The Journal of Open Source Software

Renard: A Modular Pipeline for Extracting Character Networks from Narrative Texts

Arthur Amalvy [1], **Vincent Labatut** [1], **and Richard Dufour** [2]

1 Laboratoire Informatique d'Avignon **2** Laboratoire des Sciences du Numérique de Nantes

DOI: N/A

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: 01 January 1970

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Renard (*Relationships Extraction from NARrative Documents*) is a Python library that allows to define custom natural language processing (NLP) pipelines to extract character networks from narrative texts. Contrarily to the few existing tools, Renard can extract *static* as well as *dynamic* networks. Renard pipelines are modular: the user can choose the implementation of each NLP subtask needed to extract a character network. This allows to specialize pipelines to particular types of texts and to study the impact of each subtask on the extracted network.

Statement of Need

Character networks (that is, graphs where nodes represent characters and edges represent their relationships) extracted from narrative texts are useful in a number of applications, from visualization to literary analysis (Labatut & Bost, 2019). There are different ways of modeling relationships (co-occurrences, conversations, actions…), and networks can be static or dynamic (i.e. series of networks representing the evolution of relationships through time). This variety means one can extract different kinds of networks depending on the targeted applications. While some authors extract these networks by relying on manually annotated data (Park, Kim, & Cho, 2013; Park, Kim, Hwang, et al., 2013; Yannick Rochat, 2014; Y. Rochat, 2015; Y. Rochat & Triclot, 2017), it is a time-costly endeavor, and the fully automatic extraction of these networks is therefore of interest. Unfortunately, there are only a few existing software and tools that can extract character networks (Marazzato & Sparavigna, 2014; Métrailler, 2023), and none of these can output dynamic networks. Furthermore, automatically extracting a character network requires solving several successive natural language processing tasks, such as named entity recognition (NER) or coreference resolution, and algorithms carrying these tasks are bound to make errors. To our knowledge, the cascading impact of these errors on the quality of the extracted networks has yet to be studied extensively. This is an important issue since knowing which tasks have more influence on the extracted networks would allow prioritizing research efforts.

Renard is a fully configurable pipeline that can extract static and dynamic networks from narrative texts. We design it so that it is as modular as possible, which allows the user to select the implementation of each extraction step as needed. This has several advantages:

1. The pipeline can be specialized for a specific type of texts, allowing for better performance.
2. The pipeline can easily incorporate new advances in NLP, by simply implementing a new step when necessary.
3. One can study the impact of the performance of each step on the quality of the extracted networks.

Design and Main Features

Renard is centered about the concept of *pipeline*. In Renard, a pipeline is a series of sequential *steps* that are run one after the other in order to extract a character network from a text. When using Renard, the user simply *describes* this pipeline in Python by specifying this series of steps, and can apply it to different texts afterwards. The following code block examplifies that philosophy:

```
from renard.pipeline import Pipeline
from renard.pipeline.tokenization import NLTKTokenizer
from renard.pipeline.ner import NLTKNamedEntityRecognizer
from renard.pipeline.character_unification import GraphRulesCharacterUnifier
from renard.pipeline.graph_extraction import CoOccurrencesGraphExtractor

with open("./my_doc.txt") as f:
    text = f.read()

pipeline = Pipeline(
    [
        NLTKTokenizer(),
        NLTKNamedEntityRecognizer(),
        GraphRulesCharacterUnifier(min_appearance=10),
        CoOccurrencesGraphExtractor(co_occurrences_dist=10)
    ]
)

out = pipeline(text)
```

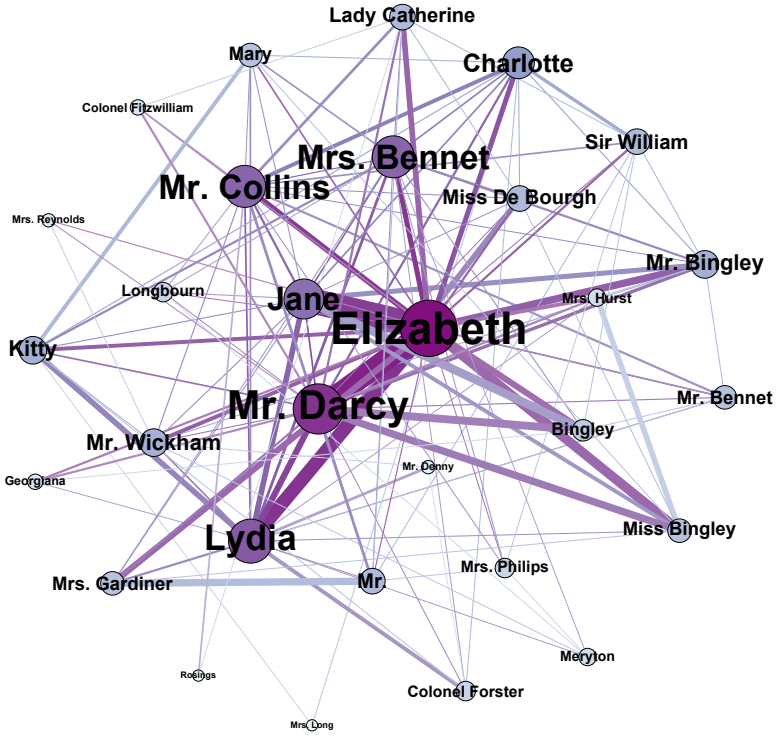

Figure 1: Co-occurrence character network of "Pride and Prejudice", extracted automatically using Renard. Node size denotes degree, while edge size denotes the number of co-occurrences between two character.

As an example, Figure 1 shows the co-occurrence character network of the 1813 novel from Jane Austen "Pride and Prejudice" extracted using the Renard pipeline above. Renard uses the NetworkX Python library (Hagberg et al., 2008) to manipulate graphs, ensuring compatibility with a wide array of tools and formats.

To allow for custom needs, we design Renard to be very flexible. If a step is not available in Renard, we encourage users to either:

- Externally perform the computation corresponding to the desired step, and inject the results back into the pipeline at runtime,
- Implement their own step to integrate their custom processing into Renard by subclassing the existing `PipelineStep` class.

The flexibility of this approach introduces the possibility of creating invalid pipelines because steps often require information computed by previously ran steps: for example, solving the NER task requires a tokenized version of the input text. To counteract this issue, each step therefore declares its requirements and the new information it produces, which allows Renard to check whether a pipeline is valid, and to explain at runtime to the user why it may not be.

Table 1: Existing steps and their supported languages in Renard.

Step	Supported Languages
`StanfordCoreNLPPipeline`	eng
`CustomSubstitutionPreprocessor`	any
`NLTKTokenizer`	eng, fra, rus, ita, spa… (12 other)
`QuoteDetector`	any
`NLTKNamedEntityRecognizer`	eng, rus
`BertNamedEntityRecognizer`	eng, fra
`BertCoreferenceResolver`	eng
`SpacyCorefereeCoreferenceResolver`	eng
`NaiveCharacterUnifier`	any
`GraphRulesCharacterUnifier`	eng, fra
`BertSpeakerDetector`	eng
`CoOccurencesGraphExtractor`	any
`ConversationalGraphExtractor`	any

Renard lets the user select the targeted language of its custom pipeline. A pipeline can be configured to run in any language, as long as each of its steps supports it. Table 1 shows all the currently available steps in Renard and their supported languages.

References

Hagbert, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics and function using NetworkX. *7th Python in Science Conference (SciPy2008)*. https://www.osti.gov/biblio/960616

Labatut, V., & Bost, X. (2019). Extraction and analysis of fictional character networks : A survey. *ACM Computing Surveys*, *52*, 89. https://doi.org/10.1145/3344548

Marazzato, R., & Sparavigna, A. C. (2014). Extracting networks of characters and places from written works with CHAPLIN. *arXiv*, *cs.CY*, 1402.4259.

Métrailler, C. (2023). *Charnetto*. https://gitlab.com/maned_wolf/charnetto

Park, G., Kim, S., & Cho, H. (2013). Structural analysis on social network constructed from characters in literature texts. *Journal of Computers*, *8*. https://doi.org/10.4304/jcp.8.9.2442-2447

Park, G., Kim, S., Hwang, H., & Cho, H. (2013). Complex system analysis of social networks extracted from literary fictions. *International Journal of Machine Learning and Computing*, 107–111. https://doi.org/10.7763/IJMLC.2013.V3.282

Rochat, Yannick. (2014). *Character networks and centrality* [PhD thesis, Université de Lausanne]. https://serval.unil.ch/resource/serval:BIB_663137B68131.P001/REF.pdf

Rochat, Y. (2015). Character network analysis of émile zola's les rougon-macquart. *Digital Humanities 2015*. https://infoscience.epfl.ch/record/210573?ln=en

Rochat, Y., & Triclot, M. (2017). Les réseaux de personnages de science-fiction : Échantillons de lectures intermédiaires. *ReS Futurae*, *10*, 1183. https://doi.org/10.4000/resf.1183