Sentence Classification using CNN

Joyfred Jesuraja A
16PD13

1. Introduction

Sentiment analysis of short texts such as single sentences and tweets are challenging because of the limited contextual information that they normally contain. Effectively solving this task requires strategies that combine the small text content with prior knowledge and use more than just bag-of-words.

Typically, CNN's are considered appropriate for image classifications. Based on past research works, certain niche architectures of CNN are considered successful for classification problems of NLP.

2. Dataset Used

Data[1] is a CSV consisting of 16,00,000 tweets with emoticons removed. Data file format has 6 fields as mentioned below:

0 - the polarity of the tweet (0 = negative, 4 = positive)
1 - the id of the tweet (2087)
2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009)
3 - the query (lyx). If there is no query, then this value is NO_QUERY.
4 - the user that tweeted (robotickilldozr)
5 - the text of the tweet (Lyx is cool)

3. Network Architecture

1. 1 embedding layer
2. 3 region sizes = (2, 3, 4)
3. 100 filters for each region size, Totally, 300 filters
4. 1 max pooling layer after each unique region sized layer
5. A Dense Layer of ReLu activation
6. A Drop-out Layer
7. A Dense Layer of Softmax activation

[1] "For Academics - Sentiment140 - A Twitter Sentiment Analysis ...." http://help.sentiment140.com/for-students.

Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded

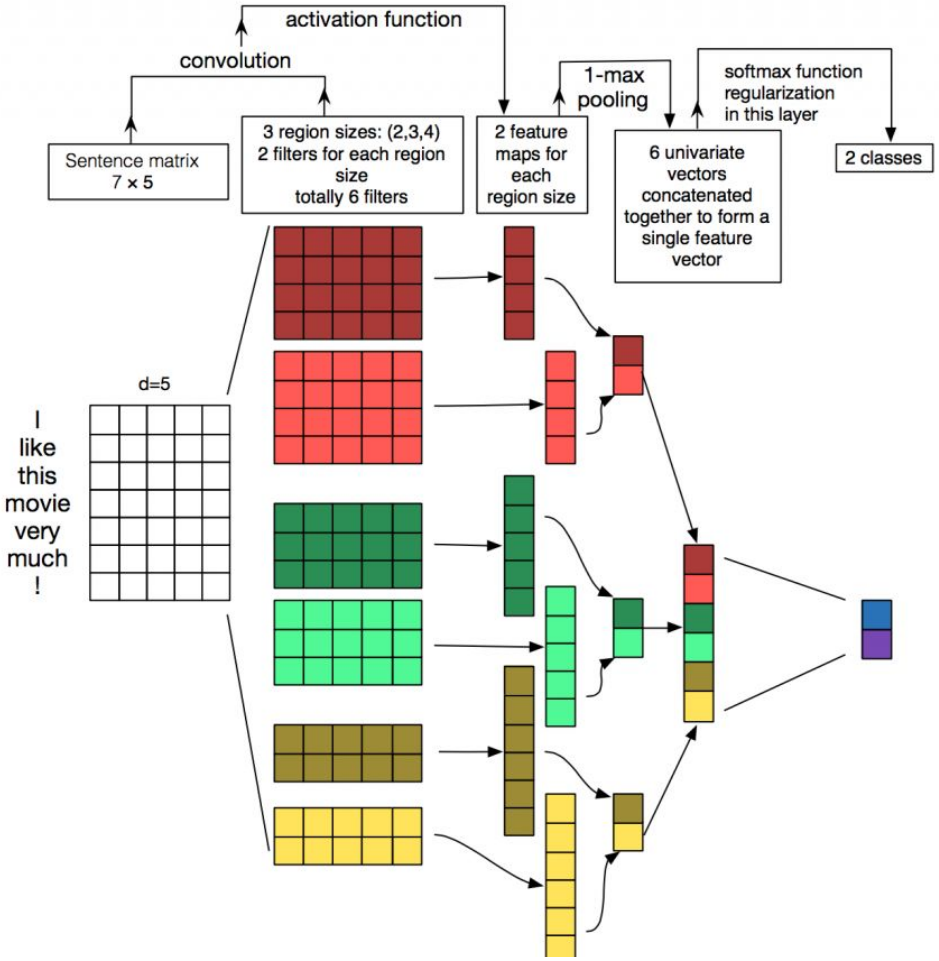

Fig: Simplified Representation(2 filter/region size & 1 dense layer) of the network architecture

4. Conclusion

Out of 1600000 tweets, 16000 are used for testing and the remaining for training. Distribution of positive and negative tweets are equal (8,00,000 each). Model was trained for 5 epochs with binary cross-entropy loss.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Finally, loss of 0.138 and accuracy of 94.8% was achieved.