Machine Learning Engineer Nanodegree

Capstone Proposal

Sentiment Analysis Using convolutional Neural Networks

Ahmed Saafan

October 12th, 2019

I. Definition

Project Overview

Problem Domain

Natural language processing (NLP) is one of the active brunches of the Artificial Intelligence/Machine Learning field that attracts lots of research and attention. NLP as the name suggests deals with the natural human language using computers to analysis and find patterns in written and spoken language. NLP applications can be found in almost all language related technologies, like Siri or Alexa (speech recognition), Google's search suggestions, text summarization and lots of other applications. But one very interesting application is sentiment analysis (SA), where a model finds the sentiment behind a text, whether it's a sentence, a paragraph or document. SA can be used to classify reviews on movies or products, whether they are positive or negative (polarity), or to find the emotion behind them, angry, sad, happy .etc. Historically NLP used rule-based algorithms that were developed by hand for analyzing text, then it used statistical methods which gave much better results, but in recent years machine learning and deep learning proved to have better potential to solve the NLP challenging problems(Liu. 2012).

Sentiment analysis can be very useful for applications that are related or affected by the "mode" of the people/users. Companies for example may be interested to know how the people are reacting to a product or the company as a whole. A politician running for elections will be interested in how people are reacting to his to the opponents campaign. SA in other words can be used as a metric on how people "feel" about something, this is made easy by the amount of huge raw data that can be found on social media, where users are talking and sharing their opinions and feelings all the time(Pang and Lee. 2008).

Project Origin

For me personally I find the NLP very interesting field that I want to explore more by starting with one of the important branches of this field with its huge potential and applications that are growing with time.

Dtataset

For this project a large movie review dataset to be used. The data is collected from the IMDB website known as Large Movie Review Dataset. The data was collected by Stanford as a new and larger benchmark dataset, the dataset contains 50K movie reviews, 25k for training and 25k for testing. Each half is divided into positive and negative reviews. The dataset also contains unlabeled movie reviews, but this set will not be used in this project. The dataset also is balanced when dealing with popular movies, as the collectors limited the review/movie to be 30 max (Maas et al. 2011)

Training data :

- 25k reviews in total, in 2 files {positive and negative}
  - 12.5k are positive reviews
  - 12.5k are negative reviews

Testing data :

- 25k reviews in total, in 2 files {positive and negative}
  - 12.5k are positive reviews
  - 12.5k are negative reviews

The dataset will be used to train and test both of the benchmark and the deep learning model performing sentiment analysis on different movie reviews. The dataset is large enough to train deep learning models as they require huge amounts of data, also the dataset is very polar, which means that each review is strongly positive or negative, which will make life easier for the classification task.

Joker 2019 movie reviews:
As a small validation set to further test the model, a small dataset of 10 reviews of the jokers film as collected, the reviews are balanced as5 highly positive and 5 highly negative (10 and 1 start). The dataset was collected from the IMDB and is attached with the projects files with links to each review.

Problem Statement

Given a movie review, the model should determine whether the review was a positive or a negative review. The reviews are real ones taken form popular movie reviews website that you usually read when looking a movie up. The reviews are tagged with their polarity (pos/neg) and the model will find patterns to classify the reviews after training, which makes this a supervised learning problem (the model is feed a pretagged data), that can be measured by the accuracy of the model.

Metrics

As mentioned before the accuracy score will be used as an evaluation metric for this task, as the task is a supervised learning task with clearly polar data that are labeled, and the dataset is balanced. The accuracy score will point the percentage of the correctly classified reviews to the total classified reviews, we are interested in how well the model can classify the data and the accuracy score gives a simple, easy yet a good metric to evaluate our task.

II. Analysis

Data Exploration

The logical point to start with any data related project is by exploring the data that will be used. The dataset is first loaded, both training and testing data are grouped in 2 text files each is 25K reviews long. First 12.5k reviews are positive and the rest is negative for both files. This is an example of one of the reviews in the training set:

> 'Very nicely done movie. It does stay in your memory. Better billed as a romance than flying or war, altho the flying parts are realistic and almost error free. Flying buffs like myself will enjoy this movie even if attracted by the airplanes, unless they have no sensitivity or have never been in love.

Fun watching early Crowe. He is good and exudes charm. His reading of "High Flight" is superb.

cheers, Boom'

As we can see that the data is not clean and contains HTML derbies

 these characters should be removed later on in the preprocessing step as this might cause unwanted noise in the training.

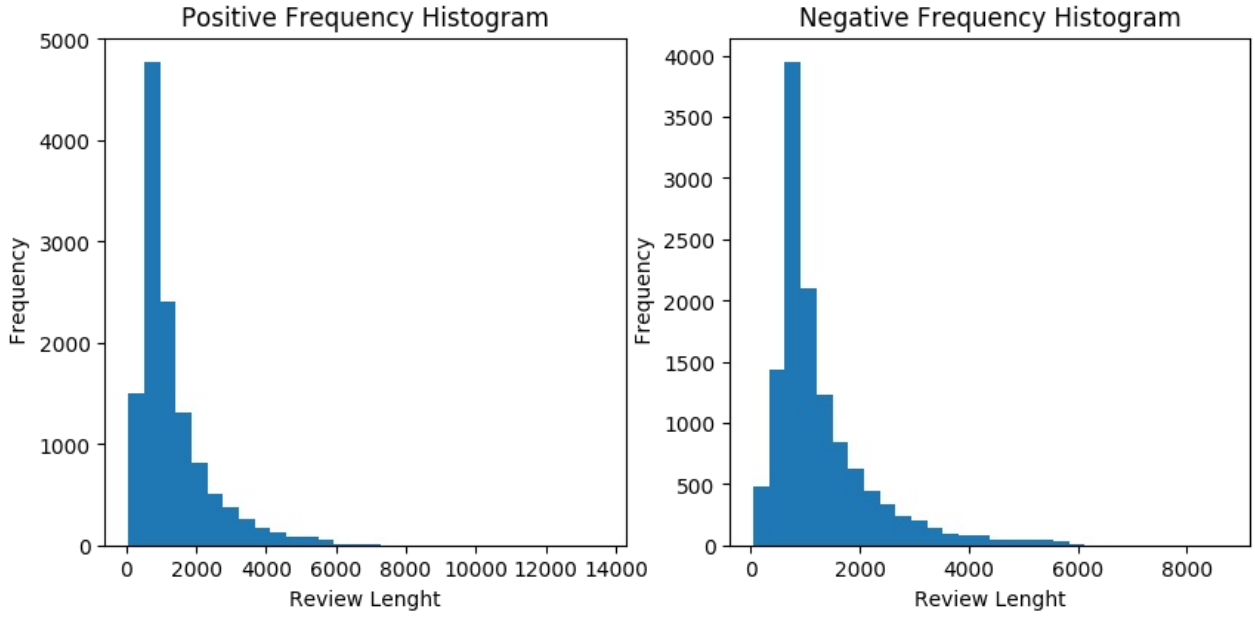
```
The stats of the train dataset
Mean: 1325.068600
Median: 979.000000
Minimum: 52.000000
Maximum: 13704.000000
Std: 1003.112772
Total # of words: 5844680
Total # of vocabulary: 251637

The stats of the positive reviews
Mean: 1347.159040
Median: 982.000000
Minimum: 70.000000
Maximum: 13704.000000

The stats of the negative reviews
Mean: 1302.978160
Median: 976.500000
Mininum: 52.000000
Maxaimum: 8969.000000
```

Next we look at descriptive statistics of the training dataset, we see that the total number of vocabulary is much less than the total number of words in the dataset, that's because words repeat in sentences and across different reviews as well. As we have 2 classes, we need to check if the dataset is balanced not just in total number of reviews but in the characteristics of each review. We can see that the statistics of the length of reviews are very similar for both positive and negative, and these statistics are generally the same for the whole training data, this means that the training dataset is quite balanced and accuracy as a metric is going to be enough.

Because of the nature of the data (text) some kind of encoding is required in order for the models to be able to work with the data, this kind of encoding will be addressed later on.

Exploratory Visualization

Another step in the exploration is to check the sentence length distribution for both the positive and negative reviews.



> Figure 1. Review length distribution for training data(postive and negative.

From the figure we can see that the training data in general is skewed and has almost the same distribution for both negative and positive, this skewness is expected as reviews in general are not quite long. From the previous section and from the plot we can see that most reviews are less than 900 words. This information will be useful when preprocessing for the deep learning model, and we will talk about it later.

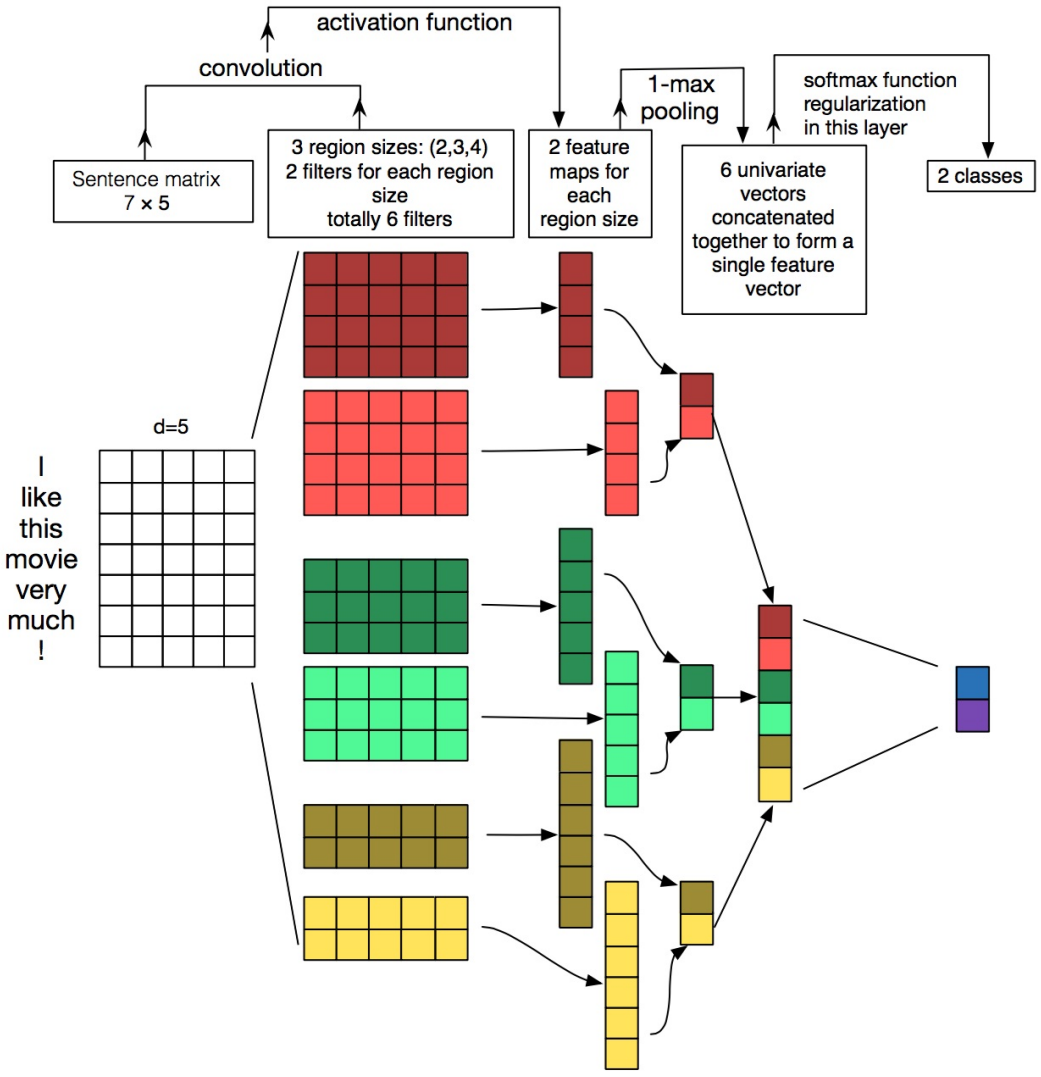
Algorithms and Techniques

There were 2 algorithms used for this project, the first one is the benchmark that utilized a Logistic Regression, and the solution model that used a deep learning convulotional neural network. Both methods required encoding to be ready to enter the model.

CNN Deep Learning Model: As a deep learning model the features are feed to the model, each feature is represented with a node in the input layer, each node is connected to the deep/hidden layers by a matrix of weights and biases. The hidden layers take these linear features and weights and biases and connect them in a non-linear way using activation functions. This introduced non-linearity is what makes DL a strong model as it's able to preform complex tasks that linear models can't handle. Finally, the features arrive at the output layer were the model tries to predict a class. Then the network compare the prediction with the true classification and start training and modifying the weights and biases using an optimization method.

Convlutional neural networks are type of layers/models used in deep learning that has proven to be effective with tasks like image classification. What makes CNN different is that the features are not connected to all the hidden layer nodes, which reduces the amount of parameters greatly. CNN were applied first to image input, using a collection of different filters that scan the image for patterns, the conv nets find/learn specific patterns in the image similar to how our brain process images. The result is a feature map, where image is reduced to pattern that is present in the image. These maps are then pooled to take the max/average of each part of the image to further reduce the dimension.

It wasn't long until CNN were applied to NLP (kim,2014), similar to image the CNN slide different filters on the input to find patterns that are learned by the model.



> Figure 2. Convolutioanl filters for text input, source (Zhang, Y., & Wallace, B. (2015))

As you can see the sentence is staked as a matrix, the filters have the same width as the input matrix, but with different height (number of words). Some filters will look for 3-words combinations, others for 4 and so on, this is known as n-gram.

I'm choosing CNN model as it's relatively simple compared with other DL models that deal with NLP. What I'm looking for is the model spotting phrases like (' liked the movie', 'movie was great', 'wasted my time') to classify the review. of course this is very simple solution for a complex problem but I think it will be better or similar to what the benchmark model. Also, this model does not require text normalization, which makes it attractive for other language that do not have the NLP resources that English has (Zhang 2016).

Benchmark

Logistic Regression: This is an ML model that is used for classification with the right parameter adjustments. The model is historically used for NLP tasks and was chosen because it's simple and easy to deploy, which is need for a baseline model. The model was trained on the training dataset and tested on the testing dataset. The model implementation details are discussed later. After

vectorization and proper preprocessing, the model achieved:

```
Training_Accuracy = 87.17
Test_Accuracy = 86.78
```

III. Methodology

Data Preprocessing

Data Preprocessing:

Tokenization: Is the process of splitting the sentence into individual words . This process is done for both benchmark and the solution model as it's very important in text processing. Both tokenizers used removed punctuation/special characters by default as they don't care any real meaning except for humans to make easier to read.

Bag of Words (BoW): is the method used to turn the text to a feature that the benchmark model could use. This technique creates a dictionary of the vocabulary of the data set, with each word there is an assigned index. Then each sentence is mapped to a binary vector (on/off) with length of the whole unique words. An example is better to clear this concept:

```
sentence =['I like dogs','Dogs don't like milk']
dict{'dogs':1,'like':2,'dont':3,'milk':4,'I':5}

first sentence becomes:
feature1=[1 1 0 0 1]

second sentence becomes:
feature2=[1 1 1 1 0]
```

From this example, the feature vector has 'dogs' as it's the first dimension, if the sentence contains the word 'dogs' it will assign 1 to the first entry, otherwise 0 is assigned. As you can see this will from a huge sparse matrix of lots of zeros, which is not computationally efficient. Other thing to note is that this technique does not reserve the word order of the sentence or number of repeated occurrences of the word. This may cause a loss of valuable information.

Word Embedding: Unlike the simple ML models, Deep learning models use more efficient ways to vectorize the text. Word embedding is one of the most popular one as it's proven to boost the performance for tasks like sentiment analysis(reference needed). Word embeddings are kind of feature reduction as they represent the test in dense word vectors that are shorter than conventional methods.

First word embedding creates a vocabulary vector based on the word counts i.e. the first word will be the most occurring word and so on. Then the sentence will be replaced with numbers, each word is assigned its index number in vocabulary dictionary. This way the structure and word order are reserved. After that a padding process will take place as each sentence will be adjusted to be same length, the sentence will be cut off or padded with zeros if it was short. The vocabulary and the sequence padding sizes are 10000 and 800 respectively. This is based on the statistics from data exploration part as the median sentence length is 900.

When the text is ready a neural layer on top of the model will reserve the input and use deep learning techniques to map semantic meaning into embedding space, that is the model will try to find dependence between different part of the text and mapped to have a vector like operations : king-man+woman=queen (Pennington et al 2014).

Data Cleaning:
we mentioned before that there were abnormalities with the dataset, mainly the HTML debris. A helper function `get_clean` using regular expressions removed these debris. This is the result of applying this function on the data:

> 'Very nicely done movie. It does stay in your memory. Better billed as a romance than flying or war, altho the flying parts are realistic and almost error free. Flying buffs like myself will enjoy this movie even if attracted by the airplanes, unless they have no sensitivity or have never been in love. Fun watching early Crowe. He is good and exudes charm. His reading of "High Flight" is superb. cheers, Boom''Very nicely done movie. It does stay in your memory. Better billed as a romance than flying or war, altho the flying parts are realistic and almost error free. Flying buffs like myself will enjoy this movie even if attracted by the airplanes, unless they have no sensitivity or have never been in love. Fun watching early Crowe. He is good and exudes charm. His reading of "High Flight" is superb. cheers, Boom'

Implementation

Logistic Regression:
The simple logistic regression model for baseline was implemented in the `Bench.ipynb` devolved and run on the Colab platform. The implementation used the `sklearn` library, from which `linear_modle.LogisticRegression` was used as the classifier. For tokenization, also from the same library I used the `CountVectorizer` to ready the data for the model. The model was then trained and tested using the model's methods and the result was reported in the Benshmark section.

CNN Deep Learning Model:
The CNN model suggested as a solution for this project is implemented in the `Modle.ipynb` file, developed and run by Colab platform. The model's architecture is based on (kim 2014) paper, which was the first paper that used convlutional networks for nlp tasks. and the `Keras` functional API. The model's architecture can be seen below:

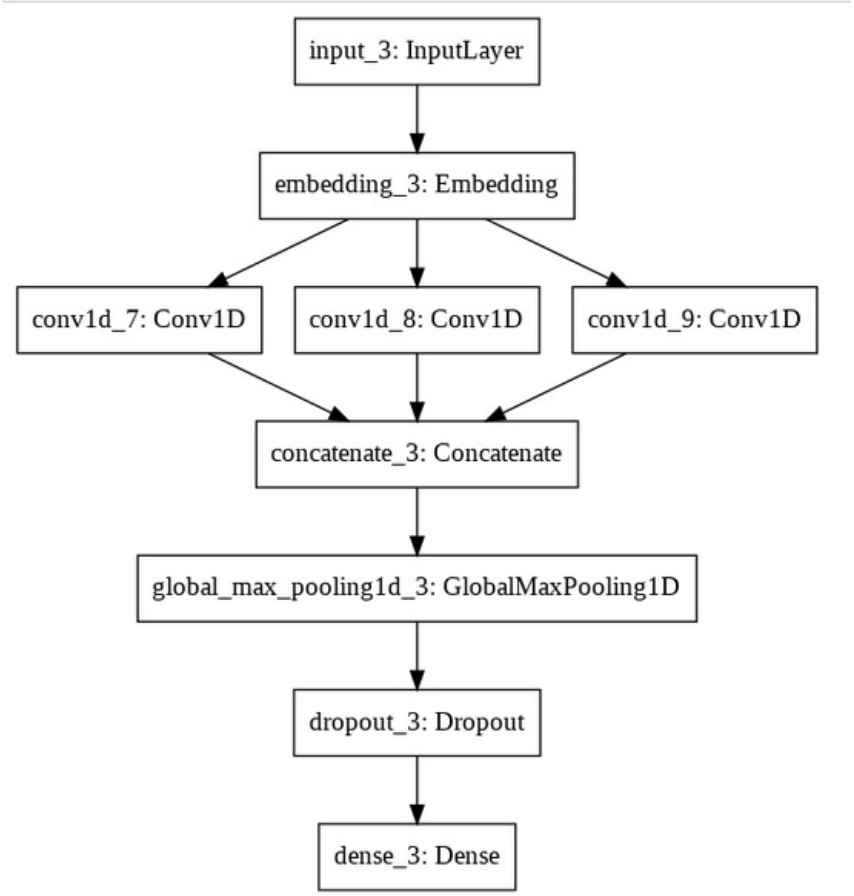
```
# First modle
from keras.utils import plot_model
from keras.layers import Input,Embedding,Conv1D,concatenate,GlobalMaxPooling1D
from keras.layers import Dropout,Dense
from keras.models import Model
from keras import regularizers


start=Input(shape=(sent_size,))
embd=Embedding(vocb_size,50)(start)

cnv1=Conv1D(250,3, padding='same' ,activation='relu')(embd)
cnv2=Conv1D(250,4, padding='same' ,activation='relu')(embd)
cnv3=Conv1D(250,5, padding='same' ,activation='relu')(embd)


mrg = concatenate([cnv1, cnv2,cnv3])

mrg=GlobalMaxPooling1D()(mrg)
mrg= Dropout(0.2)(mrg)
# mrg=Dense(10, activation='relu')(mrg)
# mrg= Dropout(0.4)(mrg)


out=Dense(1, activation='sigmoid')(mrg)
model = Model(start,out)
model.summary()
```

```
Model: "model_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            (None, 800)          0
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 800, 50)      500000      input_3[0][0]
__________________________________________________________________________________________________
conv1d_7 (Conv1D)               (None, 800, 250)     37750       embedding_3[0][0]
__________________________________________________________________________________________________
conv1d_8 (Conv1D)               (None, 800, 250)     50250       embedding_3[0][0]
__________________________________________________________________________________________________
conv1d_9 (Conv1D)               (None, 800, 250)     62750       embedding_3[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 800, 750)     0           conv1d_7[0][0]
                                                                 conv1d_8[0][0]
                                                                 conv1d_9[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_3 (GlobalM (None, 750)          0           concatenate_3[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 750)          0           global_max_pooling1d_3[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 1)            751         dropout_3[0][0]
==================================================================================================
Total params: 651,501
Trainable params: 651,501
Non-trainable params: 0
__________________________________________________________________________________________________
```

> Figure 3. The first modle' architecture

This was the first model, utilizing : an input layer, an embedding layer with output shape = 100, 3 Conv1D layers for text data using relu activation function, they are of different filter sizes(3,4and5) but same number of filters. Then the layers are pooled in a globalpooling layer, a dropout layer of 20% is added to reduce over fitting and finally the output layer is a Dense layer with 1 node and a sigmoid activation to produce the binary classification (positive of negative).

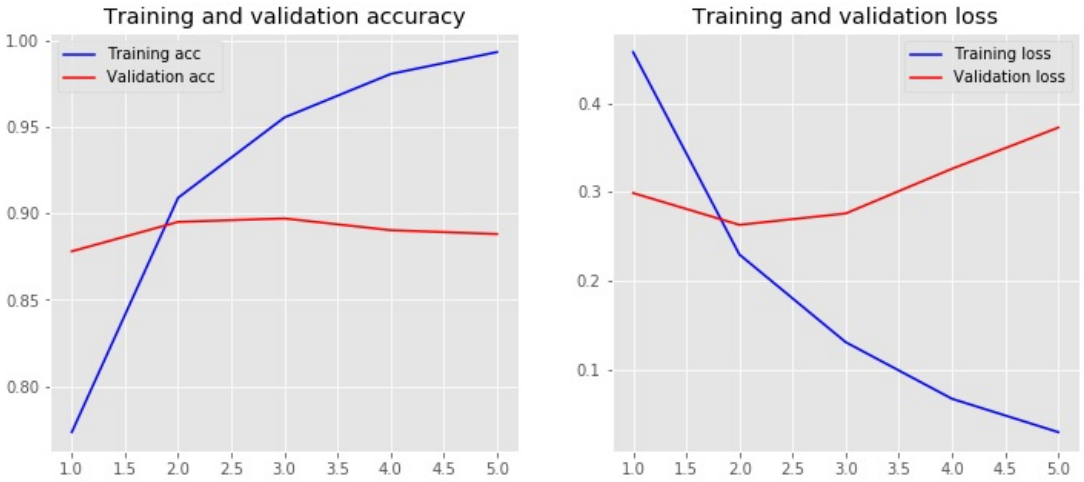
This implementation was not similar to the one developed before in the dog breeding project because the Conv layers were not sequential but worked in parallel to each other, this was quit a hard task but thanks to this post (Brownlee,2018) I manged to create the desired architecture and the next figure shows the final implementation:



> Figure 4. The first modle' architecture showing the parallal conv nets

Refinement

Working with deep learning models requires lots of adjustment and tweaking to finally get to the final model. The first problem I faced and found was common in DL model's dealing with NLP is over fitting, running the first model and getting the accuracy and loss plots of the training and validation, it was clear that the model used to over fit as can be seen below:



> Figure 5. Training history of the first modle

From the plots we can see that the model starts over fitting after 2 epochs, so the number of epochs is lowered to 2 which is similar to the suggested model from keras (1D CNN for text classification, n.d).

The number of conv layers did not affect the accuracy much and the 3 filters proposed are the best, however the number of filter was reduced from 250 to 128 and was an improvement. Also changing the embeddings output form 50 to 100 raised the accuracy but the number of parameters increased a lot as well. Changing the dropout rate to higher numbers proved to drag the performance and a small dropout rate like 20% is the best for this task.

Final model:
After a lot tweaking and following the general guidelines from (Zhang, 2016)the final model was a better implementation and a reasonable solution for the problem as we will see in the results section. The model's final architecture can be seen below:

```
# The final modle
from keras.utils import plot_model
from keras.layers import Input,Embedding,Conv1D,concatenate,GlobalMaxPooling1D
from keras.layers import Dropout,Dense
from keras.models import Model
from keras import regularizers


start=Input(shape=(sent_size,))
embd=Embedding(vocb_size,100)(start)

cnv1=Conv1D(128,3, padding='same' ,activation='relu')(embd)
cnv2=Conv1D(128,4, padding='same' ,activation='relu')(embd)
cnv3=Conv1D(128,5, padding='same' ,activation='relu')(embd)


mrg = concatenate([cnv1,cnv2, cnv3])

mrg=GlobalMaxPooling1D()(mrg)
mrg= Dropout(0.2)(mrg)
# mrg=Dense(10, activation='relu')(mrg)
# mrg= Dropout(0.4)(mrg)


out=Dense(1, activation='sigmoid')(mrg)
model = Model(start,out)
model.summary()
```

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 800)          0
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 800, 100)     1000000     input_4[0][0]
__________________________________________________________________________________________________
conv1d_10 (Conv1D)              (None, 800, 128)     38528       embedding_4[0][0]
__________________________________________________________________________________________________
conv1d_11 (Conv1D)              (None, 800, 128)     51328       embedding_4[0][0]
__________________________________________________________________________________________________
conv1d_12 (Conv1D)              (None, 800, 128)     64128       embedding_4[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 800, 384)     0           conv1d_10[0][0]
                                                                 conv1d_11[0][0]
                                                                 conv1d_12[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_4 (GlobalM (None, 384)          0           concatenate_4[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 384)          0           global_max_pooling1d_4[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 1)            385         dropout_4[0][0]
==================================================================================================
Total params: 1,154,369
Trainable params: 1,154,369
Non-trainable params: 0
__________________________________________________________________________________________________
```

> Figure 6. Final modle's architecture after the refinment process

IV. Results

Model Evaluation and Validation

The final model was trained on 80% of the dataset and the remaining 20% were used as validation. The split was totally random each time the model was run after some kind of modification.

After training the model's accuracy was tested both on training dataset and on the testdataset, that was unseen data of the same size of the dataset. As will be seen in the next section the training and testing accuracies are very close which indicates that the model has learned the task well enough.

To further ensure the robustness of the model, the model was feed a 10 reviews of the Joker 2019 movie, the small dataset was 5 positive and 5 negative reviews. The results of this small validation is reported in the next section, but the model was very successful in prediction the right sentiment, except for only one negative review.

Justification

The performance of the final project has achieved a better results compared with the benchmark model as can be seen from the table. The difference in the test accuracy is 2.65%.

Model	Training Accuracy (%)	Testing Accuracy(%)
CNN	89.82	89.38
Benchmark	87.17	86.73

> Table 1 . The accuracies of the modle and the benchmanrk

The model also achieved 90% accuracy on the joker's dataset, although it's a small dataset, but it

shows the model can handle real world data just fine.

To check whether the final solution is significant, McNemar's Test was applied to both classifiers following this tutorial(browunlee,2018). The test is at the end of the `Modle.ipynb` and uses the `statsmodels.stats.contingency_tables` model. Running the test we get the following result of the test:

> statistic=1619.000, p-value=0.000
> Different proportions of errors (reject H0)

which means the hypothesis is rejected (there is a statistical significant difference) and the proposed solution is better than the benchmark.

V. Conclusion

Free-Form Visualization

For the free-form visualization, I choose the only review that was misclassified of the 10 ten jokers' review:

> I don't see how that highly intelligent Joker on Dark Night started from this insane person. The Joker on The Dark Night is a highly intelligent person who holds deep conversations with prominent people, capable of planning a complicated bank robbery and was one step ahead of Batman, the police and intelligence. He has a strong personality free from emotionality and he laughed controllably. I can't see how this mentally unstable person on this Joker becomes that awesome Joker. 0

The zero at the end of the review indicates that the review is negative but it was classified by the model as a positive review. The review was a 1 star, which means it's very negative, but looking at it's very hard to know if it really was that negative, as a human the wording is not that negative, and I'm not sure if the reviewer only dislikes the movie or the joker's role. He is only comparing two roles from different movies in an indirect way with no mention he feels or think about the rest of the movie.

This review shows how hard the sentiment analysis is of a task, where you may face ambiguous subjective wording sometimes. This is a challenge that requires much more work on things like context and domain knowledge.

Reflection

The process of this project can be summarized as following:

i. A problem was defined and the relevent pupblic dataset was noted
ii. The dataset was downloaded, explored, cleaned and processed
iii. The data was split into training and testing datasets
iv. The benchmark model was developed and the results were found
v. The CNN model was developed, trained and validated on the training data
vi. The model was refined and results were reported
vii. The model was tested on external small real-world datasets with no preprocessing.
viii. The final model and the benchmark were compared

Point 7 were the most interesting part as I saw a real-word application to the model I worked on! I was

afraid that the training data was relatively old (2013) and the language is always evolving and some words may not be used as before. But the model worked well enough and did not require any normalization or text preprocessing, only feed the raw data to the tekonizer and then to the model, simple as that.

The difficult part was make the conv layers work in parallel, as the common way is a sequential flow, it was challenging and required more work but I'm happy with the result.

All in all this was an interesting project, I was looking to broaden my knowledge and skills developing this project and to learn and explore new territories for me. I know some of the work could be done in easier way for some parts (like loading the dataset) but I wanted to experience with every aspect of the project to learn more, I'm happy with what I've learned and see I have a long way still, the results of the model were very satisfactory and I think I did a good job.

Improvement

As I mentioned before, SA is quite complicated and requires more knowledge of the domain and context that CNN can't handle. RNN architectures proved to do just that along with LSTM which is also a good improvement and can be linked with CNN.

If my solution was a new benchmark then defiantly there are much better solutions out there, but all these solutions are state of the art and tend to be very complex. What I like about this model is that it can be easily trained and work with languages that do not have large resources like Arabic for example.

References

B. Liu, Sentiment analysis and opinion mining . San Rafael, CA: Morgan and Claypool Publishers. (2012)

B. Pang1,L. Lee Opinion mining and sentiment analysis.(2008)

A. Maas, R. Daly, P. Pham, D. Huang, A. Ng,and C .Potts. Learning Word Vectors for Sentiment Analysis.(2011). In The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

Y. Kim. Convolutional Neural Networks for Sentence Classification (2014)

Y. Zhang. A Sensitivity Analysis of (and Practitioners' Guide to) ConvolutionalNeural Networks for Sentence Classification (2014)

Keras.io. 1D CNN for text classification https://keras.io/examples/imdb_cnn/

J. Brownlee. How to Calculate McNemar's Test to Compare Two Machine Learning Classifiers https://machinelearningmastery.com/mcnemars-test-for-machine-learning/ (7/2018)

J. Brownlee.How to Develop an Multichannel CNN Model for Text Classification https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis/ (1/2018)

J. Pennington, R. Socher, C. D. ManningGloVe. Global Vectors for Word Representation. (2014) Computer Science Department, Stanford University, Stanford, CA 94305