International Institute of Information technology – Hyderabad

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

CSE471 - Statistical Methods in A.I

Course Project report

Project ID: 11 Predict movie tagline from text synopsis

Project Guide: Satyam Mittal (satyam.mittal@students.iiit.ac.in)

Project Members (PGSSP Candidates):

1) Jyothsna – 2018900063 (jyothsna.munipalle@students.iiit.ac.in)
2) Surekha – 2018900085 (surekha.medapati@students.iiit.ac.in)
3) Madan – 2018900075 (madan.nandiwada@students.iiit.ac.in)
4) Karthikeyan – 2018900074 (Karthikeyan.arumugam@students.iiit.ac.in)

Table of Contents

Abstract

Predicting movie taglines from the context of movie plot synopsis. Our model is a bidirectional encoder-decoder RNN. It was optimized for best performance with tests utilizing Long short-term memory (LSTM) cells as building blocks. Attention model for predicting each word of the tagline conditioned on the movie plot. While the model is structurally simple, it can easily be trained end to end and scales to large amount of training data. We apply our model to the IMDB movies dataset. We evaluate the predicted tagline using standard metrics like ROGUE score, showing that model can encode texts in a way that preserve its syntactic and semantic coherence.

Introduction

Context taglines for movies are beneficial for both movie audience and creators. The tag generated gives an idea on the movie. Based on the tag of the movie people can get an idea on the type of movie and what can be expected from the movie. The catchy tag for the movie is always USP for the movie. Not every movie has a tag mentioned to it. It becomes an easy task for movies related website or portal when there is a tag mentioned for every movie. A tag for the movie represents the experience which the movie goes can expect like genre, impact etc. Due to these factors the tag for the movie becomes an important point for movie recommendation. Any movie which does not have a tag will have impact on movies becoming popular. Machine Translated Taglines are unbiased.

Challenges

Data

- Missing values
  The Data set from Kaggle contains over 45000 movie details. Unfortunately, there are lot of data points where either synopsis or tagline. This renders those data points use less as we couldn't use them to train the model.
- Non UTF8 Text contents
  A lot of international movie details contains non UTF 8 Characters. These unknown chars may include trademark /copyright symbols, Latin chars. Synopsis and tagline Text which are typed in a language which not in English.
- Non-English Text Contents

  There are many data points in the Kaggle data set which contains non-English characters. Such as Movie synopsis or tagline written in regional Language fonts which are a valid English text content. *Example: Tagline given in Telugu or Malayalam*.

Unknown words in Data set

There are many out of vocabulary words in out data set. Such unknown words tends to not contributing to our training weights There is always possibility of getting a Unknown word which is not part of word embedding matrix which we created. i.e. Out of Vocabulary Words. Such out of vocabulary will be assigned as unknown word in embedding it will not help model to train better.

Examples: Name of Characters in the movie like Sivagami, Baahubali etc.

Challenges with Training model

a) Huge Volume of data to be processed

Each Plot Synopsis is a para or list of sentences each has 10-15 words per sentences.

When we convert all words in the given synopsis into word vectors. Processing entire data set with each 3-4 sentences per synopsis will leads to Spike in the Volume of data handled by the system

b) Computation complexity and Long hours of training

As we supply a huge set of movie synopsis and plot lines into model, each one of them these data points have their own complexity with data. Mathematical operations over a large matrices takes toll in system memory. Hence for many parts of the solution we should try it in google colab with GPU enabled option.

c) Risk of Losing training data while executing training for long hours

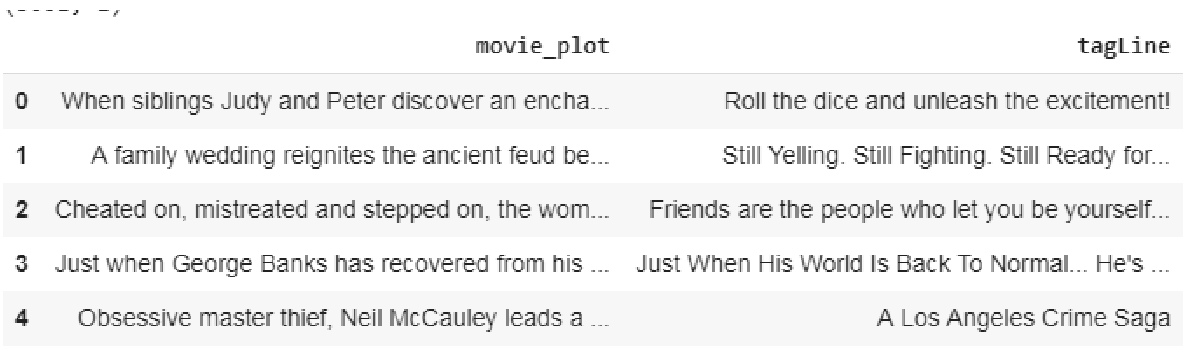
As per are planning to process data as batches , after each batch computation completion there is a challenge with holding weights in memory of the program. Any interruptions and errors may lose entire training data. Hence holding weights in memory during computation is risky.

Dataset

Data source: Kaggle Link https://www.kaggle.com/rounakbanik/the-movies-dataset

We apply our model to the IMDB movies dataset. The dataset have been split to Training, Validation and Testing dataset in the ration of 60:20:20.

Table for Data Description:



	movie_plot	tagLine
0	When siblings Judy and Peter discover an encha...	Roll the dice and unleash the excitement!
1	A family wedding reignites the ancient feud be...	Still Yelling. Still Fighting. Still Ready for...
2	Cheated on, mistreated and stepped on, the wom...	Friends are the people who let you be yourself...
3	Just when George Banks has recovered from his ...	Just When His World Is Back To Normal... He's ...
4	Obsessive master thief, Neil McCauley leads a ...	A Los Angeles Crime Saga

Data Pre-processing:

During the pre-processing step, we cleaned the Movie Dataset. In this step, we expanded the contractions which are usually present in the English language. We also removed the stop words with nltk library, but kept them for the patent titles.

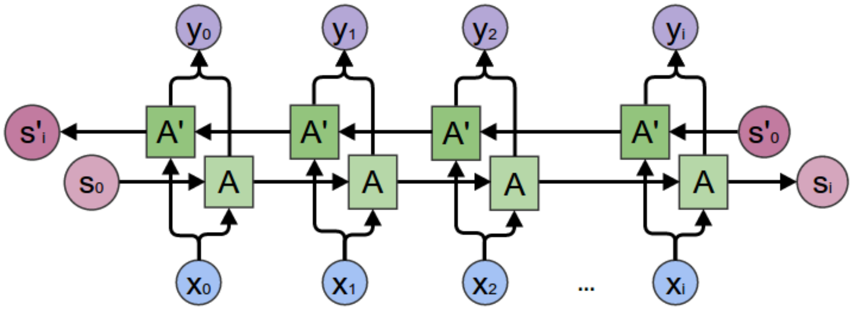
This was done to preserve the syntactic structure of the sentences that the decoder was generating. We then computed the word counts for our vocabulary words. The words not matching with Concept-number batch were replaced by <UNK>tokens.

Below special tokens were used for seq2seq model

- GO - the same as <start> on the picture below - the first token which is fed to the decoder along with the though vector in order to start generating tokens of the answer
- EOS - "end of sentence" - the same as <end> on the picture below - as soon as decoder generates this token we consider the answer to be complete (you can't use usual punctuation marks for this purpose cause their meaning can be different)
- UNK - "unknown token" - is used to replace the rare words that did not fit in your vocabulary. So your sentence my name is guotong1988 will be translated into my name is unk.
- PAD - your GPU (or CPU at worst) processes your training data in batches and all the sequences in your batch should have the same length. If the max length of your sequence is 8, your sentence My name is guotong1988 will be padded from either side to fit this length: My name is guotong1988 pad pad pad pad

Model Overview

Our model is a bidirectional encoder-decoder recurrent neural network (RNN). In bidirectional RNN, the input sequence is fed in normal time order for one network, and in reverse time order for another. This structure allows the networks to have both backward and forward information about the sequence at every time step. This can provide additional context to the network and result in faster and even fuller learning on the text. The outputs of the two networks are usually concatenated at each time step.

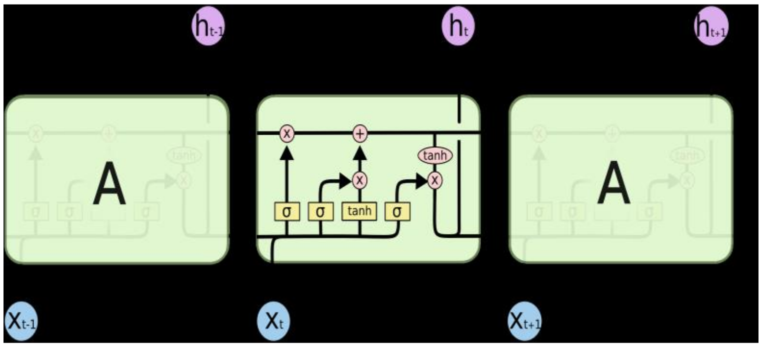


Reference: - Definition and Image taken from https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Encoder:

Our encoder-decoder is based on model proposed by (Bahdanau et al., 2014). We use an LSTM cell for the encoder (This is a special kind of RNN, capable of learning long-term dependencies. Regular RNNs suffer from vanishing gradient problem which prevents them from learning long-term dependencies. LSTMs were designed to combat vanishing gradients through a gating mechanism.

LSTM chain:



Decoder

The decoder implements the mechanism of attention. It decides which parts of the source sentence to pay attention to. Each time the model generates an output word, it searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

Experiments and Results

To train the model, we used word embedding's called ConceptNet Numberbatch (CN) which is an ensemble of many of other popular word embeddings such as Word2Vec and Glove. However, with these embeddings, the model performed sub optimally on various metrics The Concept-Net Numberbatch word embeddings consists of 1.9 million words, we used the training dataset containing 12,242,000 samples for our training and validated our model on the validation dataset containing 4,081 samples after electing our final model we tested our results on the test dataset containing 4,080 samples. We trained all the models for 100 epochs. We were also using early stopping when the error did not improve for continuous iterations.

Performance metrics

We are using Rouge method for performance metric. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially of a set of metrics for evaluating automatic summarization of texts as well as machine translation. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced).

We are currently displaying ROUGE-1 values which refers to the overlap of **1-gram** (each word) between the system and reference summaries.

Best Scores for Train Data

Accuracy Metric	**Precision**	**Recall**	**F-1**
ROUGE-1	0.32107	0.20662	0.2450

Best Scores for Validation Data

Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.2914	0.1682	0.2075

Best Scores for Test Data

Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.2636	0.1550	0.1986

Setting up Environment/Tools Used

- Programming language: Python 3.X
- Dev tools: jupyter notebook notebooks, Colab.research.google.com
- Toolkits Used: TensorFlow, NumPy, Pandas, etc.
- SCM: GitHub and Google Drive(for Large files)
- Solutions Type: Recurring Neural Network

Quick Reference Links:

GIT Hub Repository URL: https://github.com/karumugamio/SMAIProject-Team31

Google Drive URL: http://tiny.cc/team31gdriverepository

Conclusions and Future Work

Our model performed reasonably well and the chosen approach has proven to be effective. We were able to significantly improve the score by additional text pre-processing and tuning hyper parameters .There are still some problems that need to be Addressed in the future work. First is the word embeddings. There are still quite a few <UNK> tokens and repetitions of words because the Dataset is not Grammatically Sound. This can be solved by pointer-Generator model may lead to further improvement. We are getting low Metrics score because automatic metrics such as Rouge itself have limitations like They only assess content selection and do not account for other quality aspects, such as fluency, grammaticality, coherence, etc. To assess content selection, they rely mostly on lexical overlap, although an abstractive summary could express they same content as a reference without any lexical overlap.

Project Distribution

1. Web Scrapping for Dataset : Surekha and Karthik
2. Literature Review of Paper :Madan and Jyothsna
3. Schema of project and Data references: Surekha, Jyothsna
4. Pre-processing of data and Building Model of Project: Surekha, Jyothsna
5. Tuning of the Model :Karthik
6. Metric Evaluation: Surekha
7. Language Translation model: Madan
8. Documentation: All
9. Paper Presentation: Madan and Karthik