Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts

Hardik Vala[1], **David Jurgens**[1], **Andrew Piper**[2], **Derek Ruths**[1]
[1]School of Computer Science
[2]Department of Languages, Literatures, and Cultures
McGill University, Montreal, QC, Canada
hardik.vala@mail.mcgill.ca, jurgens@cs.mcgill.ca
andrew.piper@mcgill.ca, derek.ruths@mcgill.ca

Abstract

Characters are fundamental to literary analysis. Current approaches are heavily reliant on NER to identify characters, causing many to be overlooked. We propose a novel technique for character detection, achieving significant improvements over state of the art on multiple datasets.

1 Introduction

How many literary characters appear in a novel? Despite the seeming simplicity of the question, precisely identifying which characters appear in a story remains an open question in literary and narrative analysis. Characters form the core of many computational analyses, from inferring prototypical character types (Bamman et al., 2014) to identifying the structure of social networks in literature (Elson et al., 2010; Lee and Yeung, 2012; Agarwal et al., 2013; Ardanuy and Sporleder, 2014; Jayannavar et al., 2015). These current approaches have largely assumed that characters can be reliably identified in text using standard techniques such as Named Entity Recognition (NER) and that the variations in how a character is named can be found through coreference resolution. However, such treatment of character identity often overlooks minor characters that serve to enrich the social structure and serve as foils for the identities of major characters (Eder et al., 2010).

This work provides a comprehensive examination of literary character detection, with three key contributions. First, we formalize the task with evaluation criteria and offer two datasets, including a complete, manually-annotated list of all characters in 58 literary works. Second, we propose a new technique for character detection based on inducing character prototypes, and in comparisons with three state-of-the-art methods, demonstrate superior performance, achieving significant improvements in F1 over the next-best method. Third, as practical applications, we analyze literary trends in character density over 20 decades and revisit the character-based literary hypothesis tested by Elson et al. (2010).

2 Related Work

Character detection has primarily been performed in the context of mining literary social networks. Elson et al. (2010) extract character mentions from conversational segments, using the Stanford CoreNLP NER system to discover character names (Manning et al., 2014). To account for variability in character naming, alternate forms of a name are generated using the method of Davis et al. (2003) and merged together as a single character. Furthermore, the set of aliases for a character is expanded by creating coreference chains originating from these proper names and merging all coreferent expressions. Agarwal et al. (2013) also rely on the CoreNLP NER and coreference resolution systems for character detection; however for literary analysis, they use gold character mentions that have been marked and resolved by a team of trained annotators, highlighting the difficulty of the task.

He et al. (2013) propose an alternate approach for identifying speaker references in novels, using a probabilistic model to identify which character is speaking. However, to account for the multiple aliases used to refer to a character, the authors first manually constructed a list of characters and their aliases, which is the task proposed in this work and underscores the need for automated methods.

Two approaches mined social interaction net-

works without relying on dialogue, unlike the methods of Elson et al. (2010) and He et al. (2013). Lee and Yeung (2012) build social networks by recognizing characters from explicit markers (e.g., kinship) and implicit markers (e.g., physical collocation). Similarly, Agarwal and Rambow (2010) build character networks using tree kernels on parse trees to identify interacting agents.

In the two most-related works, Bamman et al. (2014) and Ardanuy and Sporleder (2014), character names are extracted and clustered under a set of constraints. In the BookNLP system developed by Bamman et al. (2014), NER-identified names are retained and merged based on animacy, determined through dependencies with "sentient" lemmas from a small dictionary (including for example, *say* and *smile*), and gender, assigned through pronomial resolution and a dictionary of gender-specific honorifics. Ardanuy and Sporleder (2014) similarly use NER to identify character name mentions. These names are grouped through the application of a series of deterministic rules, beginning with recognizing gender constraints, where gender assignments are based off of gender-specific honorifics and names. If a gender can't be assigned, then one is derived from the majority count of gender-specific pronouns (e.g. he, herself) appearing in the immediate context of the name mentions. The extracted names are then clustered, while respecting the gender impositions, based on a sieve of name variant heuristics. In the final step, any remaining ambiguous referents , i.e., those that can be matched to multiple characters, are assigned to the more prominent character in the story. The authors achieve F1-scores > 0.9 for extracting the 10 most relevant characters in a small collection of novels, but the performance on all characters is unknown.

3 Detecting Characters

We propose an eight stage pipeline for detecting characters, which builds a graph where nodes are names and edges connect names belonging to the same character. The vertices in the graph are initially populated by running NER over the corpus and also incorporating names following an honorific. Second, coreference resolution is run to identify names that occur together in a coreference chain and edges are added where two nodes' names co-occur in a chain. Stanford CoreNLP is used for both NER and co-reference. Third, we apply a series of name variation rules to link

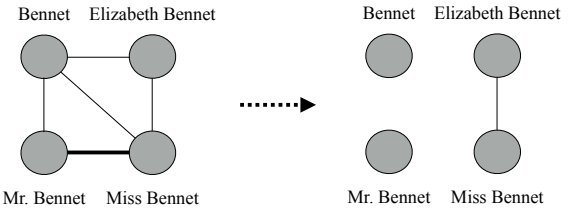

Figure 1: Resolving names in a character graph. The circles represent individual names and the thin and thick lines denote edges and anti-edges, respectively.

names potentially referring to the same character (e.g., by removing an honorific). Fourth, a gazetteer of 1859 hypocorisms for 560 names is used to link variations (e.g., Tim and Timmy).

Stages 2–4 potentially introduce edges connecting names of different characters. Therefore, in the fifth stage, three heuristics are applied to add prohibitions on merging two names into the same character. Two vertices cannot be merged if (1) the inferred genders of both names differ, (2) both names share a common surname but different first names, or (3) the honorific of both names differ, e.g., "Miss" and "Mrs." Similarly, the sixth stage inserts prohibitions by extracting pairs of names from the novel where (1) both names appear connected by a conjunction, (2) one name appears as the speaker mentioning the other name in direct speech, and (3) both names appear together in a quotation.

Together, Stage 1–6 are applied by first linking all nodes by edges and following, identifying pairs prohibited from being connected and remove the edges along the shortest path between those two nodes, effectively creating two new disconnected components in the name graph. Figure 1 illustrates this transformation on a simple character graph.

Next, the seventh step attempts to identify characters whose names may not be recognized by NER. For example, many minor characters do not appear as named entities and instead have general role-based referents such as "the governor" or "the archbishop." However, despite the lack of proper names, such characters behave and interact in similar ways as major characters, including having dialogue. Therefore, to discover such characters, we adopt a bootstrapping technique aimed at uncovering prototypical character behaviors from the novels themselves, inspired by the semantic predicate work of Flati and Navigli (2013). The Project Gutenberg fiction corpus was dependency parsed to identify all verbs in a dependency relation with nouns, where each noun was categorized as (a) a

named entity (b) having its first sense in WordNet refer to an animate entity (Fellbaum, 1998), or (c) neither of the above. All verbs associated with these were then ranked according to their ratio of types (a) and (c) to identify verbs strongly associated with character-like behaviors, which avoids including the behavior of nouns in (b) which may refer to minor characters. Ultimately, 2,073 verbs-and-dependency pairs with a ratio of $\geq$0.25 were retained as predicates selecting for character-like entities, after limited experimental testing showed this threshold extracted sensible verbs such as "rejoice," "accost," and "frown." Using this set of predicates, nouns appearing with the verb in the appropriate dependency relation are added as characters. We prohibit adding names contained in a small stop list of 22 generic nouns (e.g., "man").

Finally, the eighth and last stage removes nodes that are disconnected from the rest of the graph and represent a name that is a portion of one or more names for other nodes. These nodes are typically ambiguous first or last names. Thus, the remaining set of nodes are merged to create sets of names, each associated with a different character.

4 Experiments

Given a literary novel, our objective is to produce a list of characters, where each character may be associated with one or more names.

Datasets Two datasets are used. The first is a manually-annotated collection of 58 works with complete lists of all characters and their possible referents in texts. Of these works, 56 were generated as a part of an on-going longitudinal study of author style for all *Sherlock Holmes* stories written by Sir Arthur Conan Doyle. The remaining two works are the full length novels *Pride and Prejudice* by Jane Austen and *The Moonstone* by Wilkie Collins. Characters and their alias were manually coded by expert annotators, with multiple passes to ensure completeness. *The Moonstone* was treated as truly held-out test data and results were only generated once prior to submission.

The second dataset consists of 30 novels listed on Sparknotes (`sparknotes.com`) and their corresponding lists of characters, with supplemental naming variations of these characters provided by our annotators. These character lists often contain only the major characters in a novel; for example, their list for Pride and Prejudice contains only 17 characters, where as our manually-annotated list identifies 73 characters. Nevertheless, the Sparknotes data serves as a baseline of those characters any method should be able to detect.

Evaluation Character recognition systems produce a list of sets, each containing the names associated with one character, denoted $E = \{E_1, \ldots, E_n\}$ where E_i is a set of names for a character. These lists are evaluated against a gold standard list, denoted G, containing all naming variations for each character. To evaluate, we formalize the problem as finding a maximum bipartite matching where the sets of names in E and those in G constitute the two node types. For precision, matching is measured in the purity of an extracted set of names, E_i, with respect to the gold-standard names, G_j: $1 - \frac{|E^i - G^j|}{|E^i|}$; simply, a match is maximal when the set of extracted names is a subset of the gold standard names, with penalties for including wrong names. Recall uses a looser definition of matching with the aim of measuring whether a character G_j was found at all; matching is measured as a binary function that is 1 if $E_i \cap G_j \neq \emptyset$ and 0 otherwise.

Comparison Systems The task of character recognition has largely been subsumed into the task of extracting the social network of novels. Therefore, three state-of-the-art systems for social network extraction were selected: the method described in Elson et al. (2010), BookNLP (Bamman et al., 2014), and the method described in Ardanuy and Sporleder (2014). For each method, we follow their procedures for identifying the characters in the social network, which produces sets of one or more aliases associated with each identified character. As a baseline, we use the output of Stanford NER, where every name is considered a separate character; this baseline represents the upper-bound in recall from any system using only NER to identify character names.

Experiment 1: Accuracy Table 1 shows the results for the manually-annotated and SparkNotes corpora. The Sherlock Holmes corpus presents a notable challenge due to the presence of many minor characters, which are not detected by NER. An error analysis for our approach revealed that while many characters were extracted, the coreference resolution did not link a characters' different referents together and hence, each name was reported as a separate character, which caused a drop in performance. Nevertheless, our system provided the highest performance for character recognition.

The Pride and Prejudice novel presents a dif-

	Sherlock Holmes Stories			Pride and Prejudice			The Moonstone			SparkNotes
System	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Recall
NER Baseline	0.3327	0.6535	0.4332	0.3910	0.8356	0.5328	0.2460	0.5441	0.3388	0.6762
Elson et al. (2010)	0.3757	0.6042	0.4485	0.3100	0.5205	0.3886	0.2612	0.4931	0.3415	0.4723
BookNLP (Bamman et al., 2014)	0.6084	0.4832	0.5219	0.4855	0.5205	0.5024	0.3662	0.4706	0.4119	0.5880
Ardanuy and Sporleder (2014)	0.5744	0.4719	0.5181	0.4610	0.5108	0.4846	0.3623	0.4691	0.4088	0.5898
This work	0.5109	0.6099	**0.5404**	0.7245	0.7945	**0.7579**	0.3673	0.5735	**0.4478**	0.5990

Table 1: Accuracy of character detection on different portions of the two datasets.

	Precision	Recall	F1
Sherlock Holmes Stories	0.5910	0.5335	0.5608
Pride and Prejudice	0.7635	0.6879	0.7237
The Moonstone	0.3943	0.4613	0.4294

Table 2: Accuracy of proposed system without stage 7.

ferent set of challenges due to multiple characters sharing the same last name or the same first name. Here, coreference resolution frequently creates incorrect links between the similar names of different characters, creating a drop in precision for most systems. Our precision value particularly benefited from the heuristics for distinguishing characters by gender and stringent name-merging constraints. BookNLP and the approach of Ardanuy and Sporleder (2014) performed quite similarly in identifying characters, which is expected given the overlap in rules applied by both systems.

Moonstone contains a unique novel structure with multiple first-person narrators, group-based characters (e.g., "the jugglers") that present a challenge to co-reference systems, and 419 different names for the 78 unique characters. An error analysis of our system revealed that majority of mistakes were due to the multiple names for a character not being merged into a single identity. Nevertheless, our system performs best of those tested.

For the SparkNotes data, the NER baseline achieves the highest recall, indicating that many of the major character names listed in SparkNotes' data can be directly found by NER. Nevertheless, in reality, the baseline's performance is offset by its significantly lower precision, as shown in its performance on the other novels; indeed the baseline grossly overestimates the number of characters for the SparkNotes novels, reporting 339 characters per novel on average.

Table 2 shows our system's performance without stage 7, which involved the extraction of minor characters. Stage 7 overall improves recall with a slight hindrance to precision. For the Sherlock Holmes corpus, stage 7 is slightly detrimental to overall performance, which as we stipulated earlier is caused by missing co-referent links.

Finally, returning to the initially-posed question of how many characters are present, we find that despite the detection error in our method, the overall predicted number of characters is quite close to the actual: for Sherlock Holmes stories, the number of characters was estimated within 2.4 on average, for *Pride and Prejudice* our method predicted 72 compared with 73 actual characters, and for *The Moonstone* our method predicted 87 compared with 78. Thus, we argue that our procedure can provide a reasonable estimate for the total number of characters. (For comparison, BookNLP, the next best system, extracted 69 and 72 characters for Pride and Prejudice and The Moonstone, respectively, and within 1.2, on average, on the Sherlock Holmes set.)

Experiment 2: Literary Theories Elson et al. (2010) analyze 60 novels to computationally test literary theories for novels in urban and rural settings (Williams, 1975; Moretti, 1999). Recently, Jayannavar et al. (2015) challenged this analysis, showing their improved method for social network extraction did not support the same conclusions. While our work focuses only on character detection, we are nevertheless able to test the related hypothesis of whether the number of characters in novels with urban settings is more than those in rural. Character detection was run on the same novels from Elson et al. (2010) and we found no statistically-significant difference in the mean number of characters in urban and rural settings, even when accounting for text size. Thus, our work raises questions about how these character interact and whether the setting influences the structure of the social network, despite similar numbers of characters.

Experiment 3: Historical Trends As a second application of our technique, we examine historical trends in how many characters appear in a novel. All fiction novels listed on Project Gutenberg were compiled and publication dates were automatically extracted for 1066 and manually entered for an additional 637. This set was combined with a corpus of 6333 novels, including works such as *To The Lighthouse* by Virginia Woolf, not available on Project Gutenberg. Books were then partitioned into the decade in which they were au-

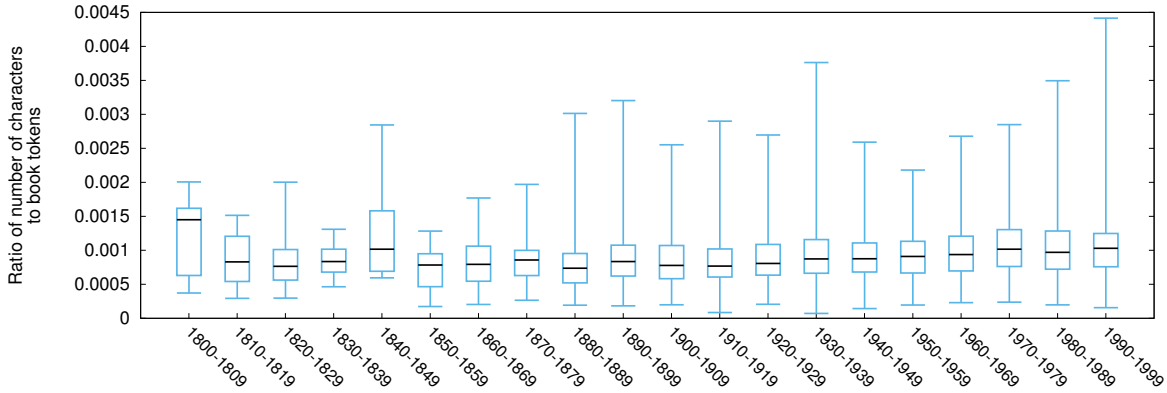

Figure 2: Distributions of the size-normalized number of characters per novel per decade.

thored. We limit our focus to trends starting in 1800 to 1990, when at least 11 books are available for each decade.

To account for variability in novel length, we normalize the novel's number of characters by its number of tokens. Figure 2 shows the box-and-whisker plot of the normalized number of characters per novel, where the box denotes the first and third quartile and the bar denotes the median. Surprisingly, we did not observe any significant change in the relative number of characters per novel, despite the underlying socio-economic changes that accompanied this time period. While novels written before 1850 had slightly more characters on average, this effect may be due to the smaller number of works available from this period. However, our finding raises many questions about whether the social networks for these characters obey similar trends in their size and density.

5 Conclusion

Although a fundamental task to character analysis, identifying the number of characters in a literary novel presents a significant challenge to current state of the art. To lay the foundation towards solving the task, we provide three contributions: (1) an annotated corpus of 58 books, (2) an evaluation framework for measuring performance on the task, (3) a new state-of-the-art method for character extraction. Furthermore, to promote future work we make all software and data available upon request.

6 Acknowledgements

We would like to thank the three annotators for their diligent reading and coding of novels.

References

Apoorv Agarwal and Owen Rambow. 2010. Automatic detection and classification of social events. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1024–1034.

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *the Proceedings of the 6th International Joint Conference on Natural Language Processing*.

Mariona Coll Ardanuy and Caroline Sporleder. 2014. Structure-based clustering of novels. In *Proceedings of the EACL Workshop on Computational Linguistics for Literature*, pages 31–39.

David Bamman, Ted Underwood, and Noah A Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 370–379.

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

Jens Eder, Fotis Jannidis, and Ralf Schneider. 2010. *Characters in fictional worlds: Understanding imaginary beings in literature, film, and other media*, volume 3. Walter de Gruyter.

David K Elson, Nicholas Dames, and Kathleen R McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Tiziano Flati and Roberto Navigli. 2013. Spred: Large-scale harvesting of semantic predicates. In *Proceedings of the The 51st Annual Meeting of the Association for Computational Linguistics*.

Hua He, Denilson Barbosa, and Grzegorz Kondrak. 2013. Identification of speakers in novels. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1312–1320.

Prashant Arun Jayannavar, Apoorv Agarwal, Melody Ju, and Owen Rambow. 2015. Validating literary theories using automatic social network extraction. In *Proceedings of the NAACL-2015 Workshop on Computational Linguistics for Literature*.

John Lee and Chak Yan Yeung. 2012. Extracting networks of people and places from literary texts. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Franco Moretti. 1999. *Atlas of the European novel, 1800-1900*. Verso.

Raymond Williams. 1975. *The country and the city*, volume 423. Oxford University Press.