Graphical Summarization of Characters and Relationships in a Novel

Saikrishna Kalahasti Karthik
Department of Computer Science
University of Illinois at Chicago

Ajinkya Upasani
Department of Computer Science
University of Illinois at Chicago

Abstract

We aim to map a layout of all the characters in a novel, while trying to identify their relationships, if any, with other characters in the novel. We focus on identifying all the characters using language processing tools and try to weed out all the conversations between to recognized entities and also the adjectives used to describe them. Furthermore, looking at the interactions and the overall tonality of the languages used by the character, we aim to identify the integrity of the character and recognize relationship between two characters as positive and negative. We also use some web scraping about the novel to extract more data from the internet. A list of characters along with their closest acquaintances in the text is tagged and produced as an output expanding upon the nature of relationships.

1 Introduction

A lot of research in the NLP field has been done in textual summarization of novels aiming to describe the entire text in a brief and concise manner. These systems try to gauge the crux of the information being portrayed in the story to build up a knowledge base. The knowledge base is then converted back into a textual block summarizing the entire text in a few sentences. These techniques focus more on knowledge but less on the sources or characters in the text that provide the knowledge. These summarization techniques, however, do not focus on the characters and their interaction with other characters to be able to generate a concrete entity-entity relationship survey.

Most of the semantic summarization techniques can be classified broadly into two categories, namely, Extractive and Abstractive. Extractive summarization works on the words tokens and sentence tokens generated from the text and picks up the most important tokens which are then sent to a summarization model. Abstractive summarizations use some deep learning techniques, like the use of sequence to sequence learning models being used along with neural networks to produce target summarization.

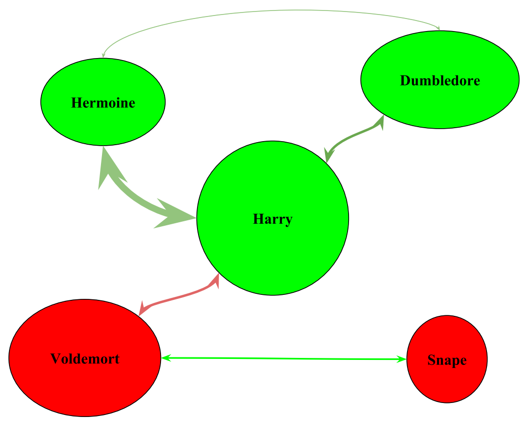

Figure 1: A Part of the Character-Relationship Graph of Harry Potter Novel

Identifying the character to character relationships in a novel can be helpful not only to summarize the novels but also to find similar works of literature from the same author. E.g. The character of *Elizabeth Bennet* in the novel *Pride and Prejudice* resembles *Elinor Dashwood* in *Sense and Sensibility*, authored again, by *Jane Austen*; more than either character resembles *Allen Quatermain* in *Allen Quatermain* authored by *H. Rider Haggard*. Austenian protagonists should resemble each other more than protagonists from books written by other authors (Bamman, 2014). Many literature studies prove that the character personalities and relationships tend to be similar when authored by the same person as compared to the works by other authors.

We have aimed to create a system which would take any novel as an input and return all the characters in the book as an output. We also show all the closest characters that a character has in the novel based on the interactions that took place in the novel. Furthermore, the system recognizes the integrity values of the character, trying to figure out the antagonists and the protagonists of the storyline.

We have taken data from Project Gutenberg, a collection of novels; and Harry Potter book series as input. As all the sentiment analysis lexicons available over the internet were primarily made for reviews, we updated and manually annotated Prof. Bing Liu's sentiment lexicon to generate 3998 words; by crowdsourcing. have aimed to create a system which would take any novel as an input and return all the characters in the book as an output as a graphical representation.

The method implemented can be broken down into 4 major parts, i.e. Character recognition in the text using, where are the characters are extracted along with their frequency. Next, the character closeness relationships are identified based on the communication between the two characters. Thirdly, the integrity of characters is computed upon by referencing the overall sentiment around the character. Finally, nature of relationship of characters through the novel is classified as good or bad based on the overall collected data.

2 Related Work

Many techniques from language processing and deep learning have been used, individually or combined, to achieve the task that we aim to achieve. Although, most of the work is not flexible to work on any input dataset and are heavily annotated or transcribed to achieve optimum results, we aim to create a system which would sustain accepting any novel that has more than a few characters and provide a generalized output.

Most of the notable techniques used are Information Retrieval, K-means clustering, probabilistic entity classification, supervised, semi supervised and unsupervised leaning techniques to learn entity-entity relationships.

Character Clustering (BookNLP): Another paper by Bamman et. al. uses clustering based on Euclidean distances and Bayesian probabilities to classify the roles of the characters. The aim of the paper is to tag the character persona, distributed over 4 dependency relations, namely, agent, patient, possessive and predicative. It does so by clustering the characters (while resolving pronouns using co-reference) using k-means algorithm and reducing the dimensions of the resulting clusters to generate a clustered representation of the character based on the aforementioned roles. Bamman et al went on to create a project called 'BookNLP' which identifies all the characters in a novel. It also identifies the closest relationships within characters in the book(Bamman, 2014).

Propp's Model : It's worth noting that most of the folktale and novel (even modern movie storylines) character breakdown frameworks follow Vladimir Propp's 'Dramatis Personae', which is an extensive breakdown of 31 narration units wrapped in 4 spheres, namely, introduction, the body of the story, the donor sequence, the hero's return. Furthermore, the characters are broadly broken down into 8 broad character types like the hero, the helper, the villain, the false hero, the donor, the dispatcher, the princess, and the princesses' father (Morphology of the Folktale , 2nd ed by Vladimir Propp).

Information Extraction : The model using Information Extraction uses the information that is extracted and tries to scheme it in to the Propp's character models (Groza,2015). Furthermore, as the system aims to extract data from folktales, an annotated database is created, which would help breakdown all character entities using a well-defined ontological based on characters often portrayed in folktales (e.g. gender, plant, animal, object, family, etc.). These annotation sets are used in correspondence with entity recognition tools to place the entity in the proper category. Furthermore, a relationship table is maintained along with the aforementioned database, which classifies the entities based on roles of the characters, like brother, sister, fiancée, etc. As all the entities are placed in their respective location in the database, a treelike structure can be viewed which connects entities based on their type and relationship with other entities in the folktale.

Semi supervised learning : Semi supervised framework is used to classify the transition of relationship between 2 characters throughout the story. This is done by classifying text blocks into one of 4 categories, namely positive, negative, ambiguous, and kinship (Fei Liu, 2015). This is done using feature engineering to search for characters, adverbs, adjective and sentiment transition, etc. E.g. the system is able to recognize the transi-

POS Tags	Bing's Lexicon	Cleaned Lexicon
Adjectives	2235	1388
Adverbs	890	543
Nouns	2533	1681
Verbs	1130	314
Spelling Errors	293	72
Total Tokens	**7081**	**3998**

Table 2: Comparison by number of tokens with different Part-of-Speech Tags in Bing's Sentiment Lexicon vs. The modified lexicon.

Annotators	0	1	2	3	Total
Annotator 1	5	715	183	97	1000
Annotator 2	3	561	269	167	1000
Annotator 3	0	630	265	105	1000
Annotator 4	10	412	341	235	998
Total	**18**	**2318**	**1058**	**604**	**3998**

Table 2: Comparison of number of annotations made between the values 0 and 3 by each annotator

tions like relationship between '*Harry and Ron*' in *Harry Potter and the Deathly Hallows*, transitioning from positive to negative and back to positive again in the end.

Supervised learning : Speaker recognition and speaker utterances are utilized to find features like distance to utterance, speaker appearance count, vocative speaker name, neighboring utterances, gender and presence matching (Hua He Demilson, 2013). Furthermore, speaker utterances are classified considering 3 types, namely, implicit features, explicit features and anaphoric features. All these features are used to create a social network to find gender, relationship type(father, mother, etc.) along with conversation between 2 characters talking about a third character.

Abstractive Summarization : Abstractive methods use most of deep learning techniques like sequence to sequence (Sutskever et al., 2014) learning models, where recurrent networks read the text, encodes it and then generate target text (Dohare, 2018). These techniques use Abstract Meaning Representation (AMR) graphs using co-occurrence resolution, anaphora resolution and network generation by creation of character nodes (Dohare, 2018) (Chaturvedi, 2016). AMR graphs are generated using node features, edge features along with coreference resolution techniques.

3 Data

As mentioned earlier, our goal was to come up with a generic model that performs well across all the genres of the literature. We handpicked several novels ranging from the genre of classic English literature to modern fantasy. Two major sources of our dataset is given below.

3.1 Project Gutenberg

Project Gutenberg is a volunteer effort that collects and distributes texts which have fallen out of copyright in the United States (Project Gutenberg). It consists of mostly literary fiction such as novels, plays and collections of poetry and short stories, along with non-fiction titles such as biographies, histories, cookbooks, reference works and periodicals (Project Gutenberg). The entire contents of the current archive has almost 57,000 documents, though the work here is mostly based on the classic works in English literature. Nearly all major canonical works of English literature published before 1923 are included in the collection. Although the choice of texts in Project Gutenberg is fairly limited compared to other publicly available corpuses such as HathiTrust, the Internet Archive, and Google Books, we chose to work with Project Gutenberg's texts because they are not merely an output of OCR technology but also have been proofread by human volunteers at least once, and some are even typed manually.

3.2 Harry Potter Collection

Harry Potter is an extremely popular series of fantasy novels by J.K Rowling. The plot of these novels revolve around the its central characters, Harry Potter (the protagonist), Hermoine Granger, Ronald Weasly and how they manage to take down the antagonist, Lord Voldemort with the help of their mentor/professor Albus Dumbledore. The work is extremely rich in characters and plot. These novels were downloaded from the web in the form of text document.

In spite of understanding the copyright issues behind using such unauthorized sources, we chose these novels anyway for a variety of reasons: it was easier to find annotators for our task because of the novels' popularity; since it is unlikely that the book-nlp package was trained on this data, we are justified to rely on its results and be certain of the generality of our work; it would be a lot easier to convey the results of our

Character ID	**Character Integrity**	**Close Relationship 1**	**Relationship Type**	**Close Relationship 3**	**Relationship Type**	**Close Relationship 3**	**Relationship Type**
53	1	33	1	170	1	95	1
33	1	170	1	53	1	95	1
170	1	33	1	53	1	176	1
95	1	33	1	170	1	53	1
119	0	53	0	95	0	129	1

Table 3: Snippet of the Validation Dataset

work to our audience; our work is purely academic with no commercial motive.

We created a dataset based on the novels collected from the abovementioned sources. We manually cleaned the headers and footers in each text, looked for spelling errors using python's autocorrect package and corrected them. We created an index of such cleaned text to load into our model.

3.3 Modified Bing Liu's Sentiment Lexicon

Bing Liu's sentiment lexicon is a collection of English words that usually influence the sentiment of a given sentence in the positive or negative way (Bing Liu's Lexicon). The lexicon consists of two lists, a set of positive words and negative words. The original list includes a word, its usual spelling variations, common spelling mistakes of the word and various part-of-speech forms of the word. For our task, we considered only the list containing the set of negative words.

Modification We removed all the erroneous spelling variations, lemmatized the words using NLTK lemmatizer and thus having only one form of each word. We then tailored this dataset of ~4000 words to match the requirement of our work. Although we had a list of negative words, there was no telling to how these words would influence the opinion of a reader about the characters in the novel when these words are associated with them. Table 1 consists information about the Bing's negative list of words and their part of speech tags breakdown against our cleaned dataset.

Labelling We used two native English speakers and two non-native English speakers to annotate each of the words on the scale of 1 – 3, with 1 being slightly negative and 3 being highly negative. The annotators were also asked to label a word 0 if they were not able to find its meaning or considered the word non-sensical. The annotators were asked to label the words with the following understanding in their mind: *If a word x is used to describe a character A in a text (particularly a novel), on a scale of 1 – 3, how bad would you think A is?* Each of the annotators were given 1150 words to annotate, 950 of which were unique to them and 200 were common for all the annotators. For those 200 words, the labels were decided based on rounding off the mean of the respective annotations from all four annotators to the nearest integer. An online tool (Geertzen, 2012) was used to compute the Fleiss kappa statistic (Fleiss, 1971) and we got a moderate intercoder agreement of 0.565. Table 2 shows the total number of annotations made between values 0 and 3 by each annotator. Annotators 1 and 2 are the native English speakers, Annotators 3 and 4 are the non-native English speakers.

3.4 The Validation Dataset

In order to quantitatively analyze the results of our model we built a validation dataset for two novels, namely *Pride and Prejudice* and *Harry Potter and the Deathly Hallows.* Again, the validation dataset was prepared by a team of annotators consisting of a native English speaker and a non-native English speaker. Both the annotators worked in unison to come up with an optimal labelling for both the novels. The results were purely subjective and left to the discretion of the annotators. The annotators were given a character ID – character name dictionary generated from the output of the book-nlp package (discussed in section 3.5). The annotators had to use this dictionary and fill the table given to them by the ID given in the dictionary. Table 3 consists of the snippet of this table. The dictionary technique was followed to avoid any sort of ambiguities that could occur while labelling the characters and to enforce uniform identification for characters across the model.

The annotators had to come up with the following data for each novel:

Step 1 *Identifying the top 10 characters* – Who according to them were the 10 most important and influential characters in the book.
Step 2 *Identifying the top 3 relationships* – In their view, for every character identified in step 1, who were the three closest characters from the book they would associate.
Step 3 *Identifying the character integrity* – For every character identified in step 1, what do they think about the ethical integrity of those characters with respect to the plot. They had to label 0 if they believed that the character was bad, 1 otherwise.
Step 4 *Identifying the relationship type* – For every relationship identified in step 2, what do they think is the nature of the relationship between those characters? They had to label 0 if they believed they shared a bad relationship, 1 otherwise.

3.5 The Book-NLP Output Format

As mentioned in the previous in the section 2, book-nlp is a package used for Coreference resolution and Anaphora resolution of the characters in a novel. The format of the output of this package can be found in book-nlp webpage. It is necessary to understand the output format because the output of this package is directly incorporated in our model. We extracted the following features from the output of the book-nlp package.

Feature Extraction from Book-NLP Output
Feature N_1 *Character Position Identification* – We formed a position dictionary that contained all the positions of occurrence for every identified character within the text.
Feature N_2 – *Speech of the Characters* – We formed a speech dictionary that contained the text of all the speeches by every identified character within the text. We performed preprocessing step 6 described in the next section on this feature text.
Feature N_3 – *Agent Actions* – We formed a verb dictionary that contained the words describing the actions of every identified character within the text.
Feature N_4 – *Character Co-Occurrence* – We created a matrix of size N x N, where N is the number of characters in the novel. In each cell$_{i,j}$ ($0<i<n+1$ *and* $0<j<n+1$) we updated a list of sentences that had the occurrence of both character i and j.
Feature N_5 – *Character Occurrence* – We searched the occurrence of a character and extracted a window of text around it

3.6 Web Scraped Data

In addition to the texts in the novel, we needed additional features derived from the texts scraped from the web to understand the true nature of the characters and relationships among the characters. This is because most of the times a writer while describing a character in his or her book, may not use explicit language but would use subtle linguistic features to move the opinions of the reader.
Preprocessing the Web Scraped Data
Step 1 *Query Building* – We build two dynamic queries, one based on the title of the book and the other based on the current character name and the title of the book.
Step 2 *Google Search* – We perform google search on these using Google's Custom Search API [11] and get top n results, where n can be user-defined.
Step 3 *Text Extraction* – We then extract the text from the n pages obtained above using inbuilt python libraries named BeautifulSoup (Beautiful soup).
Step 4 *Keyword Search of the Character* – The positions of all the tokens associated to the given character were found.
Step 5 *Finding Window Text* – For every token obtained from the previous step, we collected a set of n tokens preceding it and following it (n configurable in runtime). We call this a *window text.*
Step 6 *Filtering Non-Relevant Words* – We performed POS tagging using NLTK on the window text and retained only those words that were tagged as adjectives, adverbs and verbs.
Feature Extraction from Web Scraped Data
Feature Q_1 *Novel Title Query* – This contains the preprocessed text of the query output from Google searching the title of the novel.
Feature Q_2 – *Character Specific Query* – This contains the preprocessed text of the query output from Google searching the character's name alongside the title of the novel. *"Lord Voldermort's Character Analysis in Harry Potter and The Deathly Hallows"* can be treated as an example of such a query.

4 Task Description and Methods

Our ultimate aim of this project is to build a system that read text and outputs a graphical summarization in the form of a character-relationship graph. To achieve the goal, we split our project into four different sub-tasks.

Story ID	Total Characters in story	Total Tokens in the story	BookNLP Output				Our Model Output			
			Correct	Errors	Total	Execution time (sec)	Correct	Errors	Total	Execution time(sec)
Story 1	5	274	2	0	2	< 10	5	2	7	256
Story 2	7	389	4	0	4	< 10	7	2	9	398
Story 3	1	336	0	0	0	< 10	1	2	3	376
Story 7	6	366	3	0	3	< 10	6	2	8	493
Story 15	3	274	1	0	1	< 10	3	1	4	561

Table 4: Results of the model from Section 4.1

4.1 Identifying the Characters in the Text

The first step is to identify the characters in the text. As described before we have used book-nlp package to identify the characters in a text. The most frequently occurring characters in the novel were considered as the most important characters. We extracted top few characters based on this frequency. The number of characters to be extracted can be configured in run time. We call this number as **n_char**. The implementation is straightforward. It can be found in book-nlp. We however observed that book-nlp does not perform well for shorter stories that are usually about a page long. We built an experimental NLP pipeline to overcome this issue.

Character Identification in Shorter Texts This method uses various prebuilt NLP packages. We loaded the text into the python auto-correct package to clear any existing spelling errors. We used Spacy's NeuralCoref to resolve the pronouns in the text. We then performed dependency parsing using Stanford CoreNLP's dependency parser (Manning, 2014). We identified all the phrases that were tagged as nSubj and dObj and performed animacy detection on them using Wordnet (George Miller, 1995) and considered all the animate nouns as characters. We then did a frequency count of the characters on the pronoun resolved text to find its importance. The order of the steps in the pipeline was purely based on trial and error. The pipeline is clearly shown in the figure 1. The analysis of the performance of this model against book-nlp package can be found in the next section.

4.2 Identifying the Close Character Relationships

The next step was to understand the closeness of relationships among the characters in a novel. To find out how closely one character is associated to another, we built an N x N character-distance matrix, where N is the number of characters in the novel. Our algorithm pre-computes the distance of

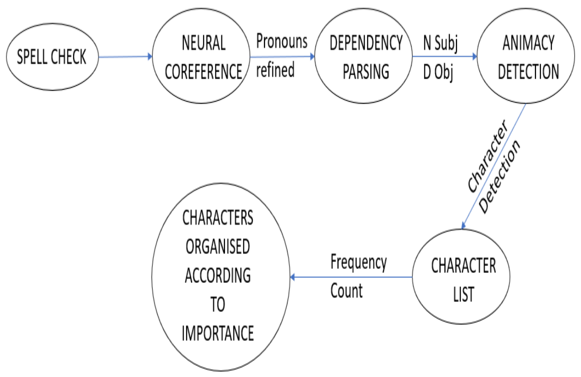

Figure 1: Character Identification Pipeline in Short Texts

every character to every other character in the text and gives the top k closest relationships for every character, with k being configurable in runtime. We call this number as k_char. To compute the distance between character A and character B, we see how often they have occurred within a specified token window using the feature N_1. We then normalize the score by the frequency of occurrence of the lesser frequent character.

4.3 Analyzing the Integrity of Characters

The next step was to identify the integrity of the characters. We fairly simplified the task into a two-class classification problem, with 0 being negative character and 1 being the non-negative character. We built four models by incrementally adding the features described in the previous section. For every model the steps of computing integrity remain the same.

Feature Sets of the Models

Model 1 N_2 ∪ N_3

Model 2 N_2 ∪ N_3 ∪ N_5

Model 3 N_2 ∪ N_3 ∪ N_5 ∪ Q_1

Model 4 N_2 ∪ N_3 ∪ N_5 ∪ Q_1 ∪ Q_2

Validation Data	Total Relationships	Correctly Classified Relationships	Accuracy
Harry Potter	30	17	0.578
Pride and Prejudice	30	14	0.467

Table 5: Results of the model from Section 4.2

Validation Data	Model 1	Model 2	Model 3	Model 4
Harry Potter	0.7	0.8	0.8	0.9
Pride and Prejudice	0.4	0.7	0.8	0.8

Table 6: Results of the models from Section 4.3

As discussed in the previous section, the modified Bing's lexicon table consists of words and their corresponding negative score. We search this table to get the score for every word in the feature set. The score is 0 if the word is not found in the table. The negative score of a character is an arithmetic mean of the scores of all the words in the feature set. This score was normalized by the table hit count of the character. Finally, we used the inverse of n_char as the threshold value to compute the label. If score > 1/n_char, the label is 0, 1 otherwise.

4.4 Analyzing the Nature of Relationships among the Characters

The next step was to identify the nature of relationships between the characters. Again, we simplified the task by making it into a two-class classification problem with 0 referring to bad relationship and 1 referring to good.

Feature Sets of the Models We used the feature N_4 for this model. When analyzing the relationship between a pair of characters, we searched the matrix and obtained the sentences where the characters co-occurred.

We ran these sentences one by one on TextBlob and NLTK's sentiment analysis package and obtained sentiment score. We then computed the arithmetic mean of the sentiment scores. The mean was normalized, and the labels were calculated in the same way described in section 4.3.

5 Results and Analysis

The performance of the model described in section 4.1 can be found in Table 4. Our model consistently outperformed the book-nlp package on shorter texts. We believe this trend could be because of the fact that a good coreference resolution model is sufficient for shorter texts, while the quality of result on the longer texts might be heavily dependent on its underlying anaphora resolution models. On the downside, this model will certainly fail on longer texts due to the limitations of NeuralCoref. Furthermore, this pipeline is extremely slow and will not be feasible on long texts.

We analyzed the performance of our models described in Section 4.2, 4.3 and 4.4 based on the validation dataset described in Section 2.

The results of the model described in the section 4.2 can be found in the Table 5. We think the results are fairly modest because of the high degree of subjectivity in the understanding and interpretation of a novel. Our model computes relationship ranking purely based on the character proximity within the novel. However, the data of the annotators could have been affected by a limited recollection. Also forcing the annotators to label only the top 3 close relationships for the characters was not a good idea. This reduced our accuracy greatly.

The analysis of the four models described in the Section 4.3 is found in Table 6. As you can observe, the results of the model kept getting better with additional features. While building an accurate model only based on the novel's text would have been interesting, we decided to include web scraped text features because, this could make the model work reasonably well even for the most enigmatic novels. However, this technique does come with its downside. This may not work very well for lesser-known novels with few online resources. To overcome this, we have made usage of web scraped data as optional. One can configure to include or exclude the web scraped data from the model.

We have given the results of TextBlob and NLTK in Table 7. We find that the NLTK package performs fairly better for this task than the TextBlob. The reason for this could be because NLTK

Validation Data	TextBlob	NLTK
Harry Potter	0.842	0.947
Pride and Prejudice	0.643	0.785

Table 7: Results of the models from Section 4.4

package might have been trained on a more generic dataset than TextBlob. We speculate that TextBlob is limited to classifying the sentiments of customer reviews, movie reviews.

6 Conclusion

We built a system which was flexible in accepting novels ranging from 10-line short stories to big novels as input. The system would return a graphical representation of all the character relationships align with their induvial character integrity throughout the story. We made use of multiple NLP techniques and toolkits and also showed that our model was better than stand-alone toolkits available online. We manually annotated data to suit our demands for novel-style textual architecture, using 2 native and 2 non-native speakers and obtained better results in sentiment analysis than the review-based sentiment lexicons available online.

Our project has a lot of scope for improvement and has potential to generate a higher accuracy than currently generated. The output of the project could be more graphical and aesthetic using data visualization. The sentiment analysis could be event better by using more astute annotation schemes suitable for a novel. Character relationships like father-son, brother-sister, etc. could also be identified along with the character closeness parameters that is calculated in the project.

7 Appendix

The workload was equally shared among both of us. Ajinkya Upasani worked on Model 1, Model 4 and collection of Validation Dataset. Saikrishna Kalahasti worked on the Model 2, Model 3 and the Modified Bing's Sentiment Lexicon.

8 References

Adrian Groza and Lidia Corde. 2015. *Information retrieval in folktales using natural language processing.* arXiv:1511.03012v1 [cs.CL], IEEE.

David Bamman Ted Underwood Noah A.Smith. 2014. *A Bayesian Mixed Effects Model of Literary Character.* In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, USA.

Fei Liu Jeffrey Flanigan Sam Thomson Norman Sadeh Noah A. Smith. 2015. *Toward Abstractive Summarization Using Semantic Representations.* The 2015 Annual Conference of the North American Chapter of the ACL Denver, Colorado.

Shibhansh Dohare Vivek Gupta Harish Karnick. 2018. *Unsupervised Semantic Abstractive Summarization.* In Proceedings of ACL 2018, Student Research Workshop,Melbourne, Australia.

Snigdha Chaturvedi, Shashank Srivastava, Hal Dame, Chris Dyer. 2016. *Modeling Evolving Relationships Between Characters in Literary Novels.* In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence.

Niels Dekker, Tobias Kuhn, and Marieke van Erp. 2018. *Evaluating social network extraction for classic and modern fiction literature.* PeerJ https://doi.org/10.7287/peerj.preprints.27263v1

Hua He Denilson Barbosa Grzegorz Kondrak. 2013. *Identification of Speakers in Novels.* In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria.

Geertzen, J. 2012. *Inter-Rater Agreement with multiple raters and variables.* https://nlp-ml.io/jg/software/ira

Fleiss, J.L.. 1971. *Measuring nominal scale agreement among many raters.* Psychological Bulletin, 76:378-382.

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.

George A. Miller. 1995. *WordNet: A Lexical Database for English.* Communications of the ACM Vol. 38.