Film Review Summary Project

Introduction

The objective of this project is to analyze film reviews and provide predictions on whether the film is good or not. The project primarily aims to gain a better understanding of natural language processing techniques and RNN and Transformer models.

Reporting

The report is divided into several sections:

- Data Collection and Processing
- Model Architecture and Training
- Experimental Results
- Conclusion
- Future Directions

Author

Tran Le Minh

Date

27 May 2023

Table

Film Summary Project

Python Library Choices

Python Library:

- Keras/Tensorflow: I aimed for code simplicity when building the model, and thus chose Keras over Pytorch.
- Matplotlib, Numpy, Pandas, Scipy: These are libraries that are commonly used in data preprocessing or analysis.
- Gensim: This library includes the Word2vec model, primarily used for word or text vectorization.
- NLTK: This is a crucial library as it provides essential tools for basic NLP.
- Transformer: This library offers advanced models for various NLP tasks.

Dataset and Preprocessing Data

Given that the primary focus of this project is to generate summaries of film reviews, categorizing them as either positive or negative using DNN and RNN-based models, or on a scale of 1-5 utilizing a Transformer-based model, the datasets employed for training and evaluation are exclusively limited to the IMDb Film Review and Amazon's Movie&TV Review dataset.

IMDb Film Review:

The IMDb film review dataset consists of 50,000 review texts, evenly divided between positive and negative sentiments. The dataset is in CSV format and follows a simple format, as follows:

- **Review:** Text review that was crawed from IMDb web
- **Sentiment:** is either Positive or Negative

When it comes to NLP tasks, there are several necessary steps to ensure that the text is thoroughly cleaned to avoid any noisy instances. Regarding this specific dataset, the initial stage involves handling special tags extracted from web crawling, such as "
" and "<html>". Additionally, punctuation marks and numerical characters are eliminated, directing focus solely towards the textual content. Subsequently, the text is transformed to lowercase to mitigate excessive word variations, while non-essential stop words, lacking significant semantic influence, are removed. However, upon performing data cleaning and analyzing the distribution of frequent words, it became evident that certain terms like "film," "movie," "really," and "one" appeared excessively in both positive and negative instances. Consequently, these terms were also discarded. Lastly, words with fewer than two characters are filtered out due to their limited impact on sentence semantics. Following this, words undergo lemmatization to revert them to their base forms. As an illustrative example, "went" → "go", "running" → "run" and "happiest" → "happy."

Due to limited computer resources, loading all 460,000 images at once is not feasible. To overcome this challenge, I utilized the k-fold cross-validation technique to train the model. This approach was appropriate given that the dataset is comprehensive and each partitioned subset is similar to one another. This consistency ensured that the model was stable across all the folds. That how load_image_data function work

Amazon's Movie&TV Review:

The Amazon product dataset is a vast collection of data containing reviews and ratings for various products sourced from the Amazon e-commerce platform. This dataset encompasses a wide range of categories, spanning from books to clothing and Movie&TV. However, for the purpose of this project, I specifically focus on a subset known as Amazon's Movie&TV Review 5-score – contains over 1 000 000 reviews. This subset is curated to include reviews from users who have contributed over five reviews, ensuring a higher level of reliability and relevance. Each review's information is encapsulated in a .json file, adhering to the following format:


```
{
        "reviewerID": "A2SUAM1J3GNN3B",
        "asin": "0000013714",
        "reviewerName": "J. McDonald",
        "helpful": [2, 3],
        "reviewText": "I bought this for my husband who plays the piano. He is having a
        wonderful time playing these old hymns. The music is at times hard to read because
        we think the book was published for singing from more than playing from. Great
        purchase though!",
        "overall": 5.0,
        "summary": "Heavenly Highway Hymns",
        "unixReviewTime": 1252800000,
        "reviewTime": "09 13, 2009"
}
```


Like the IMDb film review dataset, I only require reviewText and rating information from our dataset. Hence, I discard any extraneous data when loading the file. After loading the data, I noticed that the rating distribution was uneven, particularly at rating 4 and 5. To address this issue, I opted to retain all data with ratings of 1 and 2. Furthermore, I selectively kept reviews exceeding a word count of 200 but with ratings above 3. Upon plotting the data, it became evident that there was an ample number of reviews, consisting of more than 200 words, associated with ratings of 4 and 5. However, the number of reviews for rating 3 fell short of the desired quantity. In order to rectify this disparity, I made the decision to incorporate an additional 35,000 reviews, each containing more than 100 words and carrying a rating of 3. This meticulous approach ensured that the dataset was evenly distributed across different rating categories. Also the dataset was clean through several step to avoid noise.

Conclusion:

As the project progressed, I observed a positive correlation between the length of reviews and the credibility of their associated ratings. Furthermore, I realized that the dataset could be effectively cleaned by excluding ratings with fewer than 20 words. By doing so, the dataset becomes more reliable as it focuses on reviews that provide more substantial content and insights. This approach ensures that the analysis and models built upon the dataset are based on more meaningful and informative reviews, enhancing the overall quality of the project's outcomes.

Figure 1: Word Cloud from IMDb dataset

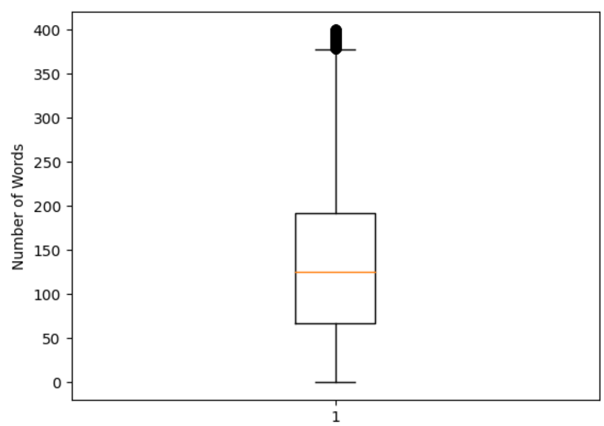

Figure 2: Review Length Distribution

Approaches to Model Building

This section covers three models that I used for NLP classification: Dense Neural Network (DNN), Recurrent Neural Network (RNN) and Transformer Neural Network (RNN). But first let talk about how data processing work.

Data Processing:

As we all know, computers can only understand and process numerical data. Therefore, to build an NLP model, we need to convert text-based language into numerical representations, which are known as word embeddings. Word embeddings capture the semantic and contextual information of words, allowing NLP models to effectively understand and manipulate textual data. In my project there are 3 method of word embedding according to 3 models, but in summary there only 2 main methods was use. One approach involves leveraging the word embeddings generated by the Word2Vec model for processing purposes, while the other approach entails employing word embeddings that adhere to the precise definition as prescribed by the BERT model.

1. Word2vec:

Word2Vec is a simple yet widely renowned model designed to generate word embedding representations. It operates on the principle that words appearing in similar contexts tend to have similar meanings. There are two primary architectures in Word2Vec: Continuous Bag of Words (CBOW) and Skip-gram. In my code both DNN and RNN model use CBOW architecture in Word2Vec to convert word to text before training (since the dataset is large enough).

In the CBOW architecture, the model predicts the current word based on the context words surrounding it. The context words are used as input features, and the central word is predicted as the output. This approach is beneficial when the focus is on smaller datasets and frequent words. The Skip-gram architecture functions in the opposite manner.

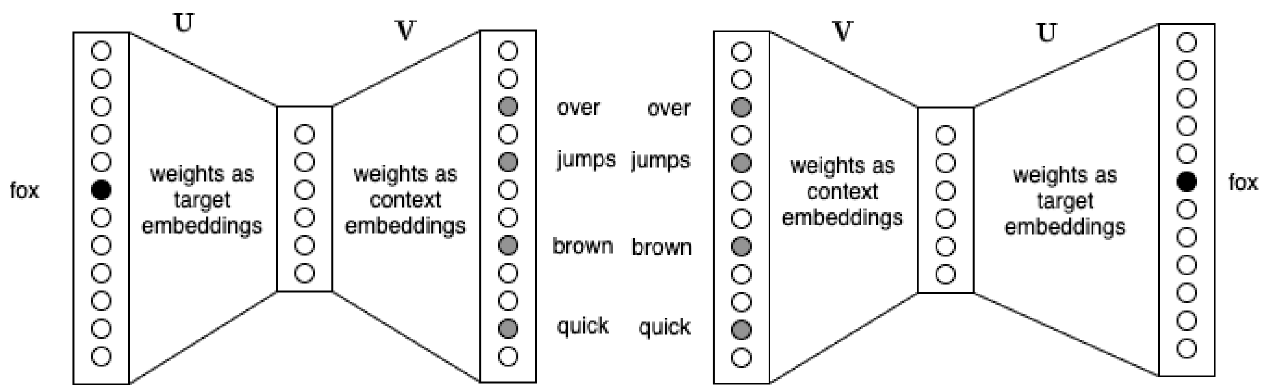

Figure 3: Skip-Gram and CBOW Model

The only distinct difference between DNN and RNN when applying this method is that DNN computes the sum of the entire text sequence and takes the average, whereas RNN treats each word as an individual embedding vector. This distinction arises because DNN models, being simpler in nature, may not capture the nuanced features present in longer text sequences. Therefore, by summing the embedding vectors and calculating the average, DNN can still effectively reflect the sentiment of the review. In contrast, RNN, particularly with LSTM layers, is more complex, capable of retaining long-term information, and recognizing intricate patterns in the text.

2. BERT Embedding:

The BERT model embedding method operates by leveraging a word “bank” tokenizer to processing input data. BERT has an advantage over models like Word2Vec because while each word has a fixed representation under Word2Vec regardless of the context within which the word appears, BERT produces word representations that are dynamically informed by the words around them. There for more information was capture by this method.

Additionally, for each sequence, there is a vector attention mask generated. This attention mask serves as a binary tensor that highlights the positions of padded indices within the sequence. By incorporating the attention mask, the model is able to effectively ignore these padded tokens during training and inference. This is particularly important when dealing with named entities or unknown words that fall outside the scope of the central word "bank." By marking these tokens in the attention mask, the model can prioritize the meaningful information within the sequence and disregard irrelevant elements.

Dense Neural Network (DNN):

As mentioned earlier, the approach of taking the average sum of word embeddings to obtain sequence embeddings can still effectively reflect the sentiment of reviews that exhibit clear positive or negative tones. Therefore, DNN models can still perform well in capturing the relevant features within sequences. By considering the collective information of the word embeddings, the DNN model can understand the overall sentiment expressed in the review and make accurate predictions. This approach proves to be a reliable method for fast and simple text classification.

Recurrent Neural Network (RNN):

RNN is a type of neural network commonly used for processing sequential data such as time series or natural language. RNNs incorporate one or more gates to process information and retain previous data rather than completely forgetting it. There are various types of RNNs, including LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) and other types such as SimpleRNN, Bidirectional RNN, etc. For my project, I have chosen to utilize LSTM for its ability to capture long-term dependencies and handle sequential data effectively.

An LSTM cell consists of three components, also known as gates, which are:

- Forget Gate: This gate determines what information to discard from the previous cell state. It takes the previous hidden state and the current input as inputs and outputs a number between 0 and 1 for each element in the cell state, representing the importance of each element.

- Input Gate: The input gate decides which values to update in the cell state. It takes the previous hidden state and the current input as inputs and outputs a number between 0 and 1 for each element in the cell state, indicating the amount of update for each element.
- Output Gate: The output gate controls what information from the cell state will be used as the output. It takes the previous hidden state and the current input as inputs, along with the updated cell state, and outputs a filtered version of the cell state.

These gates in the LSTM cell allow it to selectively retain or forget information, update the cell state, and control the flow of information through the recurrent connections. This helps LSTM cells effectively handle long-term dependencies and capture important patterns in sequential data.

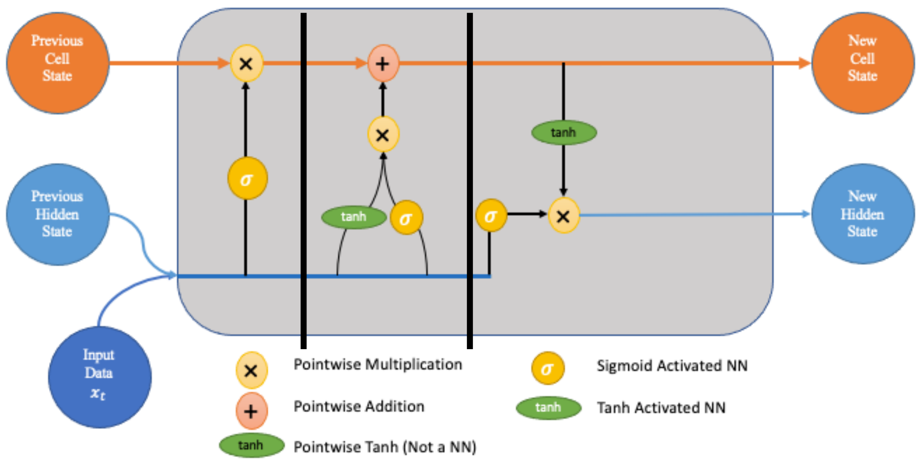

Figure 4: LSTM Cell Diagram

As previously mentioned, the intrinsic nature of LSTM eliminates the need for sequence embedding, as it can lead to potential loss of information. For instance, in the sentence "This film is not good," even if we retain the word "not" during text preprocessing, a DNN model might still mistakenly classify it as a positive review. In contrast, an LSTM model is capable of discerning that the combination of "not" and "good" signifies a negative sentiment rather than a positive one. It is worth noting that LSTM models can effectively handle multi-rating scenarios akin to Transformer-based models. However, due to limited computational resources, I refrained from utilizing the Amazon dataset for training purposes. This dataset contains considerable noise, necessitating a substantial amount of data to achieve satisfactory performance. Additionally, considering that each word would be embedded into a 75-dimensional vector, my computer would not be able to handle it efficiently.

I also added a masking layer to the model to allow it to distinguish and disregard the padded portions of the data during training. This was achieved using the `pad_sequence` function to obtain fixed-length data by padding the sequences.

Transformer Neural Network (BERT):

While recurrent neural networks (RNNs) are undeniably powerful and effective, they do possess certain limitations in terms of capturing long-term features and efficiently handling parallel processing. Recognizing these constraints, the Transformer architecture was developed as a groundbreaking solution to overcome these limitations. BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art language model that has revolutionized natural language processing tasks. It is a pre-trained model developed by Google, known for its ability to understand the context and meaning of words in a sentence.