

Evolving Character Relationships Using HMM and LSTM

Sushmita Gopalan

Haylee Ham

Chelsea Ernhofer

Abstract

The purpose of this project was to use small book synopses to predict positive or negative character relationships. This is based off of work by Snigdha Chaturvedi et al. (2015) in which evolving character relationships are modelled by training a Hidden Markov Model on key text features. Our group recreates this analysis and extends it by also using LSTM Neural Networks to capture character relationships. RESULTS SHOW SOMETHING

1 Introduction

Computational narrative studies is a field within natural language processing which seeks to understand, summarize, and generate stories. An increased ability to form narratives is not only useful when applied to literature or film, but can increase a machine's ability to communicate effectively with human users. Identification of character relationships is an important method used in computational narrative studies to identify and summarize individual character roles within a narrative and thus more fully understand the literature in question. With the ability to properly capture character identities and relationships, further work can be done to explore and understand character personality and motivation.

Current research has developed multiple methods in attempts to effectively understand stories through analysis of the characters within them. Both supervised and unsupervised methods have been applied to tasks such as role and relationship extraction and social network modeling. Our work seeks to identify evolving character relationships through supervised methods.

2 Related Work

The body of this work is based off of a paper by Snigdha Chaturvedi et al. (2015) in which researchers attempt to model the specific and dynamic relationships between characters. This goal is divergent from other character relationship projects which seek to simply predict one static relationship between characters or identify basic character roles within a narrative. This is useful since the relationship between two given characters is likely to change throughout the progression of the novel. Chaturvedi et al. used novel summaries from the website SparkNotes as data. From these summaries, sentences were extracted which contained information on a pair of characters. Human readers went through these sentences and determined whether the relationship between the two main characters mentioned was cooperative or noncooperative. This became the target variable. For the predictors, text was then preprocessed and features such as part of speech tags and dependency parses were extracted. From there, Chaturvedi et al. engineered two types of features: content based features and transition features. Content based features contain information concerning the verbs and adverbs that two characters complete together, separately, or one to another. Content features also include semantic parses which employs a number of frames to the input sentences. Transition features indicate whether the relationship between characters changes between the different extracted sentences. Chaturvedi et al. use logistic regression and decision trees as an unstructured baseline. In these models, each sentence was fed through independently of the sequence to which it belongs and the relationship between the characters in that one sentence was predicted as either cooperative or noncooperative. A second-order Markov Model

was then used to capture the potentially changing relationships between characters over time. Precision, recall, and F1 score were used as evaluation metrics. Results showed that the second order Markov Model had the highest F1 score out of all methods used with a measure of 60.76 compared to 48.54 generated by the decision tree and 51.48 from the logistic regression.

3 Data

Data from the Chaturvedi et al. paper was used for this project. Data consisted of 100 sequences comprising almost 800 single sentences. Half of these sequences were fully annotated, that is, they included indications of which characters were central in each sequence and marked their reference in each individual sentence. For the other 50 sequences which did not include such annotation, annotations were added manually after reading through the sequences and external plot descriptions.

4 Methods

As previously mentioned, our project is, in part, a recreation of the Chaturvedi et al. paper. We attempt to recreate the majority of the content based features, namely verb based interactions between characters and counts of positive and negative spanning words between characters. We then also implement a decision tree and logistic regression as baseline indicators. A first order Hidden Markov Model is used to generate predictions based off of complete sequences rather than single sentences. Finally, we extend the work of Chaturvedi et al. by including a Neural Network model to also predict character relationships, allowing for evolution of said relationships.

4.1 Feature Extraction

Positive and negative verb interaction features were extracted for three separate instances: characters verb together, one character verbs another, and character verb separately. In order to engineer these features, part of speech tagging and dependency parsing were run to capture the structure of the sentence and individual word roles. For characters verb together, verbs were included if they were ancestors of both characters in a sentence. For one character verbs another, verbs were included if it was between an object and subject and the object and subject were the two main char-

acters of that sentence. Verbs that were an ancestor to either of the characters and weren't already included in another set (either characters verb together or one character verbs another) were added to the single character verbs set. Sentiment analysis was then done on all of the verbs included and a binary indicator was created to identify whether characters negatively or positively verbed. In order to capture frames, our group used the frames created by Chaturvedi et al.

4.2 Baseline Modeling

ADD STUFF CHELSEA

4.3 HMM

A Hidden Markov Model(HMM) is a generative model, where a sequence of observable events is generated by a sequence of latent or hidden states. The model we use is a first-order HMM, i.e. we assume that transitions between latent states have the form of a first order Markov chain. The Viterbi algorithm is used for decoding probabilities.

4.4 LSTM

As hypothesized in Chaturvedi et al., relationship sequence prediction is a structured problem. The baseline methods of logistic regression and decision trees treat it as a flat classification problem. The Hidden Markov Model is the method that Chaturvedi et al. proposes be used to address the structured nature of this problem. We further propose fitting a long short-term memory network. Similarly to Hidden Markov Models, LSTM networks can make sequence predictions taking in the entire sequence of inputs and outputting a sequence of predictions about the sequences. The architecture of the LSTM network is made up of two hidden LSTM layers as well as an output layer, activated by the sigmoid function. Binary cross entropy loss was used to train the network. In order to avoid overfitting, both early stopping and dropout were incorporated into the model. There are two dropout layers in the architecture, one in between the LSTM layers and one in between the second LSTM layer and the output layer. In training the LSTM network, several hyperparameters were tuned: the number of epochs, the batch size, dropout rate, the number of hidden layers, and the number of dropout layers. The following values were tested as hyperparameters: 1 and 2 hidden LSTM layers, 1 and 2 dropout layers, 10% and 20% dropout rates, between 4 and 20 epochs, and

between 5 and 20 batch size. At the conclusion of training, 2 LSTM hidden layers, 2 dropout layers, 20% dropout rate, 10 epochs, and a batch size of 10 were chosen as the hyperparameters yielding the most accurate results.

5 Experimental Setup

5.1 Data

Data can be found on Snigdha Chaturvedi's professional website at <https://sites.google.com/site/snigdhaacademics>. Our team downloaded the data from the 2015 paper 'Modeling Evolving Relationships between Characters in Literary Novels'. Data is formatted in such a way that there exists a separate csv file for each sequence. We combined these files into one, including character IDs for each sequence in the final data frame. For the partially annotated data, we included our manual annotations at this point in the process. Sentences that had no label (were neither positive nor negative) were discarded, leaving only two possible classes: cooperative and noncooperative. This left us with 603 sentences and 100 sequences.

5.2 Modeling

The Decision Tree and Logistic Regression were performed using the sklearn library. Data were partitioned into a training and test set (with ratio 80:20) using the train test split functionality included in the sklearn library.

For the HMM, we used an implementation found <https://github.com/larsmans/seqlearn> as a guideline and corrected a minor error in the calculation of emission probabilities. Python's scikitlearn package no longer supports an API for a hidden markov model.

We split the data into training and validation sets in an 80-20 ratio. We split the sequences, rather than the sentences themselves, in order to ensure that no sequences get split up over the training and testing subsets. We ran a 100-fold cross-validation to reduce overfitting on our relatively small set of sequences.

In training the LSTM network, the data was transformed so that rather than feeding in one sentence at a time, each observation was a concatenation of all of the features for every sentence in the sequence. Similarly, the target variable was a sequence of indicators concerning whether each sentence in the sequence portrayed the two characters

in question cooperating or not. The data was then fed into a network created in the Keras deep learning framework. Data were split into a training and test set using the sklearn library's train test split functionality. The split was 80% training and 20% test set.

5.3 Model Evaluation

For each model used, three statistics were calculated: precision, recall, and F1 score. Overall model success was based on F1 score since it is a summary of both precision and recall and was the primary evaluation metric used in the Chaturvedi et al. paper.

6 Results and Analysis

Model	Precision	Recall	F1 Score
Logistic Regression	0.404	0.495	0.445
Decision Tree	0.558	0.554	0.556
HMM	0.391	0.499	0.438
LSTM	0.833	0.817	0.824

Table 1. Evaluation of Classification Models

Table 1 contains the results of each of the four models used in analysis. The models which perform the worst are the logistic regression (F1 of 0.445) and the first order markov model (F1 of 0.438). Both models were influenced by the class imbalance which led to predictions consisting only of one class. The decision tree model performs slightly better with an F1 score of 0.556.

Table here maybe... comparing accuracy between different models

Why accuracy is different between models

Why accuracy is different between their paper and our work - feature extraction?

7 Conclusion and Future Work

Conclusion stuff

CHANGING A LOT OF THINGS IN
HEREEEEEEE

8 References

Chaturvedi, Snigdha, et al. "Modeling dynamic relationships between characters in literary novels." arXiv preprint arXiv:1511.09376 (2015).