Solorme



Zeste de savoir

# Un graphe pour comprendre Tolkien ?

13 février 2023

# Table des matières



## Introduction

*Bilbo*, *Gandalf*, *Thorin* ... ces noms vous rappellent peut-être un film, un livre voire même une série autour de l'univers de **J. R. R. Tolkien**. Mais avez-vous en tête tous les personnages? Les relations que les uns entretenaient avec les autres?

**Bien sûr que non**! Mais allez-vous relire toutes les œuvres de cet univers pour répondre à ma question? **Encore moins**! La solution? Ce graphique:

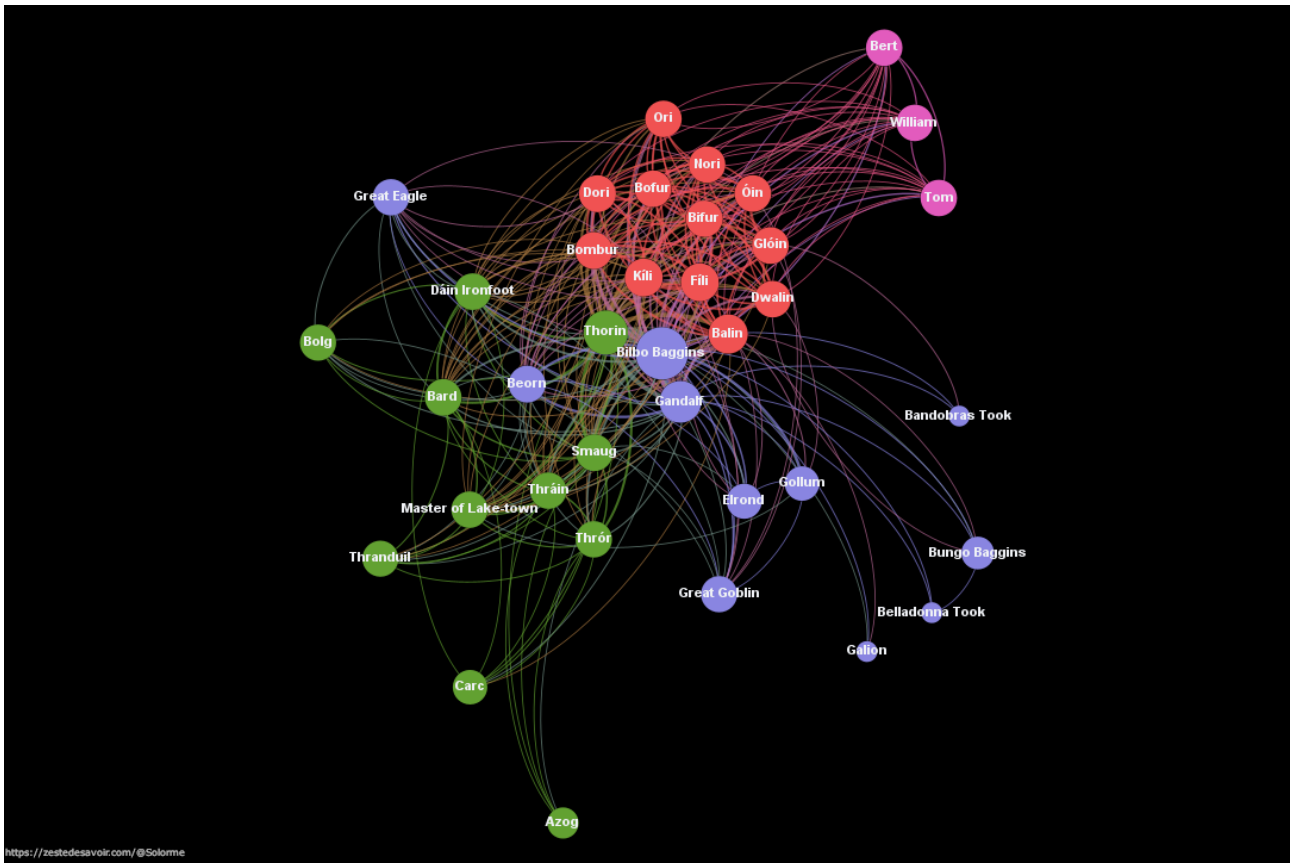

FIGURE 0.1. – Réseau d'interaction entre personnages tirés du livre Le Hobbit

Quel est sa signification? Ça vous intrigue? Je vous propose donc de me suivre afin de comprendre comment mettre en place un tel réseau, en passant par de la récolte de données, son analyse et sa mise en forme, **avec ou sans compétences préalables en informatique ou mathématique**

## 1. Un graphe pour les trouver tous

Le graphe ci-dessus représente les interactions entre les personnages du Hobbit. Mais comment ça se définit une interaction?

Vous pourrez trouver des exemples, comme cet article , où les interactions d'une personne vers l'autre ont été définis comme la **présence d'un lien hypertexte** sur la page wiki de l'un des personnages vers celle de l'autre. Cela retranscrit donc les liens que les personnes ont entre elles, **même en dehors du livre étudié**. En effet, Glóin et Bilbo sont tout deux dans *La Communauté de l'Anneau* et *Le Hobbit*, mais leur relation n'y est pas de la même nature. Cela n'est pas retranscrit avec cette méthode, qui prend en compte **toute la vie** des sujets étudiés.

Comment faire alors? On abandonne?

On va directement se baser sur le texte des livres ! On peut définir les liens entre personnages par leur présence simultanée dans un même tronçon de texte[1], par leur cooccurrence dans une même unité narrative. Ce concept variant selon les auteurs et les styles, ce papier explique que prendre un tronçon de texte d'un certain nombre de mots est une approximation facilitatrice mais que le seuil de mots reste arbitraire. C'est donc **une première limite** à ma méthode.

Intuitivement, il ***faut*** que ce nombre soit **suffisamment petit** pour ne pas mettre en relation deux personnages qui ne sont pas en contact, et **suffisamment grand** pour que deux personnages proches soient liés.

Reste ensuite à détecter les personnages dans intervalle donné. Reste? C'est en vérité la partie la moins évidente pour moi, débutant dans le domaine.

Pour détecter les personnages, il faut connaître l'ensemble des noms présent dans le livre. Certains me diraient d'utiliser la *bibliothèque python* **NLTK**, voire **Spacy**, pour détecter automatiquement les personnages qui se trouvent dans l'intervalle de caractères. Seulement, après quelques essais, il semblerait que tous les noms ne soient pas reconnus comme tels. D'autant plus que certains personnages ont plusieurs surnoms: Bilbo est parfois nommé Le Cambrioleur par exemple.: Il y a donc un risque d'avoir un écart important entre les données trouvées et réelles .

Petite aparté sur un terme qui sera récurrent dans l'article: une *bibliothèque* est un ensemble de fonctions, objets et constantes que l'on peut importer dans son code. Il est généralement plus efficace de se renseigner de leur existence plutôt que de réinventer continuellement la roue.

Une solution presque évidente serait de noter le nom de tous les personnages, ainsi que leurs surnoms… mais pour quelle consommation de café avec un peu plus de 800 personnes à vérifier?

## 2. Un script pour les rassembler

**Hors de question** de lire page par page les livres pour trouver tous les protagonistes des livres . Non, il nous faut des données déjà préparées. Et encore mieux, il nous faudrait un moyen d'automatiser la recherche des personnages qui nous intéressent. Plusieurs choix s'impose à nous : utiliser une **API** en ligne, une **base de données** accessible ou un site **encyclopédie** comme Wikipédia.

Wikipédia, d'accord je comprends … mais API? Base de données? Qu'est ce que ça veut dire?

Une **base de données** c'est un peu comme un tableur Exel ou Libreoffice Calc. On peut y avoir accès directement sur notre ordinateur et y stocker des données sous cette forme :

| Nom | Prénom | Âge |
|---|---|---|
| Bilbo | Bessac | 131 |
| Thorin | Oakenshield | 195 |

Une **API**, c'est une "interface logicielle qui permet de «connecter» un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités" [1] . En plus simple, et en lien avec notre projet, cela consisterait à envoyer des requêtes à un site qui nous enverrait les informations correspondantes sur les personnages, informations stockées dans sa base de données distante.

L'avantage de la base de données et de l'API, c'est que les informations sont déjà rassemblées et organisées, plus ou moins proprement. Le problème c'est que dans notre cas précis, celles disponible ne sont ni très complètes, ni nécessairement bien organisées. Par exemple, The One API est plutôt complète mais il y a un manque de régularité dans les données. William, Bert et Tom qui sont des Trolls dans le Hobbit sont rassemblés dans une même catégorie. C'est d'ailleurs étrangement le cas dans d'autres bases de données trouvées sur internet.

1. Vincent Labatut and Xavier Bost. 2019. Extraction and Analysis of Fictional Character Networks : A Survey . ACM Computing Surveys 52(5):89.

? Ils se sont donnés le mot ou quoi?

Il y a évidement une raison à tout cela . Et c'est lié à la troisième solution évoquée précédemment: trouver un site **encyclopédique** sur internet. L'avantage des encyclopédies participatives, comme Tolkien Gateway ou le Fandom Wiki Le Seigneur des Anneaux , c'est qu'elles sont généralement exhaustives en données et les informations les plus importantes (nom, âge, date de naissance etc) sont souvent dans une colonne à part entière. Par conséquent, on peut *facilement* extraire les données du site avec un petit script python! En l'occurrence, l'API et consorts ont probablement obtenus leurs données du même site, le **Fandom Wiki**, qui est reconnu [2] comme moins fiable que **Tolkien Gateway** et présente ces erreurs entre autres.

Pour faire ma petite analyse, je suis donc passé par le deuxième site.


FIGURE 2.2. – Toutes les infos ou presque sur Thorin II se trouvent ici!

Afin de faire du *scrapping* ou bien *data mining*, comprendre extraire les données du site, je suis passé par un script python utilisant **Selenium**. C'est une bibliothèque qui permet de contrôler un navigateur web et par conséquent accéder aux données du site, via les balises CSS ou XHTML qui forment en quelque sorte la structure des pages . Un exemple avec le code ci-dessous, pour une page personnage donnée.

```
# Les bibliothèques et codes nécessaires préalables ne sont pas spécifiées ici
# Le code ici ne sert qu'à montrer grossièrement à quoi ressemble la démarche

# On peut obtenir les données via les attributs CSS de la page ...
thorin_II = {}
race = navigateur_web.find_element(By.CSS_SELECTOR, "tr:nth-child(2) > td > a").text
thorin_II['race'] = race

# Ou bien par les attributs XHTML ...
caracteristiques_personnage = ["other names","gender", "birth", "death", "affiliation", "rule"]
for caracteristique in caracteristiques_site:
  try:
    thorin_II[caracteristique] = text = driver.find_element(By.XPATH,
    "//td[contains(.,'" + feature.capitalize() + "')]/following-sibling::td").text
  except NoSuchElementException:
    print('Données non disponible !')
```

Si les requêtes paraissent un peu mystiques, il faut bien comprendre que c'est un peu le jeu du chat et de la souris pour accéder, d'une manière ou d'une autre [3], aux informations que l'on souhaite!

? D'accord, là tu as les informations pour UN personnage quand tu es sur cette page. Mais comment *scrapper* les données de tous les personnages du Hobbit par exemple?

La question se pose effectivement! Tolkien Gateway référence les personnages selon leurs appartenances à différentes catégories : la race, la présence dans X ou Y livre, film, série. Ainsi, il suffit de naviguer jusqu'à la page qui recense les personnages présents dans le Hobbit, de récupérer leurs liens hypertextes, puis de les traiter un par un comme précédemment!

A la fin, j'obtiens un tableau qui ressemble à ça!

| name | link | race | other names | gender | birth | death | affiliation | rule |
|---|---|---|---|---|---|---|---|---|
| Azog | https://tolkiengateway.net/wiki/Azog | Orc | None | Male | None | T.A. 2799\nBattle of Azanulbizar | None | None |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Balin | https://tolkienga-te-way.net/wiki/Balin | Dwarf | None | Male | T.A. 2763 | 10 November, T.A. 2994 (aged 231)\nDimrill Dale | Thorin and Company\nBalin's | T.A. 2989 - T.A. 2994 Colony |
| Belladonna Took | https://tolkienga-te-way.net/wiki/Belladonna_Took | Hobbit | Mrs Bungo Baggins[1] | Female | T.A. 2852 | T.A. 2934 (aged 82) | None | None |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Toutes les informations ne sont effectivement pas renseignées. C'est soit qu'elles n'existent toute simplement pas (**Azog a le droit de ne pas avoir d'autres noms!**), qu'elles ne sont pas connues dans l'univers de Tolkien, ou bien qu'elles n'ont pas été entrées dans le site, ce qui m'a l'air d'être anecdotique.

> Vous remarquerez par exemple que Belladonna Took a un [1] dans la colonne "other names" et qu'il y a des \n dans la colonne death de nombreux personnages. Cela montre l'**importance de traiter les données** après le *scrapping*, en l'état ce sont des artéfacts qui pourront nous embêter dans l'analyse! Ce qui sera traité dans le prochain point.

Parfait! Nous avons maintenant une base de données avec l'ensemble des personnages du Hobbit [4] .

Mais, maintenant qu'est ce qu'on en fait? Vous pourriez établir une analyse détaillée de la mortalité des personnages en fonction de leur sexe, la distribution des races et ainsi de suite! [5]

**Ou alors se concentrer sur l'objectif initial** et mettre en forme les liens qu'ont les habitants de l'univers recensés dans le *data set* .

---

1. D'après la CNIL .
2. D'après l'avis des utilisateurs glanés ici et là, pas de métrique réelle mais une tendance assez claire.
3. L'extension Selenium pour Firefox a été d'une grande aide après de longues **tentatives** d'atteindre les résultats escomptés.
4. Ce qui pourrait s'appliquer aussi aux autres livres!
5. Vous remarquerez sur ce site , qui a fait l'analyse, que les personnages cités dans les livres ne représentent pas du tout un échantillon représentatif de la population.

## 3. Un texte pour les analyser...

Retour au point de départ. Nous avons la liste des personnages avec leurs noms et surnoms respectifs, et il se trouve que je dispose d'une version informatisée des livres. L'idée maintenant est d'étudier le livre en tant qu'un **ensemble de caractères**. Cela me facilite la détection ou non d'un personnage, comparé à une fragmentation du livre en mots. Pourquoi? 🍊

Pour faire simple, posons la liste `noms = ["Enfourcheur de Tonneaux", "Cambrioleur", "Numéro Chanceux"]`.

Posons la chaîne de caractères `phrase = "L'Enfourcheur de Tonneaux a faim, ainsi que Gandalf"` et la liste associée `phrase_liste = ["L'","Enfourcheur", "de", "Tonneaux", "a", "faim", ",", "ainsi", "que", "Gandalf"]`.

Le test `noms[0] in phrase` est *vrai* alors que `noms[0] in phrase_liste` est *faux*.

*i*

Tout de suite on se rend compte qu'en séparant les mots dans la phrase on complique la tâche pour les noms et surnoms composés de plusieurs particules. 🧙

Ensuite, je divise le livre en ses différents chapitres que je subdivise en plusieurs tronçons de taille fixée (*cf la partie précédente*). La taille est donc invariante **sauf** si la taille du dernier tronçon est inférieure à la taille habituelle, auquel cas celui-ci est concaténé au précédent tronçon. Je rajoute également **une limite à ma méthode**, car en l'état je ne vérifie pas si, pour chaque intervalle, le début et la fin de la chaîne de caractère coupe un mot ou non (et alors potentiellement le nom d'une personne)[1].

Enfin, *pour déterminer les interactions* il suffit d'appliquer la détection de personnages dans la phrase, récupérer la liste des protagonistes présents, puis affecter à chaque relation personnage X <-> personnage Y un **poids** incrémenté de un pour le nombre de tronçon où il y a effectivement **cooccurrence**. 🍊

Par exemple, dans la phrase du dessus, on obtient:

| Source | Cible | Poids |
|---|---|---|
| Bilbo | Gandalf | 1 |
| Gandalf | Bilbo | 1 |

Finalement, je passe par la bibliothèque Python **Networkx** qui permet de créer et travailler avec les réseaux. Elle me permet entre autres d'utiliser des algorithmes de **partitionnement de données**, ou de *clustering*.

?

C'est quoi ça, ça sert à quelque chose? 🍊

Le principe c'est de séparer des données en différentes catégories, sans pour autant savoir au préalable quelles catégories existent. Cela permet de visualiser rapidement des groupes qui *seraient* présents dans nos données.

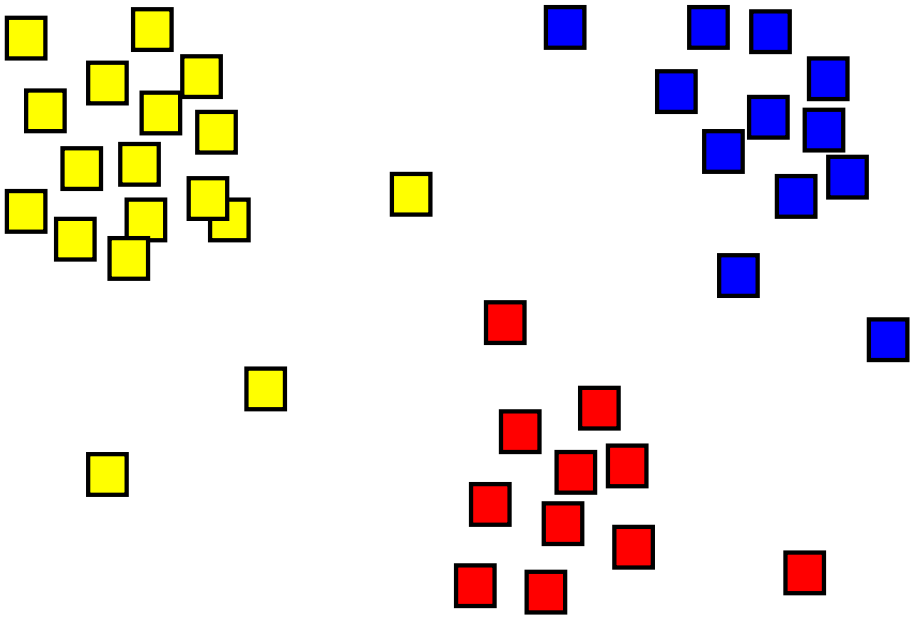

FIGURE 3.3. – Partitionnement en trois groupes (https://commons.wikimedia.org/wiki/File:Cluster-2.svg)

Typiquement, dans l'exemple ci-dessus on réalise un partitionnement en trois groupes de l'ensemble des points. Cela correspond aux trois couleurs (jaune, rouge, bleu). Cette discrimination est réalisée en fonction des distances entre les points: grossièrement on tente de regrouper les points qui se *ressemblent*.

> *i* On cherche en fait à maximiser les ressemblances **intragroupes** et les différences **intergroupes**.

Mon explication serait suffisante si nous avions des points situés sur un repère (x,y). En revanche, notre objet d'étude est un **graphe non orienté**: il y a des **nœuds** (les personnages) et des **arêtes** entre eux (le poids de leur relation). Les arêtes ne sont **pas orientées**, c'est à dire que A vers B est équivalent à B vers A.

Ça sera probablement plus clair avec un **exemple concret**. Bordeaux et Paris sont des **nœuds**, les routes les séparant sont des **arêtes** avec un poids égal au temps nécessaire pour les traverser. Intuitivement, la durée d'un trajet Bordeaux - Paris est équivalente au trajet inverse[2]. Or, comme le sens ne fait pas varier la durée, le graphe est **non orienté**.

Pour détecter des *communautés* de personnages dans les graphes, on peut utiliser la méthode de Louvain [3]. Pour faire simple, on cherche à maximiser la *modularité* notée $Q$ et que l'on calcule avec cette formule:

$$Q = \frac{1}{2m} \sum ij \left[ A_{ij} - \frac{k_i k_j}{2m} \delta(c_i, c_j), \right.$$

Pour plus de détail, je vous invite à lire le papier traitant de cette méthode. Ce qu'il faut retenir, c'est qu'il est possible d'**extraire des communautés** à partir de nos données et que la bibliothèque Python **Networkx** le fait en une seule commande! 🍊

> *i* On peut alors comparer les structures que nous identifions en tant que lecteur et celles qui sont repérées par l'algorithme!

## 4. … et décrire l'univers

En quelques lignes, je crée le graphe avec **Networkx** et je l'ouvre avec le logiciel **Gephi** pour gérer le visuel. Sans rentrer trop dans le détail, la disposition des nœuds s'est faite avec l'algorithme **Force Atlas II**. C'est ce qu'on appelle une disposition *force-based* : les arêtes jouent le rôle de ressorts et les nœuds des particules électrostatiques. Le rendu final est finalement une disposition qui présente un état stable. 🥳

1. Ce n'est pas difficile à faire, mais au moment où j'écris ces lignes je n'ai pas trouvé le temps de me remettre sur le projet.
2. Ce qui est très probablement faux
3. Blondel, Vincent D ; Guillaume, Jean-Loup ; Lambiotte, Renaud ; Lefebvre, Etienne (2008). Fast unfolding of communities in large networks , 2008(10), 0–0.doi:10.1088/1742–5468/2008/10/p10008

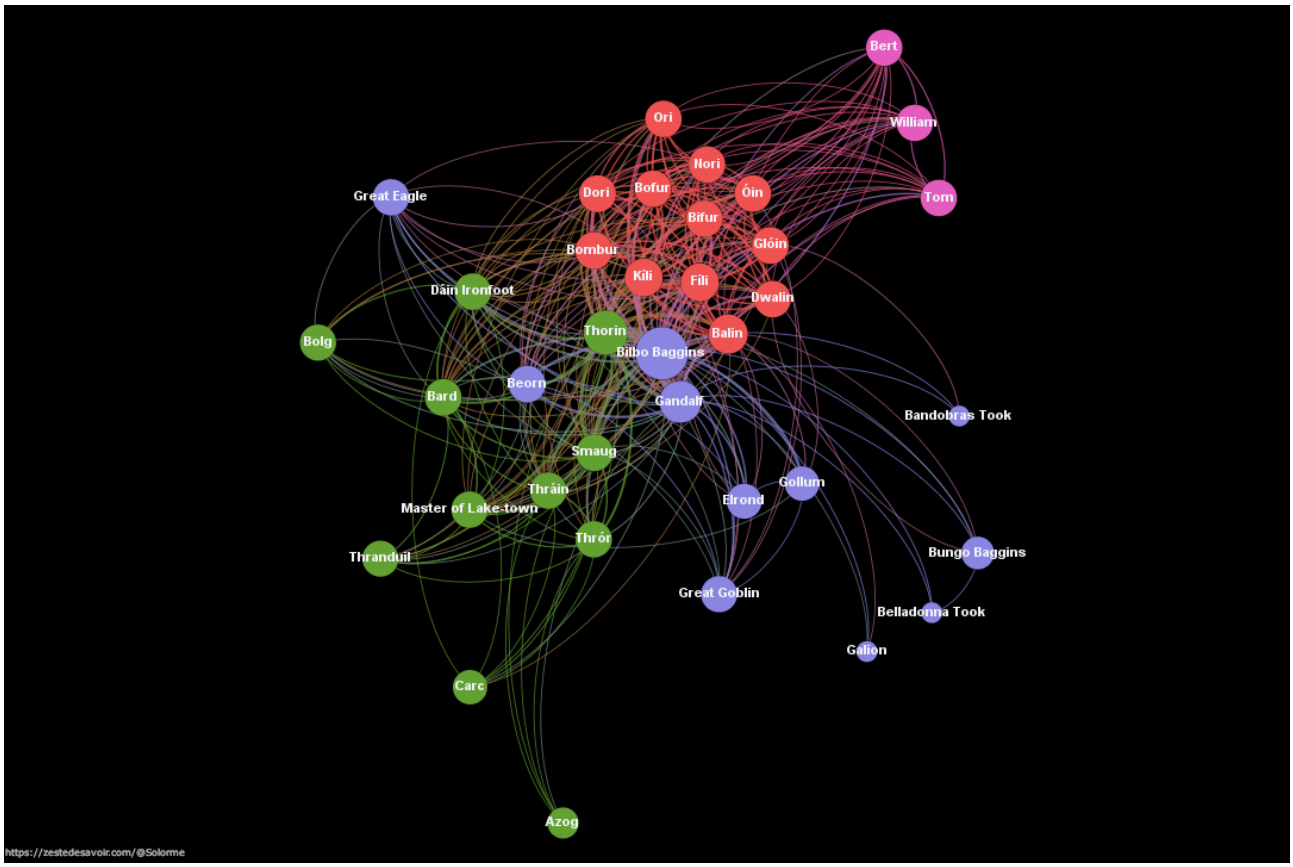


FIGURE 4.4. – Rendu avec Gephi: les couleurs indiquent des communautés différentes, plus une arête est épaisse plus le nombre de cooccurrences est élevé, la taille des nœuds traduit leur importance (suivant une courbe logarithmique)

Ce qui est assez drôle, c'est qu'en jouant sur les paramètres du Louvain et en favorisant l'apparition de plus petites communautés, on peut faire ressortir que Kili et Fili ont des liens particuliers, qui sont frères.

On dénote 4 catégories: les **rouges** sont les Nains de la compagnie de Thorin, les **roses** sont les Trolls du début du livre, les **verts** ont l'air d'être les personnages plus liés à l'histoire d'Erebor et du dragon Smaug, les **bleus** sont les autres personnages fortement liés avec Bilbo et Gandalf. C'est en tout cas l'interprétation que j'en fais, et **j'avoue être très surpris**, en bien, du résultat de la classification!

Mais ce n'est pas fini!
Grâce à la mise en place du graphe, on peut même faire des analyses plus poussées! Par exemple, l'indicateur **betweenness centrality** mesure la fréquence d'apparition de chaque nœud dans les plus courts chemins de chaque combinaison de nœuds. Dans l'image ci-dessous, il est clair que Thorin, Gandalf et Bilbo sont des personnages centraux qui ont des cooccurrences avec la plupart des personnages et jouent le rôle de pont entre les différentes communautés. D'où aussi leur position centrale dans le graphe!

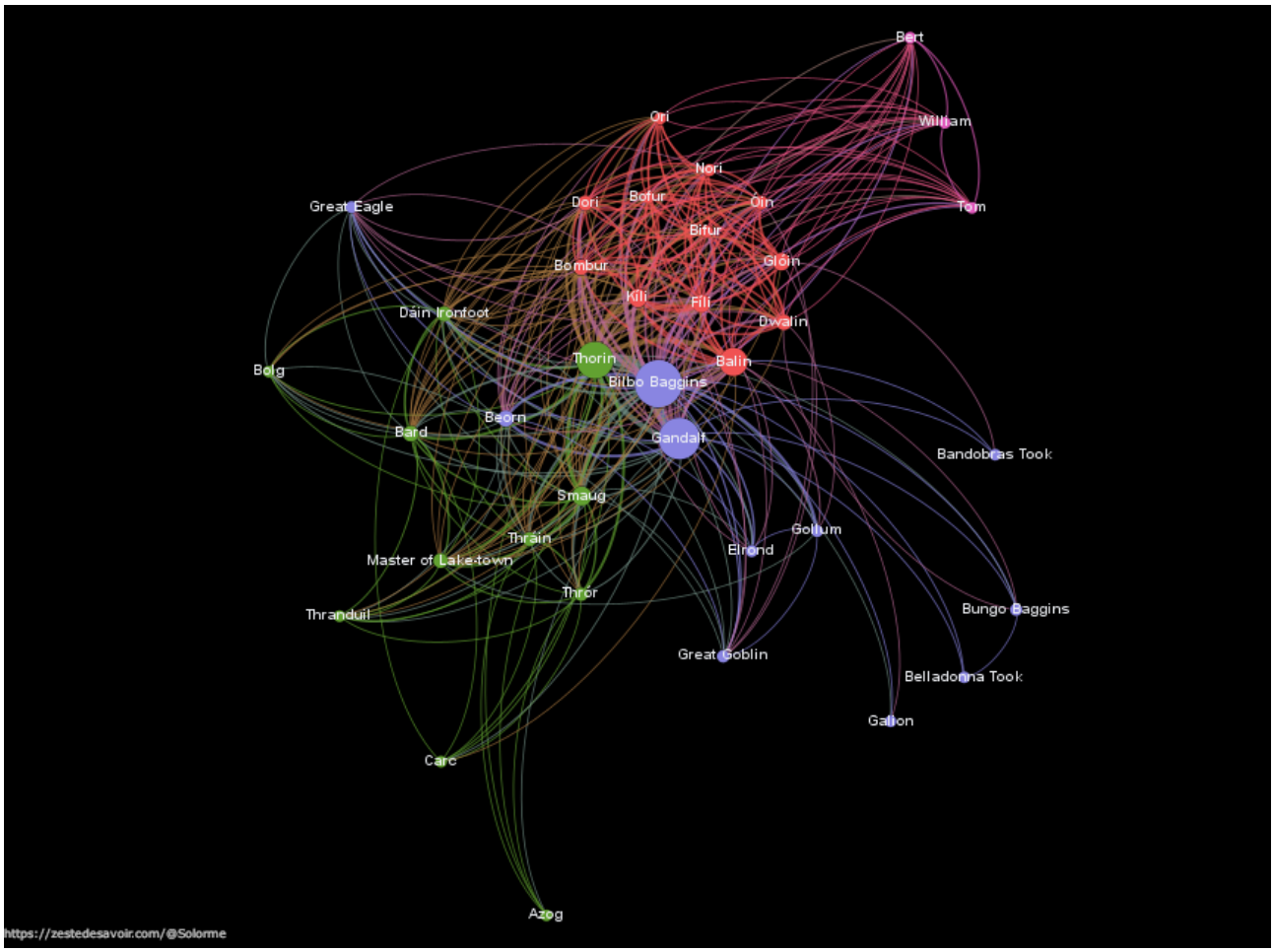


FIGURE 4.5. – Visualisation de l'indicateur Betweenness centrality dans l'univers du Hobbit

## Conclusion

Pour le Hobbit, les cooccurrences des protagonistes approximent bien les relations que l'on a en tête en tant que lecteur. Et pourtant, hormis le choix des paramètres de l'algorithme de Louvain et la valeur seuil de l'unité narrative, à aucun moment je n'ai été en capacité d'orienter le résultat vers une idée préconçue.

**Maintenant que tout le processus est mis en place**, rien ne nous empêche de travailler avec les corpus des autres livres de l'univers, de mettre en évidence des groupes selon la méthode de Louvain ou même simplement selon leur race ou groupe!

Enfin, nous avons rapidement vu que **travailler avec des graphes** permet certes d'avoir un visuel très attractif mais également donne accès à de nombreux **outils** mathématiques et **indicateurs** statistiques qui décrivent l'objet d'étude .

Je préciserai que je n'ai pas relaté toutes les difficultés du projet (comme l'obtention des corpus, le nettoyage des données…) car ce n'était pas le cœur du sujet mais pourtant presque l'essentiel du temps que j'y ai consacré.

J'espère que vous aurez apprécié cet article, n'hésitez pas à réagir dans les commentaires ci-dessous!

---

Merci à @Taurre pour la relecture et la contribution au billet! 🍊