December 7, 2015

Extracting Social Network from Literature to Predict Antagonist and Protagonist

Matt Fernandez
matthewO

Michael Peterson
mrpeters

Ben Ulmer
ulmerb

Abstract

The study of communities in literature is a relatively unexplored space, due to the many layers of analysis required to extract information from a book to a network. This paper addresses that void through a process that that involves extracting social networks from fairy tales, classifying relationships in these networks as positive or negative, and identifying the protagonist and antagonist of the fairy tale.

Introduction

We were inspired by David Jurgens, Hardik Vala, Andrew Piper, and Derek Ruths, who investigated the question of detecting characters in literary texts, to tackle a problem that built upon that task: extracting relationships between characters. Though we were unable to acquire the code for their system, we referenced their paper as we implemented our own character detection system.

Our algorithm for extracting social networks consists of four main components. First, we read in a fairy tale and prepare it for processing by removing irrelevant material such as image captions or chapter headings. Then, we identify all the characters in our story. Next, we determine which characters have relationships with each other and whether the sentiment in that relationship is positive or negative. Finally, we use this data to perform several social network analyses, such as determining the protagonist and the antagonist of the fairy tale.

We will first explore literature that exists in this area before walking through the four major components of our algorithm and the results we achieved.

Literature

As mentioned before, we referenced David Jurgens and company's paper "Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts." Their paper describes an algorithm to detect characters in literary novels. To the best of our ability, we implemented many of the features of their character detection scheme in our own system. The field of character detection in literary novels is quite unexplored, and Jurgens group provided one of the most advanced systems in this area. On top of their ideas, we then created some features of our own.

For sentiment analysis, we utilized Stanford's CoreNLP and SentiWordNet, two resources for opinion mining of text. Both provide top-of-the-line sentiment analysis performance and remain two of the best libraries available to the public. We ended up settling on using SentiWordNet for our final model because we didn't consider an entire sentence when analyzing sentiment.

Methods

We will now explore the three major parts of our algorithm in detail and show the results of each individual component, which we compared with our hand-parsed gold networks. Before we performed this algorithm, we stripped the story text of images, chapter headings, and anything else that was not the text of the story.

Character Detection and Coreference

Algorithm

Once the text is ready to be processed, we approach the problem of character detection and coreference. As a general structure, we follow a four step approach:

1. Run StanfordNLP to find coreferent mentions
2. Identify the mention chains that correspond to characters or groups of characters
3. Replace the original text
4. Post process to address clumps of characters and first person words (e.g. “I”)

For step two, we implemented from scratch the sections of Vala et. al.’s paper on character detection that were relevant to the domain of fairytales, in addition to some features of our own devising. Our features include marking all chains that are animate (a character that behaves or has the actions of a living being) as a character, and identifying all chains that are coreferent with pronouns that refer to humans (e.g. “he”). We also let all chains that do not explicitly refer to a character by name yet behave as a character would (such as chains of first person pronouns) fall through a sieve for our post-processing in step four.

In step three, we replace the mentions in all marked chains with a number. This number will be the same across all mentions for a specific chain.

In step four, we have a post-processing step that collects and identifies ambiguous chains. While a chain that includes “Little Red-Cap” or “The wolf” explicitly refer to characters, there are many chains such as “I” or “they,” which indirectly refer to characters. To merge these ambiguous sets of mentions with their appropriate character chain, we treat each chain on a case by case basis. If the chain consists of first person pronouns within a quote, we naively identify the speaker as the character closest to the edge of the quote. If the chain refers to multiple people, we identify the closest cluster of people that the chain can refer to. We also prohibit matching chains with characters that do not pass a set of rules, for example, if an ambiguous mention is part of a conjunction, that ambiguous mention cannot be coreferent with other elements of that conjunction.

Results for Character Detection

Our results were obtained by running our algorithm and comparing our list of characters with a gold version of the characters for the following fairy tales:

Book	**Precision**	**Recall**	**F1**
Hansel and Gretel	.625	1.0	.769
Rapunzel	.625	1.0	.769
The Goose Girl	.667	1.0	.800
Little Red-Cap	.833	.667	.741
The Frog Prince	1.0	1.0	1.0
Average	.750	.933	.815

We achieved an average F1 score of .815. This compares very favorably with the average f1 of .5713 score reported by Vala et. al. in their paper. Part of this discrepancy lies with the fact that Vala et. al. were

analyzing much more complex novels, such as *Sherlock Holmes* and *Pride and Prejudice*, which have much longer casts of characters and more intricate writing, but even still, our results are impressive.

Other Methods

At the beginning of our process, our character detection identified characters that did not exist in the story. To stop this problem, we experimented with various ways of determining when to mark a chain as flawed, and discard it. We tried seeing if there were two mentions of different genders within the same chain, and if there was, to throw out that chain. Unfortunately, this check proved too restrictive and left us with practically no identified characters, and we ended up either reassigning such characters to other groups (see step four above) or eliminating that chain as part of one of the checks proposed by Vala et. al.

Error Analysis

Our main error lies with precision (or incorrectly stating that a character exists), which lags behind our impressive recall. This loss of precision is due to our system incorrectly identifying multiple characters, although only one exists (for example, in the story Little Red-Cap, our system believes that there are two characters that are both Little Red-Cap, although only one Little Red-Cap exists). The lower precision score could be fixed with more aggressive merging of coreference chains.

Sentiment Analysis

Algorithm

Once we have performed coreference resolution on our document, we extract relationships between characters and identify the sentiment of those relationships. To identify relationships, we assume that two characters who appear in a sentence together have some sort of relationship. Therefore, we iterate through our document and identify sentences in which multiple characters appear. For each pair of characters in our story, we will keep track of the sentences in which they co-occur.

Then, for each pair of characters, we have a set of sentences for which these characters co-occur and thus, by our assumption, interact. Our next task is to extract the specific verb phrase that connects these two characters. Using the parse tree, we identify this verb phrase and extract it from the sentences. We now have a list of verb phrases which connect each pair of characters.

Finally, now that we have identified the specific text that unites two characters, we perform sentiment analysis on these verb phrases. We do this by utilizing a library called SentiWordNet, which is a lexical resource for opinion mining that has identified a positive and negative sentiment score for over 130,000 English words. For each word in our phrase, we take the lemma of the word and see if that lemma appears in the SentiWordNet dictionary. If it does, we extract the positive and negative scores of that word. In this dictionary, words may appear several times with different definitions -- we average the scores for all different appearances of this word to determine its positive and negative score. We do this for all words in all verb phrases connecting two characters in order to determine a final positive and final negative score for this relationship. The difference between these two scores will determine the overall sentiment of the relationship and whether it is positive or negative.

Results

To test our sentiment analysis, we check whether we classified the relationships between characters as positive or negative and compare that relationship with the gold reading. We tested on five fairy tales, for which we hand-parsed the gold solutions. The following table shows how many positive and negative relationships we correctly identified out of the ones that exist in each fairy tale:

Fairy Tale	Positive Relationships	Negative Relationships
Hansel and Gretel	2/4	6/6
Rapunzel	4/4	1/4
The Goose-Girl	7/8	3/5
Little Red-Cap	1/3	3/3
The Frog Prince	2/3	0/1
Total	16/22	13/19

Summing together all examples, both positive and negative, we find a sentiment analysis accuracy of 71%. Considering that fairy tales often involve significant amounts of deception -- Little Red-Cap, Hansel and Gretel, and The Goose-Girl all involve one character pretending to be another -- it can be very difficult to classify these relationships accurately. Additionally, fairy tales often follow a typical format of sadness and troubles followed by happiness. Considering these difficulties, we were very happy with our result.

Other Methods Tried

We experimented with using several different APIs for the actual sentiment analysis. First, we trained our own binary linear classifier on a dataset of 25,000 labeled movie reviews in order to determine weightings for various words. We then looked up each word in the verb phrase in our dictionary of weights to determine the score for each word and calculated an overall positive or overall negative sentiment score in this manner. However, because we only had a limited dataset consisting of exclusively movie reviews, this didn't turn out to be as accurate as the SentiWordNet system. In fact, a majority of the words (>50%) we looked up did not contain scores in our dictionary. Due to the lack of available data and relevant data -- we could not find any labeled sentiment data for novels -- we ultimately went with the SentiWordNet dictionary rather than our own classifier, though if we had access to labeled fairy tale data, we would absolutely train our own classifier.

Similarly, we tried using the CoreNLP sentiment analysis. Rather than training on just verb phrases, we would run the entire sentence in which characters co-occur through the CoreNLP sentiment analysis in order to determine the sentiment between these two characters. However, this turned out to return almost all negative scores. In fact, in training on the fairy tale Rapunzel, only two character relationships contained any positive scores whatsoever -- and strangely enough, one of them was the relationship between the Enchantress and Rapunzel.

Error Analysis

Several key factors remain for improvement. First, we do not try to identify cross-sentence relationships and instead rely on the assumption that relationships are confined to a sentence, which is generally, but not always, true. For example, in the paragraph "Bob was excited to go to the boat party. Jim then went on the boat. Bob hurriedly left the boat," it would be reasonable to infer that Bob and Jim have a negative relationship. Our system will not identify any relationship or sentiment between the two characters.

Additionally, just performing sentiment analysis on the verb phrase could possibly leave out crucial information. For example, the sentence “Bob loved Jim until Jim killed Bob” will extract two verb phrase relations “loved” and “killed”. The scores for those two words respectively are +1 and -0.5. After they are combined, there will be an overall positive relationship between Jim and Bob, but a reader knows that this is a negative relationship. Though just analyzing the verb allows us to perform better because it ignores other information in the sentence that could be completely unrelated, ideally we would be able to notice which other parts of the sentence are relevant and utilize just those parts for sentiment analysis.

Finally, fairy tales tend to have mostly negative plots throughout much of the story until the very end, when things become “happily ever after” in the last few lines. Thus, we tend to overclassify relationships as negative. Many fairy tales also involve deception, with one character pretending to be another. This skews the relationship between one character and the fake character.

Network Analysis

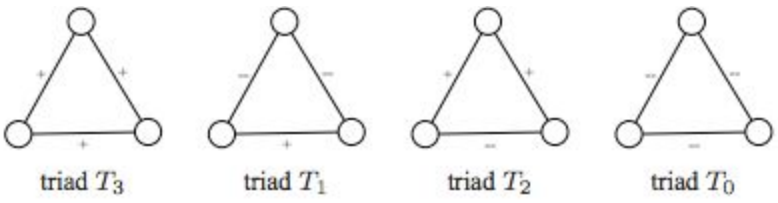
Algorithm

Our network analysis algorithms take character relationship data and plot a graph that includes information about the positivity, negativity, and significance of each character and their relationships. From this graph we set out to understand the complexity and bipartiteness of the graph. Additionally, we developed our own graph analysis algorithm to predict the identity of the protagonist and antagonist in the story. We used this as a lofty goal because it's something that isn’t innately decipherable from a story and the most telling aspect of those informations is character relationships.

Our processed data from the character identification and sentiment extraction is stored as a list of connections and the positive and negative weights corresponding to that connection. This data allows us to construct a network using SNAP. We take the edge list data and create multigraphs for positive and negative categories, which represent our social networks.

Our analysis starts with simpler layers of information and trends towards more complex measurements. The first batch of features (which takes into account positive and negative weights) include:

1. Individuals with the largest degree
2. Individuals with the largest value for sum of all edge weights
3. Largest difference between positive and negative weights to determine the most positive characters
4. Least difference (can be negative) between positive and negative weights to determine the most negative characters



Triad Types - From Leskovec

This set of features grounded us in some core ideals about the contents of the graph, but didn’t lead to significant results for predicting information about the story.

Our large innovation in graph analysis comes from triad analysis on the social networks. By examining the triads that exist in a social graph we are able to find two core pieces of information about the story. First, we can identify if the graph is strongly, weakly or not structurally balanced. This information allows us to identify stories that are

more complex and are more likely to be cases of deception or complicated characters because the relationships triads are unbalanced. The second piece of information gained from triad analysis is individual character's appearance in relation to other characters. For examples characters at the negative end of T1 triads are much more likely to be the antagonist and the protagonist will often be in T3 triads with other allies or positive characters.

This triad analysis, in addition to other features added that aren't related to triads, but add intuitive information about character's emotional valence in the story, were focused into a simple but effective rule-based classifier for identifying protagonist and antagonist. The other features are:

1. Number of times appeared in the story overall
2. Degree of the individual
3. Value for sum of all edge weights in the positive / negative direction

Many more features are possible, but we wanted to generate a system that gave a rough estimate and then scale with more data and adopt a more complex machine learning system.

Results

Best Version of Protagonist and Antagonist with no thresholding

Fairy Tale	**Protagonist ID / Error**	**Protagonist Error %**	**Antagonist ID / Error**	**Antagonist Error %**	**Balanced ID / actual**
Hansel and Gretel	correct / 0	0%	incorrect / 4	75.8%	No / No
Rapunzel	incorrect / 1	13.6%	incorrect / 4	170.5%	Weak / Strong
The Goose-Girl	incorrect / 2	1.7%	incorrect / 1	2.2%	No / No
Little Red-Cap	correct / 0	0%	correct / 0	0%	Weak / Strong
The Frog Prince	correct / 0	0%	None*	None*	Strong / Strong
Overall	3/5	3.6%	1/4	62.3%	3 / 5

* There is no villain in this story

Key:

1. Protagonist Error: Number of characters ranks away from correct prediction
2. Protagonist Error Percentage: Increasing the correct protagonist by this amount will lead to a correct prediction
3. Antagonist Error: Number of characters ranks away from correct prediction
4. Antagonist Error Percentage: Increasing the correct antagonist by this amount will lead to a correct prediction
5. Balanced: Strong = only T3 and T1 triads; Weak = T3, T1, and T0 triads; No = at least one T2 triad (We calculate the triads in two ways so we use the lowest standard)

Error analysis

Our system shows a lot of promise for predicting protagonist, antagonist and story complexity based on social network analysis. For 3 of the 5 stories, we were correctly able to identify the protagonist of the story

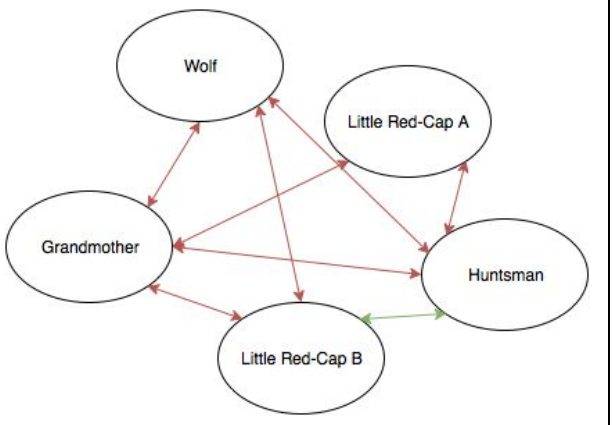
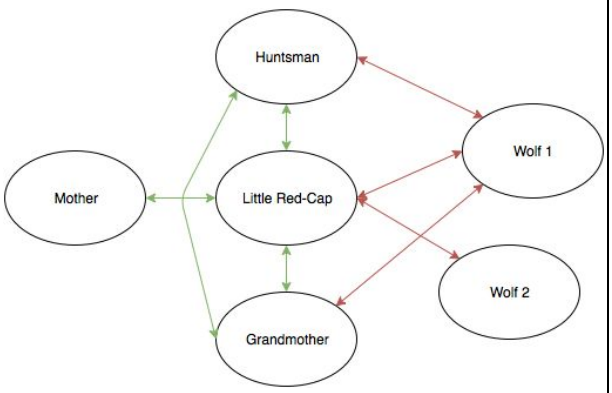
and for the other 2 we were off by only a small margin. By expanding and intelligently training our feature weights, we could likely identify the correct protagonist and for all 5 stories.

Our antagonist identification accuracy was lower because in both “Hansel and Gretel” and “Rapunzel” the villain is a complex character that has a positive relationship that turns sour. These types of relationships aren’t accurately shown in our data because we ignore any temporal aspect of the the relationships. Another issue is simply that the antagonist doesn’t always form lots of enemies (as in Little Red-Cap), so their character may not be as central which makes it harder to identify as the villain.

Because of the variety of moralities and complexities of the fairy tales analyzed its difficult to gain too much from the balance of the graph. Our system got 3 out of 5 identifications here correct, which indicates that although our network edges aren’t perfect the overall relationship network is often correct. One simple, but valuable conclusion is that the frog-prince is a positive and uncomplicated relationship story because it is strongly balanced. This feature is important for our extension into other types of stories that may have similar character archetypes. This aspect of our research is weakened dramatically by accuracy difficulties at the coreference and sentiment layers. Even though our accuracies for those tasks are comparable to current research for the same task. By staying true to a “raw text to network” model, we take a hit in our confidence level for network analysis.

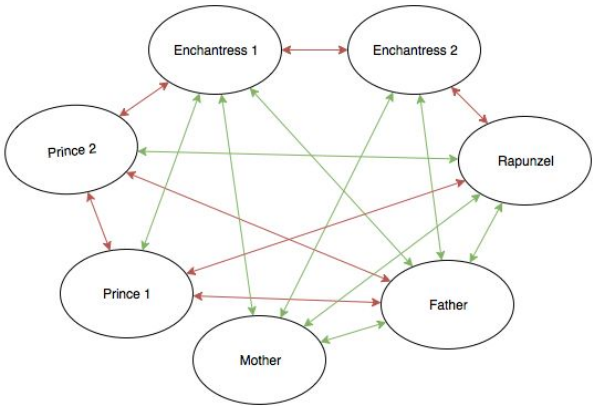
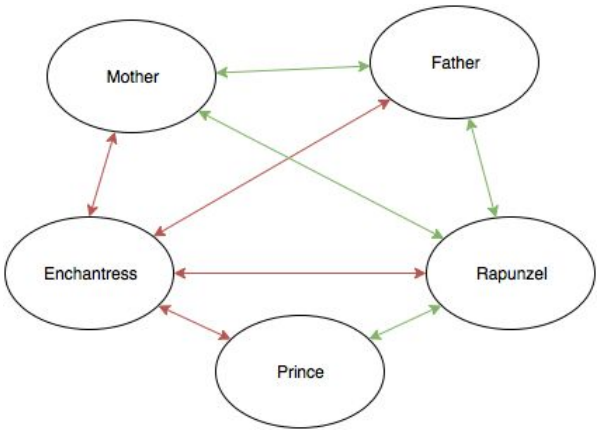
Because the focus of our project was NLP, we focused our energy away from direct tuning and adjustment of our protagonist/antagonist engine. However, with added machine learning algorithms and further statistics beyond positivity and negativity pulled from the data, we can greatly increase the accuracy and significance of those results. Additionally because we focused on fairy tales, we didn’t explore variations in types of stories and differences in authors. With more variety our algorithms can be evaluated in more interesting ways.

Network Visualization / Final Analysis

Our Network for Little Red-Cap	Gold Network for Little Red-Cap
	

Our biggest error is failing to identify the mother as a character and splitting little red-cap across two coreferent chains. However, our system correctly identifies that the wolf has negative relationships with everyone and little red-cap b is highly central so she is identified by our protagonist algorithm correctly. This network also shows our propensity to predict negative relationships in scary stories. Shared anger towards a common enemy (i.e. huntsman and little red-cap both dislike the wolf) can sometimes lead to the sentiment analysis putting a negative relationship between the two allies because they appear in negative sentences

together. Our system finds that the story has weak balance although in the gold version it is strongly balanced. Our error here is primarily due to the deception of the wolf skewing some relationships negative.

Our Network for Rapunzel	Gold Network for Rapunzel
	

Our primary errors resulted from having two extra coreferent chains, one for enchantress and the other for the prince. Without these errors our accuracy of the connections is actually quite close to correct. This error also leads to an incorrect balance identification. In reality the graph is strongly balanced, but our algorithm identifies a T2 triad between enchantress 1, prince 1 and father. Our results are promising however and the sentiment and network analysis worked well on the difficult coreferent groups.

Conclusion

Our paper appears to be the first -- or at least one of the very first -- to dive into not only extracting characters from fairy tales, but also identifying each character's social network and the sentiment of these relationships. Given the lack of available relevant training data, we believe our system performed very well, with an overall coreference F1 score of 0.815, sentiment analysis score of 71%, protagonist score of 60%, antagonist score of 25%, and balanced prediction score of 60% . Future research into this area may involve hand-labeling more data to allow for training classifiers, attempting to notice deception, identify specific attributes of relationships in social networks (e.g. best friends, lovers, brothers, etc.) or more.

Works Referenced

Borodin, A., Roberts, G., Rosenthal, J., & Tsaparas, P. (n.d.). Finding authorities and hubs from link

structures on the World Wide Web. *Proceedings of the Tenth International Conference on World Wide Web - WWW '01*.

Elson, D. (2010). Extracting Social Networks from Literary Fiction. *ACL2010*. Retrieved November 17, 2015, from http://www1.cs.columbia.edu/~delson/pubs/ACL2010-ElsonDamesMcKeown.pdf

Esuli, Andrea, and Fabrizio Sebastiani. "SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining." LREC 2006 (2006). Web. 8 Dec. 2015.

Finn, A., & Kushmerick, N. (n.d.). Learning to classify documents according to genre. *J. Am. Soc. Inf. Sci. Journal of the American Society for Information Science and Technology,* 1506-1518.

Fowler, J. (2006). Connecting the Congress: A Study of Cosponsorship Networks. *Political Analysis,* 456-487.

Free ebooks by Project Gutenberg - Gutenberg. (n.d.). Retrieved November 17, 2015, from http://www.gutenberg.org/

Kunegis, J., Lommatzsch, A., & Bauckhage, C. (n.d.). The slashdot zoo. *Proceedings of the 18th International Conference on World Wide Web - WWW '09*.

Leskovec, Jure, Daniel Huttenlocher, and Jon Kleinberg. "Signed Networks in Social Media." Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10 (2010). Print.

Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

Newman, M. (n.d.). Analysis of weighted networks. *Physical Review E Phys. Rev. E*.

Vala, H., & Jurgens, D. (2015). Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts.