Twitter'13 (paper)	5,895	648	2,734

Extension 1 Other Word-Embedding

- Model Architecture
  - 1 Convolutional layer
  - Applied word embedding:GloVe(Global Vectors for Word Representation),FastText and Google word2ve
- Performing Results

filters size	[5]	[5,5]	[3,4,5]	[5,4,3]
word2vec	0.9200	0.9357	0.9454	0.9697
6B100D	0.8812	0.8923	0.9051	0.9123
6B300D	0.9093	0.9169	0.9222	0.9433
42B300D	0.9457	0.9571	0.9697	0.9658

Extension 2 LSTM Integration

- Model Architecture
  - 1 Convolutional layer + 1-3 layers of LSTM network
  - 1-3 layers of LSTM network + Baseline model
- Best Performing Results

Dataset	Precision	Recall	F1 score
Training	91.38	88.80	89.99
Validation	93.19	90.74	91.87
Test	94.99	92.21	93.48

Extension 3 Covid19

- Covid19 Dataset
  - 772 manually Annotated Covid19 tweets
  - Annotated by 3 people, choosing the label most people agree with
- Different Model Performing Results

Dataset	F1 score
Random Baseline	0.3489
Word-Count Baseline	0.4507
CNN(Published Baseline)	0.5983
CNN+word-embedding(Extension1)	0.6007
LSTM+CNN(Extension2)	0.5930

Tweets Sentiments Trend