Sarcasm Generation with Recurrent Neural Networks

Estefan Apablaza-Arancibia, Adrien Hernandez

Objectives

Main Objective: Create an algorithm able to answer sarcastic output sentences to a neutral input sentence.

- Find an RNN architecture that fits our dataset.
- Create output sentences with good syntax.
- Create output sentences whose meaning correlates with the input sentences.

Why Sarcasm?

We were interested by building and comparing popular machine learning models with deep learning algorithms in the specific field of Natural Language Processing (NLP).

Deep learning algorithms have been increasingly outperforming more traditional models in the past
years in the task of text and language generation.

We were interested by sarcasm as it is often even difficult for human themselves to express it. Therefore, we wanted to see if this idea could be abstract for computers as well.

Our dataset contains ~500,000 sarcastic labeled English comments scrapped from Reddit.

http://nlp.cs.princeton.edu/SARC/

Related Work

Neural Joke Generation by He Ren, Quan Yang , Stanford University

- Use of encoder/decoder RNNs for Joke generation.
- GRU and LSTM outperformed simpler RNNs.
- Comparison of different models of joke generation base on human evaluation.

Knowledge Amalgam: Generating Jokes and Quotes Together, Chippada, Saha, Microsoft R&D

- (LSTM) architecture which is trained with categorical data like jokes and quotes.
- Joke generation were subjectively and objectively evaluated.
- The neural network introduces incongruity in sentences to make them funny.

Our dataset

Original feature used are:

- Comment, Score, Parent.

Added two additional features in order to control the sequence length:

- Comment Word count and Parent comment word count

Comment	Subreddit	Score	Parent comment	Comment Word Count	Parent Comment Word count
But they'll have all those reviews!	ProductTesting	0	The dumb thing is, they are risking their sell...	6	11
wow it is totally unreasonable to assume that ...	politics	2	Clinton campaign accuses FBI of 'blatant doubl...	25	8

Our Process

1. PREPROCESSING

- Text cleaning (lowercase, negation handling, …)
- Tokenized text (input & output); added <SOS> + <EOS>
- Padding (Keras works with constant-sized sequences)
- Load pre-trained Glove on Twitter data 27B 100d corpus. (word to vector)
- Word to index (ID) Mapping
- Embedding layer (vector to matrix)

2. TRAINING

- Method for training use was teacher forcing
- TPU (Google Colab) working with Keras; Tensorflow in backend
- Encoder LSTM: Dimension = 128; Regularization = L2 + dropout
- Decoder LSTM: Dimension = 128; Regularization = L2 + dropout
- Training optimizer = RMSProp; Loss = categorical cross-entropy
- Batch size = 64 ; Epochs = 20; 80% train set 20% validation set

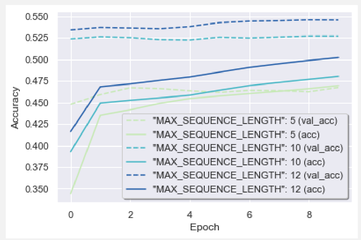
3. LOSS AND ACCURACY

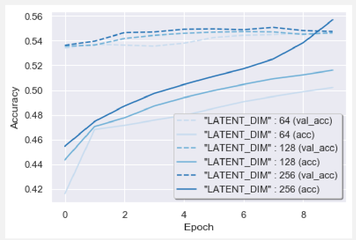
- Comparison between models

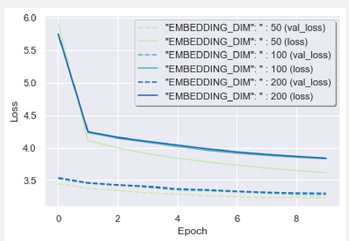
4. GENERATE SARCASTIC SENTENCES

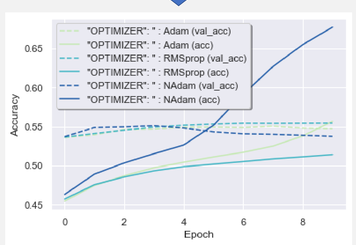
- Comparison of the quality of sarcastic outputs between models using the same input sentences each time

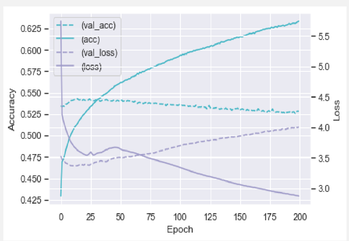
Results









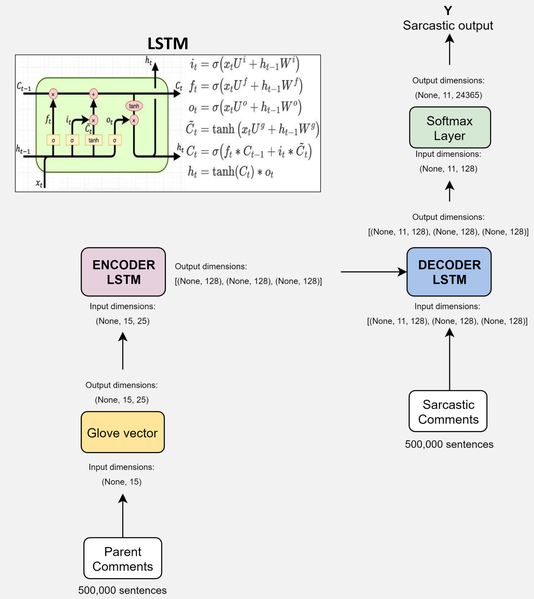


The amount of hyperparameters combination was pretty overwhelming, however, here a list of steps that help to increase the learning of the RNN:

- Maximizing the length of the each sentence to 12. (**Hardware Memory limitation)
- Minimizing the dimension of the latent (Enconder-Decoder size)
- Modifying the optimizer type to RMSprop (Adam vs RMSprop vs Nadam)
- And finally, Maximizing the embedding dimension size to 200 (50 vs 100 vs 200)

Sequence-to-Sequence Architecture

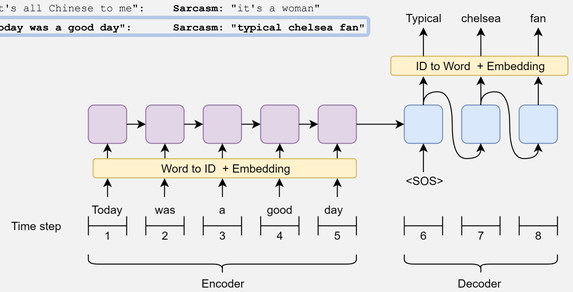
Encoder/Decoder with LSTMs



Sarcasm Generated

The level of sarcasm is something difficult to analyze!

```
Input: "What an awful human being": Sarcasm: "well it's not very american of course"
Input: "Should have gotten a bike": Sarcasm: "come on dude, we all have forgotten"
Input: "It's all Chinese to me":    Sarcasm: "it's a woman"
Input: "Today was a good day":      Sarcasm: "typical chelsea fan"
```



Future work

- Add deepness in the RNN with a bidirectional structure.
- Add attention into the LSTM.
- Try more advanced model such as Transformers and BERT models.
- The greedy search is not optimal for text generation, we could try with a beam search.