blog_post.md

Deep (Shower) Thought

Teaching AI to have shower thoughts, with on Reddit's r/Showerthoughts

tl;dr: I tried to train a Deep Learning character model to have shower thoughts, using Reddit data. Instead it learned curse words and clickbait-ing.

Background

Deep learning has drastically changed the way machines interact with human languages. From machine translation to textbook writing, Natural Language Processing (NLP) — the branch of ML focused on human language models — has gone from sci-fi to example code.

Though I've had some previous experience with linear NLP models and word level deep learning models, I wanted to learn more about building character level deep learning models. Generally, character level models look at a window of preceding characters, and try to infer the next character. Similar to repeatedly pressing auto-correct's top choice, this process can be repeated to generate a string of AI generated characters.

TODO Image of generated sequence

Utilizing training data from r/Showerthoughts, and starter code from Keras, I built and trained a deep learning model that learned to generate new (and sometimes profound) shower thoughts.

Data

r/Showerthoughts is an online message board, to "share those miniature epiphanies you have" while in the shower. These epiphanies include:

- `Every machine can be utilised as a smoke machine if it is used wrong enough.`
- `It kinda makes sense that the target audience for fidget spinners lost interest in them so quickly`
- `Google should make it so that looking up "Is Santa real?" With safe search on only gives yes answers.`
- `Machine Learning is to Computers what Evolution is to Organisms.`

I scraped all posts for a 100 day period in 2017 utilizing Reddit's PRAW Python API wrapper. Though I was mainly interested in the `title` field, a long list of other fields were available, including:

variable	type
title	string
selftext	string
url	string
ups	int
downs	int
score	int
num_comments	int
over_18	bool

variable	type
spoiler	bool

Once I had the data set, I performed a set of standard data transformations, including:

- Converted the string to a list of characters
- Replacing all illegal characters with a space.
- Lowercase-ing all characters
- Converting text into an `X` array containing a fixed length arrays of characters, and a `y` array, containing the next character.

For example `If my boss made me do as much homework as my kids' teachers make them, I'd tell him to go f...` would become the `X, y` pair: `['i', 'f', ' ', 'm', 'y', ' ', 'b', 'o', 's', 's', ' ', 'm', 'a', 'd', 'e', ' ', 'm', 'e', ' ', 'd', 'o', ' ', 'a', 's', ' ', 'm', 'u', 'c', 'h', ' ', 'h', 'o', 'm', 'e', 'w', 'o', 'r', 'k', ' ', 'a', 's', ' ', 'm', 'y', ' ', 'k', 'i', 'd', 's', ' ', ' ', 't', 'e', 'a', 'c', 'h', 'e', 'r', 's', ' ', 'm', 'a', 'k', 'e', ' ', 't', 'h', 'e', 'm', ' ', ' ', 'i', ' ', 'd', ' ', 't', 'e', 'l', 'l', ' ', 'h', 'i', 'm', ' ', 't', 'o', ' ', 'g', 'o', ' ']`, `f`.

Model

Data in hand, I built a model. Similar to the keras example code, I went with a Recurrent Neural Network (RNN), with Long Short Term Memory (LSTM) blocks. Why this particular architecture choice works is beyond the scope of this post, but Chung et al. covers it pretty well.

In addition to the LSTM architecture, I chose to add a character embedding layer. Heuristically, there didn't seem to be much of a difference between One Hot Encoded inputs and using an embedding layer, but the embedding layers didn't greatly increase training time, and could allow for interesting further work. In particular, it would be interesting to look at embedding clustering and distances for characters, similar to Guo & Berkhahn.

Ultimately, the model looked something like:

```
x = keras.Input(..., name='char_input')
x = Embedding(..., name='char_embedding')(x)
x = LSTM(128, dropout=.2, recurrent_dropout=.2)(x)
x = Dense(..., activation='softmax', name='char_prediction_softmax')(x)

optimizer = RMSprop(lr=.001)

char_model = Model(sequence_input, x)
char_model.compile(optimizer=optimizer, loss='categorical_crossentropy')
```

Unfortunately, this character level model performed quite poorly. This is perhaps due to the variety in post content and writing styles, or the compounding effect of using predicted characters to infer additional characters. In the future, it would be interesting to look at predicting multiple characters at a time, or building a model that predicts words rather than characters.

Results

While this model struggled with the ephiphanies and profoundness of `r/Showerthoughts`, it was able to learn basic spelling, a complex (and unsurprisingly foul) vocabulary, and even basic grammar rules. Though the standard Nietzsche data set produces more intelligible results, this data set provided a more interesting challenge.

Check out the repo if you're interested in the code to create the data set and train the LSTM model. And the next time your in the shower, think about this: We are giving AI a bunch of bad ideas with AI movies.