Sentiment Analysis with Recurrent Neural Network - LSTM

Aastha Patel, Rahil Bhensdadia, Priyavardhan Panchal, Yash Amethiya

August 2020

Goal: To perform sentiment analysis using LSTM.

1 Introduction

1.1 What is Sentimental analysis?

Sentimental analysis is a ML technique to understand the polarity (Positive, Negative, Neutral) within the text.
Apart from polarity, the sentiment models also focus on feelings and emotions (angry, happy, sad) and also on intentions (interested or not interested).

Types of sentiment analysis:

1. Fine-grained analysis
   When polarity precision matters, we can divide polarity as follows:
   - Very Positive
   - Positive
   - Negative
   - Very Negative
2. Emotion Detection
   It aims at detecting emotions like anger, frustration, sad, happy etc. Such systems use lexicons or complex ML algorithms.
3. Aspect- Based Sentiment Analysis
   Understanding particularly which aspects or features people are mentioning in a positive, negative or neutral manner.
4. Multilingual Sentiment Analysis
   Performing sentiment analysis in different languages.

1.2 Why perform Sentiment Analysis?

It's estimated that 80% of the world's data is unstructured. Enormous amount of text data is produced every day, but it's hard to sort, analyze and understand them. All this is very time consuming hence we do sentiment analysis on this unstructured data and make some sense out of it.

Sorting data through thousands of tweets, emails, chats manually is very time consuming and there is too much data to work upon. So sentiment analysis helps to process huge amount of data in an efficient way as well as makes it cost effective.

It's estimated that people only agree around 60-65% of the time when determining the sentiment of a particular text. Tagging text by sentiment is highly subjective, influenced by personal experiences, thoughts, and beliefs. By using a centralized sentiment analysis system, companies can apply the same criteria to all their data, helping them to improve accuracy and gain better insights.

We use RNN for sentiment analysis instead of CNN because a RNN is trained to recognize patterns across time, while a CNN learns to recognize patterns across space.

1.3 What is Recurrent Neural Network (RNN)?

RNN is capable to do tasks such as connected handwriting recognition or speech recognition due to its internal memory to process sequences of inputs. RNN remembers everything. In RNN, all the inputs are related to each other, by taking the output of each neuron and feeding it back to the model as an input with new pieces. This helps RNN to remember the previously learnt things with the new information and stores it into hidden layers which gives us delay of time to recall the information as input to the next neuron.

This type of flow of information of RNN is shown below:

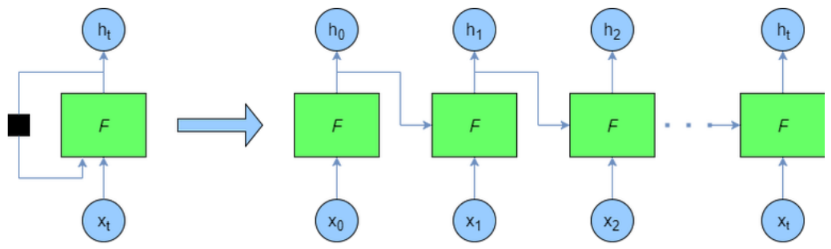

Figure 1: The flow of information in RNN

This is unrolled version of RNN. First, it takes X0 as input from sequence of inputs and then its output is stored into hidden layer h0 Now, h0 is given as an input with new piece of sequence X1 in the next step. Similarly, with h1 and X2 in the next step and so on. This way, it keeps remembering the context while training.

1.3.1 Types of RNN

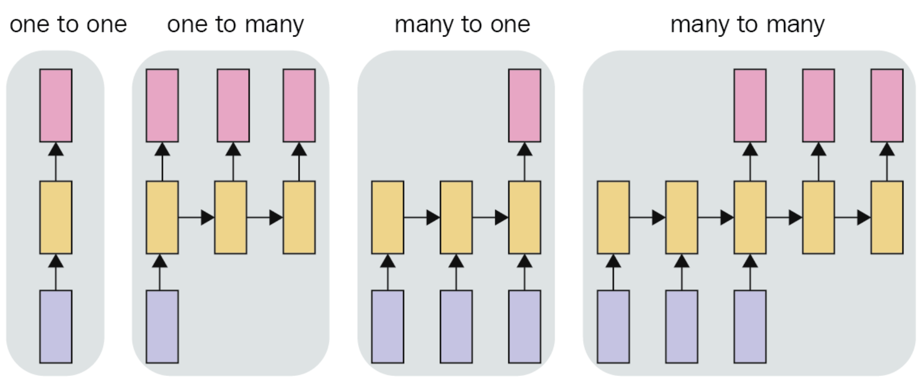

Figure 2: Types of RNN

1. One-to-One:
   RNN takes one fixed size input and gives one fixed size output which is independent of previous output/information.
   For example, Image Classification

2. One-to-Many:
   RNN takes one single input and generates sequence of data as output.
   For example, image captioning i.e., Non-label image is given as an input and it generates description of it.

3. Many-to-One:
   RNN takes sequence of words as input and generates one output.
   For example, reviewing comments about the movie whether that comment is positive or negative.

4. Many-to-Many:
   RNN takes sequence of words as input and generates sequence of words as output.
   For example, translating sequence of words from one language into another language.

2 Pre-processing Datasets

Pre-processing data means when we want to transform our data or to pre-process our data and then use the data in Machine Learning.
Here we are using imdb dataset which is already pre-processed in which our texts is changed into tokens by giving unique index or id to each unique word in our training data.

Thereafter, we are padding our data to make each data of same length for which we are using pad_sequence method.

- **pad_sequence:**
  Transform a list of sequence into 2D numpy array. The maxlen if provided or the length of the longest sequence otherwise.
  Sequences that are shorter than maxlen are padded with value provided in value attribute. Sequence that are longer than maxlen are truncated.
  Position where padding or truncating happens is determined by padding or truncating, respectively.

3 Selecting the Layers and Training the Model

- **Embedding:**
  This are the methods for learning vector representations of categorical data.
  - input_dim: Specifies the size of the vocabulary. i.e. maximum integer index +1
  - output_dim: Specifies the dimension of the embedding.
  - input_length: Specifies the length of the input sequences when it is constant. This argument is required when connecting Flatten then Dense layers upstream (without it the shape of the dense outputs cannot be computed).
- **LSTM:**
  LSTM networks are a type of RNN capable of learning order dependence in sequence prediction problems.
  This is a behavior required in complex problem domains like machine translation, speech recognition, and many more. They are a complex area of deep learning.
  - units: Specifies the dimensions of the output space.
- **Dense:**
  Dense layer is regularly deeply connected neural network layer.
  - units: Specifies the dimensionality of the output space.
  - activation: Specifies the activation function to be used. If not specified, then no activation is applied.

Model compilation

- **Compile**
  Configures the model for training.
  - optimizer: Specifies the optimizer to be used. We have used 'rmsprop' optimizer. This optimizer restricts the oscillations in vertical direction and hence improves the learning rate.
  - loss: Specifies the loss function to be used. We have used 'binary_crossentropy' loss function.
  - metrics: List of metrics to be evaluated by the model during training and testing. Each of this can be a string, function or an instance of tf.keras.metrics.Metric .

The model composed in our application has following specifications:

- Batch size: The batch size set for training the model is 20.
- Number of Epoch: 15 are used for training.
- Python packages used:

Packages
tensorflow
tensorflow.keras
numpy

4 Results

With training for 10000 reviews our model has obtained following results:

- Training statistics: For final epoch loss=0.2236 ; accuracy=.9161 ; val_loss=0.3392 ; val_acc=0.8665

5 Conclusion

We have implemented type 3 aspect based sentiment analysis for this dataset using LSTM.