Fine-grained dynamic graph for novel comprehension - Final Project Report -

GitHub repo : https://github.com/dldk-gael/novel_story_building

Gaël de Léséleuc de Kérouara
gael.de-leseleuc@student-cs.fr

Alexandre Duval
alexandre.duval@student-cs.fr

1 INTRODUCTION

1.1 Motivation

Novels often propose a complex narrative and an elaborated plot that are sometimes difficult to grasp and follow throughout the book. Visualising different aspects of the novel and its evolution across time through a dynamic graph networks seems to be an intuitive approach to improve one's comprehension of it. Many authors using character based networks to solve higher-level problems identify the dynamics of the story as key aspect that needs to be taken into account. This especially holds in long-term narratives such as a novel saga, in which relationships and characters are likely to change over time. Yet, the vast majority of methods proposed in the literature rely on static networks which represent a novel in its entirety. As a result, an important part of the narrative story, ie the dynamic information, is lost. This motivates us to shed further light on dynamic narrative by using fine-grained graph which keep all information about temporal data.

1.2 Problem definition

In this project, we investigate how to create an end-to-end tool that will produce, given a file text representing the novel, a fine-grained graph which keeps the dynamic aspect of the narrative. Then we study how the use of such a graph can lead to improve one's comprehension of the novel. We mainly focus on extracting the key properties of the dynamic structure of the narrative graph. In particular, we study the the evolution of the character through the novel: their role, their relations, their importance throughout the book. We also aimed to identify structural changes in the plot, display its core component and isolate some specific segments of the story according to one's needs. Finally, we compare two books together as well as their k-core graph with the book's summary. Therefore, given one novel, we explored a wide range of graph properties and draw conclusions from them, mainly using supervised learning.

1.3 Report organization

In section **2**, we discuss about related works that also focus on extracting a narrative graph from a novel, classification tasks and studying some of the graph's aspects. In subsection **??**, we detail the information extraction pipeline we have implemented to later construct the narrative graph (entities recognition, co-references, improved matching, etc.). In **??**, we discuss one of the core idea, which is to construct a fine-grained graph and then only derive from it particular sub-graphs that will by analysed using graph properties. Finally, **??** contains the core of the work we do on the book's character network (character importance, structural change, community detection, graph summarisation, compare two books) and present the results obtained on *Harry Potter*. Note that **??** contains the results for *Lord of the Rings*, which is used as a comparison for Harry Potter.

2 RELATED WORK

The majority of resources we have found closely related to our project mostly consider the static aspect. Among them, [4], which we saw in class, is interesting in the way it interprets the novel's (character) graph. Indeed, it uses centrality measures to detect main characters and apply clustering to detect the different communities. Similarly to [2], it constructs the network by applying Name Entity Recognition and drawing edges between two characters if they co-occur in window of like 15 words. In the later, they also detail a few tricks they used to solve co-reference. The aim of their paper is to build social networks from novel to quantify their plot and structure. They create static and dynamic character networks and extract features (graph density, clustering coef, central/isolate nodes proportion, weight of biggest and second biggest node, presence of character throughout novel...) to perform classification on genre and author. To do so, they collect 238 books from Gutenberg, written by a small set of authors and grouped into 11 genres. They find many interesting patterns and use purity, entropy and F1 measure to evaluate their findings. [5] study the difference of structures in novels' graphs, in rural and urban settings. It shows a different way to approach some parts described above. More precisely, it applies clustering on entities to solve the issue of co-reference. It also weights the edge not only by the number of time two entities appears together but also by the the length of dialogue between these two characters.

All these aspects, that is co-reference, interaction, classification tasks with handcrafted features, are equally detailed in the survey of methods [6]. However, the latter delves further into the visualisation techniques of dynamic networks. The one method we retain is to first plot the whole network as a shadow and then plot on this shadow network the graph obtained at each chapter. More information on nice and comprehensive dynamic visualisation is detailed in the following resource: [3]. They talk about representation of changes (marking event in the book's plot), temporality changes, structural tricks, visual emphasis, etc. Some of the challenges they faced concerned characters identification, changes depiction, order of events, spatial context and the number of elements for big novels. On a similar note, [7] use force-vector spatialization to place nodes in the space. The algorithm simulates a system of physical forces: nodes repulse each other, while edges act as springs attracting the nodes that they connect.

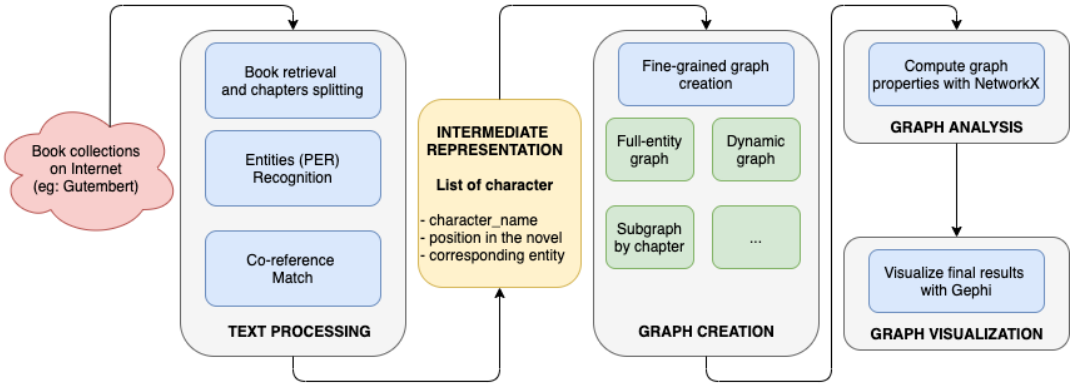

Figure 1: Representative schema of our end-to-end, from novel to graph visualization pipeline

3 METHODOLOGY

3.1 Overall overview

Before going into details, we present here the general overview of the end-to-end pipeline we created. The **figure 1** illustrate this pipeline.

- **Text processing**
  (1) Retrieve book from internet and split them by chapter
  (2) Extract character name
  (3) Match each character to entity
- **Graph creation**
  (1) Construct full fine-grained graph
  (2) Derive from it multiple subgraph of interest
- **Graph Analysis**
  (1) Plot comprehension, character importance and structural change analysis, via computing key properties of the dynamic graph using networkx.
  (2) Community detection
  (3) Compare books and compare book summary with k-core
  (4) Visualize entity and dynamic graph using gephi

3.2 Text processing

1. The goal.

The objective of the text processing module is, given a novel, to construct a list of occurrences containing multiple pieces of information. We define an occurrence as being the appearance of a character name somewhere a novel. The text processing module must spot all the occurrences in the novel and store for each of them the following information :

```
character_name: [str] name of character (eg: 'Harry')
pos: [int] nb of tokens since beginning (eg: 30490)
chapter: [int] index of the chapter (eg: 2)
entity : [str] corresponding entity (eg: 'HARRY POTTER')
```

2. Book retrieval and chapter splitting.

To conduct this project, we targeted online books, for instance using Project Gutenberg. [1]. Because we not only want to know the position of a particular occurrence in the novel (number of token since the beginning) but also the chapter in which the occurrence is located, we have to split the book by chapter. To proceed, we used the ***chapterize***[2] framework that works particularly well on books coming from Project Gutenberg. Roughly it uses a list of regular expression rules to find the chapter title and then split using this information.

3. Name entity recognition for character occurrence.

To detect the occurrence in the text novel, we used ***BERT-NER***[3] framework which is a pre-trained google-BERT specifically fine-tuned for entity recognition task. Concretely we split each chapter in batch so that each string input correspond to BERT-NER input size and browse all the novel by proceeding this way. Each time BERT-NER detect a 'PER' entity, we create a new occurrence and keep in memory the character name, the position of this occurrence in the text (we count using BERT tokens metrics) and the chapter index.

4. Co-reference for entity matching.

Then, once we have detected all occurrences of character in the novel, we must match each occurrence to a unique entity. For instance, considering *Harry Potter*, we would like that all the following occurrences: Harry, Mr. Potter, H. Potter to be linked to a single entity, in this case HARRY POTTER.

This part was particularly time consuming because there does not exist yet such tools like BERT-NER which can perform co-references on an entire book. As a result, we decided to follow the strategy developed by M.Ardanuy and C.Sporleder in **[1]**. Their can strategy can roughly divided several parts:

(1) Use ***NameParser***[4] framework to parse a character name into a generic structure. For instance, Mr. Harry Potter will be parsed into :

```
{title: Mr, first: Harry, last: Potter}
```

(2) Determine if possible the genre of the character name. Here we do not use all techniques describe in [1] but simply assess it from the title (for instance if title is 'Mr', the genre is Male) and from the first name, using databases.

[1] https://www.gutenberg.org/

[2] https://github.com/JonathanReeve/chapterize
[3] https://github.com/kamalkraj/BERT-NER
[4] https://github.com/derek73/python-nameparser

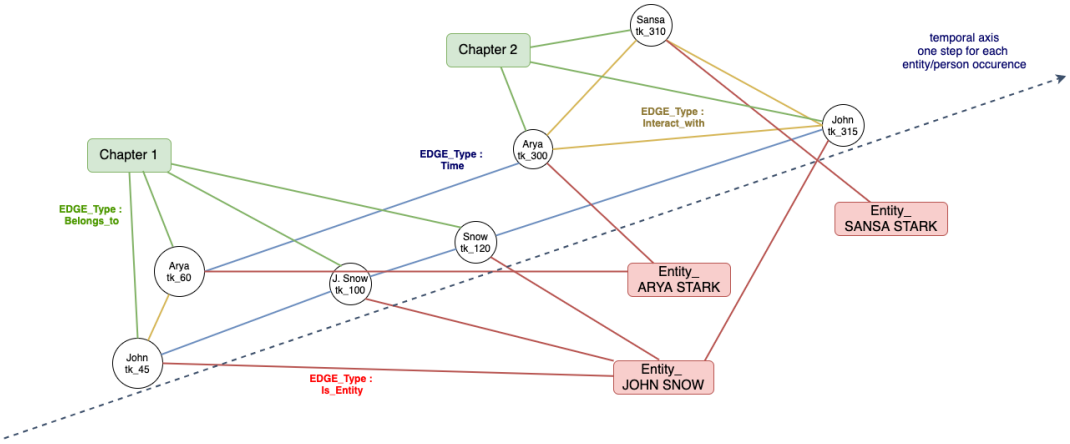

Figure 2: Schematic representation of the full graph

(3) Map an occurrence to an entity. Without delving too deep into details, we first consider all occurrences showing a title, a first name and a last name and create the corresponding entity if does not already exist (match first name, last name and genre). Then, we consider among remaining occurrences those presenting first name and last name and repeat the process. Next steps concern occurrences with title and first name, title and last name and finally first or last name. Hence, since HARRY POTTER will be created at the beginning, Harry or Mr. Potter will be associated to this same entity, when following the rules we created. The process is defined in such way that all occurrences are mapped to an entity. [1]

5. Improved matching.

We went even further in the entity matching process, by considering nicknames, initials and small mistakes of the NER model. Indeed, in the current framework, h. Potter and Duddy are distinct entities and are not mapped to Harry Potter and Dudley Dursley, as we would like them to. We solve this problem by specifying carefully specified rules and using an existing nicknames database [5], which in the end enable us to map correctly these occurences.

3.3 Graph construction

1. Full graph.

The crucial point of the strategy we use to tackle narrative graph analysis is to create a base structure graph that keeps all temporal information about the narrative dynamic. And then use this relatively big graph to derive sub-graphs of interest.

An example of such graph is represented on left-side of **figure 5**.

The full graph is composed of 3 nodes types :

- **occurrence-node** : tuple (character name, occurrence position)
- **chapter-node** : index of a chapter
- **entity-node** : string representing the entity

There are also 4 types of (un-directed) edges:

- **belong-to edge** : connect an occurrence node to a chapter node if the occurrence is localed in chapter node idx
- **is-entity edge** : connect an occurrence node to an entity node if the character name of the occurrence has been matched to the corresponding entity.
- **interact-with edge** : connect two occurrence nodes if they interact together and are not connected to the same entity node.
- **time edge** : connect two occurrence nodes that are connected to the same entity if there does not exist another occurrence node connected to this entity which is located in between these two, in the novel.

Interaction definition: we create an interaction edge between two occurrence nodes that correspond to different entities if their positions in the novel are not more than 20 token distant. This is exactly the idea of using a co-occurrence sliding window except that in our case, it is more straightforward to compute.

The main interest of this full graph is we do not lose any information on the process: in fact we simply transform the occurrence list into a graph structure that will be more easy to manipulate and process. As such, we do not compute any graph metrics neither try to visualise the full graph but we use it as a knowledge graph from which we can derive lot of a graph of interest.

2. Construction of graphs of interest.

In our pipeline, we implement several functions that take as input the full graph and output several types of entity interaction graph. An entity interaction graph is a collapsed version of the full graph which simply contains one type of node : **entity-node** and one type of edges **interact-with**. Those edges being weighted by the number of interaction between the two entities.

To illustrate how using the full graph make it very straightforward to obtain such graph, let take an example. Suppose that we want to compute the entity interaction graph for each chapter.

- **First step: retrieve the subgraph of the full graph corresponding to each chapter**

[5] https://github.com/MrCsabaToth/SOEMPI/blob/master/openempi/conf/nick$_{t}o_{n}ame.csv$

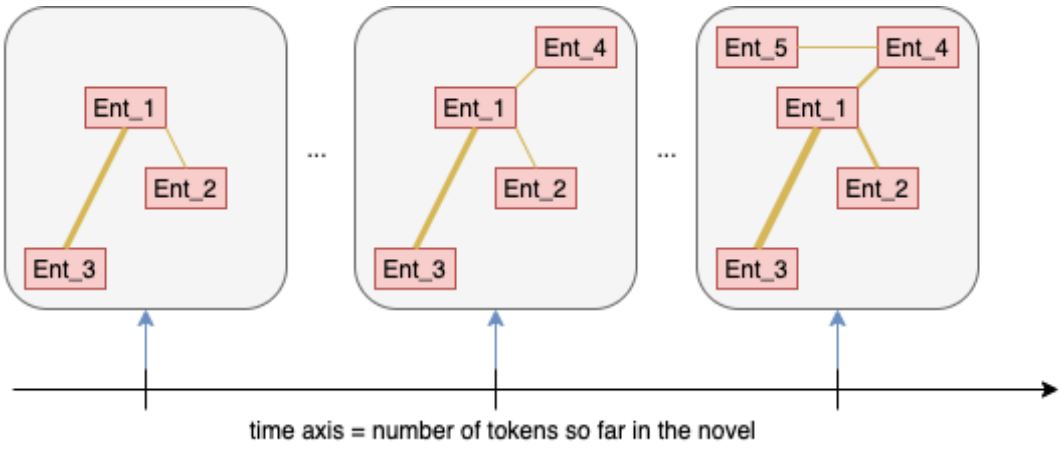

Figure 3: Dynamic representation of entity interaction

This is easy to do using neighbours function from networkX, we just need to begin by the chapter node and look at the neighbours. In pseudo code, it would look like :

```
retrieve_subgraph_for_chapter(idx):
  chap_node = full_graph.nodes[idx]
  occ_nodes = full_graph.neighbours(chap_nodes)
  ent_nodes = full_graph.neighbours(occ_nodes,
                                    type=ent)
 return full_graph.subgraph(chapter_node,
                            occ_nodes,
                            ent_nodes)
```

- **Second step: create the entity graph**.
  This is quite simple. The nodes of the entity graph are the node of the sub-graph that have the type entity. Now we just have to compute the interaction edges between two entities. Or by definition two entities ent1 and ent2 are interacting if in the subgraph they are each connected to occurence node occ1 and occ2 such that occ1 and occ2 and linked by an interaction edges. We can reformulate the definition as two entities are interacting if there is the following path in the subgraph :

  ```
  NODE(entity 1) / EDGE(IS_ENTITY) / NODE(occ 1)
  NODE(occ 1) / EDGE(INTERACT_WITH) / NODE(occ2)
  NODE(occ2) / EDGE(IS_ENTITY) / NODE(entity 2)
  ```

  As a result, two entities are interacting if there is path of length 3 between them in the subgraph and the weight of the edge is the number of such path.
  In pseudo code :

  ```
  entity_graph = graph()
  entity_graph.add_nodes(subgraph.nodes(type=ENT))

  for each ent1, ent2 in entities_nodes:
     x = full_graph.compute_path(ent1, ent2, lenght=3)
      if len(x) > 0
          entity_graph.add_edge(ent1, ent2,
                                weight = len(x))
   return entity_graph
  ```

As a final result, we would obtain a sequence of entity graph per chapter such as presented in **figure 4**

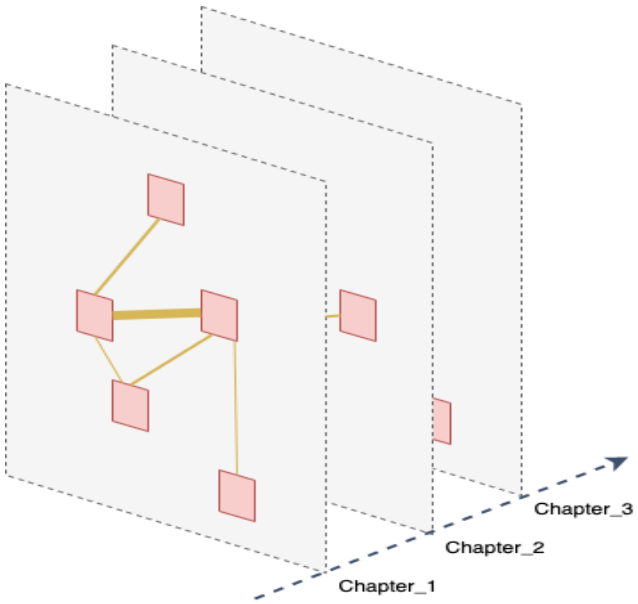

Figure 4: A sequence of entity graph per chapter, obtained by integrated the global graph on the temporal dimension and reducing the occurrences nodes by using entity nodes

3. Fine-grained dynamic graph.

Because the position of occurrence character name is encoded in our full graph representation, we are able to produce much more fine-grained dynamic graph that the simple (but still very usefull) sequence of graph by chapter. We had the idea to produce a fine-grained dynamic entity interaction graph where the timeline is simply the number of token so far in the text, ie: the 'time' goes from 0 to number of tokens in the novel. In such a graph :

- **entity-node** only appears when a character name that match the entity occurred in the text
- **interaction-edge** only appears between two entity nodes when an interaction between two character names related to those entities occurred in the text. Moreover, the edges are dynamically weighted by the number of interaction so far in the text.

This allows us to render a dynamic version of entity interaction (see **figure 3 for a schematic representation**.

3.4 Graph exploitation and visualization

To construct and compute the majority of key graph properties, we used **networkX** library. To construct the visualization (such as the dynamic version of the graph as described above), we used **gephi** software.

In addition to the graph analysis work that will be described in the next section, we were particularly focus on some visualisation techniques evoked in **Telling stories about Dynamic Networks with Graphs comics (2016)**, trying for instance to get nice and clear representations preserving the order of events or showing structural changes. We consider the visualisation aspect to be a key step towards understanding the novel. Unfortunately, these visualisations being dynamic, it is impossible to insert them in the paper. A few videos are however available on the github link and showcase the evolution of the graph, emphasising sometimes the core characters or main relationships between nodes. This of course involves a representation in *gephi* as a single dynamic graph endowed with a time axis.

4 GRAPH ANALYSIS

We have studied several aspects of the graph, always in the perspective of getting a better comprehension of the novel. This allows us to apply different techniques that we have evoked in the lectures, and that we will therefore not detail from a theoretical point of view. We will simply detail the results that we obtained when studying two books in particular: *Harry Potter and the Philosopher Stone* and *Lord of the Rings: The fellowship of the Ring*.

4.1 Character Importance

The first step of the analysis, and probably the most informative one, is to spot the most important characters of the book. To do so, we regard all entity graph by chapter, constructed in the previous section and built in networkx. We then compute for each character a centrality measure (degree, betweeness, pagerank, etc.) representative of its importance in the narrative, taking the book's entity graph. Since our graph is weighted, we favoured the pagerank metric and obtain the following results *Figure 5* for Harry Potter book. It shows the centrality metric evolution across chapters of the 10 most important characters, using entity chapter graphs. This yields excellent results. Obviously, Harry is the most central character. We also notice a trend where some influencial characters (Dudley Dursley) vanish out from chapter 4 and are replaced by others (Ron, Hermione...), representing a book's big change of setting as Harry is arriving to Hogwards.

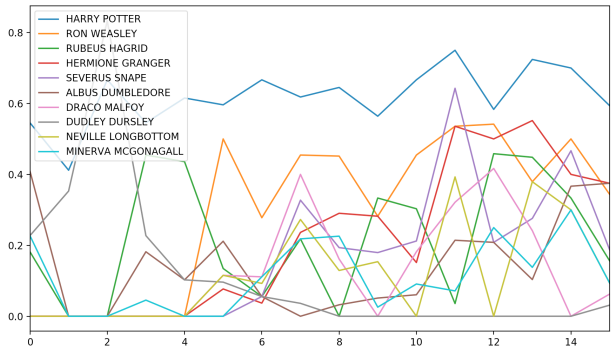

Figure 6: Character importance accross novel

4.2 Structural change

Building on the sort of change we remarked in the book at Chapter 4, we further investigate structural change in the book's narrative. To do so, we have selected a few properties, both static and dynamic, that could help us gain a better understanding of the plot evolution. We explain some of them:

- *Number of nodes and edges, number of existing characters, characters that disappear, characters that are introduced.* These variables give precious information about the number of characters in the book and their interactions. Observing *Figure 6*, we notice that there is a surge in character introductions and interactions when Harry arrives at Howards, confirming what was inferred in previous subsection. Comparing this graph with another book's also allows us to compare how two books are structured and could be use to differentiate between two authors.

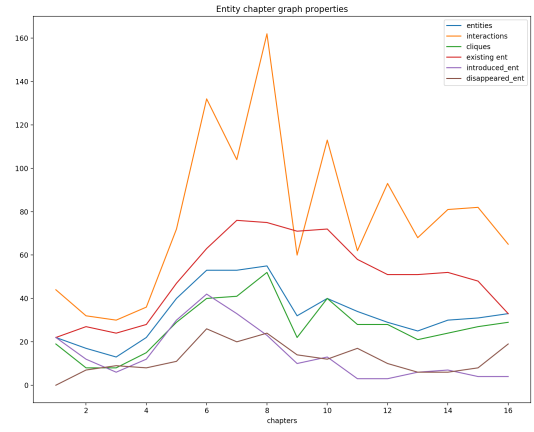

Figure 7: Graph properties - nodes and interactions

- *Cliques related properties* are features that provide information on the density of the graph, so on the relation between characters and their evolution across the book. Looking for

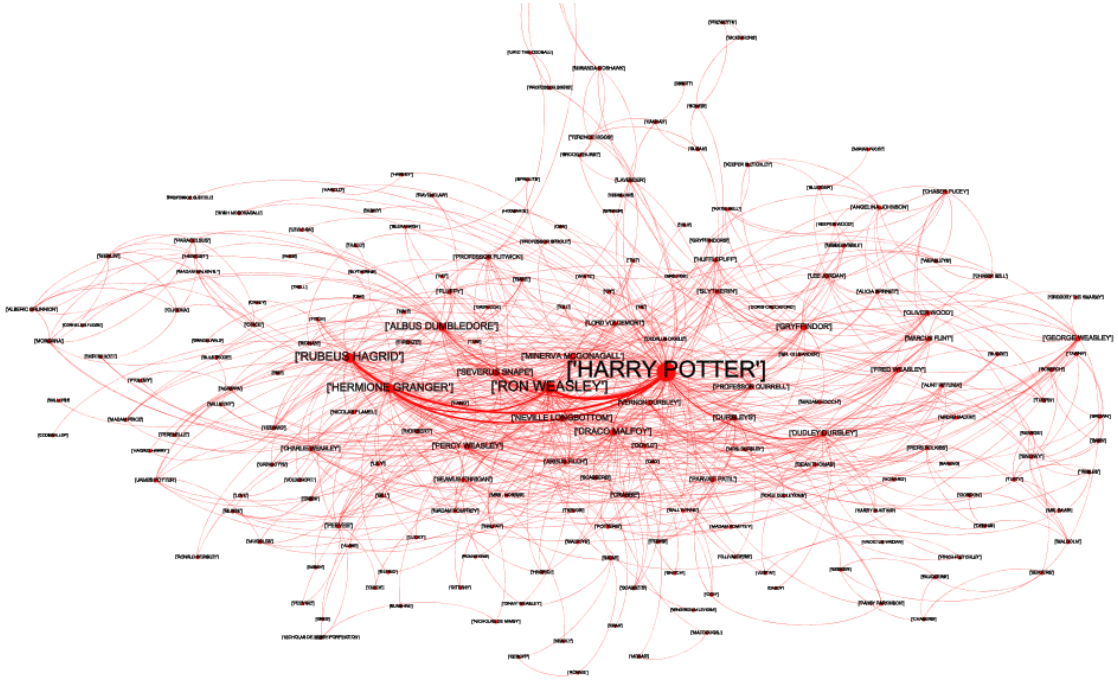

Figure 5: Visualisation of the full entity graph with Gephi

instance at the largest clique in each chapter enables to see which characters interact and what the global plot is about. Is it about the Dudleys, teacher-student interactions or Harry-Voldemort-Dumbledore. Investigating the cosine similarity of the cliques could also turn out to be interesting to see if the action often turn around the same characters (HP) or if it ofen changes (GOT, LOTR). Similar deductions can be made inside chapters by comparing cliques and the k-core (iou metric) of each chapter. **Figure 11** (at the end of the report) shows the clique for each chapter.

- Looking at the *proportion of important characters in each chapter* may help us find what chapters are important in the storyline.
- The *difference between the main and second main character importance* gives information about the type of book written, is it a one-main character or several-main characters book.

4.3 Community detection

In this subsection, we want to find the different communities that exist in the book. We apply the Girvan Newman algorithm on the full entity graph. In fact, we apply it on a subgraph obtained by deleting the nodes with too few interactions (fewer than five usually). In addition to making the graph readable, it excludes a few fake entities spotted by BERT, and only keep the real ones.

In both Harry Potter and Lord of the Rings, we spot interesting communties. In Harry Potter for instance, one community involves the quidditch team, another the Dudleys' entourage, another the friends of Dumbledore and the last and main one: the friends of Harry Potter and teachers at Hogwards. The **figure 8** shows community detection for both those books.

4.4 Comparisons

4.4.1 Summary of the book. The first one involves comparing the *k_core* graph of the full Harry Potter book with the graph obtained from the summary of the book, scrapped from the Internet. The idea behind it is that we would like the k-core decomposition of the full graph to be similar to the graph of the summary of the book. We followed a similar process, with BERT NER for entity extraction, co-referencing, improved matching and graph creation. Except that this time, only the static full entity graph matters. We have computed metrics based on nodes intersection and edges intersection, since they are labelled and we have a correspondence. We have also used some notions of graph similarity seen in class. And the results obtained are extremely satisfying. The two graphs nearly have the same number of nodes, the same connectedness and similar character relations. The entities involve differ only slightly, with an iou score of 0.5. The main differences lie with the inclusion of the Dursleys and the Weasleys. As a result, our graph obtained from the k-core of the full book could be used successfully as a summary of the book.

4.4.2 Comparison with Lord of the Rings. We compare the writing style of Harry Potter with the one of Lord of the Rings. More precisely, we proceed to the exact same analysis for both books and search for similarity and differences between both. In short, although LOTR is much longer, there is a similar number of existing entities and interactions in the two books. However, unlike Harry Potter, LOTR is not centered around one character and several plots

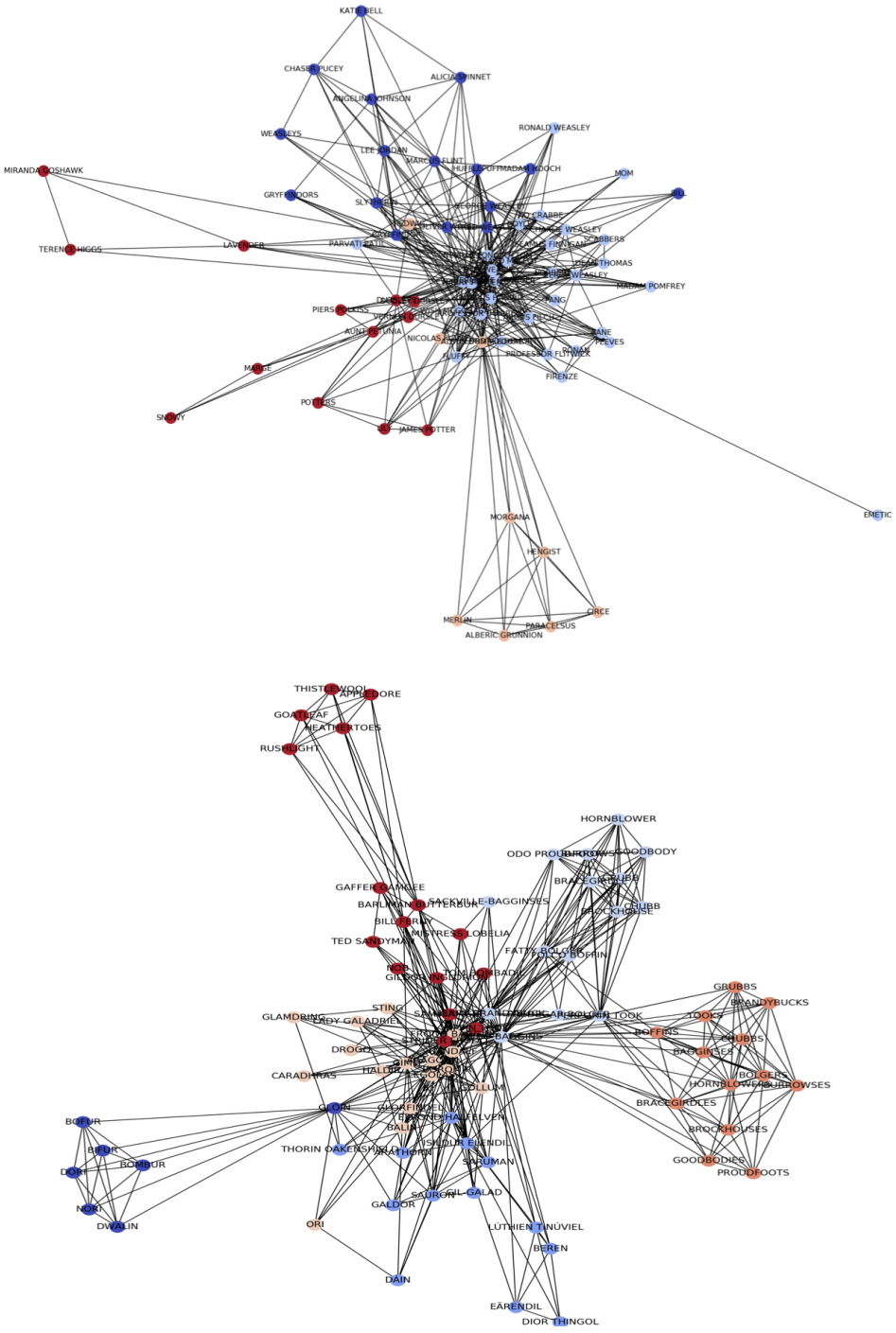

Figure 8: On top: community detection for Harry Potter first book
On bottom: community detection for Lord of the Ring first book

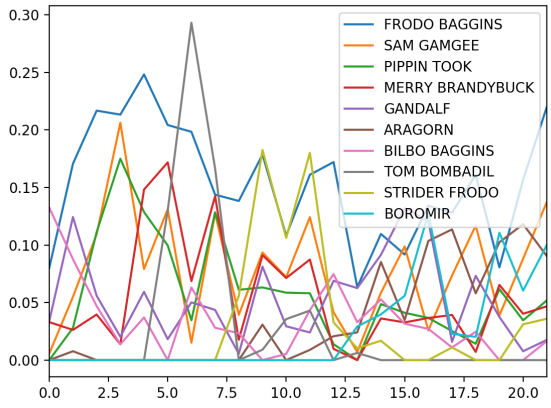

Figure 9: Variation of character importance among LOTR's chapter computed using degree centrality

take place at the same time, involving different characters from a chapter to another, as shown in the **figure 9**. We also spot a structural change in the plot in chapter 14 where the number of entities and interactions surges suddenly, this is particular visible in the **figure 10**.

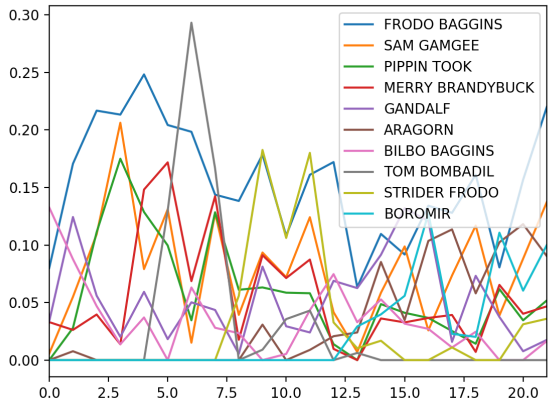

Figure 10: Structural change of entity interaction graph among LOTR chapter

It correponds to the beginning of the quest and the rise of some characters like Boromir and Aragorn. Regarding community detection, there are more distinct communities in LOTR than in Harry Potter, where the storyline is really centered around Hogwards, where everyone interacts. There often is in LOTR a key character that serves as a link between different communities, which yield several characters with high betweeness centrality, unlike Harry Potter (the betweeness centrality subgraph is plot on **figure 12** at the end of the report. Continuying on this note, the graph of character importance does not showcase a single entity dominating the others, like in Harry Potter. All graphs are also very relevant according to our knowledge of the book.

5 CONCLUSION

In this project, we aim at providing a dynamic framework to analyse a novel and more precisely its character network. We have therefore first processed the text and extracted all character occurrences using BERT NER, placing a special emphasis on co-referencing to obtain graphs as accurate as possible. We then spent some time creating a dynamic graph framework that would allow us to include a time dimension for our network. From this, we have studied some targeted subgraphs like entity graph by chapter, using *networkx* and *gephi*. We have focused on five different tasks: character importance, structural change, community detection, graph summarisation and books comparison. Studying them gave us precious information about the book's plot, the author's style and characters (importance, relations, role, etc.). Note that graph visualisation in *gephi* was a key aspect of the project, to which we dedicated quite some time.

REFERENCES

[1] Mariona Coll Ardanuy and Caroline Sporleder. [n.d.]. Structure-based Clustering of Novels.

[2] Mariona Coll Ardanuy and Caroline Sporleder. 2014. Structure-based clustering of novels. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*. 31–39.

[3] Benjamin Bach, Natalie Kerracher, Kyle Wm. Hall, Sheelagh Carpendale, Jessie Kennedy, and Nathalie Henry Riche. 2016. Telling Stories about Dynamic Networks with Graph Comics. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3670–3682. https://doi.org/10.1145/2858036.2858387

[4] Andrew Beveridge and Jie Shan. 2016. Network of Thrones. *Math Horizons* 23, 4 (2016), 18–22. https://doi.org/10.4169/mathhorizons.23.4.18

[5] David Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting Social Networks from Literary Fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, 138–147. https://www.aclweb.org/anthology/P10-1015

[6] Vincent Labatut and Xavier Bost. 2019. Extraction and Analysis of Fictional Character Networks: A Survey. *Comput. Surveys* 52 (09 2019), 89. https://doi.org/10.1145/3344548

[7] Tommaso Venturini, Liliana Bounegru, Mathieu Jacomy, and Jonathan Gray. 2017. *How to Tell Stories with Networks: Exploring the Narrative Affordances of Graphs with the Iliad.*

```
-- COMPUTE PROPETIES OF ENTITY GRAPH FOR EACH CHAPTER --

For chapter n° 0
k-core:  ['LORD VOLDEMORT', 'JAMES POTTER', 'MINERVA MCGONAGALL', 'LILY', 'RUBEUS HAGRID', 'HARRY POTTER', 'DURSLEYS', 'ALBUS DUMBLEDORE', 'POTTERS']
strongest edges ('MINERVA MCGONAGALL', 'ALBUS DUMBLEDORE')
biggest clique ['MINERVA MCGONAGALL', 'ALBUS DUMBLEDORE', 'LILY', 'LORD VOLDEMORT']
core part : {'ALBUS DUMBLEDORE', 'MINERVA MCGONAGALL'}

For chapter n° 1
k-core:  ['VERNON DURSLEY', 'TIBBLES', 'MARGE', 'MR. PAWS', 'SNOWY', 'TUFTY']
strongest edges ('HARRY POTTER', 'DUDLEY DURSLEY')
biggest clique ['VERNON DURSLEY', 'MARGE', 'MR. PAWS', 'SNOWY', 'TUFTY', 'TIBBLES']
core part : {'VERNON DURSLEY'}

For chapter n° 2
k-core:  ['DENNIS', 'DUDLEY DURSLEY', 'MALCOLM', 'GORDON', 'PIERS POLKISS']
strongest edges ('HARRY POTTER', 'DUDLEY DURSLEY')
biggest clique ['DUDLEY DURSLEY', 'DENNIS', 'GORDON', 'PIERS POLKISS', 'MALCOLM']
core part : {'DUDLEY DURSLEY'}

For chapter n° 3
k-core:  ['RUBEUS HAGRID', 'LILY', 'HARRY POTTER', 'DURSLEYS', 'JAMES POTTER']
strongest edges ('HARRY POTTER', 'RUBEUS HAGRID')
biggest clique ['HARRY POTTER', 'RUBEUS HAGRID', 'LILY', 'DURSLEYS', 'JAMES POTTER']
core part : {'RUBEUS HAGRID', 'HARRY POTTER'}

For chapter n° 4
k-core:  ['TAWNY', 'SCREECH', 'BROWN', 'BARN', 'HARRY POTTER', 'SNOWY']
strongest edges ('RUBEUS HAGRID', 'HARRY POTTER')
biggest clique ['HARRY POTTER', 'TAWNY', 'BROWN', 'BARN', 'SCREECH', 'SNOWY']
core part : {'HARRY POTTER'}

For chapter n° 5
k-core:  ['HENGIST', 'CIRCE', 'PARACELSUS', 'MERLIN', 'ALBERIC GRUNNION', 'MORGANA', 'ALBUS DUMBLEDORE']
strongest edges ('RON WEASLEY', 'HARRY POTTER')
biggest clique ['ALBERIC GRUNNION', 'CIRCE', 'MERLIN', 'PARACELSUS', 'HENGIST', 'MORGANA', 'ALBUS DUMBLEDORE']
core part : {'ALBUS DUMBLEDORE'}

For chapter n° 6
k-core:  ['ZABINI', 'SALLY-ANNE', 'MINERVA MCGONAGALL', 'RON WEASLEY', 'BLAISE', 'DURSLEYS', 'HARRY POTTER', 'PERCY WEASLEY', 'PARVATI PATIL', 'PERKS']
strongest edges ('RON WEASLEY', 'HARRY POTTER')
biggest clique ['HARRY POTTER', 'PARVATI PATIL', 'PERKS', 'DURSLEYS', 'SALLY-ANNE']
core part : {'HARRY POTTER'}

For chapter n° 7
k-core:  ['SEVERUS SNAPE', 'MINERVA MCGONAGALL', 'RUBEUS HAGRID', 'RON WEASLEY', 'DRACO MALFOY', 'FRED WEASLEY', 'GOYLE', 'HARRY POTTER', 'CRABBE', 'GEORGE WEASLEY']
strongest edges ('RON WEASLEY', 'HARRY POTTER')
biggest clique ['HARRY POTTER', 'RON WEASLEY', 'SEVERUS SNAPE', 'DRACO MALFOY', 'GEORGE WEASLEY', 'RUBEUS HAGRID', 'FRED WEASLEY']
core part : {'SEVERUS SNAPE', 'RUBEUS HAGRID', 'RON WEASLEY', 'DRACO MALFOY', 'HARRY POTTER'}

For chapter n° 8
k-core:  ['SEVERUS SNAPE', 'LAVENDER', 'MINERVA MCGONAGALL', 'RON WEASLEY', 'HERMIONE GRANGER', 'HARRY POTTER', 'PARVATI PATIL', 'PROFESSOR FLITWICK']
strongest edges ('RON WEASLEY', 'HARRY POTTER')
biggest clique ['HARRY POTTER', 'RON WEASLEY', 'HERMIONE GRANGER', 'PARVATI PATIL', 'LAVENDER']
core part : {'HERMIONE GRANGER', 'RON WEASLEY', 'HARRY POTTER'}

For chapter n° 9
k-core:  ['DEAN THOMAS', 'NEVILLE LONGBOTTOM', 'RUBEUS HAGRID', 'RON WEASLEY', 'HERMIONE GRANGER', 'HARRY POTTER', 'SEAMUS FINNIGAN']
strongest edges ('RON WEASLEY', 'HERMIONE GRANGER')
```

Figure 11: Cliques and k-core output

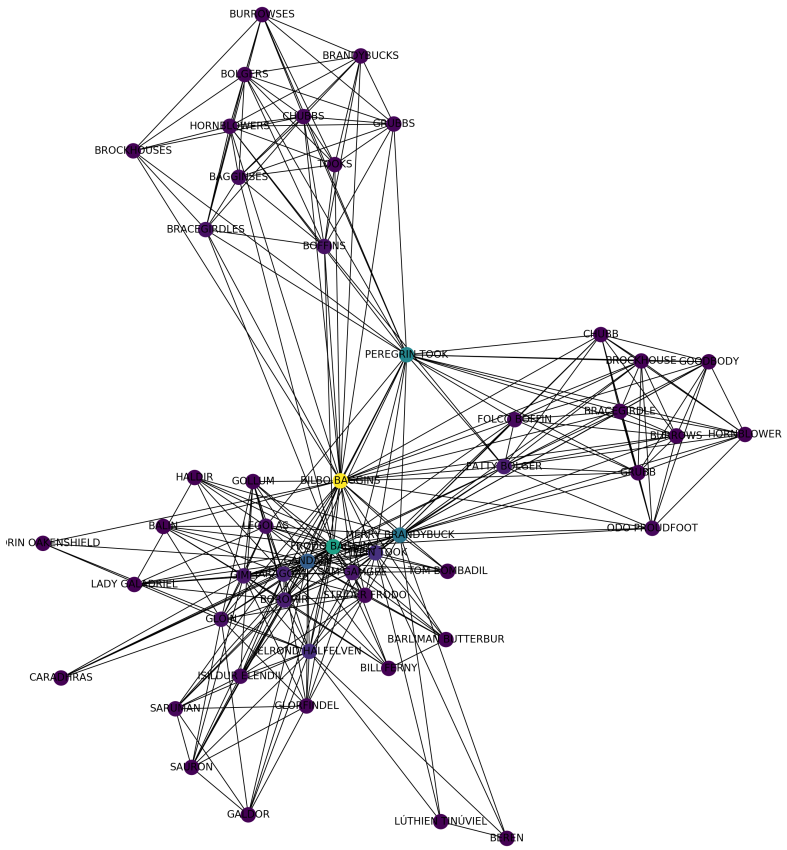

Figure 12: Betweeness centrality subgraph

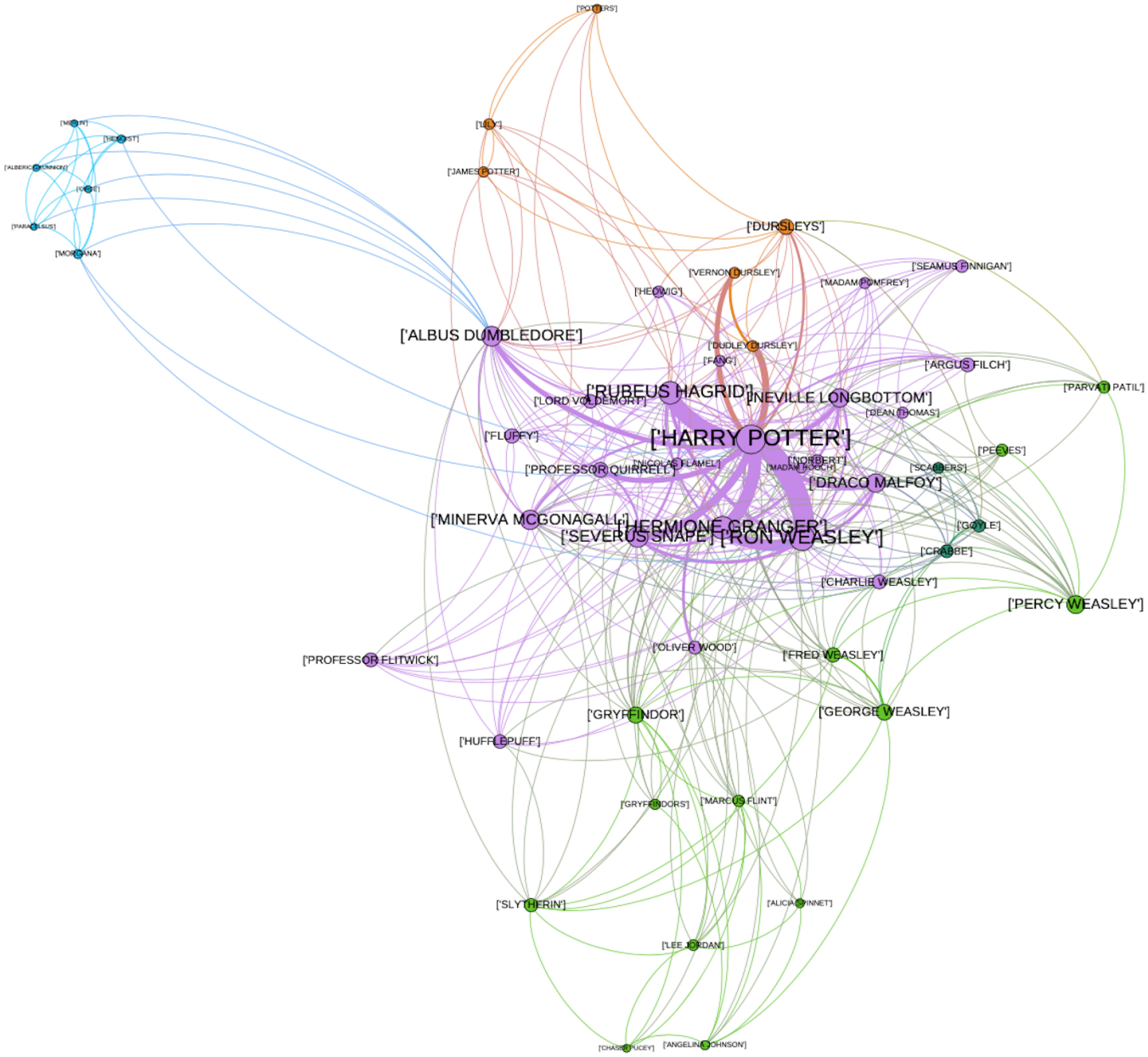

Figure 13: Community detection with gephi on subgraph of full entity graph