CS 224N: Final Project
Extracting Family Trees from Literary Texts

Alex Chin
Stanford University
Statistics Department
ajchin@stanford.edu

Evan Patterson
Stanford University
Statistics Department
epatters@stanford.edu

Abstract

We develop an NLP pipeline for extracting family trees from literary texts. To our knowledge, we are the first to attempt this task, although we build on existing work about extracting social networks from literary texts. At a high level, our pipeline has two stages. First, we identify characters in the text by clustering person mentions. Second, we attempt to link these characters by discovering family relationships. (The second stage is more difficult than the first.) In this report, we describe our system, evaluate its performance on a few texts, and suggest directions for improvement.

Note: We request that this report *not* be distributed on the public CS224N website.

1 Introduction

The past five years has seen a surge of interest in applying natural language processing (NLP) techniques to literary texts. The motivation for this work comes from both the literary and computer science communities.

For literary scholars, NLP offers an exciting new set of tools for discovering trends in literature on a large scale, over time and across genres. The early work [EDM10] offers an illustrative example. The Russian critic Mikhail Bakhtin [Bak81] has proposed the influential literary theory of *chronotopes*, which holds that a novel's temporal and spatial setting influence the quantity and quality of the characters' interactions. It is argued that novels with a rural setting, which brings the family structure to the fore, should have relatively few characters and a tight social network. By contrast, novels with an urban setting should have a larger and more diffuse social network. These empirical claims can be regarded as predictions of the theory, and it is obviously desirable to subject them to tests. Until recently, such tests have been confined to the detailed exegesis of a small number of novels by individual scholars. Natural language processing offers the possibility of larger-scale studies. As it happens, the findings of [EDM10] challenge the conventional wisdom surrounding the chronotope theory.

For the computer scientists, literary texts both provide a test bed for core NLP techniques and suggest new problems for research. For example, our project relies heavily on named entity recognition (NER) and coference resolution, two mainstays of NLP. It also offers unique challenges not encompassed by the standard tools.

The purpose of the present project is to extract family trees in an automated way from literary texts. A family tree describes the biological and marital relationships between characters. It is naturally connected to the literary hypotheses describe above. For example, do novels with rural settings have fewer and larger family trees than novels with urban settings? (Regrettably, our system is not yet sufficiently developed to address this question in a satisfactory way, but that is the sort of question we would hope to answer.)

The paper is organized as follows. In Section 2, we briefly survey some related work. In Section 3, we describe our system and its implementation. In Section 4, present results for a few literary texts. Finally, in Section 5, we summarize our work and suggest directions for future research.

2 Related work

To our knowledge, we are the first to attempt to extract family trees from literary texts. However, family trees can be regarded as a kind of social network, and there is previous work on extracting other kinds of literary social networks. We survey some of this work below.

An early work on extracting literary social networks, cited above, is [EDM10]. The object of this paper is to construct conversational networks: undirected graphs whose vertices are characters and whose edges indicate that the corresponding characters converse with each other. The weight of the edge is proportional to the amount of conversation. [HAJR12] incorporates sentiment analysis by assigning positive weight to positive interactions and negative weight to negative interactions, leading to the notion of a *signed* social network.

Other types of literary networks have been considered. For example, [LY12] extract bipartite graphs, whose vertices are people and locations and whose edges indicate that a person has been physically present at a location. (They also considers person-person edges similar to those described above.)

All of these problems—including the one considered in this paper—depend upon the reliable identification of characters in the text, as these form the vertices of the social graph. [VJPR] studies character detection exclusively. Its main contribution is to identify characters that are not designed by a proper name, but only by their role in the story. Examples might include "the archbishop" or "the governor." Earlier approaches to character detection tended to ignore such characters.

3 Methods

In the section, we describe the design and implementation of our NLP pipeline for extracting family trees from literary texts. At a high level, our pipeline has two stages. First, we identify characters in the text by clustering person mentions (Subsection 3.1). Second, we attempt to link these characters by discovering family relationships (Subsection 3.2).

Our system is implemented in Java, building on the widely used Stanford CoreNLP toolkit [Man+14]. In particular, we utilize the named entity recognition (`ner`) and deterministic coreference resolution (`dcoref`) annotators in CoreNLP.

3.1 Identifying characters

Like most of the systems discussed in Section 2, our system begins by identifying named characters that persist throughout the text. The goal is to cluster entity mentions that correspond to the same character. Of course, coreference resolution has the same purpose, except that it is not restricted to person mentions. Thus character recognition can be viewed as a specialized form of coreference resolution.

We adopt this point of view in our work. Our character recognition system is simply a higher level of clustering built on top of the CoreNLP coference system. More precisely, our system has the following steps.

1. Find *named entities* using the CoreNLP `ner` annotator.

2. Cluster those entities into *chains* using the CoreNLP `dcoref` annotator, which includes both nominal and pronominal coference resolution.

3. Cluster the chains corresponding to person entities by exploiting conventions about English-language names.

Steps 1 and 2 are standard. The remainder of this subsection is devoted to describing Step 3 in detail. Our approach draws inspiration from [DEK03; EDM10; VJPR] and the references therein, but has some unique elements.

To begin, we form a singleton cluster for every coreference chain that refers to a proper, person mention. We attempt to extract name information from a representative mention in the chain by pattern matching. We match the following patterns, in order of precedence:

1. `{honorific} {first} {middle} {last}`
2. `{honorific} {first} {last}`
3. `{honorific} {last}`
4. `{first} {middle} {last}`
5. `{first} {last}`
6. `{first|last}` (ambiguous)

Here `honorific` matches a list of common English-language honorifics ("Mr.", "Ms.", etc) and the remaining placeholders match words that are identified as singular proper nouns (`NNP`) by the part-of-speech annotator (`pos`) in CoreNLP.

The final pattern is special in that its intepretation is ambiguous. A standalone name can be either a first name or last name, with the convention depending on the author and the genre. To handle this case, we have a separate field for "ambiguous names," which we try to resolve later.

Having created singleton clusters for each chain, we now try to merge the clusters with compatible names. We do this in two passes.

In the first pass, we merge compatible clusters, provided that the merging does not involve arbitrary choices. To be more precise, the relation of compatibility of names and gender between two clusters is not an equivalence relation: it is reflexive and symmetric, but not transitive. For example, "Mr. Bennet" is compatible with "Bennet," and "Bennet" is compatible with "Mrs. Bennet," but "Mr. Bennet" is not compatible with "Mrs. Bennet." Thus, we do not merge either "Mr. Bennet" and "Bennet" or "Mrs. Bennet" and "Bennet," as that would involve an arbitrary choice.[1]

Instead, we define an equivalence relation that strengthens compatibility and take as clusters the partition induced by this relation. Say that chains A and B are *strongly compatible* if A and B are compatible and, moreover, for any third chain C, A and C are compatible if and only if B and C are compatible. It is easily checked that strong compatibility is indeed an equivalence relation. It is the coarsest equivalence relation that preserves compatibility but does not make arbitrary choices.

There is a simple algorithm that computes this partition using $O(nk)$ strong compatibility checks, where n is the number of chains and k is the number of equivalence classes. (It works for any equivalence relation, not just this one.) Thus the worst case performance is $O(n^2)$, but in our application usually $k \ll n$.

This first pass produces an initial, rather conservative clustering. In the second pass, we eliminate all clusters with an ambiguous name. The idea is that standalone names like "John" or "Doe" usually refer to a person with a more complete name, say "John Doe." For every cluster A with an ambiguous name, we find the cluster B that has the most mentions among all unambiguous clusters compatible with A, and then merge clusters A and B. (If there is no such cluster B, we do nothing.) This approach, which is entirely global, is somewhat crude and could likely be improved upon by using local information associated with specific mentions.

3.2 Identifying relationships

Our system proceeds in the construction of a family tree by acting upon the person clusters that are output by the first stage (as described in Subsection 3.1) and constructing pairwise relationships between those clusters. Each relationship is an edge in the social network graph of characters, where the edge is tagged with the type of relationship.

The approach pursued in [EDM10] is based on dialogue networks, in which relationships are identified primarily through dialogue and conversational interactions. However, this approach is heavily influenced by the narrator's perspective, especially if the novel is written in first-person. In particular, for family trees, the dialogue approach would have trouble extracting familial relationships between characters who do not interact with each other.

Instead, our system proceeds through the entire text and attempts to extract all relevant information at the sentence level. We search the text for keywords that indicate family relationships, such as "father," "sister," or "cousin." At a high level, we attach this relation to pairs of characters that are mentioned in close proximity to the family relaton keyword. In particular, we attach all pairs of characters corresponding to mentions that appear in the sentence as the keyword, after performing a few basic consistency checks. We check to exclude multiple mentions to the same character. Furthermore, we use the gender information provided in the named-entity recognition system and the person clusters to enforce gender constraints (so that male characters are not assigned to be sisters or mothers, for example).

3.2.1 Extensions

There are a number of enhancements that can be made to the relationship identification system, and indeed a well-performing implementation will likely be composed of several rule-based passes that iteratively extract relationships and perform consistency checks.

One such pass that will help to augment and complete the family tree structure is to enforce consistencies that exploit some of the symmetries that are present in the nature of familial relationships. For example, some family keywords, such as "cousin" and "sibling" are symmetric in the sense that if character A is the cousin of character B, then character B is also the cousin of character A. (This is not true for other relationships such as "mother" or "grandson.") In other cases, if we know that character A is the father of character B, then we can conclude that character B is the son or daughter of character A, depending on the gender

[1]The gender attitudes of the 19th century make it much more likely that "Bennet" refers to "Mr. Bennet" than "Mrs. Bennet," but we do not make use of such information.

status of character B. In this way, we can draw a more complete relationship network and begin to display a fuller family tree.

Another extension is to perform a more attentive search of mentions around a given keyword. For example, we may wearch neighboring sentences. Along these lines, one option is to take into account the parse structure of the sentence rather than using pure proximity in order to assign relationships. A section of text may contain mentions of characters who are simply engaged in discussion with or about other characters who belong to the relationship, and would be wrongly classified by proximity into the relationship himself or herself. Additionally, for constructions like "she is my mother," we can utilize the parse structure of the sentence to help the system determine which character is the mother and which is the son or daughter.

Finally, we may exploit some common naming conventions to improve the system, especially for literary texts whose social structure mimics traditional Western social structure, such as many novels written in the eighteenth or nineteenth century. For example, two characters that appear frequently and both have the last name Johnson, but one is usually referred to as Mr. Johnson and the other as Mrs. Johnson, are likely a husband and wife pair.

4 Results

We examine the output of our system on four texts: *Wuthering Heights* by Emily Brontë [Bro47], *Romeo and Juliet* by William Shakespeare [Sha97], *Frankenstein* by Mary Shelley [She18], and *Adventures of Huckleberry Finn* by Mark Twain [Twa85]. These texts were chosen primarily because of their variety in family structure. They were accessed from Project Gutenberg and any metadata were stripped from the file before the file was input into the system.

The number of characters and number of relationships identified by our system for each of the four novels is listed in Table 1. A significantly larger number of characters was identified in *Wuthering Heights* than in the other three texts; this is likely because of the importance of the family environment to the structure of the novel, and so familial relationships are likely explicitly expressed frequently throughout the course of the novel. *Romeo and Juliet*, on the other hand, only has three relationships identified, a likely result of the Shakesperean style of writing, which makes poetic and indirect references to familial relationships, even though the plot is centered around the two prominent, rival houses of Montague and Capulet. The specific relationships identified by our system are displayed in Tables 2, 3, 4, and 5.

Text	Characters	Relationships
Wuthering Heights	67	34
Romeo and Juliet	41	3
Frankenstein	40	12
Huckleberry Finn	114	6

Table 1: Number of characters and relationships identified

5 Conclusion

In this project we attempted to implement a system that can extract family trees from literary texts. The system consists of two largely separable tasks: extracting characters and extracting relationships. In order to extract characters, we build on Stanford CoreNLP's named entity recognizer and deterministic coreference resolver systems and tailor a rule-based system to cluster people mentions into characters. For relationship extraction, we use a keyword system to link mentions by proximity to keywords that indicate family relationships. We discuss the results of our system on four popular literary texts, and also highlight a number of ways in which the system can be improved.

References

[Bak81] Mikhail M Bakhtin. "Forms of time and of the chronotope in the novel" (1981).

[Bro47] Emily Brontë. *Wuthering Heights*. 1847.

[DEK03] Peter T Davis, David K Elson, and Judith L Klavans. "Methods for precise named entity matching in digital collections". *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*. IEEE Computer Society. 2003, pp. 125–127.

[EDM10] David K Elson, Nicholas Dames, and Kathleen R McKeown. "Extracting social networks from literary fiction". *Proceedings of the 48th annual meeting of the association for* *computational linguistics*. Association for Computational Linguistics. 2010, pp. 138–147.

[HAJR12] Ahmed Hassan, Amjad Abu-Jbara, and Dragomir Radev. "Extracting signed social networks from text". *Workshop Proceedings of TextGraphs-7 on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics. 2012, pp. 6–14.

[LY12] John Lee and Chak Yan Yeung. "Extracting networks of people and places from literary texts". *26th Pacific Asia Conference on Language, Information and Computation (PACLIC 26)*. 2012, p. 209.

[Man+14] Christopher D Manning et al. "The Stanford CoreNLP natural language processing toolkit". *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2014, pp. 55–60.

[Sha97] William Shakespeare. *The Tragedy of Romeo and Juliet*. 1597.

[She18] Mary Wollstonecraft Shelley. *Frankenstein; or, The Modern Prometheus*. 1818.

[Twa85] Mark Twain. *Adventures of Huckleberry Finn*. 1885.

[VJPR] Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. "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". *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 769–774.

Character 1	Relation	Character 2
Mr. Linton	uncle	Master Linton
Mrs. Linton	sister	Mr. Heathcliff
Mr. Heathcliff	cousin	Miss. Catherine
Mr. Earnshaw	father	Mr. Heathcliff
Joseph	father	Miss. Catherine
Mr. Hareton	father-in-law	Mr. Heathcliff
Mr. Heathcliff	cousin	Mrs. Linton
Mr. Heathcliff	father	Mrs. Linton
Mr. Heathcliff	father	Mrs. Earnshaw
Joseph	brother	Miss. Catherine
Mr. Linton	father	Catherine Linton
Mrs. Linton	cousin	Mr. Heathcliff
Master Linton	uncle	Mr. Linton
Mr. Linton	cousin	Miss. Cathy
Mr. Heathcliff	child	Mr. Hareton
Mrs. Earnshaw	parent	Mr. Heathcliff
Mr. Heathcliff	father	Mr. Earnshaw
Mr. Hareton	cousin	Mr. Heathcliff
Miss. Catherine	sister-in-law	Joseph
Mr. Heathcliff	father	Mr. Hareton
Mr. Heathcliff	husband	Mrs. Linton
Miss. Cathy	cousin	Mr. Linton
Mr. Heathcliff	father-in-law	Mr. Hareton
Mr. Heathcliff	father	Miss. Catherine
Mr. Heathcliff	father-in-law	Miss. Catherine
Mr. Heathcliff	cousin	Mr. Hareton
Mr. Linton	father-in-law	Mrs. Heathcliff
Mr. Linton	father	Miss. Catherine
Mr. Hareton	child	Mr. Heathcliff
Mr. Heathcliff	parent	Mrs. Earnshaw
Mr. Edgar	brother	Miss. Catherine
Mr. Edgar	brother	Miss. Linton
Mr. Hareton	father	Mr. Heathcliff
Mrs. Linton	mother	Mr. Heathcliff

Table 2: Relationships identified in *Wuthering Heights*

Character 1	Relation	Character 2
Montague	nephew	Lady Capulet
Montague	son	Prince Escalus
Prince Escalus	son	Montague

Table 3: Relationships identified in *Romeo and Juliet*

Character 1	Relation	Character 2
Justine Moritz	sister	Elizabeth Lavenza
Ernest	cousin	Poor William
Justine Moritz	aunt	Elizabeth Lavenza
Elizabeth Lavenza	aunt	Justine Moritz
Felix	brother	Agatha
Poor William	cousin	Ernest
Elizabeth Lavenza	cousin	Justine Moritz
Felix	father	Agatha
Ernest	father	Poor William
Elizabeth Lavenza	sister	Justine Moritz
Justine Moritz	cousin	Elizabeth Lavenza
Poor William	father	Ernest

Table 4: Relationships identified in *Frankenstein*

Character 1	Relation	Character 2
Archibald Nichols	son	Tom Sawyer
Peter Wilks	uncle	Miss. Mary Jane
Tom Sawyer	son	Archibald Nichols
Uncle Harvey	uncle	Miss. Mary Jane
Peter Wilks	uncle	Uncle Harvey
Uncle Harvey	uncle	Peter Wilks

Table 5: Relationships identified in *Adventures of Huckleberry Finn*