The Role of Natural Language Processing Tasks in Automatic Literary Character Network Construction

Arthur Amalvy, Vincent Labatut, Richard Dufour

The Role of Natural Language Processing Tasks in Automatic Literary Character Network Construction

Arthur Amalvy
Laboratoire Informatique d'Avignon
arthur.amalvy@univ-avignon.fr

Vincent Labatut
Laboratoire Informatique d'Avignon
vincent.labatut@univ-avignon.fr

Richard Dufour
Laboratoire des Sciences du Numérique de Nantes
richard.dufour@univ-nantes.fr

Abstract

The automatic extraction of character networks from literary texts is generally carried out using natural language processing (NLP) cascading pipelines. While this approach is widespread, no study exists on the impact of low-level NLP tasks on their performance. In this article, we conduct such a study on a literary dataset, focusing on the role of named entity recognition (NER) and coreference resolution when extracting co-occurrence networks. To highlight the impact of these tasks' performance, we start with gold-standard annotations, progressively add uniformly distributed errors, and observe their impact in terms of character network quality. We demonstrate that NER performance depends on the tested novel and strongly affects character detection. We also show that NER-detected mentions alone miss a lot of character co-occurrences, and that coreference resolution is needed to prevent this. Finally, we present comparison points with 2 methods based on large language models (LLMs), including a fully end-to-end one, and show that these models are outperformed by traditional NLP pipelines in terms of recall.

1 Introduction

Character networks are graphs whose vertices represent characters, and edges represent the relationships between them. They can be seen as a special case of knowledge graphs, where vertices are restricted to being characters. Such networks have multiple uses: visualize the relationships between characters in a novel, support literary analysis (Rochat, 2015; Rochat and Triclot, 2017; Elson et al., 2010), or solve downstream tasks such as recommendation (Lee and Jung, 2019) or genre classification (Hettinger et al., 2015). As indicated by the survey of Labatut and Bost (2019), many authors work on automatically extracting these networks (Sparavigna and Marazzato, 2015; Dekker et al., 2019; Elson et al., 2010), following a generic three-phase automatic extraction framework. First, identify characters present in a text, using techniques such as NER and alias resolution. Second, detect interactions between characters. One can consider multiple types of interactions (co-occurrence, conversations, actions...), hence this process differs given the targeted type. Third, given characters and their interactions, derive the relationships between characters and extract a character network.

Due to their statistical nature and the difficulty of the tasks they entail, natural language processing (NLP) cascading pipelines applied to character network extraction are bound to make errors. While such networks have been leveraged for many applications, the general question of how to better extract them has been comparatively much less explored, and certainly not in a systemic way. Since extraction is generally performed using an NLP pipeline, the first step to answering this question is to study the impact of each NLP task on the quality of the final network, as understanding it would allow the community to prioritize future research efforts. Therefore, in this article, we propose to study that impact in depth, by artificially adding errors in steps of the extraction pipeline to observe their influence. We specifically focus on NER and coreference resolution, with the latter remaining a challenging task. We perform our study on Litbank (Bamman et al., 2019, 2020), a standard English literary corpus. To support our experiments, we implement extensions to the recent modular character network extraction pipeline Renard (Amalvy et al., 2024). In order to understand whether cascading pipelines are still competitive against LLM-based extraction systems and to guide future research, we compare our pipeline against such systems. To facilitate reproducibility, we release all our code and data under a free license[1].

[1] https://github.com/CompNet/Splice

Our contributions are as follows. First, we define new measures to evaluate the quality of extracted networks. Second, we extend the existing Renard extraction pipeline, and evaluate it on a character network dataset we adapt from Litbank. Our measures and dataset are a first step towards a more systematic benchmarking of network extraction systems, something that is missing in the literature. Third, we propose a model to simulate errors for two tasks, NER and coreference resolution, in order to understand their impact on a character network extraction pipeline. Finally, we compare our pipeline to end-to-end LLM models, in order to understand how much cascading errors are detrimental to a sequential pipeline.

We organize the rest of this article as follows. In Section 2, we discuss related work by highlighting existing character network extraction pipelines and literature on NER and coreference resolution error analysis. In Section 3, we detail our methods, including the extended Renard character network extraction pipeline we use. We describe our experiments in Section 4, and discuss their results in Section 5. Finally, we review our main contributions in Section 6 and present the limits of our work in Section 7.

2 Related Work

2.1 Character Network Extraction Pipelines

Character network extraction is related to knowledge graph extraction, but with vertices restricted to characters. Both share common tasks such as entity recognition and linking, but the focus on narratives and characters implies specialized instances with specific challenges.

BookNLP (Bamman et al., 2014) is a well-known NLP pipeline specialized for novels, and is sometimes used in the literature when it comes to extracting character networks (Dekker et al., 2019; Piper et al., 2017). Other authors go further and propose pipelines specifically tailored to character network extraction, such as *CHAPLIN* (Sparavigna and Marazzato, 2015) or *Charnetto* (Métrailler, 2023). Recently, Amalvy et al. (2024) propose *Renard*, a modular character network extraction pipeline written in Python. In this article, we extend Renard to conduct a detailed study of each extraction module.

2.2 Error Analysis

Most works interested in the effect of NLP errors focus on specific tasks. For the NER task, Stanislawek et al. (2019) find that different NER models make different categories of errors, while Rueda et al. (2024) highlight recurrent errors made by models, such as the difficulty of detecting mentions unseen in the training set. For the coreference task, Martschat and Strube (2014) focus on recall errors, while Chai and Strube (2023) perform an analysis on multilingual coreference systems, and focus on two-mentions entities that they find hard to recall.

To the best of our knowledge, only Dekker et al. (2019) adopt a more global view and assess the effect of NER errors on character networks. However, no study exists on the impact of the performance of the main NLP steps required to extract a character network. Our goal in this article is to fill this existing void in the literature by proposing a first impact study.

3 Methods

3.1 Terminology

The terminology from the NER, coreference and alias resolution literature diverge and are confusing when used together. This is why, in this section, we clarify the terms that we use in this article. We use *"form"* to refer to a textual representation of a character. A form can be a proper noun (*"Lianna"*), a pronoun (*"she"*), a definite description (*"the princess"*)... Meanwhile, we use *"mention"* to refer to the occurrence of a form in the text. A NER system only extracts a subset of characters' mentions: for example, it does not extract pronouns. We refer to the form of a mention detected by a NER model as an *alias*, as it strongly identifies a character. Meanwhile, a coreference system typically detects all mentions, including pronouns and other generic constructs. We therefore distinguish two types of mentions: *alias mentions* and *generic mentions*.

3.2 Extraction Pipeline

To extract character networks, we extend the Renard extraction pipeline (Amalvy et al., 2024). We design our own pipeline for the needs of this study and contribute different modules. There are many types of interactions that we could extract to produce character networks. As a first study on the subject, we choose to focus on co-occurrence character networks: they are conceptually simple, and

are the most used type of networks in the literature (Labatut and Bost, 2019). We consider an interaction between two characters when they appear close to each other in the text, in a range we call the *co-occurrence window*. Our pipeline is divided into four main phases: NER, coreference resolution (optional step), character unification, and finally co-occurrence detection and network extraction.

We perform flat NER using the fine-tuned BERT model (Devlin et al., 2019) included in Renard, trained on the literary NER dataset introduced by Dekker et al. (2019) and later improved by Amalvy et al. (2023b). We only keep mentions of the PER class.

For coreference resolution, we use the end-to-end coreference model included in Renard based on Lee et al. (2017) and Joshi et al. (2019). The model predicts links between mentions, but also performs mention detection: this is important when extracting co-occurrence character networks, as generic character mentions (such as pronouns) are still counted as co-occurrences.

Character unification resembles *alias resolution*. In the case of our extraction pipeline, we define character unification as resolving each mention detected by the NER and coreference steps to a single character. This task could be described as a document-level version of coreference resolution, restricted to characters. To unify mentions, we base ourselves on the work of Vala et al. (2015). We construct a graph where each vertex is a character alias as detected by NER, and we employ a set of rules to connect or disconnect these vertices. While rules can introduce errors, they are often used by previous works (Vala et al., 2015; Ardanuy and Sporleder, 2014), and thus analyzing their failure modes is important. We use the following rules:

1. When two aliases have a first or last name in common, we connect them (*"Emma"* and *"Emma Woodhouse"*).

2. When two aliases are related by a hypocorism gazetteer (*"John"* and *"Johnny"*), we connect them.

3. When one of the two above rules holds for two aliases when removing titles, we connect them (*"Mr. John"* and *"Johnny"*).

4. When two aliases are coreferential, we connect them. We consider two aliases to be coreferential when they appear together in one or more coreference chains, and never appear without the other in other chains.

5. When connected aliases have the same last name but a different first name, we delete all vertices in the shortest paths between them, since they are probably different characters from the same family (*"John Smith"* and *"John Klint"*).

6. When two aliases have a different inferred gender, we delete all the edges in the shortest paths between them (*"Mr. Smith"* and *"Miss Smith"*). We infer gender using the gendered titles and pronouns in coreference chains.

After having applied all these rules, we merge the graph-connected components. Using the alias groups extracted with this algorithm, we assign each mention detected by the NER or coreference steps to a single character.

Finally, we apply the co-occurrence detection and network extraction step of Renard. This step is entirely deterministic and cannot cause any errors errors by itself: we simply consider that two character mentions in the defined co-occurrence window form an interaction, which results in an edge between these characters. To take into account the importance of each relationship, we weight edges by the number of interactions between characters.

3.3 Perturbation Analysis

To assess the impact of NER and coreference resolution errors on the extracted networks, we propose to start from a pipeline with gold-standard NER and coreference predictions, and to progressively degrade the performance of these tasks while observing the impact on the quality of the extracted networks. To degrade task performance, we add uniformly distributed perturbations to the predictions, corresponding to different types of errors.

3.3.1 NER Perturbations

As an example, we consider the following gold predictions as a starting point, and consider two types of perturbations.

Add Spurious Alias Mentions: We add false positives to the NER predictions by uniformly sampling generic spans (up to a certain span size) from the text, to reduce Precision.

Remove Correct Alias Mentions: We remove true positives from the NER predictions by uniformly sampling from the predicted alias mentions, to reduce Recall.

PER One-Eye | o looked | o at | o Goblin | o .

3.3.2 Coreference Resolution Perturbations

As an example, we consider the following gold prediction as a starting point, and consider four types of perturbations.

1 One-Eye pranced over and took a poke at 2 Goblin, trying to break 2 his concentration.

Add Spurious Mentions: We add singletons (mentions linked to no other mentions) to the predictions consisting of incorrect mentions, by uniformly sampling non-mention spans (up to a certain span size).

1 One-Eye pranced over and took a 3 poke at 2 Goblin, trying to break 2 his concentration.

Remove Correct Mentions: We remove correctly predicted mentions from the predictions by uniform sampling.

One-Eye pranced over and took a poke at 2 Goblin, trying to break 2 his concentration.

Add Spurious Links: We add incorrect coreference links between two mentions, wrongly merging coreference chains together. We uniformly sample the incorrect links in the set of all possible incorrect links.

2 One-Eye pranced over and took a poke at 2 Goblin, trying to break 2 his concentration.

Remove Correct Links: We remove correct links between predicted mentions, wrongly splitting coreference chains. We uniformly sample links among all existing correct coreference links.

1 One-Eye pranced over and took a poke at 2 Goblin, trying to break 3 his concentration.

3.4 Network Quality Measures

Since we want to measure the impact of NLP errors on the extracted network, we need a set of measures to assess the quality of this network when compared to a reference network. We base our measures on the work of Vala et al. (2015) on alias resolution.

Let $G_p = (V_p, E_p)$ be a predicted character network, and $G_g = (V_g, E_g)$ be the corresponding gold network. Let each vertex of V_p and V_g represent the set of aliases $\{a_1, a_2, \cdots, a_n\}$ of the underlying character. In order to know whether the predicted network G_p correctly contains vertices and edges similar to the gold network G_g, we first need to match their characters, since a vertex in V_p is not necessarily present in V_g and vice versa. Thus, we start by computing a maximum bipartite mapping f_V from the set of predicted vertices V_p to $V_g \cup \{v_\varnothing\}$. This mapping associates any predicted vertex $u \in V_p$ to a gold vertex $v \in V_g$ or to the null vertex $v_\varnothing$, meaning u is not associated with any character in V_g. Note that the null vertex $v_\varnothing$ represents the empty set of aliases. Symmetrically, we construct a mapping g_V from V_g to $V_p \cup \{v_\varnothing\}$. We leverage the alias sets represented by the vertices to compute Vertex Precision and Vertex Recall[2]:

$$Pre_V = \max_{f_V} \frac{\sum_{u \in V_p} 1 - \frac{|u - f_V(u)|}{|u|}}{|V_p|} \quad (1)$$

$$Rec_V = \max_{g_V} \frac{\sum_{v \in V_g} [g_V(v) \cap v \neq v_\varnothing]}{|V_g|}. \quad (2)$$

We define Vertex F1 ($F1_V$) as the harmonic mean between Vertex Precision and Vertex Recall.

For edges, we use mappings f_V and g_V to construct mappings f_E and g_E, which map similarly edge sets E_p and E_g:

$$f_E(\{u, v\}) = \begin{cases} \{f_V(u), f_V(v)\}, & \text{if } f_V(u) \neq v_\varnothing \\ & \text{and } f_V(v) \neq v_\varnothing \\ v_\varnothing, & \text{otherwise.} \end{cases}$$

Based on f_E and g_E, we compute Edge Precision and Edge Recall:

$$Pre_E = \max_{f_E} \frac{\left|\{f_E(e) : e \in E_p\} \cap E_g\right|}{|E_p|} \quad (3)$$

$$Rec_E = \max_{g_E} \frac{\left|E_p \cap \{g_E(e) : e \in E_g\}\right|}{|E_g|}. \quad (4)$$

We define Edge F1 ($F1_E$) as the harmonic mean of Edge Precision and Edge Recall.

We also introduce weighted variants of these network measures ($WPre_E$, $WRec_E$ and $WF1_E$) in order to take into account the weights of the network edges, that correspond to the number of interactions between connected characters. Before computing the measures, we normalize the weights by

[2]We employ the Iverson bracket notation, where $[P] = 1$ if proposition P is true, and 0 otherwise.

dividing by the maximal number of co-occurrences in the network. We compute Precision and Recall as follows:

$$WPre_E = \max_{f_E} \frac{\sum_{e \in E_p} 1 - \left|w(f_E(e)) - w(e)\right|}{|E_p|} \quad (5)$$

$$WRec_E = \max_{g_E} \frac{\sum_{e \in E_g} 1 - \left|w(e) - w(g_E(e))\right|}{|E_g|}, \quad (6)$$

where $w(e)$ is the function that computes the normalized weight of edge e. $w(e)$ is 0 when $e = e_\varnothing$. Weighted measures evaluate the quality of the distribution of weights in the predicted network, and are always less than or equal to their unweighted counterparts.

4 Experiments

4.1 Literary Corpus

We perform all of our experiments on the NER and coreference layers of the Litbank literary corpus (Bamman et al., 2019, 2020). Since it is designed for nested NER while the Renard NER step performs flat NER, we flatten the Litbank annotations using an algorithm we implement (see Appendix B for details). We use coreference chains of PER mentions as the ground truth for the character unification step, since coreference resolution on characters is equivalent to character unification. As the network extraction step cannot cause errors on its own, we extract gold character networks using these annotations only.

Litbank is composed of excerpts from 100 novels of approximately 2,000 tokens each. Since these excerpts can be short, we restrict our analysis to the 30 novels involving at least 10 characters. This prevents high deviation of network quality measures when a vertex or an edge is modified in the prediction. We use the remaining 70 novel excerpts to train coreference resolution models: we use 63 of these 70 novels (90%) as a training set, and the remaining 7 as a development set.

4.2 Pipeline Performance

We apply our character network extraction pipeline to the 30 excerpts we select for analysis. Since multiple coreference resolution measures exist and none of these are entirely satisfying or measure the same thing, we report a large set of measures including MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF (Luo, 2005), BLANC (Recasens and Hovy, 2011) and LEA (Moosavi and Strube, 2016). We also report the performance of a pipeline without the optional coreference resolution step, in order to assess the usefulness of the task. We consider a co-occurrence window of 32 tokens.

4.3 LLM-based Approaches

Since character network extraction is usually performed using cascading pipelines, errors may propagate and degrade performance. Meanwhile, LLM-based approaches are less modular and explainable, but are not subject to cascading errors by design. Therefore, inspired by the recent advances in LLMs, we survey their capability to be used as character network extractors, and compare them to our cascading Renard pipeline. We introduce two different LLM extraction methods:

LLM-Coref We remark that the last network extraction step, co-occurrence detection, cannot cause errors on its own: we simply create an edge between two characters if some of their mentions are in the same co-occurrence window. Therefore, in the case of a co-occurrence network, we can define the extraction problem as span extraction, where each extracted span must be assigned to the correct character. This problem definition could also be viewed as coreference resolution restricted to characters only. To tackle the problem this way, we prompt LLMs to mark character mentions in the text with a unique character ID.

LLM-E2E Given an input text, we simply prompt LLMs to produce the corresponding character network in a simplified version of the XML-based Graphml format.

See Appendix A for the exact prompts. We use few-shot prompting by providing examples of the task to the model. For both methods, we make an effort to parse the LLM output in order to fix slightly incorrect output format. We survey two proprietary models, GPT3.5 Turbo (Brown et al., 2020)[3] and GPT4o[4], and a recent open weights model, Llama3-8b-instruct (Touvron et al., 2023).

5 Results

5.1 Pipeline Performance

Table 1 shows the NER and coreference resolution performance of our extended Renard pipeline.

[3] GPT3.5 checkpoint: `gpt-3.5-turbo-0125`

[4] GPT4o checkpoint: `gpt-4o-2024-05-13`

NER performance is below the reported state-of-the-art on datasets from other domains such as CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) where the best systems obtain F1 scores higher than 90. While part of this lack of performance may be due to the way we transform the nested Litbank dataset to a flat NER dataset, we observe a high disparity of performance between novels, with F1 ranging from 51.16 to 97.30. This matches the observations from Dekker et al. (2019) and Amalvy et al. (2023a), indicating challenges that are specific to some novels. Meanwhile, the performance of coreference resolution is lower than what Bamman et al. (2020) reports on Litbank (for example, we report 78.45 MUC while Bamman et al. (2020) reports 84.3). This may be due to the lower number of excerpts in our training set (63 vs. 80), which is required for our analysis.

Task	Measure	Mean	Min	Max
NER	F1	79.58	51.16	97.30
Coref	MUC	78.45	64.31	88.75
	B3	54.87	41.53	70.40
	CEAF	47.04	34.31	59.75
	BLANC	60.82	40.64	79.82
	LEA	28.84	19.44	43.95

Table 1: Performance of our pipeline on NER and coreference resolution. We compute the Mean, Min and Max values on the series formed by the measures of the 30 novel excerpts of our analysis set.

Table 2 shows the performance of our pipeline on our test corpus of 30 excerpts, depending on whether we add the optional coreference step or not. Vertex and edge F1 are higher when omitting coreference information, likely because the performance of the coreference resolution algorithm is not high enough, leading to the detection of spurious mentions, which misleads both the character unification and co-occurrence detection steps. We discuss the question of the utility of coreference resolution in more detail in Section 5.1.1. In general, Edge Recall is quite low, meaning many character interactions are missed.

Meanwhile, using coreference information proves to be important to increase edge recall measures. Even though coreference resolution leads to a compromised network structure overall, it allows the pipeline to detect more character mentions, leading to a better estimation of the relative strength of their interactions.

Measure	w/ coref	w/o coref
$F1_V$	57.64	**70.39**
$F1_E$	40.19	**44.93**
$WF1_E$	**33.53**	30.55
Pre_V	59.32	**68.99**
Pre_E	48.07	**62.07**
$WPre_E$	38.40	**39.91**
Rec_V	57.77	**74.00**
Rec_E	**39.37**	37.81
$WRec_E$	**33.17**	26.18

Table 2: performance of our pipeline on network extraction with or without the coreference resolution step.

5.1.1 Coreference Resolution Performance

Given the negative impact that coreference resolution can have, as seen in Table 2, a question that naturally arises is whether this task is useful when extracting character networks. Co-occurrence networks extracted with alias mentions might only be a sufficiently good approximation of a network extracted with all mentions. In that case, performing a coreference resolution to extract additional mentions would not be critical.

Measure	Rec_E	$WPre_E$	$WRec_E$
Value	54.39	63.20	35.66

Table 3: Network quality measures for gold networks extracted by ignoring coreference mentions. Only affected measures are presented.

To understand whether this is the case, we extract gold networks from Litbank with and without coreference-extracted mentions, and compute network quality measures by considering networks with coreference mentions as the reference. Results can be found in Table 3. While only some measures are affected (Edge Recall, Weighted Edge Precision, and Weighted Edge Recall), ignoring coreference mentions proves to severely impact performance. This shows that coreference mentions are a crucial part of relationships extracted using co-occurrence.

To try and understand the minimal performance needed for coreference resolution to be useful, we perform an experiment where we inject the gold coreference information in the pipeline, and slowly degrade performance. To do so, we combine all the coreference perturbations we described in Section 3.3.2, by uniformly sampling a degradation and applying it. We repeat this process for a fixed

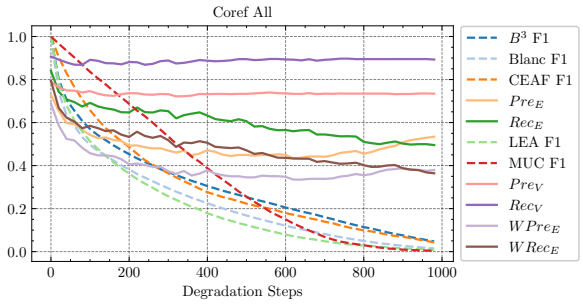

Figure 1: Network extraction performance when applying the coreference perturbations from Section 3.3.2.

number of steps and observe the effects in terms of coreference and network quality measures. The results of this experiment can be found in Figure 1.

When applying perturbations, overall performance for all measures starts to decrease until a certain point, after which it starts reaching a plateau (except for edge recall measures). This effect occurs because after this point, coreference resolution recall on alias mentions starts being so low that it does not affect the character unification algorithm anymore, and other rules based on character aliases start to have more influence. Meanwhile, there is a big discrepancy between vertex and edge measures: while vertex measures can stay close to their original values when coreference resolution is highly degraded, this is not the case for edge measures. This is because coreference resolution is crucial when detecting co-occurrence, as it is the only way to detect generic mentions.

5.1.2 NER Perturbation Analysis

Figure 2 shows the NER perturbation results.

Add Spurious Alias Mentions Unsurprisingly, reducing NER Precision has a direct effect on Vertex Precision, which plummets as more NER false positives are added. Edge Precision also sharply decreases as a result, while Vertex and Edge Recall slowly decrease down to a plateau.

Remove Correct Alias Mentions All recall measures sharply decrease when removing correct alias mentions. Precision measures become unstable and finally undefined, as no vertices or edges are predicted when NER Recall reaches 0.

Unsurprisingly, NER performance has a high impact on network quality. Since NER performance varies greatly depending on the novels (as seen in Table 1), enhancing performance for challenging novels is an important concern that should be addressed by future research.

5.1.3 Coreference Resolution Perturbation Analysis

Figure 3 shows the coreference resolution perturbations results.

Add Spurious Mentions Adding spurious singletons does not affect our character unification algorithm, and therefore has no impact on the quality of the extracted network.

Remove Correct Mentions Removing correct mentions mainly affects edge measures: characters are still recognized correctly, but some co-occurrence interactions are lost due to missing character mentions, leading to fewer edges.

Add Spurious Links Adding wrong coreference links sharply decreases all network extraction performance measures: characters are harder to recognize, and interactions are missing. This is driven by rules 4 and 6 of our character unification step (Section 3.2), that are both fed wrong information. While it would be possible to not apply these rules, rule 4 is the only one that allows linking two mentions with completely different forms.

Remove Correct Links Network edge measures are affected the most by coreference links removal, while vertex measures stay somewhat stable.

Not all coreference resolution errors prove to be equal in terms of impact on network quality. Adding spurious links is the most harmful error, while adding spurious singletons does not affect our character unification algorithm. Meanwhile, other errors mainly impact edge extraction performance. Therefore, if one's concern is to extract characters only, a conservative coreference algorithm with low linking recall, but high linking precision might give sufficient performance. However, when extracting edges between characters, both high precision and recall are necessary, in terms of mention detection as well as linking.

5.2 LLM-based Approaches

Results of our LLM-based extraction methods *LLM-Coref* and *LLM-E2E* can be found in Table 4.

LLM-Coref If we observe F1 scores, GPT4o performs the best amongst the LLMs we survey, followed by Llama3-8b-instruct and GPT-3.5 Turbo. LLMs particularly struggle with recall, with GPT3.5 Turbo and Llama3-8b-instruct missing a

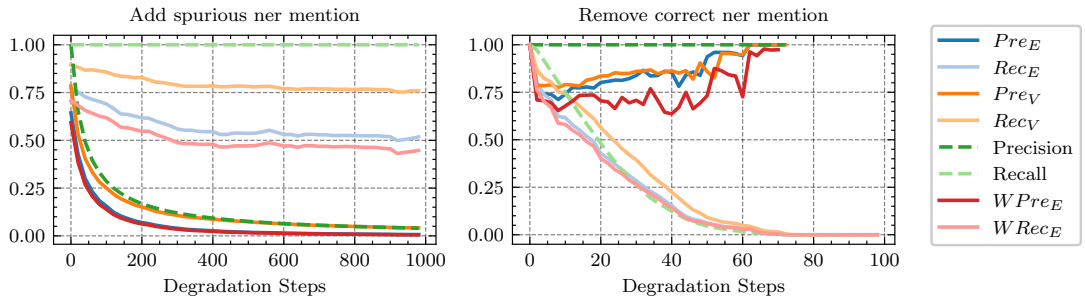

Figure 2: Network quality measures versus number of degradation steps for "add spurious alias mention" and "remove correct alias mention" perturbations.

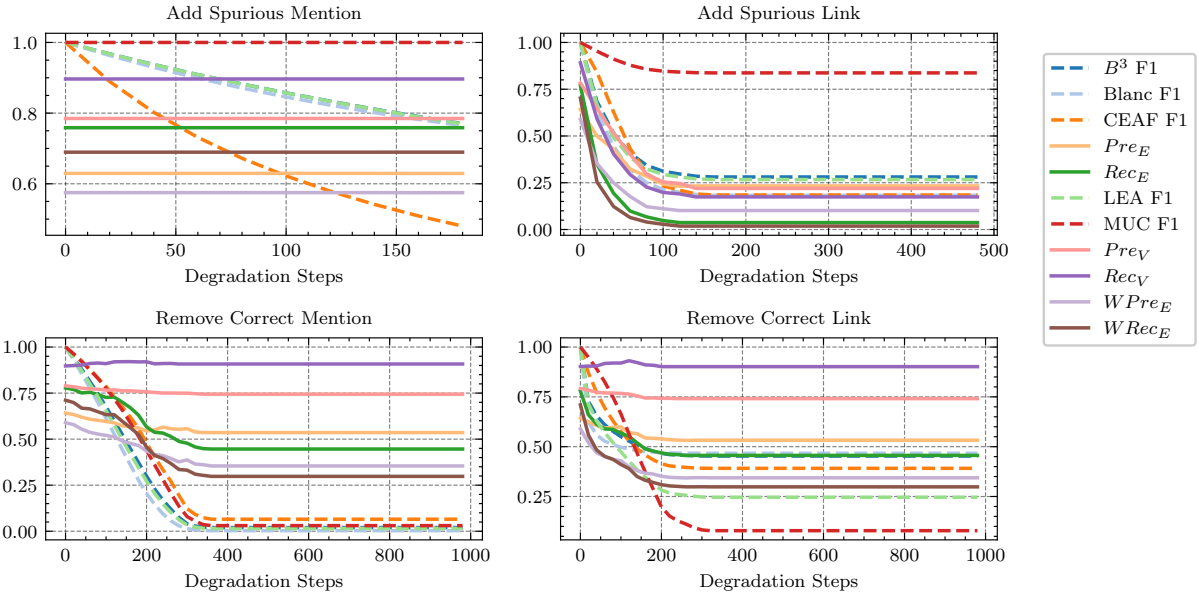

Figure 3: Network quality measures versus the number of coreference resolution degradation steps.

lot of character occurrences. Qualitatively, both of these models either miss a lot of mentions, or start hallucinating a lot of mentions before continuing that way through the output, influenced by their previous predictions. Llama3-8b-instruct also has trouble respecting the output format, sometimes generating invalid output. GPT4o is much more consistent, although it still misses many mentions. When compared against our extended Renard pipeline, LLMs results are generally lower, except for Edge Precision and for weighted edge measures where GPT4o trades precision for recall.

LLM-E2E Results are generally higher than with *LLM-Coref*, highlighting the importance of task formulation. If we focus on F1 scores, we observe the same ranking between LLMs, with GPT4o beating Llama3-8b-instruct and GPT3.5 Turbo. Generally, LLMs display high vertex and edge precision, even surpassing our pipeline sometimes. However, their recall still lags behind Renard.

6 Conclusion

In this article, we presented a study on the impact of NLP tasks on character network extraction. We show that NER performance is crucial to detecting characters, but that it depends heavily on the considered novel: therefore, future research should focus on improving the low NER performance on difficult novels. Additionally, we show that not tackling the challenging coreference resolution task implies missing co-occurrence relationships between characters. This task is important to extract correct co-occurrence edges, particularly when it comes to edge weights. Since it remains difficult in general, our extraction pipeline exhibits relatively low edge extraction performance. We also show that not all coreference errors have the same impact: adding spurious coreference links between mentions has the strongest negative impact of all the errors we

	LLM-Coref			*LLM-E2E*			*Renard*
Measure	**Llama3**	**GPT-3.5T**	**GPT4o**	**Llama3**	**GPT-3.5T**	**GPT4o**	
$F1_V$	37.93	28.99	52.32	56.87	44.26	62.98	**70.39**
$F1_E$	23.20	16.96	38.85	29.35	20.91	30.42	**44.93**
$WF1_E$	15.17	11.39	**32.72**	20.17	14.33	24.35	30.55
Pre_V	42.86	52.50	68.78	67.96	61.04	**77.96**	68.99
Pre_E	57.85	**65.78**	62.62	59.01	64.39	65.76	62.07
$WPre_E$	34.69	37.25	51.46	40.56	44.75	**53.97**	39.91
Rec_V	25.12	22.15	32.28	53.18	37.87	34.24	**70.39**
Rec_E	10.34	9.18	23.38	21.32	13.24	13.12	**37.81**
$WRec_E$	6.86	6.34	19.77	14.86	9.14	10.44	**26.18**

Table 4: Comparison between the network extraction performance of LLM-based extraction methods and our Renard pipeline. *Llama3* stands for Llama3-8b-instruct, *GPT-3.5T* for GPT-3.5 Turbo.

surveyed. Developing systems able to make conservative predictions when it comes to coreference linking might be a good research direction to create better character network extraction systems. Unfortunately, developing coreference models at the scale of a novel remains difficult due to the lack of fully annotated ones for training and benchmarking, which prompts the development of datasets with long documents.

Even though errors propagate in cascading pipelines, our pipeline generally outperformed LLM-based approaches. However, the performance of these approaches is encouraging given the fact that we only evaluated the few-shot prompting setting. Fine-tuning LLMs on character network extraction is therefore a promising direction of research, even though pipelines remain more interpretable.

7 Limitations

- While the character network extraction pipeline we use is inspired by the generic framework outlined by Labatut and Bost (2019), we still had to make implementation choices. Other pipelines may behave differently regarding task errors, although we hypothesize that similar architectures should yield similar results.
- Our perturbation analysis methodology may not reflect the distribution of errors from existing models. However, it allows considering the different types of possible errors.
- For end-to-end extraction using LLMs, we only survey the few-shot prompting setting due to resource limitations. Fine-tuning a model may yield better results. However, the excerpts on which we perform analysis are short (approximately 2,000 tokens) compared to full-scale novels: the performance of LLMs in that setting may not be as high as the results we report in our study.

References

A. Amalvy, V. Labatut, and R. Dufour. 2023a. Learning to rank context for named entity recognition using a synthetic dataset. In *2023 Conference on Empirical Methods in Natural Language Processing*, pages 10372–10382.

A. Amalvy, V. Labatut, and R. Dufour. 2023b. The role of global and local context in named entity recognition. In *61st Annual Meeting of the Association for Computational Linguistics*, page 714–722.

A. Amalvy, V. Labatut, and R. Dufour. 2024. Renard: A Modular Pipeline for Extracting Character Networks from Narrative Texts. *HAL*.

M. C. Ardanuy and C. Sporleder. 2014. Structure-based clustering of novels. In *3rd Workshop on Computational Linguistics for Literature*, pages 31–39.

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *1st International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

D. Bamman, O. Lewke, and A. Mansoor. 2020. An annotated dataset of coreference in English literature. In *12th Language Resources and Evaluation Conference*, pages 44–54.

D. Bamman, S. Popat, and S. Shen. 2019. An annotated dataset of literary entities. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2138–2144.

D. Bamman, T. Underwood, and N. A. Smith. 2014. A bayesian mixed effects model of literary character. In *52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 370–379.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

H. Chai and M. Strube. 2023. Investigating multilingual coreference resolution by universal annotations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10010–10024.

N. Dekker, T. Kuhn, and M. van Erp. 2019. Evaluating named entity recognition tools for extracting social networks from novels. *PeerJ Computer Science*, 5:e189.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186.

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

L. Hettinger, M. Becker, I. Reger, F. Jannidis, and A. Hotho. 2015. Genre classification on german novels. In *26th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 249–253.

M. Joshi, O. Levy, L. Zettlemoyer, and D. Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5803–5808.

V. Labatut and X. Bost. 2019. Extraction and analysis of fictional character networks : A survey. *ACM Computing Surveys*, 52:89.

K. Lee, L. He, M. Lewis, and L. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

O.-J. Lee and J. J. Jung. 2019. Modeling affective character network for story analytics. *Future Generation Computer Systems*, 92:458–478.

X. Luo. 2005. On coreference resolution performance metrics. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

S. Martschat and M. Strube. 2014. Recall error analysis for coreference resolution. In *2014 Conference on Empirical Methods in Natural Language Processing*, pages 2070–2081.

N. S. Moosavi and M. Strube. 2016. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 632–642.

C. Métrailler. 2023. Charnetto.

A. Piper, M. Algee-Hewitt, K. Sinha, D. Ruths, and H. Vala. 2017. Studying literary characters and character network. In *Digital Humanities*.

M. Recasens and E. Hovy. 2011. BLANC: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17:485–510.

Y. Rochat. 2015. Character network analysis of Émile Zola's Les Rougon-Macquart. In *Digital Humanities 2015*.

Y. Rochat and M. Triclot. 2017. Les réseaux de personnages de science-fiction : échantillons de lectures intermédiaires. *ReS Futurae*, 10:1183.

A. Rueda, E. Alvarez-Mellado, and C. Lignos. 2024. CoNLL#: Fine-grained error analysis and a corrected test set for CoNLL-03 English. In *2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3718–3728.

A. C. Sparavigna and R. Marazzato. 2015. Analysis of a play by means of chaplin, the characters and places interaction network software. *International Journal of Sciences*, 4(3):60–68.

T. Stanislawek, A. Wróblewska, A. Wójcicka, D. Ziembicki, and P. Biecek. 2019. Named entity recognition - is there a glass ceiling? In *23rd Conference on Computational Natural Language Learning*, pages 624–633.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *7th Conference on Natural Language Learning*, pages 142–147.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. 2023. Llama: Open and efficient foundation language models. *arXiv*, cs.CL:2302.13971.

H. Vala, D. Jurgens, A. Piper, and D. Ruths. 2015. Mr. bennet, his coachman, and the archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Conference on Empirical Methods in Natural Language Processing*, pages 769–774.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *6th Message Understanding Conference*.

A Large Language Models Prompt

We use the following prompt for our *LLM-Coref* and *LLM-E2E* extraction methods respectively:

SYSTEM PROMPT You are an expert in literature and natural language processing.
USER PROMPT Given a text, you must extract characters and their mentions. Your answer must be the original text, where character mentions are tagged with the following format: [CHARACTER_ID]CHARACTER MENTION[/CHARACTER_ID]. You must tag character mentions only.

Here are some examples of this task:
Example 1:
Input: Elric was riding his horse . Alongside Moonglum , the prince of ruins was looking for his dark sword .
Output: [0] Elric [/0] was riding [0] his [/0] horse . Alongside [1] Moonglum [/1] , the [0] prince of ruins [/0] was looking for [0] his [/0] dark sword .

Example 2:
Input: Princess Liana felt sad , because Zarth Arn was gone . The princess decided she should sleep .
Output: [0] Princess Liana [/0] felt sad , because [1] Zarth Arn [/1] was gone . [0] The princess [/0] decided [0] she [/0] should sleep .

SYSTEM PROMPT You are an expert in literature and natural language processing.
USER PROMPT Given a text, you must extract a co-occurrence character network where vertices represent characters and edges represent their relationships. Each edge must have a weight corresponding to the number of interactions between two characters. Two characters without any interactions do not share an edge. An interaction between two characters occurs when two of their mentions occur within a distance of 32 tokens. Your answer must be in a simplified Graphml-like format. Vertices must have an 'alias' attribute with the list of aliases of a character, separated by semicolons.

Here are some examples of this task:
Example 1:
Input: Elric was riding his horse . Alongside Moonglum , the prince of ruins was looking for his dark sword .
Output:

```
<graph>
<node id="n0"
 aliases="Elric;prince of ruins">
</node>
<node id="n1" aliases="Moonglum">
</node>
<edge id="e0" source="n0"
 target="n1"
 weight="2">
</edge>
</graph>
```

Example 2:
Input: Princess Liana felt sad , because Zarth Arn was gone . Liana decided she should sleep .
Output:

```
<graph>
<node id="n0"
 aliases="Princess Liana;Liana">
</node>
<node id="n1" aliases="Zarth Arn">
</node>
<edge id="e0" source="n0" target="n1"
 weight="2">
</node>
</graph>
```

B Adapting Litbank to Flat NER

In the main body of this article, we perform our experiments on the Litbank dataset (Bamman et al., 2019). However, Litbank NER annotations are nested, while the Renard NER step we employ performs flat NER. Annotations guidelines are also different between Litbank and the original NER dataset on which the Renard NER step is trained (Amalvy et al., 2023b). In particular, Litbank annotates many generic mentions (such as *"an honourable man"*) as alias mentions. We there-

fore flatten Litbank annotations using an algorithm we implement, while trying to respect the original annotation guidelines as much as possible.

Litbank annotations consists of 4 layers of nesting, where annotated alias mentions can overlap. Our flattening algorithm works as follows: first, we try to cut annotated mentions to a form that would be accepted as an alias mention in the dataset of Amalvy et al. (2023b). We do so by removing leading determiners (*"the"*) and cutting the content of a mention after the first comma. Afterward, we filter mentions that are still deemed generic, by checking if their constituting tokens are capitalized (except for some stopwords). If some alias mentions are still overlapping at this point, we select the outermost ones.

To give an example, the annotated mention *"the Lord High Chancellor"* would have been shortened as *"Lord High Chancellor"*, and then accepted as an alias mention since all of its tokens are capitalized (provided it does not overlap with a larger mention). By contrast, the generic mention *"an honourable man"* would have been discarded since its tokens are not capitalized.