BACHELOR THESIS

Zuzana Urbanová

Quote Attribution and Character Networks in Novels

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis: Mgr. Rudolf Rosa, Ph.D.
Study programme: Computer Science
Study branch: General Computer Science

Prague 2021
I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ............. date ............. ....................................

...................                      .................

Author’s signature
I would like to thank Rudolf Rosa for weekly motivation and support.

I would like to thank David Jurgens for sending me their sources on Character Extraction that I could build on.

An most of all, I would like to thank Josef Svoboda for his valuable advices that I didn’t listen to until it was almost too late, for his unending fight against my urge to work without a break, and simply for the fact that he exists.
Title: Quote Attribution and Character Networks in Novels

Author: Zuzana Urbanová

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Rudolf Rosa, Ph.D., Institute of Formal and Applied Linguistics

Abstract: This thesis focuses on extracting information from literary works using tools for language analysis. Our goal is to automatically extract a conversational network of the characters in a novel. We divide the work into three subproblems and solve them separately: Character Extraction, Quote Attribution and Network Creation. The result is an end-to-end tool that gets a text of a novel in English and outputs a visual representation of the character network. Our work is based on existing literature. It presents new ideas and compares the accuracy of various methods for each subproblem.

Keywords: character networks NLP quote attribution
Contents

Introduction 3

1 Preliminary Problems 4
1.1 Word Tokenization and Parsing 4
1.2 Named Entity Recognition 5
1.3 Coreference Resolution 5
1.4 Gender and Animacy 6

2 Detecting Characters 7
2.1 What is a Character 7
2.1.1 Our Definition 8
2.2 Existing Approaches 8
2.2.1 Pipeline for Graph Edges 8
2.2.2 Features from Coreference Chains 9
2.3 Our Approach 9
2.3.1 Edge Types 10
2.3.2 Component Extraction 11

3 Quote Attribution 13
3.1 Definitions and Conventions 13
3.1.1 Definitions 13
3.1.2 One Speaker per Paragraph 13
3.1.3 Quote Detection 14
3.2 Existing Work 14
3.2.1 Supervised Machine Learning Approach 14
3.2.2 Deterministic Sieve Approach 15
3.3 Our Work 16
3.3.1 Differences to Existing Work 16
3.3.2 Quote-Mention 17
3.3.3 Mention-Speaker 18

4 Character Network 19
4.1 Overview and Terminology 19
4.2 Existing Work 20
4.2.1 Novel2Graph 20
4.2.2 Conversational Networks 20
4.3 Our Networks 22
4.3.1 Experiments 22
4.3.2 The Method 22
4.3.3 Difference to Existing Approaches 23

5 Implementation 24
5.1 Preprocessing and Tagging 24
5.1.1 Annotator 24
5.1.2 FalseAnnotator 25
5.2 Character Extraction 26
Introduction

There are more than 120 million different books in the world with 2 million new being published every year. With the average of reading 12 books per year, there is no way to go through even a tiny fraction of the pile in your lifetime.

The impossibility of reading every written work in the world leads researchers to invent more effective ways to extracting information from the books.

For novels, the goal can either be to provide a summary of the plot to let the reader choose to read what they desire, or to extract larger volumes of data to allow for studying differences and similarities of various works on a larger scale.

This thesis aims to extract the most basic information from novels: the list of characters. Using this list, we are heading for our ultimate goal of extracting the relations between the characters. Since the pillars of relationships are conversations, we also solve the task of attributing speakers to utterances in the books.

The first chapter serves as an introduction to Natural Language Processing and provides an overview of standard procedures and approaches to dealing with human language.

In the second chapter, we will look closely at the problem of detecting characters in literary texts. We will show not only how to detect their occurrences but also how to recognize that two names belong to one character. Simple as it may seem, it is not so easy to tell that Elizabeth Bennet, Lizzy, and Miss Eliza all refer to one person in Jane Austen’s Pride and Prejudice.

In the third chapter, we will describe how to connect characters to what they say. We will describe existing approaches to the problem of Quote Attribution and improve one of them.

The fourth chapter introduces the Character Networks. The Character Networks can show the strength of the relations between characters. By comparing the networks built with different types of interactions, we can get insight into the characters’ traits. What type of person can a character be whose name appears often in connection with others but who rarely talks to anyone?

The fifth chapter describes the technical implementation of our approach. The performance of our program is subsequently evaluated and compared to other methods in the sixth chapter.
1. Preliminary Problems

The goal of this work is to extract information from a text written in human language. Natural Language Processing, or NLP, has been studied for a long time. Many related problems have been identified and studied separately. In our work, we encounter some of these problems as well. We describe them in this chapter. It also serves as an introduction to NLP terminology.

We use the spacy library for solving most of the following problems. Some of the following approaches and definitions may differ when using other tools.

1.1 Word Tokenization and Parsing

The first step in processing a text is usually tokenization. This means breaking the text down into smaller units such as words, abbreviations, punctuation marks. Each of these units is called a token. Figure 1.1 shows an example of tokenization by spacy. Other tokenizers might consider tokens not as whole words but as shorter parts of the words, or add additional tokens to the beginning or the end of the text.

He said, “We’re going to N.Y.!”

A number of procedures can follow the tokenization. One of them is lemmatization. A Lemmatizer finds the base form (the lemma) of the tokens. For example, say is the lemma of saying.

Tokens can be tagged with part-of-speech labels, such as noun, verb, adjective, or more fine-grained labels such as proper noun singular, superlative adverb, past tense verb. The labels depend on the given context. For example, the word jump can be either a noun or a verb, depending on the neighboring tokens.

Dependency Parser finds the syntactic dependencies of the tokens. That is, the dependencies as a nominal subject or adverbial modifier. It also finds the parent token to which the token is connected. An example of a dependency parsed sentence is shown in Figure 1.2, as well as part-of-speech tags at every token.

Shortcuts are used for the dependencies by spacy: poss means possession modifier, nsubj is nominal subject, dobj is direct object and amod is adjectival modifier in Figure 1.2. We will not use these shortcuts in the following text, except for nsubj.

A noun chunk is a phrase that has a noun as its head. In the example of 1.2 the noun chunks are My sister and delicious cakes.

https://spacy.io/
1.2 Named Entity Recognition

Consider the following sentence:

Elon Musk, the founder of SpaceX, was born in South Africa.

A human reader can deduce that Elon Musk is a person, SpaceX is an organization, and South Africa a country. The task of computer recognition of these entities is called Named Entity Recognition, shortened as NER.

There can be various types of entities, depending on the particular system. The most common one is the PERSON entity, which we use in our work. Other types of entities include:

- ORG, names of organizations
- GPE, geopolitical entities such as countries, cities, states
- LOC, non-GPE locations, mountain ranges, bodies of water
- DATE, absolute or relative dates or periods of time
- WORK_OF_ART, titles of books, songs, etc.

1.3 Coreference Resolution

Coreference occurs when two expressions in the text refer to the same entity. For example, literary characters are usually introduced by their name and then referred to as he or she in the next few sentences. A coreference chain is a sequence of expressions referring to the same entity.

Coreference presents a big challenge for language processing; it often requires a deep understanding of the text. Consider the following pair of sentences [Levesque et al., 2012]:

The trophy doesn’t fit into the suitcase because it’s too large.

The trophy doesn’t fit into the suitcase because it’s too small.
In the first case, the \textit{it} refers to \textit{the trophy}. Since the trophy doesn’t fit into the suitcase, the trophy is the large item. In the second sentence which differs only in one word, the \textit{it} refers to \textit{the suitcase} because the suitcase must be small if the trophy doesn’t fit in it.

The Winograd schema challenge proposed by \cite{Levesque2012} provides many examples of such pairs of sentences.

Coreference resolution is crucial for both Character Detection and Quote Attribution. The accuracy of the coreference chains we find has a high impact on our performance on said tasks.

1.4 Gender and Animacy

The final section of this chapter introduces the \textit{gender} and the \textit{animacy} of nouns. Of course, nouns in English don’t have a grammatical gender themselves, such as in Czech or German. However, some nouns clearly represent a male person: \textit{archbishop, businessman, suitor}. Other nouns, on the contrary, represent female persons: \textit{abbess, nymph, queen}. The gender of a noun can be determined either by resolving the coreference to a pronoun \textit{he} or \textit{she}, or by using a predefined list of gendered nouns\footnote{We use the list from \url{https://github.com/ecmonsen/gendered_words} }.

Some nouns can represent an \textit{animate being}. We call those nouns \textit{animate}. All of the gendered nouns from the previous example are animate since they refer to a person. Animate nouns include also nouns referring to animals or mysterious creatures. Inanimate nouns are those that refer to things, places, or abstract concepts.

\footnotetext[2]{We use the list from \url{https://github.com/ecmonsen/gendered_words}}
2. Detecting Characters

Imagine that you want to process a novel automatically to extract some information about the story. What type of information would you start with? The list of literary characters that appear in the novel is perhaps among the top choices for most people.

Most literary characters have a name. Gregor Samsa, Hermione Granger or Samwise Gamgee – we know them well. These characters can be found with the help of the tools for Named Entity Recognition.

However, NER alone doesn’t suffice to recognize all characters. There can be characters that don’t have a proper name. Consider, for example, The Little Prince, Frankenstein’s monster, the quantities of unnamed Kings and Queens, or other characters who are determined rather by their appearances or occupation than the name.

On the other hand, some characters have more names than one. They can be called by their first name, last name, or both. They may also be called by multiple name variants, depending on who is talking to them and on what occasion. Without the context, it is hard to tell that Elizabeth Bennet, Lizzy and Miss Eliza all refer to one person in Jane Austen’s Pride and Prejudice.

2.1 What is a Character

To define the problem of detecting characters and be able to evaluate our approach, we must first define what a character is. Consider the following example (Jahan et al. [2020], slightly modified):

Once upon a time, a little girl named Mary and her dog Fido lived together in a small house. Mary and Fido were best friends. Mary played with Fido whenever she could. Fido helped Mary in her daily chores and brought letters to Mary from the post office. One sunny afternoon, Mary and Fido were walking through town. They saw a crowd gathered around a fruit vendor who was excitedly gesticulating some news. An ugly man just crossed the path to join the crowd. Suddenly, another dog barked at Fido.

Most people would agree that Mary and Fido are characters in this short story. However, it is not clear whether they are the only characters or not. They are not alone in the story: apart from them, there is also the fruit vendor, an ugly man, another dog, or the crowd. These are all animate beings (or, in the case of the crowd, a group of animate beings). Does it make them also characters?

Jahan et al. [2020] define a character as “an animate being that is important to the plot”. They further explain that we can test the animate being’s importance to the plot by asking if their actions or presence are critical to the progression of the plot.

However, this leads to another problem of defining what is critical and what is not. As Jahan et al. [2020] describe, main characters are easy to identify thanks to numerous mentions and close involvement in the main events of the plot. However, there is no ultimate decision boundary on what is a minor character.
and what is not a character at all. What one person might see as an important contribution to the plot, another person might find rather insignificant.

2.1.1 Our Definition
For the purpose of performing and evaluating the task of Character Detection, we present our own definition of a character.

*Character* is an animate being that meets one of the two following criteria:

1. It has one or more names (e.g. Gregor Samsa, Hermione, Mr. Bennet) and is referred to by one of its names at least three times.

2. It is referred to by an anaphor (e.g. the housekeeper, the queen; we don’t consider ambiguous anaphors such as her mother, that man) and the same anaphor is used at least three times.

2.2 Existing Approaches
We describe two of the recent approaches to detecting literary characters. In Section 2.2.1, we describe the approach of Vala et al. [2015], who connect different names of one character using a set of deterministic rules. Jahan et al. [2020] use a different approach described in Section 2.2.2. They identify coreference chains and train a model to recognize if the coreference chain represents a character.

2.2.1 Pipeline for Graph Edges
Vala et al. [2015] point out that some characters don’t get recognized by pure NER, and others get found multiple times under different names. To solve this problem, they propose an eight-stage pipeline. Throughout the pipeline, a graph is being built, where the nodes represent names and edges connect the names belonging to the same character. All the eight stages are as follows:

1. Use the names found by NER as vertices. Include also the names following an honorific (such as Mr. Bennet) which are not found by NER.

2. Resolve coreference and add an edge if the names occur in the same coreference chain.

3. Apply name variation rules (e.g. add edge where names differ just by omitting an honorific).

4. Use a gazetteer to link name variations (e.g. Tim and Timmy).

5. Prohibit merging of different characters by the following heuristics:

   (a) The inferred genders differ.

   (b) The names share a common surname but have different first names.

   (c) Honorifics of both names differ.
6. Prohibit merging of different characters through dependency parsing and quoted speech recognition if one of the following holds:

(a) Names are connected by a conjunction.
(b) One name appears as the speaker mentioning the other name.
(c) Both names appear together in a quotation.

7. Identify unnamed characters not recognized by NER and add them to the graph.

8. Remove nodes that are disconnected from the rest of the graph and represent a name that is a portion of another name (typically, those are ambiguous first or last names).

Stage 1 introduces the nodes. In stages 2–4, edges are added. Due to potentially connecting names of different characters, stages 5–6 on the contrary remove edges: if two nodes are prohibited from being connected, the edges along the shortest path between them get removed.

Stage 7 introduces characters not previously recognized by NER. This is done by recognizing character-like behavior. A list of verb-and-dependency pairs that are commonly associated with characters is extracted: the list of 2073 pairs includes verbs such as speak, cough, or forget, with a nominal subject being a character. If a noun referring to an animate entity is connected to one of these verbs by a corresponding dependency, it is added as a character.

The final stage removes family names and ambiguous first names that are not characters on their own. After the last stage, we get a graph where individual components represent all names of one character.

2.2.2 Features from Coreference Chains

Jahan et al. [2020] first use coreference resolution to find coreference chains, and keep only those that are marked as animate. Then, several features are obtained for each coreference chain, and a model is trained to predict whether the coreference chain represents a character or not. The features include the normalized coreference chain length and information whether the head of the coreference chain appears as a semantic subject to a verb, whether it is a Named Entity in the category person, or whether it is a descendent of Person in WordNet.

The approach by Jahan et al. [2020] is not optimal. It depends fully on the output of the animacy detector and coreference resolution. Also, the proposed features do not fully encompass the importance of the animate being to the plot. We believe that they indicate whether the coreference chain represents a person, rather than whether the person is really important. However, we will use the key idea of training a model to recognize characters in our approach.

2.3 Our Approach

We will adopt the idea of Vala et al. [2015] to create a graph where edges connect different names of the same character. Unlike Vala et al. [2015], we will use
multiple types of edges with various weights, instead of a single edge between two nodes. We will train a model to predict if the nodes refer to the same person, using different types of edges as features and their weights as the values.

Our approach consists of the following steps:

1. Add nodes to the graph, both for named entities and unnamed characters.
2. Add edges for entities connected by coreference and edges for unconnected entities. Add name-variant and dependency edges.
3. Predict which edges remain in the graph, based on the features.
4. Remove the least probable edges from paths connecting nodes that should not be connected.
5. Extract the connected components.

The first step is similar to Vala et al. [2015]. However, we also add unnamed character nodes already in the first step. To find the unnamed characters, we use the list of verb-and-dependency pairs provided by Vala et al. [2015]. If a noun chunk appears at least three times connected to one of the verbs in the list by a proper dependency, we add it as a character.

The second step adds edges of various types and weights and is described in detail in the next subsection. In the third step, we predict whether the nodes correspond to the same character, according to our trained model. The fourth step removes edges from large components and is described in Section 2.3.2.

After the steps 1-4, we obtain a graph whose connected components correspond to individual characters. The final step outputs the characters, which can be either used in the next module, Quote Attribution, or evaluated on its own.

### 2.3.1 Edge Types

Our graph contains multiple edge types, the complete list is in Table 2.1. We use them as the features of the model; the weights are their values.

Unlike Vala et al. [2015], we consider not only cases when two nodes are connected by coreference, but also the cases when they are not. In coreference resolution, we use a sliding window approach: we resolve coreference for shorter parts of the novel and overlap these parts to gain higher precision. It might therefore happen that we connect the entities in one window but in another window, we do not. We believe that using both types of edges adds valuable information for the model and might make up for the coreference inaccuracy.

The gender of the characters is derived in two ways: First, we count how many times a node was connected by coreference to a male pronoun and how many times to a female one. The majority then determines the gender. Second, we use a list of common male and female names to guess the gender of the character if its first name is detected. Then, we compare both derived genders between both nodes and add a *Same Gender* edge for each agreement, and *Different Gender* edge for each difference.

The edges *Honorific Differ*, *Name Subset*, *First Name Variants*, and *Names Differ* have binary value and are only applicable for nodes found by NER that
Table 2.1: Edge types

<table>
<thead>
<tr>
<th>Edge type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coreference (connected)</td>
<td>Number of occurrences of the two nodes in the same coreference cluster</td>
</tr>
<tr>
<td>Coreference (unconnected)</td>
<td>Number of occurrences of the two nodes in different coreference clusters</td>
</tr>
<tr>
<td>Same Gender</td>
<td>The inferred genders are equal</td>
</tr>
<tr>
<td>Different Gender</td>
<td>The inferred genders are different</td>
</tr>
<tr>
<td>Connected by Conjunction</td>
<td>The names are connected by conjunction in the novel text</td>
</tr>
<tr>
<td>Honorific Differ</td>
<td>Both names contain honorific and they are different</td>
</tr>
<tr>
<td>Name Subset</td>
<td>One name is a subset of the other, e.g., a honorific is omitted</td>
</tr>
<tr>
<td>First Name Variants</td>
<td>The first names are variants of the same name (using a gazetteer obtained from [Vala et al. [2015]])</td>
</tr>
<tr>
<td>Names Differ</td>
<td>Both names contain a first name and the first names differ, or both names contain a last name and they differ</td>
</tr>
<tr>
<td>Name Substring</td>
<td>One name is a substring of the other (can be used also if the parts of the name are not correctly parsed)</td>
</tr>
</tbody>
</table>

can be correctly parsed to an honorific, first and last name. The edge Name Substring is similar to Name Subset but is applied also to the nodes representing unnamed characters.

2.3.2 Component Extraction

We use a trained model to predict whether an edge connects two names of one character or two different characters. We add all edges with the predicted probability of at least 0.5 to the graph. However, the components of such a graph cannot yet be declared as characters. An example of such an incorrect component is Mr. Philips – Philips – Mrs. Philips. There is a high probability that Philips and Mr. Philips refer to the same character because the names differ only by missing honorific. The same holds for Philips and Mrs. Philips. However, Mr. and Mrs. Philips are clearly different characters.

To prevent these errors, we examine every component and if there are two nodes with the probability of being connected less than 0.1, we remove the least probable edge on the shortest path between these two nodes (if the probability of the edge is not greater than 0.9).

Figure 2.1 shows the largest component of Pride and Prejudice before and after the edge removal. In the first graph, all edges are shown and their probability is expressed by the edge color. In the second graph, only the edges that have remained in the graph are shown, and the components of this graph represent the final detected characters.
Figure 2.1: The largest component before and after the edge removal
3. Quote Attribution

The goal of Quote Attribution lies in identifying the speaker of each utterance of direct speech.

This task is closely related to the processing of literary texts. Once the speaker of each utterance is recognized, we can build conversational networks [Labatut and Bost, 2019] and study relations between the characters. Quote Attribution can also be applied on its own, for example, to automatically generate audiobooks where each character is distinguished by their tone of voice [He et al., 2013].

3.1 Definitions and Conventions

In this section, we introduce the terminology used in this chapter and our conventions for the detection of the quotes.

3.1.1 Definitions

**Utterance** An utterance is a connected text that can be attributed to a single speaker [He et al., 2013]. Sometimes, a speaker tag is inserted into the middle of an utterance, as in the following example (from *Pride and Prejudice* by Jane Austen):

“Dining out,” said Mrs. Bennet, “that is very unlucky.”

In such case, we consider both parts as a single utterance. We use the noun *quote* as a synonym to *utterance*.

A paragraph without any utterances is called a *narrative*. A series of utterances together with related narratives is called a *dialogue* [He et al., 2013].

**Mention, Speaker, Entity** A mention is a hint that allows the reader to recognize the speaker of an utterance. There are several types of mentions, identified by Muzny et al. [2017] and shown in Table 3.1. A mention is resolved to an entity, which is our representation of a character. In our case, an entity is a set of names and a gender, belonging to one character.

3.1.2 One Speaker per Paragraph

We assume that every utterance within a paragraph belongs to the same speaker. This is a correct assumption for most novels: He et al. [2013] identified only five different cases in *Pride and Prejudice*, which were usually caused by one character citing another or reading a letter that contains quotations. Muzny et al. [2017] point out *The Steppe* as a counterexample that uses more complex conversational structures. The example provided by Muzny et al. [2017] contains a character who tells a story and cites other characters while speaking. Muzny et al. [2017] identify the cited speech as an utterance by the cited character. However, we consider the nested speech as told only by the storyteller because he is the one who says the words aloud now, even though they have been originally said by someone else.
Table 3.1: Types of speaker mentions [Muzny et al., 2017]

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>“Money! My uncle!” cried Jane, “what do you mean, sir?”</td>
<td>Jane Bennet</td>
</tr>
<tr>
<td>Anaphoric (pronoun)</td>
<td>“You are uniformly charming!” cried he, with an air of awkward gallantry;</td>
<td>Mr. Collins</td>
</tr>
<tr>
<td>Anaphoric (other)</td>
<td>“I see your design, Bingley,” said his friend.</td>
<td>Mr. Darcy</td>
</tr>
<tr>
<td>Implicit</td>
<td>“How long did you say he was at Rosings?”</td>
<td>Mr. Wickham, Elizabeth Bennet</td>
</tr>
<tr>
<td></td>
<td>“Nearly three weeks.”</td>
<td>Mr. Wickham, Elizabeth Bennet</td>
</tr>
<tr>
<td></td>
<td>“And you saw him frequently?”</td>
<td>Mr. Wickham, Elizabeth Bennet</td>
</tr>
<tr>
<td></td>
<td>“Yes, almost every day.”</td>
<td>Mr. Wickham, Elizabeth Bennet</td>
</tr>
</tbody>
</table>

We found 14 examples of such nested conversations in *The Steppe* and five more paragraphs with utterances by more than one character. Out of 460 paragraphs with any utterances at all, this is a negligible amount and our assumption is justified by it.

3.1.3 Quote Detection

To attribute speakers to utterances, we must first detect the utterances.

Direct speech is usually marked by quotation marks. They can, however, also be used as scare quotes, or to quote someone else’s speech in the narrative.

Apart from that, different quoting styles are used in American and British English. The former uses double quotes (“...”), while single quotes are preferred in the latter (‘...’) [Trask, 1997]. In both American and British English, opening and closing quotes are different. However, in our data (books from Project Gutenberg1), straight quotes are used for both the opening and closing part (’ or “).

In this work, we consider only double quotes (“...” or "...") to mark direct speech. We also assume that every text in one paragraph between two double quotation marks is a direct speech.

3.2 Existing Work

We will describe the work of [He et al., 2013] and [Muzny et al., 2017]. Both authors use a predefined list of characters including all the names by which they can be referred, to identify speakers. [He et al., 2013] developed a trainable model with a ranking approach, predicting the probability of every possible speaker. [Muzny et al., 2017] on the other hand adopt a deterministic approach and yield similar results.

3.2.1 Supervised Machine Learning Approach

[He et al., 2013] created the state-of-the-art system for quote attribution. First, they extract explicit speakers by focusing on speech verbs as say, speak, reply, add

1https://www.gutenberg.org/
that appear before, after, or between quotations. The speaker’s name is extracted from the dependency relation \texttt{nsubj} to the verb. If the speaker is anaphoric, their gender can be inferred in some cases, for example from the phrase \textit{she said} we know that the speaker is female. If the speaker is not clear yet, it is predicted based on the features we will describe shortly.

\textit{He et al.} [2013] note that the task is difficult to model as a sequential prediction because the set of tags (speakers) is different for every novel. Therefore, they propose a ranking approach. A set of candidate speakers is chosen and each speaker is ranked individually.

\textit{He et al.} [2013] introduce the speaker alternation pattern by making two observations: First, the speakers of consecutive utterances are usually different. Second, the speaker of the \( n \)-th utterance is likely to be the same as the speaker of the \( (n-2) \)-th utterance.

Based on their observations, \textit{He et al.} [2013] propose the set of features for the ranking model as shown in Table 3.2. Beside others, the features represent whether there is a vocative speaker name or the speaker name in utterance. For example, the speaker of the utterance “I hope Mr. Bingley will like it, Lizzy.” is neither Mr. Bingley, nor Lizzy. However, the vocative of Lizzy indicates that she is likely to be the speaker of the next utterance. They also include a supervised actor-topic model that predicts the most likely speaker by considering the content of an utterance.

Their model \texttt{NEIGHBORS}, which incorporates not only features from the utterance to be predicted but also from the neighboring utterances, reaches an accuracy of 82.5%, 74.8%, and 80.3%, respectively on the novels \textit{Pride and Prejudice}, \textit{Emma} and \textit{The Steppe}.

### 3.2.2 Deterministic Sieve Approach

\textit{Muzny et al.} [2017] use a two-step process to determine the speaker. First, they identify the mention that corresponds to the speaker of the quote. Then, they resolve the mention to an entity.

Both steps are done deterministically, using a series of sieves. The key idea is to identify the speakers in the easiest cases first and use the already identified speakers in the neighboring utterances to resolve more challenging utterances.

The sieves from the first step are described in Table 3.3.

In the second step, \textit{Muzny et al.} [2017] construct a list of speakers which appear around the target quote, ordered from the most common speaker. Then, they match the mention to one of the speakers in the list using one of \texttt{Exact Name Match}, \texttt{Coreference Disambiguation}, \texttt{Conversational Pattern}, or \texttt{Family Noun Vocative Disambiguation}. If no other speaker is attributed to the mention, the \texttt{Majority Speaker} is chosen from the list of speakers. The only unintuitive sieve
Detected mention

Mentions in patterns like Quote-Mention-Verb, e.g. “...,” she said.

An animate noun that is nsubj to a common speech verb (e.g. say, cry, reply).

If there is only a single mention in the non-quote text in the paragraph.

The vocative in the preceeding quote.

Final mention in the preceeding sentence.

If the n-th and the two preceeding paragraphs are quotes, links the mention to the one in the (n − 2)-th paragraph.

The same as above without the restriction on the paragraphs being quotes.

Table 3.3: Sieves in mention detection by Muzny et al. [2017]

is the Family Noun Vocative Disambiguation. This sieve is used if the mention is a vocative of a family relation (e.g. “papa”). Then, the first speaker that matches the last name of the previous speaker is chosen from the list of speakers.

The results are comparable to the results of He et al. [2013]. The accuracy on the same novels Pride and Prejudice, Emma and The Steppe is 85.1%, 75.9% and 72.9%, respectively.

3.3 Our Work

In this section, we will describe our approach to the task of Quote attribution.

We use a deterministic sieve approach, following Muzny et al. [2017]. Since we attribute speakers to whole paragraphs instead of individual utterances, we had to modify some of the sieves to reflect the differences. We also introduce two new sieves. All the sieves we use are summarized in Table 3.4.

3.3.1 Differences to Existing Work

Contrary to both of the cited papers, we do not rely on a predefined character list but extract the characters automatically. This, of course, introduces some errors. However, our goal is to create a tool that can extract information from novels without requiring user interaction at all.

Moreover, our assumption is that main characters are easier to recognize than the minor ones since they appear most frequently in the novel, and they are attributed most utterances (in Pride and Prejudice, 617 out of 1154 utterances are uttered by only three characters, as labeled by Muzny et al. [2017], and 90% of the quotes are attributed to only 12 characters – compare to the 73 characters in total identified by Vala et al. [2015]). Therefore, the errors in recognizing minor characters should not have a great impact on the overall performance.

We succeeded in our goal of creating a fully automatic tool and even surpassed the results of Muzny et al. [2017] on The Steppe. The results are analyzed in Section 6.2.
3.3.2 Quote-Mention

The first task we had to solve was how to detect mentions. Muzny et al. [2017] don’t elaborate on how to decide if a string is a mention. We consider a string to be a mention if it is contained in the list of character names, if it is linked by coreference to a character name, if it is a nominative third-person pronoun, or if it is a common mention such as man, sister, boy.

Conversational Sieves  We noticed that the Conversational Pattern sieve is unnecessary because all mentions that it finds are also found by the Loose Conversational Pattern sieve. However, we decided to keep it to be able to test the sieves individually and experiment with turning the sieves on and off.

We also noticed that the Mention-Speaker Conversational Pattern sieve partially substitutes the two conversational sieves in Quote-Mention. Instead of attributing the mention following the conversational pattern, it attributes directly the speaker. However, It is not completely equal to use the conversational pattern on mentions and on speakers. For example, if the mention is anaphoric and has to be resolved using the list of top speakers, it can be resolved differently when different speakers lists are used on the same mention. If we skipped the conversational sieves in Quote-Mention, the same speaker would be attributed in all cases. We decided to keep all the sieves in our program and test their performance. The results are presented in Section 6.2.3.

<table>
<thead>
<tr>
<th>Quote-Mention Sieves</th>
<th>Mention-Speaker Sieves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigram Matching</td>
<td>Exact Name Match</td>
</tr>
<tr>
<td>Determined by Previous Paragraph</td>
<td>Coreference</td>
</tr>
<tr>
<td>Common Speech Verb</td>
<td>Conversational Pattern</td>
</tr>
<tr>
<td>Single Mention</td>
<td>Closest Name Before</td>
</tr>
<tr>
<td>Vocative in Previous Paragraph</td>
<td>Majority Speaker</td>
</tr>
<tr>
<td>Final Mention</td>
<td></td>
</tr>
<tr>
<td>Conversational Pattern</td>
<td></td>
</tr>
<tr>
<td>Loose Conversational Pattern</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4: Our sieves, including the two novel sieves

Previous Paragraph Sieve  We created a new sieve which recognizes a mention if the speaker of the next utterance has been stated in the previous paragraph. Consider the following sentence (from The Steppe by Anton Chekhov):

* Dymov not very readily raised himself on his elbow and said: *

The reader understands quickly that the utterance that follows is said by Dymov. The utterance is anticipated due to the final colon. Our sieve checks the last sentence of the previous paragraph and if it ends with a colon, it marks the subject of this sentence as the mention. We place this sieve right after the Trigram Matching because we expect this to be quite precise way of finding mentions. In The Steppe, we found only one case out of more than 50 sentences ending with a colon where the subject of the sentence was not the correct mention.
to the utterance that followed. The one case was this: “From somewhere floated
the mournful cry of the bird:”. In this particular case, the sentence subject cry is
not considered a mention. Even if we cannot correctly identify the mention the
bird, we do not introduce errors by choosing a wrong mention.

All the sieves are summarized in Table 3.4.

3.3.3 Mention-Speaker

In the second part of our approach, we omit the Family Noun Vocative Disam-
biguation sieve that Muzny et al. [2017] use. We believe that this sieve can be
useful for 19th century English romantic novels such as Pride and Prejudice in
some cases but not so much useful for most other novels. Even Pride and Prej-
udice contains family noun vocatives rarely; and our other sieves are likely to
resolve the correct speaker anyway.

We propose another sieve instead: Closest Name Before. This sieve finds the
closest name before the given quote within two previous paragraphs. We place
this sieve in the penultimate position, as shown in Table 3.4. We wanted to test
whether this would be more precise than the Majority Speaker sieve. This did
not prove to be true. However, neither did the opposite. Using the Closest Name
Before yields similar results to not using it.

The last three sieves Conversational Pattern, Closest Name Before, and Ma-
jority Speaker assign the speaker only if it is different from the speaker before
and after this quote. We follow the observations of He et al. [2013] that the
neighboring speakers are likely to be different.

For the list of top speakers, we tried to use different metrics than Muzny et al.
[2017]. While they count the speakers only by the number of their occurrences in
the range from 2000 tokens before to 500 tokens after the quote, we considered
also how far the occurrences were, so that nearer occurrences would have greater
weight than the more distant. We tried to use the weights \( \frac{1}{\text{distance}(x, \text{quote})} \) and
2000 – distance(x, quote). However, none of these metrics worked better than
pure counting of occurrences.
4. Character Network

Character Network is a graph that represents characters by its nodes and interactions between them by its edges [Labatut and Bost, 2019]. The survey by Labatut and Bost [2019] cites more than 300 articles that are related to creating Character Networks out of various types of media (novels, plays, movies, TV series). In this section, we will present an overview of the task of creating Character Networks, describe two existing approaches, and introduce our method.

4.1 Overview and Terminology

An overview of a generic Character Network extraction process is illustrated in Figure 4.1 [Labatut and Bost, 2019]. It consists of three parts: identifying characters, detecting interactions, and extracting the graph. In our work, we first identify characters as well. Detecting interactions is partially done by Quote Attribution in our case but we describe also another approach in Section 4.3.1.

Labatut and Bost [2019] identify multiple types of character interactions. They are described in Table 4.1. Networks created by using co-occurrence are called co-occurrence networks, networks that use conversations are called conversational networks.

The networks can be static or dynamic. Static networks represent interactions between the characters in the whole narrative at once, whereas dynamic networks show interactions during specific temporal windows and change over time.

Character networks can be used to test literary theories, detect roles of individual characters, recommend similar books to the users, or to generate new stories [Labatut and Bost, 2019].

Figure 4.1: The process of Character Network extraction [Labatut and Bost, 2019]
<table>
<thead>
<tr>
<th>Type of interaction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-occurrences</td>
<td>Characters occur in the same narrative unit (e.g. sentence, paragraph, chapter).</td>
</tr>
<tr>
<td>Conversations</td>
<td>One character talks to the other (requires also identification of the addressee; usually, the addressee is assumed to be the next speaker).</td>
</tr>
<tr>
<td>Mentions</td>
<td>A character talks about another character.</td>
</tr>
<tr>
<td>Direct actions</td>
<td>One character performs an action on another (e.g. thinking about someone), or two characters perform an action jointly (e.g. fighting).</td>
</tr>
<tr>
<td>Affiliations</td>
<td>Relations as being blood-related, being married, belonging to the same social group.</td>
</tr>
</tbody>
</table>

Table 4.1: Types of character interactions [Labatut and Bost, 2019]

4.2 Existing Work

In this section, we will describe two existing approaches to creating character networks. Kanjirangat and Antonucci [2019] create co-occurrence networks, while Elson et al. [2010] build conversational networks.

4.2.1 Novel2Graph

Kanjirangat and Antonucci [2019] focus on extracting the sentiment of characters and their relations (that is, whether they are perceived positively, or negatively). They create a tool called Novel2Graph which outputs a graphical summary of the plot out of the input text. Figure 4.2 shows the output on the fifth book of the Harry Potter series.

Kanjirangat and Antonucci [2019] identify the characters by using the Stanford Named Entity Recognizer and cluster them using the DBSCAN clustering algorithm to group together all aliases of the same character.

They extract phrases in which a character or a pair of characters occur. Then, they identify the positive or negative sentiment in these phrases, and use it to color the output nodes and edges blue or red. They also aim to extract relation types, such as friends, enemies, family. However, they currently focus only on the sibling relations, which are denoted by dashed edges in Figure 4.2.

4.2.2 Conversational Networks

The approach of Elson et al. [2010] is similar to ours. First, they identify characters using NER and cluster them together using generated name variants (for example, names created by keeping only the first name or omitting the honorific from the full name). They develop their own system for quote attribution as described in their other article, Elson and McKeown [2010]. They annotate their own corpus with the golden speakers. However, Muzny et al. [2017] find that 57.8% of the quotes in the corpus have no speaker label or the speaker cannot be linked to a known character.

[https://nlp.stanford.edu/software/CRF-NER.shtml](https://nlp.stanford.edu/software/CRF-NER.shtml)
Figure 4.2: NOVEL2GRAPH output of *Harry Potter and the Order of the Phoenix* [Kanjirangat and Antonucci, 2019]
Elson et al. [2010] focus on validating two hypotheses from literary theory on the 19th-century British novels through the networks.

4.3 Our Networks

We create a conversational network where the weight of an edge is the number of cases when the first character speaks directly after the second one or vice versa. The detailed description is provided in Section 4.3.2. We propose two experiments with the networks in Section 4.3.1 In Section 6.3 the evaluation can be found.

4.3.1 Experiments

We propose two experiments with the networks.

**Conversation vs. Co-occurrence** We will create a co-occurrence network next to the conversational network. We will compare the two networks and determine whether the differences can be justified in the story. We will also compare the conversational network we created using our extracted characters to the conversational network that uses the gold quote labels by Muzny et al. [2017].

**Gender Bias** We will create a conversational network that considers the character genders. This allows us to compare whether there are more men or women talking to each other in the novel. Gender equality is a popular topic in these days and is being demanded in many spheres of life. This experiment shows whether books are also biased against women.

The results of these experiments are discussed in Section 6.3. The visualizations of the generated networks can be seen there as well. We include more examples of networks in the attachment. The networks generated for the first experiment can be found in Attachment A.1 and A.2. The networks generated for the second experiment are in Attachment A.3.

4.3.2 The Method

In this section, we will describe the creation of the conversational networks using the characters we extracted. We will also describe possible variants to this approach. The approach is as follows:

1. Add all the extracted characters as nodes to the graph.
2. Add the edges, using the extracted speakers from Quote Attribution.
3. Reduce the number of characters.
4. Normalize edge widths and visualize the network.

In the first experiment proposed in the previous section, we include two variants of the second step: we use the golden speakers instead of the ones we find, or we add the edges using co-occurrence instead of conversation.
For co-occurrence, we add an edge between two characters for every paragraph where both characters appear. The characters are taken from the Character Detection module: it marks the character id of every token that refers to a character either by the name, or by coreference.

For the interactions, directed edges could be used instead of undirected (e.g. Moretti [2020]). However, we decided to keep the edges undirected because we wanted the network to be more simple. Also, the undirected edges seem to be easier to visualize.

Reducing the number of characters in the third step is done because a large network can be confusing and conceal some information that could be seen when looking at a smaller one. We decided to limit the number of characters in the network to 20. This number is large enough to show the most interesting relations, and small enough to not get lost in the network. If there is more than 20 characters in the network, we keep only the characters that have the most interactions. That is, the nodes with the largest sum of the weights of the adjacent edges.

In the final step, we normalize the edge widths. The number of interactions usually depends on the length of the book. There are more interactions in total in longer books than in the shorter ones. To avoid too thick edges in the networks of longer books and empty-looking networks of shorter books, we let the heaviest edge have a fixed width, the same for every network, and scale other edges proportionally.

We label the nodes of the network by the character names. If a character has more names, we use the name that appears most often in the book.

The created graphs are shown in Section 6.3 and Attachments A.1, A.2 and A.3.

4.3.3 Difference to Existing Approaches

The main difference of our approach to the approaches that already exist is that our tool is fully automatic. Most existing automatic tools build rather co-occurrence networks than conversational ones (such as Kanjirangat and Antonucci [2019]). Falk [2016] for example, builds co-occurrence networks but proceeds manually and collects the data by hand.

Elson et al. [2010] build conversational networks automatically. However, we believe that our approach might be more precise due to improving the performance of the first two steps (Character Extraction and Quote Attribution).
5. Implementation

The content of this thesis consists of three main tasks: Character Extraction, Quote Attribution, and Network Creation. The individual tasks are mostly independent of each other, we thus created an individual module for each of these tasks. This also allows us to improve each module independently, or to change the implementation easily. We will describe each module in one of the following sections.

The data flow between the components can be seen in Figure 5.1. The key data structure is a list of docs, which is vital for all the three main components. The docs are created in the Annotator, as described in Section 5.1.1. Every doc contains the text of one paragraph, as well as many attributes added along the way.

For the implementation, we chose python. It is one of the most widespread languages in Natural Language Processing, and the availability of various libraries makes it easy to use.

We use the spacy library for most of the text processing. It provides a variety of functions for Tokenization, Named Entity Recognition, and is extensible with custom components and attributes.

5.1 Preprocessing and Tagging

The input to the program comes in the form of a file that contains the text of a novel. The texts that we worked with come from Project Gutenberg. The paragraphs of these texts are broken into multiple lines to not exceed the maximum line length, and separated by a blank line. Therefore, we require all input files to have paragraphs separated by a blank line as well. We extract all paragraphs to one list and hand the list of paragraphs to an Annotator.

5.1.1 Annotator

The class Annotator first loads a spacy language model en_core_web_trf and runs a pipeline on the list of paragraphs. The pipeline contains standard spacy modules such as Tokenizer, Lemmatizer, or Named Entity Recognizer, and two of our own modules which are described later. The pipeline outputs a doc out of each paragraph. Each doc is basically the tokenized paragraph. Each token in the doc contains attributes that were added in the pipeline modules: the Named Entity Recognizer adds attributes determining whether the token is inside or outside a named entity, the Lemmatizer adds the token’s lemma. There can also be attributes and properties for the whole doc.

The spacy’s Named Entity Recognizer recognizes only names; if a name contains also an honorific, such as Mr. Bennet or Miss Elizabeth, it doesn’t bind the honorific to the entity. However, the honorifics are often very important for character recognition: it allows us to distinguish Mr. Bennet from his wife, for example. Our pipeline module entity_modifier therefore modifies the entities previously found by Named Entity Recognizer to contain also honorifics from a
predefined list. It also adds attributes to the tokens, determining the gender of some words.

Our second module, quote parser, finds sections of direct speech in the paragraphs and marks them in the doc attributes.

After the docs are created, we run a coreference resolution on it. We use the coreference model introduced by Toshniwal et al. [2020]. The given maximum token length for this model is 512, so we use it on smaller slices of the docs list instead of on the whole book. We didn’t encounter any errors when using the model for longer sequences. Therefore, we use this limit loosely. The coreference model uses different tokenization than our pipeline and we had to find a mapping between both representations.

The last task of the Annotator is to find characters that are not found by the NER. We use a list of verbs-and-dependency pairs obtained from Vala et al. [2015] that denote behavior typical for characters. We create a list of nouns that can possibly be unnamed characters. We check the animacy of the proposed nouns via WordNet [Fellbaum, 1998], by checking if the word organism is a hyperonym of the given word. If the noun is animate and occurs at least three times together with one of the verbs typical for characters, we mark it as an entity of type NAMELESS_CHAR. Finally, if the novel is narrated by a first-person narrator, we mark all singular, first-person pronouns that are not inside a direct speech as an entity of type NARRATOR.

5.1.2 FalseAnnotator

The coreference model by Toshniwal et al. [2020] is said to run in linear time in the length of the document. However, the absolute runtime is quite long. We witness
the average runtime of about 6 s per chunk of paragraphs with the total length not exceeding 512 tokens. With the overlapping windows, the total runtime reaches tens of minutes by longer novels. To not have to annotate each novel multiple times, we can save the annotated docs to a file, and retrieve them later with a FalseAnnotator. This class retrieves the docs from a DocBin file (DocBin is spacy’s container for serialization of docs) and sets the correct extensions on Tokens, Spans and Docs that the Annotator otherwise sets.

5.2 Character Extraction

The main component of Character extraction is the class CharacterExtractor. The method extractCharacters provides a high-level overview of the underlying procedures. First, the CharacterUnificationGraph creates an initial graph with the edges being features for the model. Then, edge probabilities are predicted by the CharacterUnificationModel, and the final graph is created by the CharacterExtractor, along with a complete list of characters.

5.2.1 Character Graph

The purpose of the class CharacterUnificationGraph is to create a complete graph of named entities and collect the features for edges to determine which edges connect two names of the same character.

The method createGraph is the main method of this class and it builds the graph through other functions. First, it adds nodes to the graph. The nodes can be of three types, according to the type of the entity from which this graph is created: PERSON, NAMELESS_CHAR, or NARRATOR. Entities of the type PERSON are parsed by NameParser to find the first name, last name, and determine the person’s gender.

Then, different types of edges are added to the graph, as described in Section 2.3.1. The graph is then passed back to CharacterExtractor.

5.2.2 Unification Model

The class CharacterUnificationModel is responsible for predicting the probability that an edge connects two different names of one character.

This class is also responsible for obtaining the model. If we want to train the model on the data we previously created by CharacterExtractor, this class retrieves the data from files and trains the model.

The model we use is the MLPClassifier from the scikit-learn library. We use this model with the default settings.

5.2.3 Character Extractor

We have already described the creation of the initial character graph. The method extractCharacters of the class CharacterExtractor continues by creating the final graph where the connected components will denote individual characters. First, all nodes from the initial graph are copied to the final graph. Then, probabilities for edges are predicted through the CharacterUnificationModel, and all
edges with the predicted probability of at least 0.5 are added to the graph. Edges on the shortest path between two nodes that are connected with the probability of less than 0.1 are removed from the graph, as described in Section 2.3.2.

We now have the character extracted. But this is not the end: we must check if the characters meet the definition we set in Section 2.1. If the total number of occurrences of all names in a component is less than 3, we delete this component.

After the component extraction, we rename some nodes. Specifically, the NAMELESS_CHAR nodes have been represented as a sole noun throughout the process. We rename these characters to the most common noun chunks. For example, in the book The Little Prince, we so find the character the little prince instead of prince. Also, we merge all NARRATOR entities to a single narrator character.

The final task of the method extractCharacters is to mark character ids of the extracted characters in the token attributes. This simplifies coreference resolution later in the Quote attribution.

The other significant method of CharacterExtractor is saveWeights. This method only creates the initial graph with features for the model and instead of proceeding with the character extraction, it saves the weights to a file from which they can be later loaded for the model training.

To allow for evaluation of the Quote Attribution with the gold standard set of characters, we created the FalseCharacterExtractor. This class takes the set of characters from a file and only adds the correct attributes to the tokens, without extracting our own characters.

### 5.3 Quote Attribution

The task of Quote Attribution is covered by the main class QuoteAttributor. This class joins the two steps of Quote Attribution: the attribution of mentions to quotes, which is done by the class QuoteMention, and the resolution of mentions to speaker entities, which is done by the class MentionSpeaker.

#### 5.3.1 Quote-Mention

The class QuoteMention contains the main method solveMentions. This method creates the eight sieves and runs the sieves one after another on the docs.

Each sieve is a class derived from the main class MentionSieve. This class contains functions and attributes that are common for multiple sieves. The functions include isMention, which decides if a given text can be a mention, or getSubjIndices, which returns the indices of the syntactic subject bound to a given verb. MentionSieve also contains some attributes, such as the list of common speech verbs by Muzny et al. [2017] which we extended by six more verbs.

Each sieve contains the function run. This function takes a doc and its index in the list as an argument. If the sieve finds the correct mention, it is marked in the doc attribute mention. Note that this mention can originate from a previous doc, as is the case in the sieves Previous Vocative Detection or Conversational Pattern.
The sieves share a common calling procedure in MentionSieve. The method iterates through the docs and calls the function run on each doc that contains direct speech but doesn’t have assigned any mention yet. If the sieve assigns a mention to the doc, the name of the sieve is stored in the doc attribute mention_sieve.

5.3.2 Mention-Speaker

The class MentionSpeaker is mostly similar to the class QuoteMention. It creates sieves of type QuoteSieve, and runs the sieves one after another on the docs.

Apart from that, MentionSpeaker contains the method makeSpeakersLists that creates the lists of the names mentioned around every doc, sorted from the name with the most mentions. This method is called before calling the first sieve.

After running through all the sieves, every doc has assigned a speaker_id and the speaker_sieve as an attribute.

5.3.3 False Quote Attributor

The class FalseQuoteAttributor doesn’t assign the golden speakers instead of the extracted ones, such as FalseCharacterExtractor does. It calls the original class CharacterExtractor to extract the speakers and adds the golden speakers additionally to the attribute gold_speaker. The gold_speaker attribute contains a string name of the correct speaker and the gold_match_id contains the id of the corresponding character that we extracted.

The extraction of the golden speakers requires complicated parsing of the annotated file. This is already done by the QuotesEvaluator, as described in Section 5.5.2. Therefore, the QuotesEvaluator is created for the attribution of the gold speakers by the FalseQuoteAttributor as well.

5.4 Network Creation

The Network Creation is done by the class NetworkCreator. It has two separate functions for creating networks. The first one is createNetworks that creates a co-occurrence and a conversational network and, if golden data is provided, also a conversational network that uses the golden characters. Creating all three networks at once is necessary because we want them to have the same nodes at the same positions to be able to compare them visually.

The second function for creating networks is createGenderNetwork which, as the name suggests, creates a network that allows us to observe gender biases.

5.4.1 Outputting the Networks

The NetworkCreator outputs the networks in the form of a networkx graph. The file out_formatter contains a set of functions to output neatly looking images of the networks. For the outputting of the co-occurrence and conversational network, it uses the function outputThreeNetworks. The function outputGender takes care of outputting the gender network.
5.5 Evaluator

The biggest challenge of the evaluation was to parse annotated data from different sources. We will describe the evaluation of Character Detection and the evaluation of Quote Attribution separately.

5.5.1 Characters

The class CharacterEvaluator evaluates the characters that we found. It has two derived classes, one for evaluation in each metric as described in Section 6.1.1. Both the derived classes contain functions getPrecision and getRecall that compute the precision and the recall in the given metric.

The classes can also parse the lists of characters from a file in a csv format as set up first by Vala et al. [2015].

5.5.2 Quotes

The Quote Attribution is evaluated by the QuotesEvaluator. This class has two derived classes as well: One for the evaluation using an annotated novel from the QuoteLi3 corpus by Muzny et al. [2017] and one for the evaluation using a novel from the Colombia Quoted Speech Corpus (CQSC) by Elson and McKeown [2010]. However, we decided afterwards to not use CQSC in the evaluation. Each corpus uses a different labeling of the quotes. Moreover, we only attribute speakers to whole paragraphs, whereas every single quote is labeled in both corpuses.

The function parse parses the annotated file and splits it to paragraphs. If the paragraph contains more quotes, we choose the speaker with the most quotes as the true speaker of this paragraph. The function evaluate computes the accuracy as explained in Section 6.2.2.

5.6 Modes of Execution

Our program has multiple modes of execution which can be accessed through command line arguments. The default mode is run; this mode annotates the input file (or an example short story A Scandal in Bohemia by Arthur Conan Doyle if no file is specified), extracts characters, assigns speakers to quotes, and creates the character network. The input file can also be a DocBin file which contains already annotated docs to speed up the process. Extracted characters and the networks are by default outputted to the folder out.

The mode collect annotates all text files in a given folder and saves the docs and the weights of the edges. These can be later used in the mode train to train a model on the annotated data, using the corresponding golden sets of characters.

The last mode, evaluate, takes a DocBin file and a file containing either the golden set of characters, or the file with annotated speakers of quotes, depending on what we want to evaluate. This mode extracts the characters or the speakers from the DocBin file and evaluates the accuracy.
5.7 Project Structure

The project with our code is attached to this thesis. The source code is located in the `src` folder. This folder contains the subfolders `character_extraction`, `quote_attribution` and `network_creation` for the three modules of our program. Moreover, there are also folders `text_preproc`, `annotation`, `evaluation` and `output_format` that contain the code that takes care of the additional tasks we had to solve. The file `main.py` is the entry point to the program and connects all the parts together.

The folder `vocab` contains the lists of words we use during the annotation or gender prediction: a list of honorifics, common family relations, a list of common male and female words, or a list of gendered English words.

The folder `models` is the default folder to save the trained models for edge prediction. We included two already trained models: `all_data.model` trained on all data we had, and `no_pp.model` trained on all data excluding the *Pride and Prejudice*.

The folder `data` contains four folders. The folder `example` with the example book text and a list of its characters, the two folders `data_vala` and `data_muzny` with the book texts and corresponding golden data for Character Detection and Quote Attribution by Vala et al. [2015] and Muzny et al. [2017], and the folder `golden_characters` containing the lists of characters of some novels that we completed by the character gender and the true number of their occurrences.

At the root of the project, there is a `README.md` file that explains how to set up and run the program and provides some examples of execution.
6. Evaluation

Since the three main modules of our work are rather individual problems, we will evaluate them separately. Character Detection is evaluated in Section 6.1, Quote Attribution in Section 6.2, and the creation of Character Networks in Section 6.3.

6.1 Character Detection

The first module, Character detection, gets a list of annotated book paragraphs on the input. It outputs a list containing one record for each character. Each record contains the detected gender, unique character id, and all possible names of the character, along with the number of occurrences of each name.

6.1.1 Metrics

To measure the accuracy of our extracted list, we use two different metrics. The first one is taken from Vala et al. [2015], the second one is modified to encompass the importance of each character.

**Unweighted Metric** Vala et al. [2015] formalize the problem as finding a maximum bipartite matching between the set of extracted characters $E$ and a gold standard list $G$. An edge exists between $E_m$ and $G_n$ if the two sets of names (denoting one character) have a non-empty intersection. For precision, they measure the matching in the purity of the extracted set: the weight of the edge between $E_m$ and $G_n$ is $1 - \frac{|E_m \setminus G_n|}{|E_m|}$. That means, the match is maximal when the extracted set of names is a subset of the gold standard set. For recall, they check whether the character $G_n$ has been found at all and all edges have weight 1.

We see a deficiency in the approach of Vala et al. [2015] in not distinguishing the importance of characters. They punish not finding a minor character who makes their appearance once or twice the same way as not finding the main character who is present throughout the whole book. Given that it is often questionable whether such a minor character is a character at all, we decided to implement our own metric.

**Importance-Weighted Metric** We will measure the importance of a character’s name as the number of its occurrences, and the importance of a character as the sum of occurrences of all the character’s names. The correct number of occurrences can differ from the occurrences we found. Therefore, if the name is in the golden set, we use the number of occurrences from the golden set. Otherwise, we use the number of extracted occurrences.

We use the previous idea of Vala et al. [2015] of finding a maximum bipartite matching but with different weights. We denote the total number of extracted characters as $M$ and the total number of golden characters as $N$. Equation 6.1 defines the edge weight for measuring precision and Equation 6.2 defines the edge weight for measuring recall.
We do not include the narrator character in this metric because the number of the first-person pronouns is much higher than the number of occurrences of any other character (for example, in *The Moonstone*, we found 6221 narrator pronouns, compared to the 698 occurrences of the most frequent proper character, Franklin Blake).

\[
\text{weight}_P(E_m, G_n) = \frac{\sum_{x \in E_m \cap G_n} \text{importance}(x)}{\sum_{i=1}^{N} \text{importance}(E_i)}
\]

(6.1)

\[
\text{weight}_R(E_m, G_n) = \frac{\text{importance}(G_n)}{\sum_{i=1}^{M} \text{importance}(G_i)}
\]

(6.2)

The weight equations for precision and recall are similar to those defined by Vala et al. [2015]. We only replace the binary value of whether a name is present by the importance of this name. Therefore, we call this metric Importance-Weighted.

### 6.1.2 Datasets

The books that we used come from Project Gutenberg. For evaluation, we used golden lists of characters provided by Vala et al. [2015]. The data include the novels *Pride and Prejudice* by Jane Austen, *The Moonstone* by Wilkie Collins, and a collection of 58 stories of *Sherlock Holmes* by Arthur Conan Doyle. For the Importance-Weighted metric, we completed the character lists of the two novels by the number of occurrences of each name. There are no other sources with complete lists of characters available. Vala et al. [2015] used lists of main characters from SparkNotes to measure recall. However, we don’t believe that this way of measuring recall brings valuable information. The lists of main characters often include only full names of the characters, which are either mentioned only a few times in the book or not at all. Characters are usually called only by their first name or by their last name. For example, the full name of Elizabeth Bennet is mentioned only once in *Pride and Prejudice*, and five more times following the honorific Miss; whereas Elizabeth appears about 550 times.

Searching for the full names of main characters is a different task than we are trying to solve. We search for sets of names that belong to individual characters, as they appear in the book.

### 6.1.3 Results

We evaluated our approach on the novels *Pride and Prejudice* and *The Moonstone* in both metrics. The model we used was trained on the collection of *Sherlock* stories and *The Moonstone* for evaluation of *Pride and Prejudice*, and on *Sherlock* and *Pride and Prejudice* for evaluation of *The Moonstone*. The results are presented in Tables 6.1 and 6.2 and analyzed separately for both metrics. Moreover, we also evaluated our success in finding unnamed characters in *The Little Prince*, with the results presented in Table 6.4.

We compared our approach to the approach of [Vala et al. 2015], and to the baseline of NER. We also included two variants of our approach: NO\_REMOVAL, which doesn’t remove characters with less than three occurrences from the final list, and ALL\_EDGES, which considers all edges for removal from a component as described in Section 2.3.2 instead of only those whose probability is less than 0.9.

**Unweighted Metric** The NER Baseline outputs Named Entities of type PERSON found by spacy’s Named Entity Recognizer. The results are worse than the results of NER Baseline reported by [Vala et al. 2015], who reach F1 score of 53.28 and 33.88 on Pride and Prejudice and The Moonstone respectively. The difference in NER could have negatively affected our approach.

The results of our work measured by the Unweighted Metric are surprisingly low on Pride and Prejudice. However, we identified that 21 out of 73 characters in the gold standard list of [Vala et al. 2015] appear only once in the novel. In our work, we remove characters that appear less than three times in the novel to agree with our definition of a character, so that we miss these 21 characters. Therefore, we included the NO\_REMOVAL version of our approach, where we did not remove any characters. The precision decreased a little but the recall jumped up by almost 30 points and ended up on the exact same value as the recall of [Vala et al. 2015].

We could not reach the precision of [Vala et al. 2015] with our approach. The error analysis shows that our program outputs names that are not characters. We get names of places as Pemberley, Longbourn, Netherfield, and names of families as Gardiners, Collines, Bingleys. All of these names occur more than three times, so they don’t get removed from the final character list. Removing these names manually causes the precision to increase to 66.88. Moreover, the golden list of characters also contains names as Sally, Sarah, Richard, which occur once in the story and our NER doesn’t recognize them. Therefore, we can say that the difference in precision between our approach and the approach of [Vala et al. 2015] is mostly caused by the used Named Entity Recognizer.

The results on The Moonstone are comparable to the results of [Vala et al. 2015], except that we reach higher precision in this case. Not removing characters doesn’t increase the recall as significantly as in the previous case because the characters that we miss are mostly unnamed characters with longer descriptions (e.g. one of my clerks, The boy with the ill-secured eyes, my Hindoo friends) instead of named characters as in Pride and Prejudice. On the contrary, not removing characters reaches much lower precision in this case. It is caused by the fact that our NER finds many foreign words that it believes to be names (Mussulman, Brahmah), and other words that are not characters (Co., Death). The version ALL\_EDGES of our approach reaches mostly similar results to the original approach on both novels.

**Importance-Weighted Metric** Measured by the Importance-Weighted Metric, the results of our approach are more similar to the results of [Vala et al. 2015] than in the previous case.

In Pride and Prejudice, the recall is almost similar to the recall of [Vala et al. 2015] but the precision is 10 points lower. The error analysis shows that the main source of inaccuracy is the largest component including the names of all Bennet.
sisters. They appear to be connected through the name Miss Bennet, which can refer to each of the sisters. Apparently, the connections are quite strong since they don’t get removed during the edge removal phase. Therefore, we included the all_edges approach, where we relaxed the condition of removing only edges whose probability is at most 0.9. This approach correctly splits the sisters into individual characters and surpasses the approach of Vala et al. [2015] by far in all three scores.

In The Moonstone, the all_edges approach doesn’t score as good as in Pride and Prejudice. In this novel, most of the edges with the probability of more than 0.9 are apparently correct and should not be removed. However, the no_removal approach surpasses the recall and the F1 score of Vala et al. [2015].

The results measured by the Importance-Weighted Metric can vary depending on the random seed given to the model. These results were evaluated with the random seed 13.

We tested the model with 10 different random seeds and evaluated it on the two novels using the base approach. The average results can be seen in Table 6.3.

The variance in recall on Pride and Prejudice is especially high. The lowest recall we encountered was 79.42, the highest was 97.13. This is probably caused by the Bennet sisters. They can be connected through the name Miss Bennet that can refer to each of the sisters. Moreover, the Bennet sisters and their friends often appear together and it is not unusual that names of different persons get
connected by coreference. If they stay together in one connected component after the Character Extraction, the recall is low because the component can be connected only to one of the golden characters. Other characters that appear in this component are counted as not found. On the other hand, if the component breaks into little pieces, all the characters are found and the recall can be high.

In *The Moonstone*, there are not many names that can refer to multiple characters and the variance in recall is only 1.1.

Better identification of the names that can refer to more than one character could reduce the variance. The chosen method for Character Detection doesn’t handle these names specially and this deficit is revealed on *Pride and Prejudice*. Having more training data would also make the model more stable.

**Unnamed Characters** We evaluated our approach on two novels. However, all the main characters of these novels have a name, so we could not clearly compare the results of finding the unnamed characters. In the golden list of characters in *The Moonstone* by Vala et al. [2015], there are some minor unnamed characters. However, they are usually identified by multiple descriptions that can be ambiguous and don’t clearly identify the character. For example, they include the following descriptions of one character as names:

- the boy from Mr. Bruff’s office
- his attendant sprite with the gooseberry eyes
- a small boy
- that boy
- my lad
- the small Gooseberry

All of these descriptions appear only once in the novel and don’t agree with our definition of a character (or a character name).

We evaluate our approach on *The Little Prince* by Antoine de Saint-Exupéry. It has a small set of characters, and every character has a single most common description. The NER finds exactly zero entities of type PERSON.

Table 6.4 shows the characters we detected, and the correct characters of *The Little Prince* from SparkNotes with their alternative names in different translations. As discussed in Section 2.1, the distinction between what is and what is not a character is unclear. All the presented characters appear at least three times referred to by the same anaphor. However, this might vary in different translations of the book, especially by the less frequent characters. Moreover, *the little prince’s echo* isn’t animate and doesn’t thus fulfill our definition of a character. However, we decided to include all characters from SparkNotes because this public list might be used by other authors for comparison with our system.

We detected 10 correct characters out of 18 in total, and two incorrect characters (*a sheep* and *a general*). Given the inapplicability of NER in this case and the fact that we, unlike Vala et al. [2015], output full noun chunks (as *the little prince* instead of *prince*), we believe this to be the best result of a universal character extractor on *The Little Prince*. 

35
### Table 6.4: Character detection in *The Little Prince*

<table>
<thead>
<tr>
<th>Detected characters</th>
<th>Correct characters</th>
<th>Occurences</th>
</tr>
</thead>
<tbody>
<tr>
<td>THE NARRATOR</td>
<td>THE NARRATOR, the pilot</td>
<td>865</td>
</tr>
<tr>
<td>the little prince</td>
<td>the little prince</td>
<td>141</td>
</tr>
<tr>
<td>a sheep</td>
<td>the fox</td>
<td>41</td>
</tr>
<tr>
<td>the fox</td>
<td>the fox</td>
<td>28</td>
</tr>
<tr>
<td>the king</td>
<td>the king</td>
<td>26</td>
</tr>
<tr>
<td>the geographer</td>
<td>the geographer</td>
<td>23</td>
</tr>
<tr>
<td>the flower</td>
<td>the rose, the flower</td>
<td>19</td>
</tr>
<tr>
<td>the snake</td>
<td>the snake</td>
<td>16</td>
</tr>
<tr>
<td>the lamplighter</td>
<td>the lamplighter</td>
<td>12</td>
</tr>
<tr>
<td>the businessman</td>
<td>the businessman</td>
<td>10</td>
</tr>
<tr>
<td>the tippler</td>
<td>the drunkard, the tippler</td>
<td>8</td>
</tr>
<tr>
<td>a general</td>
<td>the vain man, the conceited man</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>the roses in the rose garden</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>the salesclerk, the merchant</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>the three-petaled flower</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>the little prince’s echo</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>the Turkish astronomer</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vala</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>83.33</td>
<td>85.50</td>
</tr>
<tr>
<td>55.56</td>
<td>73.89</td>
</tr>
<tr>
<td>66.67</td>
<td>79.27</td>
</tr>
</tbody>
</table>

6.2 **Quote Attribution**

The input to Quote Attribution comes in form of the annotated docs and a list of characters. Each doc contains information about one paragraph, as explained in Section 5.1.1. The Quote Attribution assigns a speaker from the character list to every doc that contains direct speech.

6.2.1 **Datasets**

For evaluation, we use the dataset QuoteLi3 annotated by Muzny et al. [2017]. It contains three novels (*Pride and Prejudice* and *Emma* by Jane Austen, and *The Steppe* by Anton Chekhov).

Since we only attribute speakers to whole paragraphs, we deleted multi-paragraph quotes from QuoteLi3. They were in all cases contents of a letter read aloud, and it was often not even possible to determine whether the speaker should be the writer of the letter, or the recipient. There were 12 multi-paragraph quotes in *Pride and Prejudice*. The two other novels contained less or no such quotes.

The dataset uses slightly different versions of the novels than available on Project Gutenberg. To ensure a correct alignment between the annotated data...
For the evaluation of *Pride and Prejudice*, we use slightly modified list of characters by Vala et al. [2015]. For the evaluation of the other two novels, we use the character lists by Muzny et al. [2017]. We corrected the name of Ivan Kuznitziov in *The Steppe* (Muzny et al. [2017] misspells the last name as Kuzmichov). In *Emma*, we connected the name Mr. Knightley to George Knightley because he seems to be called by this name more often than his brother John.

### 6.2.2 Metric

Since the set of the extracted characters is different than the golden set of characters, we have to create a bijection between these two sets to be able to evaluate our data. We obtain this bijection by finding maximum bipartite matching between these sets, where the weight of an edge between an extracted character $A$ and a golden character $B$ corresponds to the number of quotes which were attributed to $B$ in the golden data and to $A$ by our algorithm.

Our observation shows that this matching usually connects the corresponding characters correctly. The incorrect matches can be those that are not attributed any utterances or are only attributed incorrect ones.

We measure our algorithm first with the golden set of characters from Vala et al. [2015] and Muzny et al. [2017] as stated above, and then with our extracted characters. For extraction, we use the approach with the highest precision: `all_edges` with a model trained on *The Moonstone* and the collection of *Sherlock* stories for *Pride and Prejudice*, and the base approach with a model trained on all data for all other novels.

### 6.2.3 Sieves

First, we will measure the precision and recall using different sieves. We evaluate the average precision and recall of Quote Attribution on the three novels from QuoteLi3, using the golden set of characters. We add the sieves one by one to observe the recall increase and the precision decrease with adding new sieves.

Table 6.5 shows the eight Quote-Mention sieves in the rows and the five Mention-Speaker sieves in the columns. In each cell, there is the average precision and recall we obtained using all the previous sieves up to the one in the same row or column.

Using only the `Trigram Matching` sieve in Quote-Mention and `Exact Match` sieve in Mention-Speaker, we were able to obtain the precision of 99.2%, having a low recall of only 15.3%. Adding more Quote-Mention sieves, the recall increased up to 41.5%, while the precision dropped to 89.3%. The precision drop is likely caused by incorrect mention identification: as we use more Quote-Mention sieves, wrong mentions are more likely to be reported. As we add the `Coreference` Mention-Speaker sieve, the trend is similar: the precision decreases while the recall increases up to 61.6% when using all the sieves.

With the `Conversational Pattern` Mention-Speaker sieve, the precision increases along with the recall when adding the Quote-Mention sieves. This is be-
cause the conversational sieve benefits from having more information: the higher the absolute number of correct speakers, the more other correct speakers it can detect. The increase stops when adding the last four sieves. The last two conversational sieves do in general the same thing as the Conversational Pattern Mention-Speaker sieve, so the lack of improvement is clear. The low improvement when adding the Previous Vocative and Final Mention sieves might mean that these sieves don’t provide much new or precise information.

The precision when using the Conversational Pattern sieves both in Quote-Mention and Mention-Speaker is slightly lower than when using the Conversational Pattern sieve only in Mention-Speaker. We have discussed the differences between these sieves in Section 3.3.2. This proves that omitting the Quote-Mention conversational sieves and replacing their functionality through the Mention-Speaker conversational sieve might be beneficial.

Using the last two Mention-Speaker sieves is only guessing since the quotes that reach these sieves usually don’t have any mention assigned. However, these sieves increase the recall up to a maximum of 74.5%. The precision and recall both reach almost the same values in the last column. This means that we attribute some speaker to almost every quote in the end.

The best precision we reached was 74.5% using all the Mention-Speaker sieves and excluding the two conversational sieves from Quote-Mention. We didn’t test a different ordering of the sieves or using different subsets of the sieves. However, we believe this would not reach significantly better results.

For the following detailed evaluation of individual novels, we used all sieves except the Closest Name Before sieve from Mention-Speaker. Both the average precision and recall of this combination are 74.0% which is slightly less than the best reported result. However, we first tested the sieves only on Pride and Prejudice where this combination was the one with the best performance.
6.2.4 Results

In Table 6.6, we compare the numbers of speakers we found by their type to those reported by Muzny et al. [2017] (they provide the numbers both for all quotes they consider and for the collapsed version of one speaker per paragraph that we assume). The first thing to notice is that the total numbers of speakers found are slightly different, even though we use the same QuoteLi3 corpus. This might have two explanations: First, the multi-paragraph quotes we removed (especially in Pride and Prejudice). And second, Muzny et al. [2017] might have used a different version of the corpus in the paper, and improved the corpus later. The individual types of speakers are roughly the same.

<table>
<thead>
<tr>
<th>Quote Type</th>
<th>Muzny et al. 2017</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P&amp;P</td>
<td>Emma</td>
</tr>
<tr>
<td>Explicit</td>
<td>326</td>
<td>128</td>
</tr>
<tr>
<td>Anaphoric (pronoun)</td>
<td>241</td>
<td>73</td>
</tr>
<tr>
<td>Anaphoric (other)</td>
<td>68</td>
<td>15</td>
</tr>
<tr>
<td>Implicit</td>
<td>655</td>
<td>357</td>
</tr>
<tr>
<td>Total</td>
<td>1290</td>
<td>558</td>
</tr>
</tbody>
</table>

Table 6.6: Identified speaker types

We evaluated our approach separately on Pride and Prejudice, Emma, and The Steppe. For each novel, we computed the accuracy for each of the different speaker types. The results are shown in Tables 6.7, 6.8, and 6.9.

<table>
<thead>
<tr>
<th></th>
<th>ES</th>
<th>AS(p)</th>
<th>AS(o)</th>
<th>IS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muzny et al. 2017</td>
<td>98.4</td>
<td>77.3</td>
<td>42.9</td>
<td>82.3</td>
<td>85.1</td>
</tr>
<tr>
<td>This work –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>golden characters</td>
<td>95.4</td>
<td>78.6</td>
<td>45.2</td>
<td>67.0</td>
<td>75.6</td>
</tr>
<tr>
<td>This work –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>extracted characters</td>
<td>94.1</td>
<td>69.2</td>
<td>29.3</td>
<td>53.9</td>
<td>65.2</td>
</tr>
</tbody>
</table>

Table 6.7: Accuracy on Pride and Prejudice

Pride and Prejudice In Pride and Prejudice, the most challenging speaker type showed up to be the implicit speaker. As you can see in Table 6.7, our results on all other speaker types are very similar to the results of Muzny et al. 2017, but the implicit speaker results fall behind by more than 15 points.

The error analysis shows that 72 out of the 191 incorrectly attributed paragraphs with an implicit speaker were determined by the Conversational Pattern sieve. Therefore, the original failure in these 72 paragraphs lies in one of the prior paragraphs whose speaker has been identified incorrectly. Another 57 wrongly identified implicit speakers have its origin in the Majority Speaker sieve. This is the catch-all sieve that increases recall and is not expected to be very precise. The remaining 62 incorrect implicit speakers were determined by Exact Name Match or Coreference Disambiguation. However, the mention in almost all of these paragraphs originates from a previous paragraph through the Conversational Pattern
or *Loose Conversational Pattern* sieve. Therefore, the original mistake lies in some prior paragraph in most cases of the incorrectly identified implicit speakers.

We identified the common failures that cause the wrong speaker identification:

- A narrative that interrupts the conversation. There is a conversation between Mr. and Mrs. Bennet at the beginning of *Pride and Prejudice*. Mrs. Bennet tells her husband that certain Mr. Bingley moved into their neighborhood. The dialogue is sometimes interrupted by short narratives. Some of them substitute a direct speech and the following speaker is the same as the previous one, following the conversational pattern: *Mr. Bennet made no answer.* However, other narratives are only inserted between the utterances of Mr. and Mrs. Bennet and disturb the pattern: *This was invitation enough.* The identified speakers thus can get exchanged.

- Complicated dialogues with more than two participants. Consider the following dialogue:

  “Can I have the carriage?” said Jane.
  “No, my dear, you had better go on horseback, because it seems likely to rain; and then you must stay all night.”
  “That would be a good scheme,” said Elizabeth, “if you were sure that they would not offer to send her home.”
  “Oh! but the gentlemen will have Mr. Bingley’s chaise to go to Meryton; and the Hursts have no horses to theirs.”
  “I had much rather go in the coach.”

  The speaker of the last utterance is *Jane*. However, this follows only from the content of the utterance which we don’t consider. We identify the speaker incorrectly as *Elizabeth*.

- Speaker determined in the previous paragraph. For example, *A short pause followed this speech, and Mrs. Hurst began again.* Our *PreviousParagraph* sieve doesn’t recognize the mention of Mrs. Hurst because the sentence doesn’t end with a colon and the following paragraph contains only the quote.

- Speakers mentioned in a long narrative. It happens especially at the beginning of a chapter that there is a long narrative which introduces the participants of the next dialogue. Our approach doesn’t recognize the correct speakers and uses the *Majority Speaker* sieve which has a low success rate.

The results on the extracted characters are lower than the results on the golden characters, as expected. The lower anaphoric speaker accuracy probably affected also the implicit speaker accuracy. The total accuracy reaches about 86% of the accuracy using the golden characters, which roughly corresponds to the 88% precision of the characters we extracted.
**Emma**  In *Emma*, we face similar problems as in *Pride and Prejudice*. It might be because both novels were written by *Jane Austen*.

A difference to *Pride and Prejudice* is that our version of *Emma* contains dashes quite often. For example:

> “Such sweet lines!” continued Harriet– “these two last!– But how shall I ever be able to return the paper, or say I have found it out?– Oh! Miss Woodhouse, what can we do about that?”

These dashes cannot be correctly parsed by our tokenizer. The tokenizer outputs *Harriet–* as one token, which is later recognized as the mention to this utterance. However, it doesn’t get connected to the correct name *Harriet* which doesn’t contain any dashes.

The total accuracy on the extracted characters reaches only 52.5%. The poor performance corresponds to the poor performance of character extraction: our extractor doesn’t connect *Emma* to *Miss Woodhouse*, connects *Mr. Knightley* to *John Knightley* (which might be correct but, as said earlier, *Mr. Knightley* refers more often to *George*) and doesn’t find *George Knightley* at all.

<table>
<thead>
<tr>
<th></th>
<th>ES</th>
<th>AS(p)</th>
<th>AS(o)</th>
<th>IS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Muzny et al. [2017]</strong></td>
<td>92.1</td>
<td>62.5</td>
<td>35.0</td>
<td>71.5</td>
<td>75.9</td>
</tr>
<tr>
<td>This work – golden characters</td>
<td>90.7</td>
<td>61.4</td>
<td>46.7</td>
<td>61.9</td>
<td>66.9</td>
</tr>
<tr>
<td>This work – extracted characters</td>
<td>82.2</td>
<td>42.1</td>
<td>45.1</td>
<td>45.5</td>
<td>52.5</td>
</tr>
</tbody>
</table>

Table 6.8: Accuracy on *Emma*

<table>
<thead>
<tr>
<th></th>
<th>ES</th>
<th>AS(p)</th>
<th>AS(o)</th>
<th>IS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Muzny et al. [2017]</strong></td>
<td>97.5</td>
<td>67.0</td>
<td>14.9</td>
<td>60.4</td>
<td>72.7</td>
</tr>
<tr>
<td>This work – golden characters</td>
<td>95.3</td>
<td>80.0</td>
<td>63.0</td>
<td>70.8</td>
<td>80.2</td>
</tr>
<tr>
<td>This work – extracted characters</td>
<td>90.2</td>
<td>76.6</td>
<td>37.3</td>
<td>68.0</td>
<td>74.1</td>
</tr>
</tbody>
</table>

Table 6.9: Accuracy on *The Steppe*

**The Steppe**  In *The Steppe*, our accuracy surpassed the accuracy of *Muzny et al. [2017]* both for *implicit speakers* and all speakers in total. We believe that the main cause of this success is our sieve *PreviousParagraph*. In *The Steppe*, it is common that the speaker of the next utterance is mentioned at the end of the previous paragraph ending with a colon. This is recognized by our sieve. Without the *PreviousParagraph* sieve, the total accuracy would be 72.6, similar to *Muzny et al. [2017]*.

Here, the performance on the extracted characters is almost as good as the performance on the golden characters, and even surpasses the approach of *Muzny et al. [2017]* who use the golden characters.
6.3 Character Networks

The last module gets the annotated docs on the input along with a list of characters. It outputs a graph of interactions between the characters.

6.3.1 Experiments

We perform two experiments with the networks. First, we check how much the conversational network differs from the co-occurrence network. Second, we include character genders in the network and test whether the novels are gender-biased.

6.3.2 Conversation vs. Co-occurrence

We created three different character networks from *The Steppe*:

- Co-occurrence network where the weight of an edge corresponds to the number of paragraphs where the two characters occur together (Figure 6.1).

- Conversational network using the golden speakers (Figure 6.2). We will call this network golden conversational network.

- Conversational network using the speakers we extracted (Figure 6.3). We will call this network extracted conversational network.

We matched our characters to the golden speakers as in Section 6.2 and kept only those pairs of characters that mutually agree on more than one quote, to ensure that the characters are paired correctly. The golden conversational network has different labels than the two others but the positions of the nodes correspond to the matching characters from our character list.

We normalized all the three networks to have equally weighted the heaviest edge.

**Co-occurrence network** The co-occurrence network is quite dense. This shows that the characters are connected in the plot, either by really being together in the same place, or by thinking and talking about each other. *Yegorushka* has connections to all other characters, so we can say he is probably one of the main characters. He indeed is the main character: *Yegorushka* is a young boy who is being taken to a boarding school by his uncle *Kuzmitchov* and the priest *Father Christopher*. *The Steppe* describes his journey.

**Golden conversational network** The golden conversational network is, opposed to the co-occurrence network, very sparse. *Varlamov*, who was connected to 12 out of the 16 other characters in the co-occurrence network, doesn’t have any connections here. *Varlamov* is an elusive wool-merchant who is sought by multiple characters in the story. They talk about him as they try to find out where they can meet him, so that his name occurs in the same paragraph with others often. However, he appears only once in the story in person. The three utterances *Varlamov* says are separated by narratives, so that he doesn’t get connected to any other character in the golden conversational network.
Apart from Varlamov, we can see a significant difference in the edge between Yegorushka and Dymov. This edge belonged to the thickest in the co-occurrence network but is very thin here. This can be explained by the fact that Yegorushka and Dymov travel together, so that they often occur together in the narrative. However, Yegorushka finds Dymov violent and does not seek to talk to him.

In the golden conversational network, we can more clearly see the separation of the graph to two subgraphs joined through Yegorushka. The first subgraph includes Kuzmitchov, Father Christopher and Moisey Moisevitch. The second subgraph includes Panteley, Dymov, Konstantin and others. If it hadn’t been for the edge between Panteley and Kuzmitchov, Yegorushka node would have been an articulation connecting these two components. This has an explanation: Yegorushka first travels with Kuzmitchov and Father Christopher and encounters the characters that would belong to the first graph component. Halfway through the journey, Yegorushka’s uncle hands his nephew over to a driver of a long wagon train and they travel separately for some time. Yegorushka then gets to know all the wagon travelers who would belong to the second graph component.

It is possible to have two characters that are connected by co-occurrence but not by conversation – this is common in our two networks. However, it is also possible the other way around. There can be characters who talk to each other.
but whose names don’t appear in one paragraph together. This seems to be the case of Nastasya Petrovna and Ivan Kuzmitchov in the golden conversational network. However, it is only caused by imprecise merging of the names of the characters, as explained later.

Extracted conversational network The extracted conversational network looks similar to the golden conversational network but contains some edges that are not present in the golden network (e.g. between Yegorushka and Emelyan or Varlamov). These incorrect edges might have been added through wrong coreference resolution or the Majority Speaker sieve that uses the names that appear often in the text surrounding the quote. It is, therefore, likely that these names will also be connected by co-occurrence. We believe that the more imprecise the extracted conversational network is, the more similar to the co-occurrence network it can look.

Another differences between the extracted conversational network and the golden one are caused by the imprecise connection of the character names. For example, our algorithm doesn’t connect Kuzmitchov to Ivan Ivanitch, even though both names belong to the same character. In the golden network, Nastasya Petrovna talks to Ivan Kuzmitchov but our algorithm identified Nastasya’s conversa-
tional partner as Ivan Ivanitch, who is considered to be a different character than Kuzmitchov. Ivan Ivanitch is not shown in our graph and Nastasya is not connected to any character in the extracted conversational network. The full graph including both Ivan Ivanitch and Kuzmitchov is shown in Section 6.3.3.

Even though the extracted conversational network is not as precise as the golden one, the previously mentioned properties can be seen here as well. Varlamov is connected to Yegorushka here but the difference in the number of edges is still significant. The edge between Yegorushka and Dymov looks the same as in the previous network. The two components are also visible here.

The three networks for Pride and Prejudice and Emma are available as Attachment A.1 and A.2.

![Figure 6.3: Extracted conversational network of The Steppe](image)

### 6.3.3 Gender Study

The second experiment we performed was a gender study to recognize gender biases in literary works. We created a conversational network where the characters we identified as female are colored red and male characters are colored blue. Similarly, the conversational edges between two female characters are colored red
Figure 6.4: Characters by gender in *The Steppe*

and the edges between two male characters blue. The edges between a male and a female character are colored gray.

This network is created using the extracted speakers. We include only the characters identified as speakers of some utterance. If the novel has more than 20 such characters, we use only the 20 most frequent.

These networks can be used to perform the *Bechdel–Wallace test*.[2] This test has been originally proposed to measure the representation of women in movies. It has three criteria to pass:

1. The movie has at least two female characters.
2. The female characters talk to each other.
3. They talk about something other than a man.

This test can be applied to literary works as well. We can use our networks to evaluate the first two points of this test; a proper semantic analysis would be required to evaluate the third criterion.


46
As you can see on Figure 6.4, *The Steppe* fails this test. It has only three female characters and none of them talk to each other.

Both *Pride and Prejudice* and *Emma* pass the test, as you can see in Figures 6.5 and 6.6. This is not surprising, as both of these novels were written by a female author and have a female protagonist.

Three more gender networks are included in Attachment A.3.

### 6.3.4 Future Work

We extracted the strength of mutual relations between the characters as depicted in the novels through dialogues and co-occurrence. However, it might be interesting to identify the type of the relations as well (e.g. friends, family, enemies). Kanjirangat and Antonucci (2019) succeeded in identifying the sibling relation which might be a good first step.

The networks might also be improved by including character descriptions. Extracting the descriptions would allow a reader who is not familiar with the story to obtain more information from the network. For example, knowing that Mr. Darcy is described as *proud* and *arrogant* can make the reader wonder what
makes him talk to Elizabeth so often. This can hint on the romantic interests Mr. Darcy holds to her.

Figure 6.6: Characters by gender in Emma
Conclusion

In this thesis, we solved three problems related to information extraction from novels: Character Detection, Quote Attribution, and Character Network creation. Character Detection is the task of extracting the list of characters from a book. We presented a novel approach for Character Detection combining hand-picked features and Machine Learning. We introduced a new metric to measure the accuracy of the extracted set of characters. Using this metric, our system beats the existing approaches. Having more training data would have increased the precision of our system and make it more stable.

In Quote Attribution, our goal was to correctly assign speakers to each utterance. We reimplemented an existing approach and improved it to obtain better results on one of the three tested novels. The error analysis shows some suggestions for further improvement.

In the last module, Character Networks, we created networks that show the interactions between the characters. We compared co-occurrence networks to conversational networks and discovered the gender bias in *The Steppe*. We presented some suggestions for improving the networks to provide more information for a reader who is not familiar with the novel.

This thesis connects the three modules together. While each of these tasks can be viewed as a separate problem, we created one tool that can process the text of a novel to extract the character list, use this list to find the speaker of each utterance, and finally use the speakers to create a character network and visualize it.

We believe that this thesis can serve as a basis for further work. The performance on the individual problems can be further improved and more interesting results can be obtained from studying the relations between the characters.
Bibliography


<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Example of tokenization</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Dependency parsed sentence as by spacy</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>The largest component before and after the edge removal</td>
<td>12</td>
</tr>
<tr>
<td>4.1</td>
<td>The process of Character Network extraction</td>
<td>19</td>
</tr>
<tr>
<td>4.2</td>
<td>NOVEL2GRAPH output of Harry Potter</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>Data flow in the program</td>
<td>25</td>
</tr>
<tr>
<td>6.1</td>
<td>Co-occurrence network of The Steppe</td>
<td>43</td>
</tr>
<tr>
<td>6.2</td>
<td>Golden conversational network of The Steppe</td>
<td>44</td>
</tr>
<tr>
<td>6.3</td>
<td>Extracted conversational network of The Steppe</td>
<td>45</td>
</tr>
<tr>
<td>6.4</td>
<td>Characters by gender in The Steppe</td>
<td>46</td>
</tr>
<tr>
<td>6.5</td>
<td>Characters by gender in Pride and Prejudice</td>
<td>47</td>
</tr>
<tr>
<td>6.6</td>
<td>Characters by gender in Emma</td>
<td>48</td>
</tr>
<tr>
<td>A.1</td>
<td>Co-occurrence network of Pride and Prejudice</td>
<td>54</td>
</tr>
<tr>
<td>A.2</td>
<td>Golden conversational network of Pride and Prejudice</td>
<td>55</td>
</tr>
<tr>
<td>A.3</td>
<td>Extracted conversational network of Pride and Prejudice</td>
<td>56</td>
</tr>
<tr>
<td>A.4</td>
<td>Co-occurrence network of Emma</td>
<td>57</td>
</tr>
<tr>
<td>A.5</td>
<td>Golden conversational network of Emma</td>
<td>58</td>
</tr>
<tr>
<td>A.6</td>
<td>Extracted conversational network of Emma</td>
<td>59</td>
</tr>
<tr>
<td>A.7</td>
<td>Gender network of Alice’s Adventures in Wonderland</td>
<td>60</td>
</tr>
<tr>
<td>A.8</td>
<td>Gender network of Great Expectations</td>
<td>61</td>
</tr>
<tr>
<td>A.9</td>
<td>Gender network of The Adventures of Tom Sawyer</td>
<td>62</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Edge types</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Types of speaker mentions</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Features by He et al.</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Sieves in mention detection by Muzny et al.</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Our sieves, including the two novel sieves</td>
<td>17</td>
</tr>
<tr>
<td>4.1</td>
<td>Types of character interactions</td>
<td>20</td>
</tr>
<tr>
<td>6.1</td>
<td>Evaluation of Character Detection (Unweighted)</td>
<td>34</td>
</tr>
<tr>
<td>6.2</td>
<td>Evaluation of Character Detection (Importance-Weighted)</td>
<td>34</td>
</tr>
<tr>
<td>6.3</td>
<td>Average results with 10 different random seeds</td>
<td>34</td>
</tr>
<tr>
<td>6.4</td>
<td>Character detection in <em>The Little Prince</em></td>
<td>36</td>
</tr>
<tr>
<td>6.5</td>
<td>Evaluation of the sieves</td>
<td>38</td>
</tr>
<tr>
<td>6.6</td>
<td>Identified speaker types</td>
<td>39</td>
</tr>
<tr>
<td>6.7</td>
<td>Accuracy on <em>Pride and Prejudice</em></td>
<td>39</td>
</tr>
<tr>
<td>6.8</td>
<td>Accuracy on <em>Emma</em></td>
<td>41</td>
</tr>
<tr>
<td>6.9</td>
<td>Accuracy on <em>The Steppe</em></td>
<td>41</td>
</tr>
</tbody>
</table>
A. Character Networks

A.1 Conversation vs. Co-occurrence – *Pride and Prejudice*

Figure A.1: Co-occurrence network of *Pride and Prejudice*
Figure A.2: Golden conversational network of *Pride and Prejudice*
Figure A.3: Extracted conversational network of *Pride and Prejudice*
A.2 Conversation vs. Co-occurrence – *Emma*

Figure A.4: Co-occurrence network of *Emma*
Figure A.5: Golden conversational network of *Emma*
Figure A.6: Extracted conversational network of Emma
A.3 Gender Networks

Figure A.7: Gender network of *Alice’s Adventures in Wonderland* by Lewis Carroll
Figure A.8: Gender network of *Great Expectations* by Charles Dickens
Figure A.9: Gender network of *The Adventures of Tom Sawyer* by Mark Twain