A Language Model for Romance Movie Screenplays

Connie Hsueh
chsueh@stanford.edu

1 Introduction

The aim of this project was to create a statistical language model—or a probability distribution for a sequence of words or tokens—based off of a corpus of English-language romance movies. Then, this language model could be queried to generate new screenplays. This follows a similar idea to *Sunspring*[1], a 2016 sci-fi movie that was written using a long short-term memory (LSTM) recurrent neural network. The resulting blockbuster was mostly nonsensical though with short bursts of clarity. As a first guess, romance movies can be more homogeneous than sci-fi movies, as most are set on Earth with a predictable entourage of characters and a predictable plotline. This may make them more amenable to modeling. Additionally, romance movies are not bound to traditional logic, so some nonsensical elements may not seem quite as out of place. For these reasons, I hoped that this approach would be more successful than *Sunspring*, and I also hoped to explore recent advances in language modeling neural networks such as attention and transformers.

Language modeling is a significant statistical tool in Natural Language Processing and assigning probabilities to words or phrases is central to tasks like machine translation, speech recognition, QA, and more[2]. LSTMs feed inputs through a recurrent unit such that it preserves a time-ordering and a memory of previous computations in its hidden states. This has had a natural relevance to modeling sequential data. Improvements on LSTMs such as Embedding from Language Model (ELMo)[3], which comprises a forward and backward language model to capture contextual embeddings, have been instrumental to creating pre-trained models for downstream, supervised-learning tasks. Transformers[4] are a network architecture that do not use recurrent units to preserve sequential information as in LSTMs, but instead include the relative positioning in the vector embedding of each word/token. It also relies on an attention mechanism to allow for longer and variable length dependencies for determining which words in a sequence are most relevant. In transformers, the stacking of encoder-decoder units allows for great parallelizability and as a result have been widely adopted in the last few years for large datasets[5].

2 Dataset and Features

For the data, I took about 60 scripts listed under the Romance genre from `http://IMSDb.com`. Some features are characteristic of scriptwriting: for example, settings are capitalized, as are character names to indicate the speaker in dialogue. Actions and scene notes are unindented while dialogue is indented. In preprocessing, I chose to preserve capitalization but strip the indentation. The resulting dataset is 7 MB. Counting up the word frequencies in the dataset suggests about 35,000 unique words, though this is not adjusted for misspelled words, non-English words, or different forms of the same lexeme (e.g. sit/sat/sitting). At first glance, this was surprising as the scripts are primarily composed of dialogue, and some sources suggest the average adult vocabulary is 20,000–35,000 words[6]. This suggests the lexical diversity in these scripts is high, which may make it more difficult to capture with a small corpus.

To point out just one example to demonstrate that the language distribution here is unlikely to be a uniform sampling from the general English distribution, the world “love" is the 114th most

CS230: Deep Learning, Spring 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

common word in these scripts, appearing 1307 times (and 260 of those as part of the phrase, “I love you."). According to the New General Service List[7] which analyzes from the Cambridge English Corpus,“love" is the 229th most common English word. With a different probability distribution, we can expect other distribution-level statistics such as entropy to differ, and it may be interesting to see how those develop with training.

3 Methods/Experiments

My first attempt was to use a minimal character-based LSTM model as implemented by Eli Bendersky[8, 9]. Finding the results underwhelming, I looked instead to OpenAI’s GPT-2[10], a transformer-based decoder-only model that has demonstrated impressive success in generating text based on prompts. Part of its strength lies in its massive dataset, and though the full GPT-2 model isn’t made available due to ethical concerns, I used the implementation of GPT-2 Small [11] with 124M parameters through Huggingface’s `transformer` package in Pytorch[12].

I attempted two experiments with the GPT-2 architecture. First, I loaded the pre-trained GPT-2 model weights and then fine-tuned the weights using just the corpus of scripts described above. The other test was to train a completely new GPT-2 model using the scripts. Both minimized the causal language modeling loss which optimizes the model to predict the next token given a context. Both used the GPT-2 tokenizer for a full vocabulary size of 50,257 tokens, and both were trained on 8 epochs through the dataset.

I demonstrate a sample of generated text for each model, alongside the GPT-2 with no fine-tuning. They were sampled with top-K value 100, top-p value 0.95, and temperature 1.1. They were prompted on classic dialogue from *Casablanca* (1942), which was not part of the training set.

Prompt

```
ILSA
But what about us?

RICK
We’ll always have Paris. We didn’t have, we, we lost it until you came to
Casablanca. We got it back last night

ILSA
When I said I would never leave you
```

Pre-trained GPT-2 model without fine-tuning

```
, she had tears in her
eyes, I thought of
getting you up there
on the roof of the
building, I could still
lay flowers.

RICK
We’re friends now and
you love us, she said.
I had an easy life, we
had fun together...

But her heart swells
with a sense of loss, a
sense of betrayal and
sadness.

RORY
The only hope left for
you and your dog is not
to go home,
```

Model fine-tuned from GPT-2 model

```
, you. So I’m so mad
at you! Your name’s in
this meeting. I want
to call to you! I’m
the man who’s gonna
kill you. (Shows the
guy the sign.) I wanna
fly! (Shows the sign.)
We’re gonna go, man!
We’re gonna - oh, you
have my body. Man, we
can handle it. A real
man! We’re gonna shoot.
(Shows the sign.) I
wanna live
```

Model from scratch

```
! I! We!!! For to go!
What to! We have! Let!
Let! You! Be to In!
You aIDE I haven! They?
You just! You could Did
to go! You’re You need!
I ever For for! It had
to do so Not. You are
I want to turn to stop
into or What even shot
a army a break for hung
a hand! We go down!
Who to show a fight one
cast"
```

The latter samples don't look very good, and it's underwhelming that any adjustments to the original pre-trained model seem to only make it worse (and substantially so). I next sought to look at how quickly the models degenerated or if there was a way to quantify that. I recovered a model from an earlier checkpoint during the training, when it had only iterated through 1 epoch of the sample set.

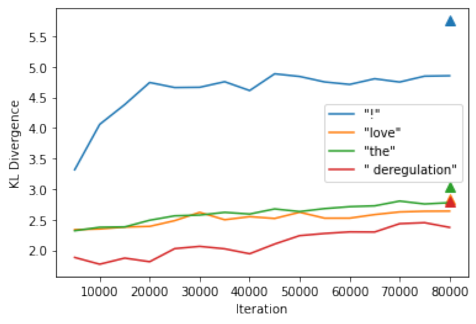
Fine-tuned model after 1 epoch

```
so I said the night before, ‘‘I would leave for you." I really was thinking,
You know, because that’s -so much you’d been through - but, it’s not.  I’m
very happy with you.  We got a deal, he’s the one who gave me that three
and your idea, that’s what we did.  So we talked, we talked, we drank, we
married.  We’ve always talked, we’ve done.  The idea is
```

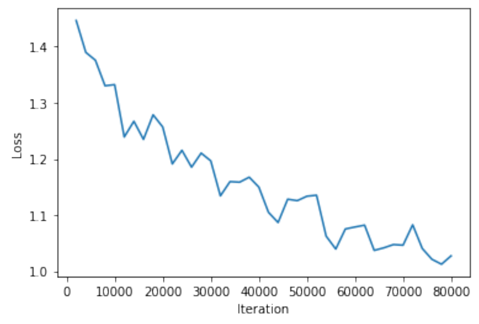
To quantitatively compare the language models, I chose to look at their Kullback-Leibler divergence (or relative entropy), which is a measure of how different two probability distributions are. It can also be informally thought of as an abstract distance between two distributions. For discrete distributions P and Q defined on probability space $\mathcal{X}$, the KL divergence from Q to P is[13]

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)}$$

For some prompt, the language model generates a probability distribution for the next token. I looked at the distributions produced by a few single-token inputs: “!" because it appeared frequently in the generated samples; “love" because it's a word that appears disproportionately frequently in the training set; “the" because it's a common enough word that I guess its language model representation could be robust against the addition of the scripts; and “ deregulation" (including the space), an arbitrarily-chosen token that does not appear in the training set at all. I looked at how these distributions changed with additional training iterations, and calculated their KL divergence relative to the original pre-trained model (i.e., iteration number 0). Note that each epoch is about 10,000 iterations.



(a)



(b)

Figure 1: (a) Kullback-Leibler divergence from the fine-tuned model to the original GPT-2 model as a function of training iteration number. The solid triangles are the KL divergence from the from-scratch model to the original GPT-2 model after 8 epochs of training. (b) Model loss (calculated as causal language modeling cross-entropy loss) of the fine-tuned model as a function of training iteration.

4 Analysis

I make a few qualitative observations from the generated samples shown above. First, only in the pre-trained original model does the generated text reproduce the formatting in the prompt, including newline characters and capitalized names. Even with less than an epoch's worth of additional fine-tuning on the scripts dataset, the model cannot reproduce this. Once the scripts are introduced, it seems like the generated text is choppier, more colloquial, and more exclamatory, which may be the influence of the types of sentences present in the script dataset. Present tense action such as “Shows the guy the sign" sounds like a cue in a screenplay. The from-scratch sample is not able to capture capitalization, grammar, punctuation, or diction very well, but given the small corpus size, this is ultimately unsurprising.

Figure 1 shows the probability distribution from “!" has a rapid divergence in the first 2 epochs before settling into much more modest growth. The magnitude of the divergence is larger than that of the other tokens. This strong influence of exclamation points in the training set is consistent with how frequently they appear in the from-scratch sample. The other three distributions more steadily diverge from their original distributions. “ deregulation" has the lowest magnitude, which is consistent with the thought that its embedding within the model should be less affected by the additional fine-tuning, since the token does not appear in the training set. Some finite amount of change is expected though as model weights shift in response to other training tokens.

Using the original GPT-2 probability distribution as a proxy for the “true" distribution of the English language, we see that the additional training drives the language model further and further from English. The samples shown above were hand-picked for coherence, but certainly there were other generated samples where the model seemed to unlearn some spelling, grammar, and logic as it read more romance.

5 Conclusion

In this work, I used the transformer-based GPT-2 architecture and pre-trained model to investigate a corpus of romance movie scripts. Training an entirely new language model from scratch on this small dataset was not immediately realizable. Starting with a pre-trained model and fine-tuning it based on the dataset sounded more promising, but in practice seemed to push the language model away from some of the pre-learned linguistic requisites, as demonstrated in some of the generated samples. I investigated the change in relative entropy of certain probability distributions related to the model as a function of training length and found a unique trend for “!", which shows up unusually frequently in the generated samples. The immediate (within 1 epoch) degradation of the language model upon fine-tuned training is surprising and piques further interest.

Code

I used Huggingface’s `transformer` package in Pytorch[12] and used Google Colab’s GPU hardware. An IPython notebook is available at `https://github.com/cohsueh/cs230`.

Contributions

This was a solo project.

References

[1] http://www.thereforefilms.com/sunspring.html

[2] https://nlpoverview.com/

[3] https://arxiv.org/abs/1802.05365

[4] https://arxiv.org/abs/1706.03762

[5] http://jalammar.github.io/illustrated-transformer/

[6] https://www.economist.com/johnson/2013/05/29/lexical-facts

[7] http://www.newgeneralservicelist.org/

[8] https://github.com/eliben/deep-learning-samples/blob/master/min-char-rnn/min-char-lstm.py

[9] Karpathy, Andrej. “The Unreasonable Effectiveness of Recurrent Neural Networks." May 21, 2015. https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[10] https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[11] https://github.com/openai/gpt-2

[12] https://github.com/huggingface/transformers

[13] https://en.wikipedia.org/wiki/Kullback-Leibler_divergence