Sentiment Analysis Online Movie Reviews

Al Applications

© Kenneth Foo. All rights reserved

Al Applications Student | NLP Graded Assignment

Honor Pledge for Graded Assignments

"I affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own."

Signature:





Summary

Data Pre-Processing

Model Creation

Prediction Results

 Difficulties Encountered & Learning Points

Process Text

Perform Tokenization

from nltk.tokenize import sent_tokenize, word_tokenize

• Remove Punctuation

stop = nltk.corpus.stopwords.words('english')

Lemmatize

from nltk.stem import WordNetLemmatizer

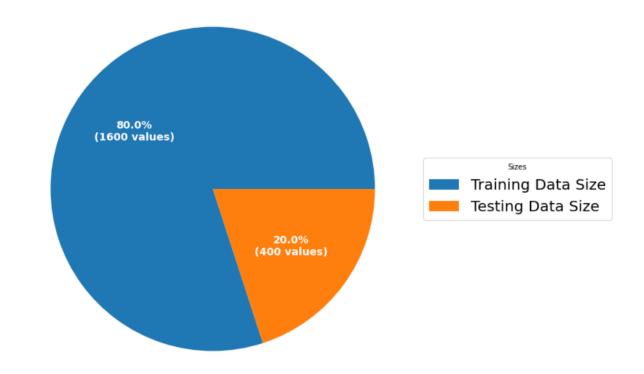
Train Test Split

• 80% Training

• 20% Testing

from sklearn.model_selection import train_test_split

Training & Testing Size Comparison



- Vectorization
- TF-IDF

$$w_{i,j} = t f_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

- TfidVectorizer
- Build TF-IDF function

Term Frequency: TF of a term or word is the number of times the term appears in a document compared to the total number of words in the document.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

Inverse Document Frequency: IDF of a term reflects the proportion of documents in the corpus that contain the term. Words unique to a small percentage of documents (e.g., technical jargon terms) receive higher importance values than words common across all documents (e.g., a, the, and).

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

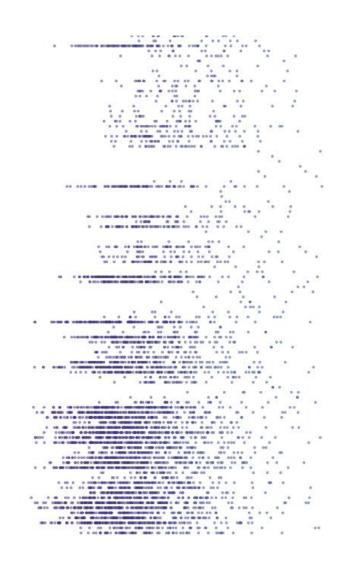
The TF-IDF of a term is calculated by multiplying TF and IDF scores.

$$TF$$
- $IDF = TF * IDF$

- Visualization using Altair
- Heatmap

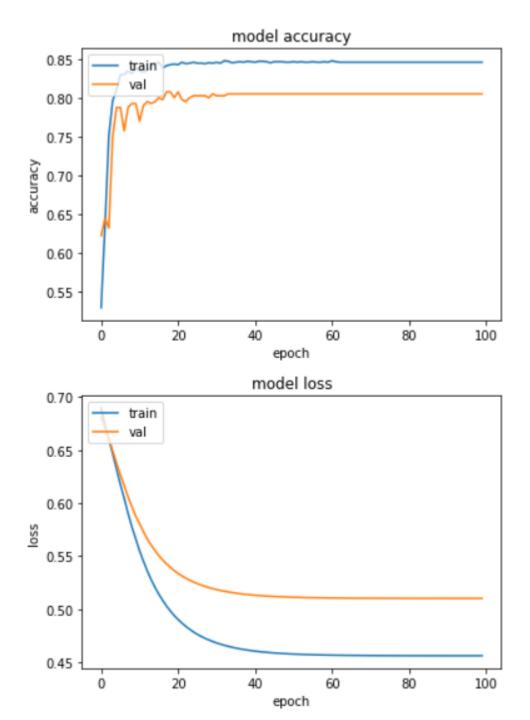
| 0 - | detail | emotional | alone | told | complete | e njo y | art | oscar | forced | taken |
|------|--------------|--------------|--------------|--------------|----------------|-----------------------|----------|--------------|------------|-----------|
| 1- | turned | actress | screenwriter | white | plan | form | quickly | the | single | robert |
| 2- | hilarious | twist | girlfriend | oh | alone | leave | saw | whether | chance | age |
| 3- | detail | near | hold | premise | in | working | release | guess | eventually | forced |
| 4- | important | twist | earth | serious | oh | deep | recent | million | giving | future |
| 5- | hilarious | surprisingly | deep | expect | o sc ar | eventually | dream | peter | whether | success |
| 6- | motion | expect | told | appears | leave | chance | white | particularly | beautiful | dark |
| 7- | whether | tale | mr | worse | quality | version | number | fine | beginning | light |
| 8- | hell | project | appears | team | daughter | sometimes | taking | obviously | truly | thriller |
| 9- | important | motion | easily | easy | recent | tale | across | CO | daughter | quality |
| 10 – | premise | among | oh | alone | told | age | dark | light | drama | WO |
| 11 - | emotional | read | actress | CO | daughter | flick | form | five | possible | number |
| 12- | earth | 10 | oh | premise | taken | bo <mark>fi</mark> ng | type | bring | talk | single |
| 13- | hilarious | emotional | turned | among | whether | across | white | particularly | involving | the |
| 14- | deep | in | complete | screenwriter | enjoy | middle | taken | yes | type | late |
| 15- | killer | in | involved | taking | particularly | none | type | quickly | robert | five |
| 16- | surprisingly | premise | project | release | in | future | stay | forced | worse | break |
| 17- | surprisingly | oh | read | among | score | in | mr | co | white | plan |
| 18- | in | middle | tale | present | strong | break | room | upon | coming | son |
| 19- | detail | 10 | alone | complete | town | return | surprise | heart | school | son |
| 20 – | hilarious | 10 | premise | working | stay | appears | peter | saw | whether | extremely |

- TF_IDF Concentration Clusters
- Scatter Plots
- Majority of terms appeared in the range of 0.1 to 0.2, rarely above 0.3. Total range is 0 to 1.
- This shows that there are a lot of repetitive terms that appeared in multiple documents.



Model Creation

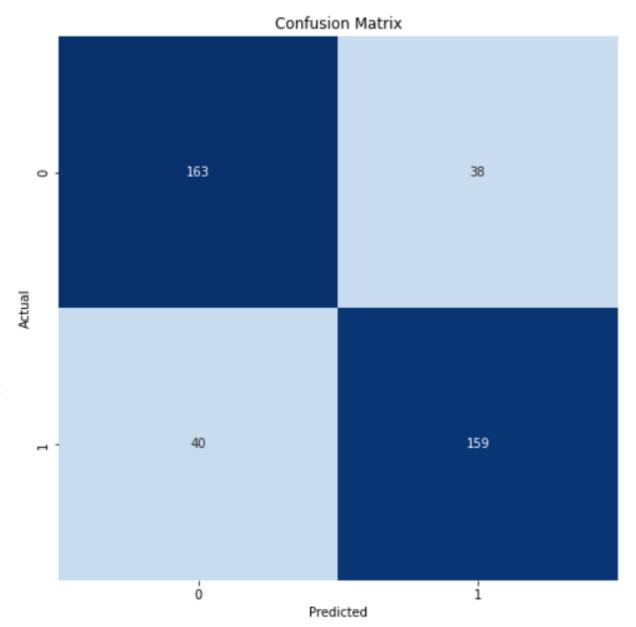
- Neural Network
- 2 Dense Layers
- Optimizers = Adam Optimizer
- Loss = binary_crossentropy
- Metrics = accuracy
- Callbacks = [
 EarlyStopping,
 ModelCheckpoint,
 LearningRateScheduler
]



- Evaluation acc = 80%
- Precision label 0 = 0.80
- Precision label 1 = 0.81

Classification Report:

precision recall f1-score support 0.80 0.81 0.81 201 0.81 0.80 0.80 199 0.81 400 accuracy 0.81 0.80 0.80 400 macro avg weighted avg 0.81 0.81 0.80 400



Test for Unseen Text

Positive text

"This is going to go down as one of 2022's most entertaining motion pictures"

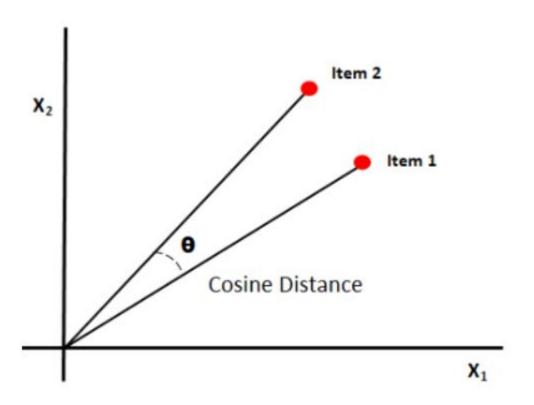
Predicted Sentiment: POSITIVE

Negative text

"Just when you think you've seen the worst movie ever made, along comes this pile of toxic waste."

100%| 100/100 [00:01<00:00, 97.20it/s]
Predicted Sentiment: NEGATIVE

Apply Cosine Similarity to Find Similar Text



- Compute cosine similarity for each word in unseen texts against each word in training dataset.
- Sort values from descending order and display the relevant word for the top cosine similarity values.

Apply Cosine Similarity to Find Similar Text

from sklearn.metrics.pairwise import cosine_similarity

Similar words for positive text:

| | Similar words | Cosine Similarity |
|---|---------------|---------------------|
| 0 | several | 0.26121273222675995 |
| 1 | nice | 0.17099611612487206 |
| 2 | beginning | 0.16436803502996014 |
| 3 | mean | 0.16434677982006513 |
| 4 | show | 0.16399003084068903 |
| 5 | taken | 0.16192629754923127 |

Similar words for negative text:

| | Similar words | Cosine Similarity |
|---|---------------|---------------------|
| 0 | come | 0.2535885476337849 |
| 1 | turn | 0.22630730810113545 |
| 2 | screenwriter | 0.2120569879127968 |
| 3 | think | 0.2087952832181084 |
| 4 | violence | 0.19622243100038428 |
| 5 | involving | 0.18709544095036856 |

Difficulties Encountered & Learning Points

Dr Chang | PhD, Computational Biology, Bioinformatics





"Friend of mine, Dr Chang, gave advices on how to build the tfidfvectorizer. He corrected my mistake specifically on how I used the fit_transform method. I had to be careful when building the vectorizer. Should only **fit transform** on the **X train** then use transform on **X_test**. The reason for not using fit_transform on **X_test** is because *fit transform* chooses the best words you provide. So even though you may have equal amount of vocabs in both sets, using *fit transform* may result in the mis-alignment in the arrays(because the vocabs are different). It would render your validation set useless because your model is validating against nonsense. He also used an analogy of describing the TF-IDF function like a mother function, it gave birth to the vectorizer, then you can use it subsequently."

End of Presentation