Narrative Similarity Using Sentiment Trajectories

Ali Arsalan Yaqoob

This is a sample of my writing. It is the project report of my final project in my Modeling and Machine Learning II (DSI 5660) Class at Vanderbilt University.

Abstract

Humans are able to identify similarity in stories by identifying the similarities within different elements of a narrative. For example, the classic story about the rabbit and the hare is the classic example of a story where the underdog wins unexpectedly. Previous work[1] explored identifying similar stories in terms of similarities in the plot events and resemblances between characters and their social relationships. This project aims to explore another dimension - the sentiment trajectory. To achieve this, two models were trained on the TweetEval dataset to be able detect the sentiment of a tweet. The best model (Facebook's RoBERTa[2]) was then used to extract the sentiment trajectory of a given story and then compared to other stories to find stories similar in how their narratives evolved temporally.

Introduction

Our ability to understand the deeper context within stories is linked to our effectiveness in being able to draw similarities between stories. For example, halfway through a movie, and after seeing how the plot evolves, you may be able to tell how it is going to end. This is because the plot of the movie being watched may be similar to the plot in other previously watched movies. This ability to identify similarity in narrative plots allows us to draw from our previous understanding of how similar narratives evolve, and allows us to make predictions about how a movie is going to end.
Being able to design systems that are able to understand the similarity in narratives is an important task for information retrieval, QA tasks and to develop tools that are able to understand the context behind these narratives.

Problem Description

This project works on identifying similarities between narratives by extracting their sentiment trajectories. It builds up on previous work in computational narratology by examining the dimension of similarity in terms of the sentiment trajectories. Previous work in extracting sentiment trajectory involved getting the sentiment associated with words in different regions of the story and then plotting the sentiment trajectory based on the words used in those regions[3]. However, this project details a different methodology where we train sentiment classifiers (LSTM and Pretrained RoBERTa) on the TweetEval[4] dataset to detect sentiments in text. The best classifier is then used to get the prediction on text within a sliding window that moves across the story text. The predictions are used as measurements of sentiments across a story and then a third order polynomial curve is fit to the points.

Dataset(s) Description

Due to a lack of a dataset with narratives and the emotional ratings associated with them, I

---

[1] "Where Have I Heard This Story Before? Identifying Narrative ...." https://www.aclweb.org/anthology/N18-2106. Accessed 7 Dec. 2020.

[2] "RoBERTa: An optimized method for pretraining ... - Facebook AI." https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/. Accessed 15 Dec. 2020.

[3] "The emotional arcs of stories are dominated by six basic shapes." 26 Sep. 2016, https://arxiv.org/abs/1606.07772. Accessed 7 Dec. 2020.

[4] "Evaluation Datasets for Twitter Sentiment Analysis - CEUR ...." http://ceur-ws.org/Vol-1096/paper1.pdf. Accessed 7 Dec. 2020.

decided to use the Tweet Eval dataset. This dataset has benchmark performance for various models which will be later used to evaluate the performance of models I created.

	Positive	Neutral	Negative
Train	17,752	20,569	7068
Test	2,352	5,743	3,811
Val	803	896	301

Table 1: Number of positive, neutral and negative tweets in the training, testing and validation dataset under Tweet Eval

Table 1. Provides some basic statistics about the dataset. The tweets also contained emojis and hashtags, and required further preprocessing steps as described below.

Preprocessing

In order for the model to be trained appropriately on text that would appear in stories, the text from the tweet dataset had to be preprocessed so that it would look like text appearing in stories.

Steps:

1. Removing URLs
2. Removing symbols
3. Removing HTML tags
4. Removing punctuation (only for LSTM, RoBERTa learns from the punctuation)
5. Lemmatizing the words

This was then used to train the models and evaluate their performances.

Model 1: Bidirectional LSTM

In order to explain LSTMs, it is good to understand what Recurrent Neural Networks (RNNs) are. RNNs are commonly used in analyzing sequences. The input of RNN, in the case of our task, would be a sequence of words one at a time. An RNN produces a hidden state for each word which is recurrently fed back in with the next word.

$$h_t = RNN(x_t, h_{t-1})$$

Here x_t is the word at a given time step and $h_{t-1}$is the hidden state produced from the previous word. These go into the RNN to produce the current hidden state. After the final hidden state is computed, we feed it through a linear later f, also known as a fully connected layer, to receive our predicted sentiment.

LSTMs

A Long Short-Term Memory (LSTM) is a type of recurrent neural network. This is better than a standard RNN as it does not suffer from the vanishing gradient problem[5]. LSTMs overcome this problem by having an extra state apart from the hidden state, called a cell. The cell introduces the concept of memory into the model and uses gates within the LSTM architecture to control the flow of information.

$$(h_t, c_t) = LSTM(x_t, h_{t-1}, c_{t-1})$$

The final hidden state is then used once again to make the prediction on the sequence.

In my implementation, a bidirectional LSTM is used. This simply processes the sequence from beginning to end, and from end to beginning, to produce two final hidden states which are then fed into the final layer for making the prediction.

$$\hat{y} = f(h^{\leftarrow}, h^{\rightarrow})$$

Below are the summarized model parameters.

[5] "Overcoming the vanishing gradient problem in plain recurrent ...." 18 Jan. 2018, https://arxiv.org/abs/1801.06105. Accessed 7 Dec. 2020.

Model Type	LSTM
Number of layers	2
Dropout	0.5
Bidirectional	True
Batch Size	32

Table 2: First Model Details

For the training process, I used the ADAM optimizer with a learning rate of 0.001 and betas set to 0.9 and 0.999. For the input of this model, I used the word embeddings using GloVe[6]. In the end, this model produced a test accuracy of 56.2%. This is about 2% short of the best benchmark performance on this dataset using an LSTM.

Model 2: RoBERTa Pretrained

One major breakthrough in language technology (understatement) is the development of transformers. These were introduced in the popular paper called "Attention Is All You Need"[7]. Like the LSTM, transformers also take a sequence of words. However they are different from LSTMs because they use a technique known as Multi-Headed Attention to apply a mechanism known as self-attention.

A transformer can be broken down into an encoder and a decoder. The encoder maps an input into an abstract continuous representation that holds all the learned information from that input. The decoder then takes that continuous information and generates a single output from it, while also being fed the previous output. The encoder uses multi-headed attention to apply self attention. The self attention mechanism creates associations of words with other words in a given sequence. For example, "Ali told Jeanie that she was not doing well in class." Here the model may learn to associate the word "she" with Jeanie and may also learn the dependency structure underlying language as well as the grammatical rules within it.

Without going into too much detail about the inner workings of transformers (which is beyond the scope of this paper), transformers usually outperform LSTMs because they are able to get a much bigger picture of the context within a given text.

In my implementation, I used Facebook's pretrained RoBERTa[8] (A Robustly Optimized Pretraining Approach) model from the Hugging Face[9] library with the fast.ai library. I used transfer learning, which is a method where a model developed for a task (in our case, a language model) is reused as the starting point for a model on a separate task. This methodology saves time while also allowing for better performance.

For my model implementation, I tried two methods. The first method involved adding a final linear layer to the RoBERTa[10] base model from Hugging Face. This produced a test accuracy of only 60.5%. The second method, however, involved using gradually unfreezing multiple layers of the Roberta For Sequence Classification model from Hugging Face. This resulted in a test accuracy of 70.1%. However,

[6] "GloVe: Global Vectors for Word Representation." https://nlp.stanford.edu/projects/glove/. Accessed 7 Dec. 2020.
[7] "Attention Is All You Need." 12 Jun. 2017, https://arxiv.org/abs/1706.03762. Accessed 7 Dec. 2020.
[8] "RoBERTa: An optimized method for pretraining self ...." https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/. Accessed 7 Dec. 2020.
[9] "hugging face/transformers: 🤗Transformers: State-of ... - GitHub." https://github.com/huggingface/transformers. Accessed 7 Dec. 2020.
[10] "RoBERTa — transformers 4.0.0 documentation - Hugging Face." https://huggingface.co/transformers/model_doc/roberta.html. Accessed 7 Dec. 2020.

it is to be noted that further retraining could be done to achieve benchmark performances.

Summary of Results

Model	Train	Test	Bench
LSTM	73.1	56.1	58.3
RoBERTa Base With Final Linear Layer	62.1	60.5	71.3
Gradual Unfreezing and Retraining of RoBERTa For Sequence Classification	72.0	70.1	72.1

Table 3: Summary of model performances

Sentiment Trajectory

In order to extract the sentiment trajectory of the story, I moved a sliding window of size 0.25 times the number of words in the story. The step of the window was 5% of this size. These could be tuning parameters in our sentiment similarity extraction methodology.

The model is used to predict the sentiment on the text within the window. The model results in three probabilities, one for each class of sentiment (positive, neutral and negative). A single digit for sentiment representation is extracted from this using this equation.

$$(P(positive) \times 2) + (P(neutral) \times 0) + (P(negative) \times -$$

This number is then averaged over two because the neutral rating will always be zero in any case. The sentiment values are then taken as points and a third order polynomial is fit on them to represent the arc of the story.

In order to find the similarity between arcs, I used the Frechet Distance[11], which is a measure of similarity between curves. The smaller this number, the more similar the curve.

Experiment

In order to try this methodology, I tried inputting my own story and trying to find similar stories from a database created in this paper[12]. This is the story I tried:

```
'Ali was very happy that day but then he suddenly realized that he was forgetting something. He looked at his calendar and realized that he had unfortunately forgotten about his final exam. He ran to the building where his exam was supposed to be and the building was locked. Thankfully, his calendar was wrong and it was a Saturday.He was very relieved and happy again.'
```

As we can see (Figure 1), the model produces results that represent the general shape of the arc of the story.

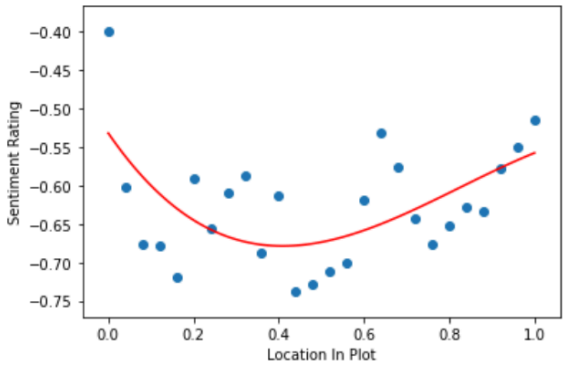

Figure 1: Sentiment trajectory of the story.

I used this arc to then find another story which is similar to this. Given below is the story with the most similar arc.

[11] "Fréchet distance - Duke Computer Science." 5 Apr. 2007, https://www.cs.duke.edu/courses/spring07/cps296.2/scribe_notes/lecture23.pdf. Accessed 7 Dec. 2020.

[12] "Where Have I Heard This Story Before? Identifying Narrative ...." https://www.aclweb.org/anthology/N18-2106. Accessed 7 Dec. 2020.

'Sandy Ricks is a young boy living in the Florida Keys who befriends a dolphin injured by a harpoon. His father, fisherman Porter Ricks is upset, as dolphins compete for fish, which jeopardizes the family income and is upset Sandy neglects his chores. After "Flipper" recovers from the wound, the dolphin puts on a show to entertain the neighborhood children. Later, however, the animal devours Porter\'s entire catch of pompano. Porter harshly berates Sandy for allowing Flipper to jump into the holding pen of valuable fish waiting to go to market, "What's wrong with you boy? How old are you, 12,— almost in your teens, or are you five, — a child who doesn\'t have the sense to know what his next meal depends on?" Reduced to tears, Sandy retreats to his bedroom as Porter\'s wife Martha remonstrates that "he\'s only a boy". Determined to make up for the loss, Sandy sets off to find more fish, and is led by Flipper to a large school of fish near a reef. Later, Sandy is rescued from a threatening shark by Flipper, and the grateful father draws closer to his son. Porter Ricks is finally convinced there are enough fish for both the local residents of the area and the dolphins.'

Both of these stories start with happy beginnings, get darker, and then get happier in the end again. The matching technique seems to be performing as we expected it to.

Limitations

Since the model was trained on a Twitter dataset, we cannot expect it to perform that well in the context of extracting sentiment from stories. However, we can try and take into account the amount of bias this type of model would produce. More work needs to be done in refining the models and adding more details to the narrative similarity between plots.