

## TABLE DES MATIERES

1. Remerciements .....	vi
2. Introduction .....	1
3. État de l'art .....	2
3.1. Introduction.....	2
3.2. Introduction aux fanfictions et exploration du genre .....	2
3.2.1. Introduction aux fanfictions et leurs communautés .....	2
3.2.1.1. Qu'est-ce qu'une fanfiction ? .....	2
3.2.1.2. Un espace queer et féminin .....	3
3.2.1.3. Fanfiction : remise en question et renversement des normes de la société ? .....	4
3.2.2. Un réseau interconnecté toujours en mouvement .....	6
3.2.2.1. Une collaboration en temps réel .....	6
3.2.2.2. Des sous-genres sans règles et sans frontières .....	6
3.2.3. Les sous-genres en fanfiction et leur classification .....	7
3.2.3.1. Sous-genres en fanfictions : les <i>fandoms</i> au centre d'une formule particulière.....	7
3.2.3.2. La place centrale des personnages et leurs relations .....	8
3.3. Panorama des méthodes et outils d'analyse de genre .....	9
3.3.1. Le topic modeling .....	10
3.3.1.1. Présentation et outils .....	10
3.3.1.2. Exemple d'application : analyse de pièces de théâtre .....	12
3.3.1.3. Proposition d'un prototype de visualisation des résultats du topic modeling .....	13
3.3.2. La textométrie .....	13
3.3.2.1. Présentation et outils .....	13
3.3.2.2. Exemple d'application .....	14
3.3.3. La stylométrie .....	15
3.3.3.1. Présentation et outils .....	15
3.3.3.2. Exemples d'application.....	17
3.3.4. L'apprentissage automatique .....	18
3.3.4.1. Présentation et outils .....	18
3.3.4.2. Exemples d'application.....	21
3.4. Précédentes applications du TAL dans l'analyse de fanfiction.....	23
3.4.1. Étude sur les paratextes de fanfiction .....	23
3.4.2. Analyse du succès et de l'impact des fanfictions auprès des fans .....	24

3.4.2.1. Prédire le succès d'un personnage à l'aide d'un modèle d'apprentissage automatique .....	24
3.4.2.2. Rapport entre style et succès d'une fanfiction.....	25
3.4.2.3. Impact des fanfictions sur les compétences de rédaction et d'analyse littéraire des fans .....	26
3.4.3. Décéler les différences avec les œuvres originales .....	27
3.4.3.1. Reconnaissance de personnages et leurs mentions dans le texte .....	27
3.4.3.2. Différences dans les couples de personnages .....	28
3.4.3.3. Différences dans la caractérisation des personnages .....	29
3.4.4. Point de vue social : l'influence de la société sur les fanfictions .....	30
3.4.4.1. Pourcentage des personnages en fonction de leur genre .....	30
3.4.4.2. Impact des évènements LGBTQ+ .....	32
3.5. Conclusion et contribution personnelle .....	33
4. Méthodologie.....	35
4.1. Introduction.....	35
4.2. Choix des données .....	35
4.2.1. Généralités sur le corpus.....	35
4.2.2. Choix des tags .....	35
4.2.3. Hypothèses sur les données .....	36
4.3. Collecte des données.....	36
4.4. Prétraitement des données.....	39
4.4.1. Fusion et premiers nettoyages des fanfictions .....	39
4.4.2. Nettoyages des textes et limites maximales de longueur.....	40
4.4.3. Stop-words.....	41
4.4.4. Tokenisation et lemmatisation .....	41
4.4.5. Longueur minimale .....	42
4.5. Vectorisation des données .....	42
4.6. Construction des corpus de développement et de test .....	43
4.7. Entraînement des algorithmes de classification .....	44
4.7.1. Objectifs de la classification.....	44
4.7.2. Préparation des données .....	45
4.7.3. Classification sur les trois tags principaux .....	45
4.7.3.1. Entraînement et validation croisée .....	45

4.7.3.2. Première analyse des caractéristiques discriminantes .....	45
4.7.3.3. Impact de la suppression de la ponctuation .....	46
4.7.3.4. Evaluation sur le corpus de test.....	46
4.7.4. Classification sur les cinq tags.....	46
4.7.4.1. Déséquilibre des classes .....	46
4.7.4.2. Stratégies d'équilibrage des classes .....	46
4.7.4.3. Évaluation sur les données de test .....	47
4.8. Analyse stylométrique avec pydistinto .....	47
4.8.1. Préparation des données .....	47
4.8.2. Installation et modifications du code .....	48
4.8.3. Paramétrage de pydistinto .....	48
4.8.4. Analyses effectuées .....	48
4.9. Conclusion .....	49
5. Résultats et analyse .....	50
5.1. Introduction.....	50
5.2. Performances de la classification automatique .....	50
5.2.1. Impact des prétraitements et des différents algorithmes .....	50
5.2.2. Impact de la ponctuation .....	51
5.2.3. Performances sur les données de test .....	52
5.3. Analyse des mots discriminants et vocabulaire des tags.....	52
5.3.1. Thèmes caractéristiques des tags .....	52
5.3.2. Similarités et différences entre tags .....	61
5.3.2.1. Analyse des matrices de confusion .....	61
5.3.2.2. Analyse des résultats de pydistinto en un contre un .....	64
5.4. Longueurs des fanfictions .....	72
5.4.1. Moyennes et écart-types .....	72
5.4.2. Diagramme à moustaches .....	74
5.5. Conclusion .....	75
6. Conclusions et perspectives .....	76
7. Bibliographie .....	77
8. Annexes .....	82
8.1. Annexe 1 : Portions de code modifiées dans le script pour la collecte d'identifiants d'AO3scraper .....	82

8.2. Annexe 2 : Portions de code modifiées dans le script pour la collecte des fanfictions et leurs métadonnées d'AO3scraper .....	83
8.3. Annexe 3 : Code suppression des noms de personnages .....	87
8.4. Annexe 4 : Entraînement et validation croisée de la classification automatique .....	88
8.5. Annexe 5 : Paramètres optimaux avec GridSearchCV .....	89
8.6. Annexe 6 : Entraînement sur les cinq tags avec pondération des classes .....	89
8.7. Annexe 7 : Entraînement sur les cinq tags avec rééchantillonnage .....	90
8.8. Annexe 8 : Entraînement sur les cinq tags avec pondération et rééchantillonnage .....	91
8.9. Annexe 9 : Exemple du contenu d'un fichier metadata.csv utilisé pour l'analyse par pydistinto .....	91
8.10. Annexe 10 : Exemple du contenu d'un fichier parameters.txt pour l'analyse par pydistinto .	92
8.11. Annexe 11 : Graphes des 15 mots les plus discriminants de chaque tag suite à la classification automatique à l'aide de LinearSVC.....	93

## **1. Remerciements**

---

Je souhaite tout d'abord remercier mon directeur de mémoire M. RUIZ pour son accompagnement et ses conseils pour ce mémoire mais aussi plus généralement pour son implication tout au long de ces deux années de master.

Je veux par ailleurs remercier l'ensemble des enseignants du master Technologies des langues, dont les cours ont nourri ma réflexion, m'ont fait progresser à travers la découverte de nouveaux outils et concepts et m'ont ainsi permis de mener à bien ce travail.

Merci aussi à ma famille qui m'a toujours soutenue et notamment dans mes études.

Je tiens évidemment à remercier mes camarades de promotion pour ces deux dernières années. Merci pour le soutien, l'entraide, la bonne humeur et les bons moments passés pendant les cours et en dehors.

Enfin, merci au monde des séries et les personnes qui le peuplent. Un monde où j'ai toujours aimé m'évader et sans lequel ce mémoire n'existerait pas. Et bien entendu, je remercie plus particulièrement les auteurs de fanfictions et leur créativité qui ont constitué la matière première de cette étude.

## 2. Introduction

---

La fanfiction constitue aujourd’hui un volet important de la production littéraire amateur, avec des millions d’œuvres publiées en ligne sur des plateformes dédiées. Ces récits, écrits par des fans pour des fans, explorent des univers et des personnages issus de franchises populaires en y ajoutant de nouvelles perspectives, relations ou intrigues. Afin de structurer cette production foisonnante et d’aider les lecteurs à naviguer parmi les œuvres, les auteurs utilisent un système de tags, qui permet d’indiquer les thématiques, les dynamiques relationnelles ou encore les tonalités émotionnelles des récits.

Parmi ces tags, certains reviennent fréquemment et semblent qualifier des genres spécifiques de fanfictions. Toutefois, leur signification et leur utilisation restent relativement peu étudiées d’un point de vue quantitatif. Ce mémoire propose ainsi d’analyser cinq tags particulièrement populaires : *Angst*, *Fluff*, *Enemies to Lovers*, *Friends to Lovers* et *Hurt/Comfort*. L’objectif est d’examiner dans quelle mesure ces tags influencent la nature des récits, que ce soit à travers le vocabulaire employé, la longueur des textes ou la capacité à les différencier automatiquement via des modèles de classification.

Dans ce mémoire, le terme sous-genre ne renvoie pas aux genres littéraires classiques (roman, théâtre, poésie), mais aux catégories thématiques et relationnelles définies par les communautés de fans à travers le système de tags de la plateforme AO3Ces tags (par ex. *Fluff*, *Angst*, *Hurt/Comfort*) fonctionnent comme des regroupements sémantiques hybrides, à la fois descriptifs du contenu, du ton émotionnel et des relations entre personnages.

Dans un premier temps, un état de l’art reviendra sur les recherches et outils existants en linguistique et en analyse computationnelle des textes littéraires, en mettant en avant les spécificités de la fanfiction. Ensuite, la méthodologie employée sera détaillée, de la collecte des données à l’application d’outils d’analyse textuelle et de classification. Cette méthodologie sera suivie de l’étude des résultats qui permettra de mettre en lumière les différences lexicales et structurelles entre les tags sélectionnés. Enfin, une conclusion soulèvera les limites et perspectives futures de l’analyse des tags en fanfiction.

### **3. État de l'art**

---

#### **3.1. Introduction**

Les fanfictions sont ces récits écrits par des fans en s'inspirant d'œuvres canoniques et selon leurs propres désirs et envies. Elles constituent une forme littéraire encore assez récente qui s'est développée et répandue de manière exponentielle ces dernières années, notamment grâce à l'entrée dans l'ère numérique et l'avènement des réseaux sociaux. Bien que ces récits puissent parfois être décriés par certains, ils constituent un matériau littéraire contemporain qui ne peut être ignoré. Ne répondant pas aux normes et conventions existantes tant sur la forme que sur le fond, les fanfictions représentent en effet un nouveau lieu d'étude dans de nombreux domaines comme la littérature, les sciences sociales, la linguistique, ou encore le traitement automatique des langues (TAL).

Dans un premier temps, cet état de l'art cherche à offrir une introduction au monde des fanfictions en présentant leur origine et fonctionnement. Il montre que les fanfictions se créent en complément voire en opposition au canon et qu'elles offrent donc une opportunité d'étudier un nouveau point de vue sur la société. Il aborde aussi le système innovant et atypique que l'univers des fanfictions adopte pour définir et classifier ses récits en genres et sous-genres bien particuliers. Cette première partie permet aussi de mettre en avant certaines hypothèses et conclusions faites par plusieurs études non issues du TAL.

Dans un second temps, l'état de l'art vise à donner un panorama des différents outils et méthodes de TAL disponibles pour l'analyse textuelle de fanfiction. En effet, le TAL permet d'étudier efficacement de vastes ensembles de données textuelles provenant des fanfictions. Il permet de s'intéresser entre autres aux styles d'écriture, aux thèmes ou dynamiques narratives de différents textes. Cet état de l'art s'attarde plus particulièrement sur quatre approches : le topic modeling, la textométrie, la stylométrie et l'apprentissage automatique. Pour chacune de ces approches, une définition ainsi qu'un inventaire d'outils et méthodes à disposition sont données, suivies d'une liste d'exemples de mise en pratique de ces outils et méthodes.

Dans un troisième temps, cet état de l'art liste et détaille plusieurs exemples de précédentes études de TAL appliquées aux fanfictions. Ces études permettent notamment d'infirmer ou de confirmer plusieurs points soulevés dans la première partie de l'état de l'art. Elles offrent également de nouvelles observations sur différents éléments qui touchent à l'univers des fanfictions ainsi que de nouvelles perspectives à explorer.

La dernière partie de l'état de l'art apporte un récapitulatif et une conclusion des différents points abordés tout en énonçant les contributions personnelles que ce mémoire tentera d'apporter au sujet.

#### **3.2. Introduction aux fanfictions et exploration du genre**

##### **3.2.1. Introduction aux fanfictions et leurs communautés**

###### **3.2.1.1. Qu'est-ce qu'une fanfiction ?**

Xanthoudakis (2021) définit la fanfiction comme un objet participatif ou collaboratif, fruit de l'interaction entre l'œuvre initiale d'un auteur et les contributions créatives des fans. C'est un texte de fiction qui est écrit par un ou plusieurs fans et qui s'inspire de l'univers et des personnages d'une œuvre déjà existante, comme une série télévisée, un film, un livre, etc. Les fans auteurs reprennent des éléments de l'œuvre originale, les modifient ou en créent de nouveaux, souvent pour explorer des aspects qu'ils auraient souhaité voir développés davantage ou autrement et pour répondre à des

attentes narratives qu'ils estiment qu'elles ont été laissées de côté voire sont complètement manquantes. Goldmann (2022) explique d'ailleurs que les fanfictions naissent de l'imagination des fans qui se posent la question « Et si ? » (« Et si ce personnage n'était pas mort ? », « Et si cette histoire se passait dans une autre époque ? Dans un autre lieu ? », « Et si tel ou tel événement n'avait pas eu lieu ? », etc.), et qui écrivent la réponse à ces « Et si ».

Les fanfictions sont souvent le fruit de l'imaginaire de différentes *fandoms* et sous-fandoms (Xanthoudakis, 2021). Les *fandoms* sont le nom que l'on donne aux communautés de fans qui se forment autour d'une entité de la culture populaire. Quand on parle de *fandoms* et de fanfictions, on pense souvent à séries, films ou livres, cependant il faut aussi noter que les *fandoms* et fanfictions peuvent aussi se créer autour d'une célébrité, d'une marque, d'une équipe sportive, etc. Au sein d'une *fandom*, plusieurs sous-fandoms peuvent se former autour de différentes spécificités de l'œuvre qui intéressent tout particulièrement certains fans. Les personnages et les relations entre les personnages d'une œuvre représentent un des éléments les plus populaires à l'origine de nombreuses sous-*fandoms*, et qui suscitent le plus de passions et débats au sein des *fandoms* et sous-fandoms. C'est ainsi que certaines *fandoms* se créent parfois presque en opposition ou désaccord avec des faits établis par l'œuvre originale ou des idées supportées par d'autres *fandoms*. On peut alors donner comme exemple ce que certains appellent les « guerres de *ships* » – un *ship* étant un nom utilisé pour parler d'un couple supporté par certains fans – où différentes *fandoms* sont en désaccord sur quel personnage devrait être en couple avec quel personnage, ce qui peut parfois mener à de vifs débats sur les réseaux sociaux. Un autre exemple sont les *fandoms* qui se forment quand une partie des fans souhaite qu'un personnage soit queer, mais qu'il ne l'est pas (ou du moins pas explicitement) dans l'œuvre originale. Les fanfictions reflètent toute cette myriade de différents points de vue et opinions.

### 3.2.1.2. Un espace queer et féminin

Puisque de nombreuses *fandoms* et fanfictions se construisent en opposition, ou du moins sur la base de points de vue divergents avec l'œuvre de base, il n'est pas étonnant qu'elles forment un espace regroupant majoritairement des groupes de personnes qui sont souvent moins représentés ou moins mis en avant dans la société et dans les œuvres du canon. C'est ainsi que Xanthoudakis (2021) et Busse (2017), entre autres, s'accordent à dire que les fanfictions sont un espace plutôt féminin et queer. Xanthoudakis (2021) cite notamment les travaux de (Hellekson & Busse, 2014), (Jamison, 2013), (Jenkins, 2013) et (Bacon-Smith, 1992) pour expliquer que les fanfictions ont vu le jour dans les années 60 autour d'œuvres et *fandoms* de science-fiction puis ont continué de se développer dans les années 70 sous l'impulsion de fans féminins qui s'en sont détachées pour former leur propre *fandom*, en réponse notamment au sexism et élitisme présent dans les œuvres et *fandoms* originales de science-fiction.

Pour Rouse (2021), les *fandoms* et fanfictions peuvent constituer un refuge pour certains fans qui ne se sentent pas à leur place dans leur vie de tous les jours. Ecrire des fanfictions et appartenir à une *fandom* peuvent leur apporter une expérience sociale positive qu'ils ne retrouvent pas ailleurs. Rouse interviewe une autrice de fanfiction qu'elle nomme Ruby. Cette dernière explique que d'intégrer le monde des *fandoms* lui a permis d'explorer sa propre identité, de créer des liens avec des personnes parfois à l'autre bout du monde et de pouvoir librement discuter de plusieurs sujets, notamment liés à la sexualité dont elle n'osait pas forcément discuter ailleurs.

En plus de créer un espace où les fans se sentent à l'abri et libre d'être eux-mêmes, Busse (2017) considère aussi qu'elles offrent une plus grande visibilité à la communauté queer, et plus particulièrement les femmes queer qui occuperaient un plus grand pourcentage dans la population des *fandoms* qu'elles n'occupent dans la population en général. Les fanfictions qu'elles écrivent leur permettent d'explorer des questions de genre et de sexualité. Aussi, pour Busse, les questions

tournant autour du genre, de la sexualité mais aussi de l'égalité sont un élément central à de nombreuses fanfictions qui explorent des relations romantiques et/ou sexuelles entre plusieurs personnages qui partagent souvent une même identité de genre.

Cette grande présence des minorités ou groupes moins mis en lumière en temps normal, notamment LGBTQ+, parmi les communautés de fanfictions a suscité l'intérêt de différents chercheurs, par exemple en sciences sociales, pour étudier la vision que pourraient avoir ces fans sur la société et ses normes.

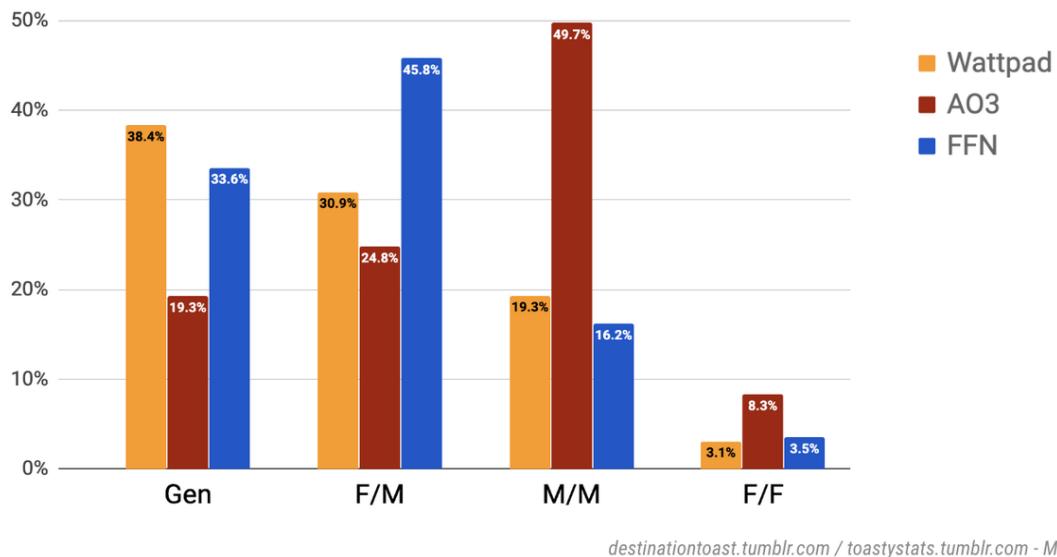
### **3.2.1.3. Fanfiction : remise en question et renversement des normes de la société ?**

Les fanfictions étant un espace plutôt queer et féminin, et complètement libre où les fans peuvent s'exprimer comme ils le veulent sans restriction, on pourrait s'attendre à ce que les fanfictions et leur contenu viennent challenger les normes de la société (par exemple, des concepts rattachés à l'hétéronormativité ou au patriarcat), et pourtant plusieurs études ont observé que ce n'est pas toujours le cas.

Tout d'abord, bien que les fanfictions semblent être un espace plutôt queer et féminin, peuplé majoritairement d'acteurs s'identifiant comme féminins et/ou LGBTQ+, on remarque que le pourcentage de fanfictions F/F (traitant d'une relation entre deux personnages féminins) est bien plus faible que le pourcentage de fanfictions M/M (traitant d'une relation entre deux personnages masculins). En effet, le site Fandom stats. (s.d.) qui collecte des données concernant plusieurs sites de fanfictions, a posté en 2019 un graphique représentant la répartition des différents types de *ships* dans les fanfictions disponibles sur ses sites (Figure 1). les *ships* en question sont les suivants : Gen (fanfictions qui ne traitent pas d'une relation romantique et/ou sexuelle), F/M (traitant d'une relation entre un personnage féminin et un personnage masculin), M/M et F/F.

Figure 1 - Répartition des différents *ships* de relation sur les sites de fanfiction

## Shipping across platforms



[destinationtoast.tumblr.com](http://destinationtoast.tumblr.com) / [toastystats.tumblr.com](http://toastystats.tumblr.com) - May 2019

1

On peut voir que sur les trois sites traités (AO3, Wattpad, FanFiction.net), les fanfictions F/F arrivent en dernière position, représentant seulement 3 % des fanfictions de Wattpad et Fanfiction.net et 8 % des fanfictions d'AO3. Sur Wattpad et Fanfiction.net, les fanfictions F/M arrivent devant les fanfictions M/M, avec une proportion d'environ 25 à 30 % contre une proportion d'un peu moins de 20 % pour les fanfictions M/M. Sur AO3, ce sont les fanfictions M/M qui arrivent en tête, représentant presque 50 % des fanfictions de la plateforme. Au final, bien que la proportion de fanfictions LGBTQ+ (environ 33 %, en comptant les fanfictions M/M et F/F) dépasse le pourcentage de personnes s'identifiant comme LGBTQ+ dans la population générale (un peu plus de 10 % selon un rapport Ipsos (2022)), les rapports de force restent globalement similaires, si ce n'est pour le fort pourcentage de fictions M/M sur AO3.

Goldmann (2022) étudie les fanfictions, et plus particulièrement le sous-genre *Slash* qu'elle voit comme un sous-genre qui laisse la place à l'exploration du monde queer pour ses auteurs et lecteurs. Pour elle, ce sous-genre permet aux auteurs, qu'ils s'identifient comme queer ou non, d'explorer leurs désirs et fantasmes librement. Ainsi, toutes les fanfictions *Slash* offrent la possibilité d'adopter certains aspects codés comme queer dans leur histoire si l'auteur le souhaite, et ce peu importe que leur fanfiction traite ou non d'un couple considéré comme queer. Ces critiques reprochent à certains auteurs, d'une part une appropriation parfois maladroite voire malsaine de codes queer dans des histoires hétéronormées ; d'autre part la reproduction de stéréotypes hétéronormatifs dans des couples queer quand ils pourraient au contraire mettre en avant une façon de penser ou d'être différente. Cependant, Goldmann rappelle aussi que les fanfictions ont pour objectifs premier le plaisir des fans et que ce plaisir passe parfois par un certain confort dans la réutilisation de stéréotypes sociaux. De plus, ramener un élément un tant soit peu politique dans les fanfictions viendrait gâcher ce plaisir de les lire et écrire.

---

<sup>1</sup> Source : Fandom stats.

Busse (2017) s'intéresse aussi au sous-genre *Slash*. Selon elle, ce sous-genre permet de se pencher sur des questions de genre et sexualité en renversant les codes et normes de la société. On y trouve des fanfictions sur des changements de sexe et/ou genre, des grossesses masculines, des fécondations forcées et autres ; des fanfictions qui remettent en question certains codes, comme les codes vestimentaires par exemple ; des fanfictions qui adoptent des narrations queer et transgenre, etc. Les fanfictions du sous-genre *Slash* explorent parfois des questions féministes et s'interrogent notamment sur ce qui fait un homme, une femme, et ce à la fois d'un point de vue biologique et social. Pour Busse, un problème réside dans le fait que ces questions et leurs réponses sont souvent abordées en usant de stéréotypes. Ces stéréotypes sont parfois même exacerbés au point qu'ils peuvent en devenir presque offensants. Busse évoque presque une forme de voyeurisme, en mentionnant que beaucoup d'auteurs féminins, en général s'identifiant comme hétérosexuelles, écrivent des fanfictions plus ou moins explicites sur des couples entre deux hommes. Cela pourrait d'ailleurs expliquer que les fanfictions M/M soient si populaires alors que les fanfictions F/F restent sous-représentées. Busse met aussi en avant qu'un bon nombre de fanfictions à succès mettent en scène des relations violentes et/ou toxiques, souvent glorifiées.

### **3.2.2. Un réseau interconnecté toujours en mouvement**

#### **3.2.2.1. Une collaboration en temps réel**

En littérature, en dehors des activités de communication ou de promotion (telles que des conférences, entretiens ou sessions de questions-réponses organisées par exemple), la principale connexion entre les lecteurs et l'auteur reste le texte lui-même. La particularité des sites de fanfictions est qu'ils permettent aux fans, lecteurs et auteurs, d'interagir directement pendant le processus de création.

Une fanfiction n'est en général pas publiée d'une traite comme un livre, mais chapitre par chapitre sur le site de fanfictions. Cela laisse la possibilité aux lecteurs de laisser des commentaires après chacun de ces chapitres. Les auteurs peuvent lire ces commentaires et s'en inspirer pour la suite de l'écriture. Ils peuvent même demander des conseils aux autres fans et chatter avec eux en temps réel sur les sites de fanfictions ou les réseaux sociaux qui sont très utilisés par de nombreux fans (Goldmann, 2022).

Comme les fanfictions se construisent et sont publiées dans le temps, ce ne sont pas des œuvres figées, mais des œuvres qui évoluent en fonction de ce qui les entourent. Elles sont influencées, comme évoqué précédemment, par les retours et discussions entre fans. L'évolution de l'œuvre source joue aussi un rôle dans la construction des fanfictions, notamment lorsque l'œuvre source n'est pas finale (que des épisodes, livres, films, etc. continuent de sortir par exemple). Des événements de société peuvent laisser une marque sur certaines fanfictions. Bien sûr, des changements au sein de la *fandom* concernée auront aussi un impact. Ceci est tout autant de critères qui font que les fanfictions évoluent en permanence. Et si les fanfictions sont toujours en mouvement, leurs sous-genres le sont aussi.

#### **3.2.2.2. Des sous-genres sans règles et sans frontières**

Les sous-genres de la fanfiction s'apparentent à des *tags*. Contrairement à la production littéraire classique, où des genres préexistants encadrent la catégorisation des œuvres (même si de nouveaux sous-genres peuvent émerger progressivement, notamment sous l'impulsion de la critique ou de la recherche académique), en fanfiction ce sont les auteurs eux-mêmes qui *taggent* leurs œuvres et définissent ainsi leurs propres sous-genres. Les tags ne sont limités par aucune règle de forme ni de

contenu. Les fans peuvent décrire et classer leurs œuvres comme bon leur semble, avec autant de mots ou caractères qu'ils le veulent par tag et autant de tags qu'ils le veulent par fanfiction.

Busse (2017) explique aussi que des nouveaux tags ou sous-genres de fanfictions peuvent se créer à partir d'autres tags déjà existants. En effet, les fans écrivent des fanfictions en partant souvent d'une idée ou structure de sous-genre générique ou préétablie auquel ils viennent ajouter leur touche personnelle, une nouvelle caractéristique par leur propre créativité. Ce faisant, ils créent un nouveau tag ou sous-genre qui peut être plus ou moins proche d'un sous-genre préexistant. On peut presque dire qu'avec chaque nouvelle fanfiction, un nouveau sous-genre apparaît même s'il est parfois très proche voire répétitif d'un sous-genre déjà présent. Cela est rendu possible par le fait qu'il n'y a pas de codes ou contraintes à respecter quand on écrit de la fanfiction, l'auteur est complètement libre.

Cette absence de limites dans la création et dans la réutilisation de tags crée un système de classification géant et dans lequel n'est pas toujours évident de s'y retrouver. Certaines fanfictions peuvent compter des dizaines et des dizaines de tags différents et aux appellations parfois obscures (Figure 2).

Figure 2 - Exemple des tags associés à une seule fanfiction



2

Pour décrire comment les fanfictions et leurs sous-genres se répandent, Xanthoudakis (2021) utilise l'image du mycélium, un réseau sous-terrain tentaculaire de champignon. « Le Mycélium modifie son environnement et est modifié par son environnement »<sup>3</sup> (Xanthoudakis, 2021, p. 62). Les fanfictions et sous-genres se répandent en partant d'un post d'une photo, d'un commentaire ou autres, qui est repris et encore repris, et parfois modifié en route. Ils n'ont pas de frontières, certains se répandent plus vite et plus loin que d'autres et obtiennent un statut plus important et reconnu à travers les différentes *fandoms*.

Le réseau que forment les fanfictions et leurs tags et sous-genres est donc tentaculaire, fait de multiples branches, nœuds et sous-réseaux tous interconnectés, sans frontières et est en mouvement constant. Cela fait tout l'intérêt mais aussi toute la difficulté qui y a à l'étudier.

### 3.2.3. Les sous-genres en fanfiction et leur classification

#### 3.2.3.1. Sous-genres en fanfictions : les *fandoms* au centre d'une formule particulière

Les auteurs de fanfictions suivent des « formules » basées sur des clichés ou idées préconçues et sur des attentes présentes au sein d'une *fandom* ou sous-fandom particulière (Goldmann, 2022). En

<sup>2</sup> Source : ao3.org

<sup>3</sup> Traduction personnelle de « mycelium shapes and is shaped by its environment ».

général, les auteurs de fanfictions ne sont pas là pour créer une œuvre originale qui n'aurait alors pas d'intérêt pour les fans, mais pour concrétiser un imaginaire déjà présent et répandu parmi ces fans. Goldmann précise que si l'histoire n'est pas forcément là où s'exprime la créativité de l'auteur, celle-ci compte tout de même beaucoup dans le succès ou non d'une fanfiction. C'est bien le style et la touche que l'auteur va apporter, tout en étant capable de rester fidèle à l'œuvre source et plus particulièrement à l'essence de ses personnages, qui va faire la qualité d'une fanfiction pour les lecteurs.

Ce sont ces « formules » que les fans utilisent pour créer ou modifier leurs *tags*. Ces tags représentent alors une promesse pour les lecteurs, la promesse d'y trouver la réponse à certaines attentes. Ces « formules » que certains voient parfois comme un manque de créativité et parfois comme de la redondance sont en fait le fondement des fanfictions et ce qui sert de base à leur classification en différents sous-genres et ce qui permet de les différencier.

On peut dire que les sous-genres en fanfictions sont faits pour et par les fanfictions elles-mêmes en s'inspirant de l'imaginaire et des désirs d'une *fandom*. Ainsi la *fandom* source a une forte influence et importance dans les *tags* et fanfictions qui vont lui être associés. Ainsi, pour comprendre les différents sous-genres ou tags de certaines fanfictions, il faut être familier avec la *fandom* concernée, car certains *tags* se développent pour répondre exclusivement aux besoins d'une *fandom* particulière et usent parfois de références que seuls les fans de cette *fandom* peuvent comprendre (Xanthoudakis, 2021).

Certains *tags* assez versatiles sont parfois très populaires, sont repris à travers plusieurs *fandoms* et deviennent des sous-genres établis et reconnus par tous ou presque (Xanthoudakis, 2021). Goldmann (2022) explique par ailleurs qu'un genre se construit sur la représentation stéréotypique que l'on en a et cite Tudor qui dit que la définition d'un genre est l'interprétation collective qu'un groupe de personnes en fait et qu'une certaine interprétation ne devient vraiment un genre que si elle est reconnue et utilisée par un nombre assez important de personnes.

### 3.2.3.2. La place centrale des personnages et leurs relations

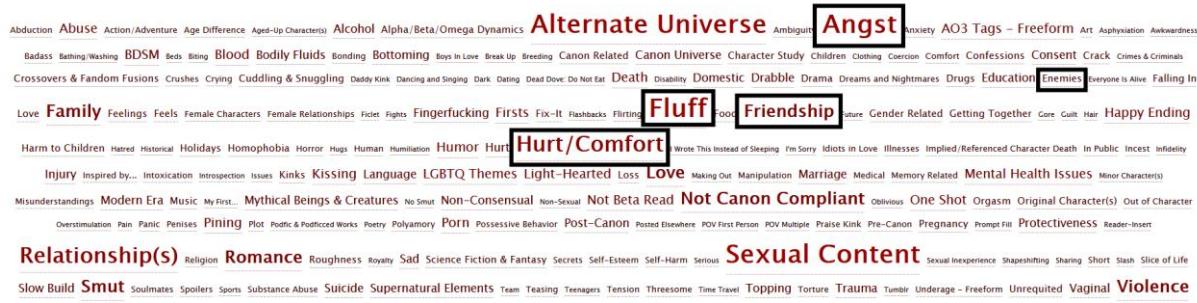
Les sous-genres en fanfictions ne sont pas les genres « traditionnels » qu'on retrouve en littérature mais des sous-genres qui font souvent référence à la relation qu'entretiennent les personnages de la fanfiction. Goldmann (2022) note qu'un critère cher aux fans et donc déterminant dans le succès ou non d'une fanfiction est le fait de retrouver l'esprit et la façon d'être du personnage de l'œuvre originale dans l'écriture de la fanfiction.

Goldmann (2022) explique aussi que la plupart des fanfictions suivent le schéma suivant de classification : Au premier niveau de classification se trouve la *fandom*. Les lecteurs ciblent leur recherche d'abord en fonction de l'œuvre source sur laquelle la fanfiction se base. Au deuxième niveau se trouve la relation entre les personnages. Les fanfictions sont souvent axées sur la romance ou relation particulière qu'entretiennent deux personnages ou plus. Au troisième et dernier niveau se trouve éventuellement le ton ou l'ambiance de la fanfiction. C'est ce qu'on pourrait presque assimiler aux sous-genres littéraires « traditionnels » tels que le thriller, la comédie, le policier, l'aventure, etc. Goldmann donne une exception à ce schéma : les fanfictions *Gen* qui ne se construisent pas autour de la relation entre ses personnages. Pour elle, il existe trois types de fanfictions différents : *Gen*, *Slash* (focus sur des relations queer) et *Het* (focus sur des relations hétérosexuelles). Ainsi même si les fanfictions *Gen* ne correspondent pas au schéma précédent, cela confirme quand même une façon de classer les fanfictions selon la relation (ou non relation) entre les personnages.

Ainsi, une assez grande majorité des tags de sous-genres utilisés sur les sites de fanfiction, sont des tags de relations que les fans utilisent pour décrire le type de relation qu'entretiennent les

protagonistes. Aussi, au milieu du système tentaculaire des sous-genres de fanfiction, des *tags* ont parvenu à s'imposer comme des tags de référence reconnus par presque tous les fans. Beaucoup de ces tags de référence sont des *tags* de relation. Parmi les plus populaires on peut notamment citer les romances *Fluff* que Goldmann (2022) défini en citant Busse comme « des histoires légères qui se soucient moins de leur intrigue que d'avoir un impact réconfortant »<sup>4</sup>; *Angst*, définies de même comme « des fanfictions qui torturent émotionnellement ou physiquement leur(s) personnage(s) »<sup>5</sup>; *Hurt/Comfort* qui sont des « fanfictions qui mettent en scène un personnage blessé qui est consolé par un autre personnage »<sup>6</sup>.

Figure 3 - Les tags les plus populaires de AO3



7

Sur le nuage des tags populaires d'AO3 ci-dessus (Figure 3) on peut aussi remarquer les *tags Friendships* et *Enemies*. Ce sont des tags parapluie qui regroupent plusieurs tags qui traitent de relation entre amis ou entre ennemis. Néanmoins, quand on explore de plus près ces deux sous-genres sur le site d'AO3, on remarque qu'une très grande majorité du sous-genre *Friendships* correspond à des fanfictions *taggées* en tant que *Friends to lovers* – qui comme son nom l'indique sont des fanfictions où deux amis tombent amoureux l'un de l'autre – et qu'une grande majorité du sous-genre *Enemies* correspond à des fanfictions *taggées* en tant que *Enemies to lovers* – des fanfictions où deux ennemis ou rivaux finissent par tomber amoureux.

On peut donc conclure que les personnages et leurs relations sont centraux dans la création, succès et classification d'une fanfiction. Leur prise en compte est donc quasi nécessaire lorsque l'on décide d'étudier et d'analyser des fanfictions. C'est pourquoi ce mémoire s'intéressera à des tags de relation qui sont les cinq tags présentés précédemment : *Fluff*, *Angst*, *Hurt/Comfort*, *Friends to lovers* et *Enemies to lovers*.

### **3.3. Panorama des méthodes et outils d'analyse de genre**

Pour mener à bien une analyse de sous-genres de fanfiction, plusieurs méthodes et outils de TAL sont disponibles.

<sup>4</sup> Traduction personnelle de « [A]n often light story that usually seeks to make a tender emotional impact rather than put forward a plot” ».

<sup>5</sup> Traduction personnelle de « fan fictions that torment their character(s) physically or emotionally ».

<sup>6</sup> Traduction personnelle de « Those fan fictions revolve around a character that is being injured and another character comforting him »

<sup>7</sup> Source : ao3.org

### 3.3.1. Le topic modeling

#### 3.3.1.1. Présentation et outils

Un premier exemple est le topic modeling. Le topic modeling est une méthode d'analyse textuelle qui vise à découvrir les *topics* ou sujets présents dans un corpus de textes. Le principe de base du topic modeling est de supposer que chaque document du corpus est constitué d'un nombre prédéfini de *topics* qu'il cherche à identifier, puis à calculer leur poids ou importance dans chaque document. L'un des modèles de topic modeling les plus utilisés est le Latent Dirichlet Allocation (LDA).

Blei (2012) explique que le topic modeling repose sur des algorithmes qui permettent de dégager les *topics* principaux de larges recueils de documents (pouvant être de formats et de types différents) et de classer ces documents en fonction des *topics* découverts. Ces algorithmes de topic modeling reposent sur des méthodes statistiques d'analyse des mots d'un texte, qui ont pour but de découvrir des *topics*, des liens entre ces différents topics et la façon dont ils évoluent avec le temps.

De son côté, Schöch (2017) définit le topic modeling comme une « méthode non supervisée utilisée pour découvrir des structures sémantiques latentes dans une large collection de textes sans user de ressources lexicales ou sémantiques comme des dictionnaires électroniques »<sup>8</sup> (2017, p. 4). C'est donc une méthode indépendante de toutes ressources extérieures potentiellement biaisées, ce qui signifie que l'algorithme détecte les *topics* sans connaissances préalables sur le contenu des documents.

Pour identifier les différents *topics* ou sujets dans un corpus, les méthodes de topic modeling analysent les mots de chaque document et les répartissent en fonction de leur importance relative dans différents *topics*. Le résultat se présente sous la forme de listes de mots associées à un poids. Ce poids représente l'importance de chaque mot pour un *topic* particulier. Chaque liste de mots représente un *topic* du corpus et contient les termes les plus représentatifs de ce sujet. Les mots sont triés selon leur poids, ce qui permet de repérer les mots clés les plus importants pour chaque *topic*.

Pour interpréter les résultats, on peut extraire les N premiers mots d'une liste en fonction de leur poids, ce qui donne une idée des thèmes centraux du *topic*. Par ailleurs, chaque *topic* est associé à l'ensemble des documents du corpus, mais dans des proportions qui varient selon la pertinence de chaque *topic* pour chaque document. En dernier lieu, c'est à l'utilisateur d'analyser les listes de mots et de déterminer une dénomination pour chaque *topic* en fonction des termes qui en ressortent.

Schöch (2017) explique que le topic modeling repose sur l'hypothèse que le sens d'un mot dépend de son contexte, c'est-à-dire des mots qui l'entourent et qui apparaissent fréquemment ensemble dans les textes. En identifiant ces co-occurrences, les méthodes de topic modeling peuvent découvrir des groupes de mots qui révèlent des thèmes récurrents dans le corpus. Cette approche statistique permet de repérer des thèmes potentiels sans avoir besoin d'une connaissance préalable du contenu, ce qui en fait un outil puissant pour explorer des corpus de grande taille.

L'algorithme LDA, décrit par Blei (2012), est une méthode fondamentale et relativement simple de *topic modeling*. Cet algorithme modélise chaque document comme un mélange de *topics* selon des proportions spécifiques. Pour chaque document, il commence par attribuer aléatoirement un certain poids à chaque *topic*. Ensuite, pour chaque mot du document, LDA choisit un *topic* en fonction de cette proportion puis sélectionne un mot dans le vocabulaire associé à ce *topic*. Ce

---

<sup>8</sup> Traduction personnelle de « an entirely unsupervised method which discovers the latent semantic structure of a text collection without using lexical or semantic resources such as electronic dictionaries »

processus se répète un grand nombre de fois, et à chaque itération, le modèle affine ses estimations, ajustant les proportions des *topics* et les mots qui y sont associés pour mieux représenter le contenu du corpus. Au fil des itérations, le modèle devient de plus en plus précis dans sa manière d'assigner des mots aux *topics*, révélant ainsi les thèmes sous-jacents du texte.

Ces algorithmes de *topic modeling* fonctionnent sans connaître les *topics* au départ ; ils les déduisent uniquement à partir du texte en observant les répétitions et les regroupements de mots. Leur objectif est de découvrir les thèmes cachés dans le corpus, qu'on appelle aussi des "variables latentes", en se basant sur les mots présents. Parmi les outils informatiques populaires pour réaliser cette analyse, on trouve MALLET (McCallum, 2002), codé en Java, ainsi que des bibliothèques en Python, comme Gensim (Řehůřek & Sojka, 2010) , ou le module lda (Griffiths & Steyvers, 2004), qui permettent d'exécuter et de personnaliser ces algorithmes pour différents types de corpus et d'analyses. En plus du LDA, d'autres modèles avancés et des extensions permettent d'adapter le *topic modeling* à des cas spécifiques, comme la gestion des évolutions thématiques dans le temps ou l'intégration d'informations supplémentaires.

Blei (2012) donne aussi des extensions possibles du LDA. Le LDA peut être facilement modifié et réadapté pour des tâches plus complexes. Par exemple, le LDA considère que l'ordre des mots dans un document n'est pas important, ce qui n'est pas vrai. Cela peut parfois causer des problèmes en fonction de l'analyse qu'on souhaite faire. Blei explique que des extensions de la LDA ont été créées pour remédier à ce problème, notamment par Wallach (2006) ou par Griffiths et al (2005).

Le LDA considère aussi que l'ordre des documents ne compte pas, ce qui peut aussi fausser certains résultats (notamment quand on analyse des documents qui s'étendent sur plusieurs siècles). Une solution à cela est le topic model dynamique. Le LDA considère que le nombre de topics est fixe et prédéterminé. Une solution est le *Bayesian nonparametric topic model*. D'autres extensions au LDA sont le *correlated topic model* et le *pachinko allocation machine* qui permettent aux *topics* d'être corrélés ; le *spherical topic model* qui permet à des mots d'être rattachés à des *topics* auxquels ils semblent peu probables d'appartenir ; les *sparse topic models* imposent une structure supplémentaire dans les distributions de *topics* ; les « *bursty* » topic models offrent un modèle plus réaliste sur les fréquences des mots. De nombreux topic models permettant d'inclure des métadonnées (auteur, titre, origine...) ont aussi vu le jour : l'*Author-topic model* qui permet d'inclure les auteurs, les proportions des *topics* sont associées aux auteurs ; le *relational topic modeling* de Chang et Blei (2009) permet d'intégrer des liens et hyperliens. Enfin, Blei rappelle que le LDA peut aussi être adaptée pour d'autres formats de donnée. Par exemple des enquêtes, de l'audio, de la musique, du code informatique, des données de réseaux sociaux, etc.

Herrmann et al. (2022) donnent quant à eux trois mises en garde ou commentaires quant à l'utilisation du topic modeling. Tout d'abord, il faut être capable de comprendre un minimum l'algorithme et ses paramètres pour l'utiliser à bon escient. Il n'y a pas de conventions ou recommandations quant à la manière de l'utiliser pour obtenir les meilleurs résultats possibles. Enfin, Il est quasiment impossible de vérifier la validité des résultats produits par le topic modeling, car il n'y a pas forcément une bonne ou une mauvaise réponse quand on parle du sens d'un texte ou de la classification de ses genres. Il y a une part d'interprétation et de subjectivité dans ce qui est à considérer comme plus important qu'autre chose.

Pour finir, une piste d'amélioration des modèles de topic modeling réside dans la visualisation des résultats (Blei, 2012). La présentation sous la forme d'une liste de topics, eux-mêmes associés à une liste de mots n'est pas forcément optimale ou intuitive. Présenter ces résultats sous formes d'étiquettes associées à différentes parties d'un document pourrait permettre aux lecteurs de plus

facilement accéder au contenu qui l'intéresse. On pourrait aussi réfléchir à comment mettre en exergue les connexions entre les différents termes et différents documents.

### 3.3.1.2. Exemple d'application : analyse de pièces de théâtre

Un exemple de mise en pratique du topic modeling est illustré par l'étude de Schöch (2017) sur un corpus des drames de la littérature française classique et des Lumières.

Le but de l'étude est de déterminer si le topic modeling peut être efficace et utile à l'analyse de genres en littérature. La notion de genre en littérature est très complexe, car elle dépend de nombreux éléments (synopsis, thème, structure, style, etc.). De plus certains genres et sous-genres semblent parfois se superposer ou même se contredire.

L'étude cherche tout d'abord à déterminer quels thèmes ou topics ressortent du corpus. Elle cherche ensuite à analyser si des sous-genres dramatiques différents présentent différents *topics* dominants et si oui, sont-ils surprenants ou non ? Vagues ou précis ? Abstraits ou concrets ? Elle s'intéresse aussi à déterminer si des thèmes liés au synopsis ressortent des différents sous-genres dramatiques et si oui, dépendent-ils de ces sous-genres ou non. Enfin, elle cherche à vérifier si les résultats obtenus par topic modeling sont en accord avec les conventions et définitions déjà établies dans la littérature. Plus précisément, les hypothèses de départ posent les questions suivantes :

- Est-ce que les thèmes qui vont se dégager seront en équation avec ce qu'on pourrait attendre de drames datant du XVII<sup>e</sup> et XVIII<sup>e</sup> siècles (par exemple, élite contre bourgeoisie, ou amour contre devoir) ?
- Quelle sera la place de la tragicomédie ? Ressortira-t-elle comme un mélange de tragédies et comédies ou présentera-t-elle ses propres thèmes et caractéristiques ?
- Les tragédies se terminent souvent par des scènes violentes et la mort du personnage principal, alors que les comédies se terminent plutôt par des scènes d'amour joyeuses et de mariages. Ces thèmes seront-ils présents, et prendront-ils plus de place au fur et à mesure que l'on progresse dans le texte ?

En tenant compte des sous-genres préétablis en littérature, les résultats de l'étude sont les suivants.

Des nuages de mots avec les quarante mots ayant la plus grande valeur dans chaque *topic* sont créés et analysés. Les *topics* observés sont cohérents et ne sont pas vagues, même ceux avec les plus hauts scores de probabilité (qui sont souvent les *topics* les plus généraux et donc les plus vagues). Certains *topics* n'ont qu'un nombre très restreint de mots clés, cela est dû à la structure interne des *topics*, c'est-à-dire l'hyperparamètre alpha.

En s'intéressant aux *topics* les plus et les moins représentatifs et présents dans chaque sous-genre (comédie, tragédie, tragicomédie), il est possible de remarquer que la comédie et la tragédie possèdent toutes deux des *topics* prédominants distincts. La tragicomédie quant à elle semble bien être un mélange de comédie et tragédie et non un genre à part.

Pour chaque *topic* trouvé, une analyse de leur poids en fonctions des différentes parties des pièces (découpées selon les actes) de tragédie et de comédie est effectuée. Pour la plupart des *topics*, il n'y a pas de fortes variations tout le long des pièces, sauf quelques *topics* qui chutent ou augmentent en fonction de la partie de la pièce et si c'est une comédie ou une tragédie. Par exemple, les *topics* autour de la mort augmentent fortement vers la fin des tragédies.

Dans l'ensemble, les clusters observés semblent confirmer les classifications en sous-genres préétablis dans la littérature.

Etonnamment, en comparant les résultats d'une approche basée sur le topic modeling (avec une prise en compte du contexte), contre une approche basée uniquement sur les mots les plus fréquents (souvent utilisée en stylométrie, une méthode évoquée plus tard dans cet état de l'art), la seconde approche obtient de meilleurs résultats. On pourrait s'attendre à l'inverse car la seconde approche ne fait pas de tri préalable des mots, en supprimant par exemple les mots grammaticaux qui sont plus souvent le reflet du style d'un auteur plutôt que du contenu d'un texte et qui pourraient donc fausser les résultats. Cependant, ce résultat est compatible avec des études citées par Schöch, notamment celles de Rybicki et Eder (2011) et Kestemont et al. (2012), qui ont montré qu'il n'existe pas de séparation nette entre les signaux liés au style d'auteur et ceux liés au genre dans les distributions des fréquences des mots.

### **3.3.1.3. Proposition d'un prototype de visualisation des résultats du topic modeling**

On a vu que le topic modeling est utile pour déterminer les *topics* ou sujets des documents d'un corpus, cependant comme mentionné précédemment, peu d'études ont été faites pour développer une façon agréable et visuelle de présenter les résultats obtenus.

Jähnichen et al. (2017) proposent un prototype d'analyse visuelle permettant à la fois de déterminer les thèmes obtenus par le topic modeling et de les présenter visuellement. Le prototype est basé sur l'outil de visualisation OpenWalnut (OpenWalnut Development Team, 2010). Ce prototype permettra aux utilisateurs de parcourir les documents, *topics* et mots du corpus comme bon leur semble. La présentation des thèmes se fera sous la forme de nuages de mots de tailles différentes. Leur position sur le plan se base sur les similarités des *topics* considérés deux par deux.

## **3.3.2. La textométrie**

### **3.3.2.1. Présentation et outils**

La textométrie est une approche d'analyse de données textuelles. Elle se concentre sur des caractéristiques lexicales et linguistiques des textes et tente d'y repérer des structures, tendances ou motifs à l'aide de méthodes basées sur les statistiques et l'informatique.

Herrmann et al. (2022) définissent la textométrie comme une approche statistique de l'analyse textuelle ayant vu le jour en France dans les années 1970. Elle s'inspire de la lexicométrie, qui est une méthode statistique d'étude des particularités lexicales de textes littéraires et ajoute la possibilité de cartographier les mots et textes du corpus en fonction de leurs similarités. La textométrie est une approche à la fois quantitative (réduction des mots à des chiffres et probabilités) et qualitative (prise en compte du contexte).

Hermann et al. mentionnent deux outils de textométrie qu'ils considèrent comme les plus pertinents : Hyperbase et TXM. Ils expliquent que ce genre d'outils génériques peuvent être très utiles lorsque l'on possède déjà une idée claire de ce que l'on souhaite vérifier ou analyser dans un corpus de textes. En effet, l'utilisateur joue toujours une part importante dans le processus d'analyse. Ce genre d'outils textométriques n'effectuent pas le travail seuls, ils permettent seulement de calculer et mettre en évidence des résultats en suivant les instructions fournies par l'utilisateur. C'est à l'utilisateur de générer les hypothèses ainsi que les différentes étapes nécessaires à la mise en œuvre de l'analyse et à l'interprétation des résultats.

Pincemin (2020) détaille les différents outils et fonctionnalités du logiciel TXM et explique ce qu'ils permettent de faire. L'outil de *concordance* permet d'étudier des mots choisis ou prédéterminés par l'utilisateur dans leurs contextes. L'outil de *cooccurrences* permet d'étudier et déterminer quels mots sont les plus souvent susceptibles d'être utilisés ensemble. TXM offre la possibilité de diviser son corpus en sous-parties sur la base de différents critères, ainsi que la possibilité de classer les mots du corpus en groupe ou listes de mots. L'outil de *spécificités* permet de d'analyser l'usage et la fréquence de certains mots ou groupes de mots choisis ou constitués par l'utilisateur dans chacune des sous-parties créées dans le corpus. L'outil de *progression* permet d'étudier l'évolution de la présence et fréquence de mots à travers l'entièreté d'un document du corpus. L'outil *d'analyse factorielle des correspondances* permet de visualiser quels mots sont proches et quels mots sont éloignés les uns des autres, ainsi que quels mots sont les plus fréquents ou importants dans un texte. Ces résultats sont présentés sous la forme d'un graphique à nuage de points où les mots qui sont proches les uns des autres sont des mots, soit souvent utilisés ensemble (similarité sémantique), soit souvent utilisés de la même manière (similarité syntaxique). Les mots proches du centre du graphique sont les mots les plus importants du texte. Cet outil permet donc de dégager des thèmes en fonction des regroupements de mots alors observés. Enfin, l'outil de classification permet de relier selon différents niveaux et différentes distances les différents documents d'un corpus textuel en fonction de leurs similarités. Cela permet de classer et réunir les documents dans différents groupes et de dégager à nouveau des possibles genres ou thèmes.

Hermann et al. (2022) soulignent cependant quelques possibles limites aux logiciels tels que TXM. En effet, TXM possède des modules qui tokenisent et lemmatisent les corpus textuels automatiquement. La plupart des analyses pouvant être réalisées à l'aide de ce logiciel dépendent de ces étapes automatisées sur lesquelles l'utilisateur n'a pas la main. Ainsi, des erreurs dans cette automatisation peuvent venir fausser les résultats. Il faut donc rester averti et penser à vérifier cela avant de prendre pour véridiques ou de réutiliser les résultats obtenus. Cependant, il est possible d'importer des textes déjà tokenisés dans TXM, par exemple au format TreeTagger (mot-forme, POS, lemme), ce qui permet d'utiliser des tokens, POS et lemmes spécifiques au lieu de ceux générés automatiquement par TXM.

### 3.3.2.2. Exemple d'application

Un exemple d'utilisation de méthodes et outils de textométrie est exposé dans le travail de Hermann et al. (2022).

L'objectif de l'étude est d'explorer le style de poèmes d'Apollinaire d'un point de vue diachronique, c'est-à-dire dans le temps, à l'aide du logiciel TXM. L'hypothèse de départ est que les œuvres d'Apollinaire présentent une évolution de style, plus précisément une évolution syntaxique dans la structure des œuvres.

Un premier obstacle réside dans le fait que les dates de conception des poèmes réunis dans un même recueil sont parfois très hétérogènes. Il faut donc indiquer la date de chacun des poèmes d'un recueil, car dans TXM, les poèmes sont traités à part et non plus perçus comme appartenant à un recueil spécifique, ce qui est nécessaire pour obtenir une analyse non biaisée par des liens prédéfinis entre ces différents poèmes à travers le temps.

L'étude s'intéresse plus particulièrement aux configurations syntaxiques dans les poèmes d'Apollinaire et va donc axer l'analyse textuelle effectuée par TXM sur les connecteurs logiques, comme les conjonctions et pronoms relatifs. C'est là que l'importance de la lemmatisation des textes par TXM entre en jeu. L'outil de spécificités est appliqué aux connecteurs logiques repérés dans les textes, en fonction de la date de chaque poème.

Les résultats montrent une augmentation de l'utilisation de connecteurs logiques dans les œuvres de Guillaume Apollinaire au cours du temps.

### 3.3.3. La stylométrie

#### 3.3.3.1. Présentation et outils

La stylométrie peut être considérée comme une approche connexe à la textométrie, qui s'intéresse plus particulièrement au style des textes.

La stylométrie peut être perçue comme une approche connexe à la textométrie, car elle utilise les mêmes outils d'analyse statistique et computationnelle pour explorer des textes. Toutefois, alors que la textométrie se concentre sur des caractéristiques générales du texte (telles que la fréquence de certains mots ou expressions, les cooccurrences, et les schémas lexicaux dans un corpus), la stylométrie s'attache spécifiquement aux aspects individuels du style d'écriture, comme les préférences lexicales et syntaxiques d'un auteur. La stylométrie vise par exemple à identifier des signatures stylistiques pour déterminer l'auteur d'un texte, suivre des évolutions stylistiques, ou détecter le plagiat.

Elle ne se limite toutefois pas à l'attribution d'auteur : elle peut aussi s'appliquer à des groupes d'œuvres définis selon d'autres critères, comme le genre littéraire ou thématique. Dans ce cas, il ne s'agit plus de chercher ce qui différencie un auteur d'un autre, mais d'identifier ce qui, stylistiquement, caractérise un genre spécifique, en dépit de variations individuelles. En somme, si la textométrie fournit une vue d'ensemble statistique d'un corpus, la stylométrie affine cette approche en se focalisant sur les particularités du style individuel ou collectif dans l'expression écrite.

Herrmann et al. (2022) expliquent que comme son nom l'indique, la stylométrie analyse le style d'un texte en se basant sur une approche statistique du texte, en s'intéressant notamment à la fréquence des mots dans le texte. Elle repose donc principalement sur des algorithmes de comptage de mots.

Herrmann et al. soulèvent que plusieurs critiques se sont fait entendre quant à la stylométrie, en lui reprochant justement de ne pas aller plus loin qu'un simple comptage de mots. Ces critiques soulignent qu'il serait intéressant de développer des méthodes qui prendraient en compte d'autres éléments rattachés au texte, comme la perspective narrative, le lecteur, la syntaxe, etc.

Une méthode prenant en compte la syntaxe et ayant généré de bons résultats a été proposée par Hirst et Feiguina, mais n'a finalement pas obtenu beaucoup de succès (Herrmann et al., 2022). Cela est sûrement dû au fait que des précédentes méthodes similaires avaient déjà été développées et proposées mais n'avaient jamais surpassé les méthodes traditionnelles plus simples et qui restent favorisées par le plus grand nombre. Herrmann et al. font par ailleurs référence à des exemples de compétitions de stylométrie PAN qui portaient sur l'analyse de fanfictions. Lors de ces compétitions, les méthodes simples (basées sur les n-grams) avaient été bien plus performantes que les méthodes qui avaient été développées pour prendre en compte les étiquettes syntaxiques quand il s'agissait d'identifier les auteurs des différents textes. On peut se demander pourquoi ces tentatives de méthodes plus avancées échouent à faire mieux que les méthodes basiques déjà établies, et d'où viennent leurs contre-performances. Ces dernières peuvent être dues à des erreurs dans la lemmatisation qui se répercutent sur le processus entier. Une autre explication pourrait venir de l'instabilité du modèle due à sa forte complexité. De plus, les outils de traitement automatique du langage sont encore assez pauvres dès que l'on sort de l'anglais, cela explique qu'une méthode basée sur le comptage de mots reste efficace peu importe le langage, mais dès que l'on intègre un traitement

plus poussé du texte, il n'est pas toujours possible de l'appliquer efficacement à d'autres langues que l'anglais.

C'est seulement récemment que de vraies avancées ont été faites dans ce domaine. En 2019 un modèle intégrant *stemming*, distorsion de texte et étiquetage grammatical en plus du comptage de mot est développé. Le *stemming* « consiste à réduire un mot dans sa forme "racine" ». « Le but du *stemming* est de regrouper de nombreuses variantes d'un mot comme un seul et même mot. Par exemple, une fois que l'on applique un *stemming* sur "Chiens" ou "Chien", le mot résultant est le même. Cela permet notamment de réduire la taille du vocabulaire dans les approches de type sac de mots » (Fabien, 2019). Ce modèle a obtenu une précision de plus de 95% sur des textes de fanfictions lors d'une compétition PAN, avec pour objectif principal l'identification de l'auteur (Herrmann et al., 2022).

Un exemple d'outil de stylométrie est pydistinto (Du et al., 2021), une bibliothèque python développée pour l'analyse stylométrique et la comparaison textuelle entre deux corpus différents. Il analyse leurs caractéristiques linguistiques et stylistiques, en s'appuyant notamment sur des techniques statistiques. Ces caractéristiques peuvent être des mots, des expressions, ou des structures syntaxiques spécifiques qui distinguent les textes ou les styles d'auteurs. Pydistinto fonctionne spécifiquement pour des comparaisons par paires : il compare deux corpus à la fois (un corpus cible par rapport à un corpus de référence), et évalue les spécificités lexicales de chaque corpus l'un par rapport à l'autre.

L'objectif de pydistinto est d'identifier des marqueurs de style dans un corpus. Il est donc particulièrement utile pour des tâches telles que l'attribution d'auteurs, l'analyse de variations linguistiques dans le temps ou selon différents groupes sociaux, ou la comparaison entre différents genres.

Pydistinto fonctionne en mesurant les "distinctivités lexicales" : il évalue dans quelle mesure certains mots ou expressions sont caractéristiques d'un texte ou d'un groupe de textes, par opposition à d'autres.

Les résultats de l'analyse par pydistinto se présentent sous la forme d'une liste de vocabulaire discriminant pour chacun des deux corpus entrés. Pour chaque mot ou expression dans cette liste, pydistinto fournit aussi un score de distinctivité qui indique à quel point le terme concerné est spécifique à un des deux corpus par rapport à l'autre. Un score de distinctivité élevé signifie donc que le terme est plus typique et représentatif du style ou genre du corpus concerné.

Le fonctionnement de pydistinto repose sur plusieurs étapes : le prétraitement des textes, le calcul de la distinctivité, puis l'attribution des scores de stylométrie. Pydistinto commence par tokeniser le texte, c'est-à-dire qu'il segmente le texte en unités de base et élimine les mots-outils. Ensuite, l'outil utilise diverses métriques de distinctivité pour calculer la fréquence relative des mots dans un texte par rapport à d'autres textes du corpus. Ces métriques permettent de repérer les mots caractéristiques de chaque texte analysé.

Pydistinto propose plusieurs mesures de discriminativité, chacune offrant une perspective différente sur la distinctivité lexicale. Parmi celles-ci, on trouve notamment zeta\_sd0 (Zeta) qui repose sur la comparaison de la proportion de segments dans lesquels un terme apparaît dans deux groupes de textes. Après division des corpus en segments de taille fixe, la proportion de segments contenant un terme est calculée pour chaque groupe. La mesure Zeta d'un terme est ensuite obtenue par la différence entre ces proportions :

$$\zeta(t) = sp_t(G_1) - sp_t(G_2)$$

où  $sp_t(G)$  désigne la proportion de segments de  $G$  contenant le terme  $t$ . (Schöch & Schlör, 2018). Cette approche ne tient pas compte de la fréquence du terme dans le segment, seulement de sa présence ou de son absence.

Schöch & Schlör (2018) énoncent plusieurs variantes de Zeta, par exemple, zeta\_sd2 qui applique une transformation logarithmique pour stabiliser l'influence des mots très fréquents, ou encore zeta\_sr0 et zeta\_sr2 qui prennent en compte les fréquences relatives des mots dans les segments au lieu des proportions binaires.

Les expériences de Schöch & Schlör (2018) montrent aussi que zeta\_sd2 permet d'obtenir de meilleures performances dans des tâches de classification automatique, notamment avec des classifieurs SVM linéaires. Toutefois, zeta\_sd0 conserve un avantage en termes d'interprétabilité, car elle repose sur des comptages simples de segments. Ce compromis entre performance et lisibilité est particulièrement important dans le cadre d'analyses exploratoires, justifiant ainsi l'utilisation de zeta\_sd0 dans ce mémoire pour l'analyse des styles textuels des fanfictions.

### 3.3.3.2. Exemples d'application

Un exemple d'application de méthode de stylométrie est résumé par Dudar (2023). Elle mentionne les travaux de Coll Ardanuy et Spordeler sur le sujet. Elles ont réuni un corpus de 238 romans. Elles ont décrit leurs synopsis et leurs structures de manière statique – c'est-à-dire qu'elles ont résumé le roman entier, puis de manière dynamique – soit un résumé par chapitre – à l'aide de réseaux de co-occurrences. De ces réseaux, elles ont extrait des vecteurs représentant des caractéristiques pertinentes pour chaque roman. Elles ont utilisé des méthodes de clustering pour grouper ces vecteurs en fonction de leurs similarités. Elles ont ensuite étudié ces différents clusters en s'intéressant aux genres et auteurs représentés pour vérifier si le modèle obtenait des résultats attendus ou non, c'est-à-dire par exemple, si les livres de mêmes genres et mêmes auteurs se retrouvaient répartis ensemble.

Un exemple d'application de pydistinto est disponible dans le GitHub de l'outil (Du et al., 2021). Pydistinto est utilisé pour comparer deux corpus distincts : un corpus de textes « detective\_yes » (des romans de type policier, par exemple) et un corpus de textes « detective\_no » (des textes qui ne sont pas du genre policier). Le but de l'étude est d'identifier des marqueurs de style propres aux deux corpus, et plus particulièrement aux romans policiers. L'étude vise donc à révéler un vocabulaire distinctif des textes de type « detective\_yes » en opposition aux textes de type « detective\_no ». L'idée est d'observer comment certains termes se démarquent dans le genre policier et d'identifier les spécificités linguistiques associées à ce genre.

Les textes sont tout d'abord organisés dans deux corpus principaux : « detective\_yes » et « detective\_no ». Chaque corpus contient plusieurs textes représentatifs de leur genre (policier contre neutre). Pydistinto est ensuite utilisé pour calculer les scores de distinctivité des mots de chaque corpus, en utilisant la mesure zeta\_sd0. Cette mesure permet d'identifier les mots qui apparaissent dans une plus grande proportion de segments d'un corpus par rapport à l'autre, reflétant ainsi la dispersion lexicale spécifique à chaque groupe de textes. Dans ce cas, cela permet notamment de relever un vocabulaire spécifique aux narrations et intrigues policières. Les mots distinctifs identifiés permettent de dresser un profil stylistique pour chaque corpus. Dans le corpus « detective\_yes », les résultats montrent une distinctivité accrue de termes liés aux enquêtes (« mystery », « murder »), aux personnages typiques (des noms de détectives comme « Sherlock » et « Holmes »), ou des termes désignant des criminels comme « murderer » et « criminal »), et aux éléments narratifs (des lieux ou objets qu'on retrouve souvent dans ce genre comme « revolver » ou « police »). En revanche, le corpus

« no\_detective » se caractérise par un vocabulaire plus général, sans les marqueurs spécifiques au genre policier.

Cette étude met donc en lumière les différences stylistiques entre les textes « detective\_yes » et « detective\_no ». Ce type d'analyse peut être utile pour des applications plus larges en classification automatique de genre littéraire, en permettant d'identifier automatiquement les caractéristiques lexicales de chaque genre. C'est un outil qui peut donc être très utile pour identifier des vocabulaires spécifiques à différents sous-genres de fanfiction et que j'utiliserais dans mon étude.

### 3.3.4. L'apprentissage automatique

#### 3.3.4.1. Présentation et outils

L'apprentissage automatique, et plus particulièrement les algorithmes de classification, permet d'entraîner des modèles pour reconnaître des motifs récurrents dans les données et classer des documents dans des catégories définies. Un modèle de classification peut ainsi prédire le genre d'un texte en fonction de ses caractéristiques stylistiques ou lexicales. Un objectif des algorithmes d'apprentissage automatique est par exemple d'être capable d'associer chaque document d'un corpus à une classe qui le représente. Les classes, en général définies par l'utilisateur, vont être associées à un ou plusieurs documents et vont servir à les classer dans différentes catégories selon les classes choisies. Pour cette tâche, plusieurs algorithmes sont couramment utilisés, chacun ayant des modes de fonctionnement différents et plus ou moins adaptés à différentes situations, en fonction des caractéristiques des données.

Il existe deux grandes méthodes d'apprentissage automatique : l'apprentissage supervisé et l'apprentissage non supervisé. L'apprentissage supervisé repose sur des algorithmes qui s'entraînent à l'aide d'un ensemble de données pré-annotées où chaque groupe de données est associé à une étiquette ou classe. L'objectif du modèle est de généraliser les informations contenues dans ces données d'entraînement pour être capable de prédire la classe de nouvelles données non préétiquetées.

L'apprentissage non supervisé lui ne repose pas sur des étiquettes ou classes préétablies. Le modèle tente de détecter des structures et motifs dans des données qui ne sont pas préétiquetées et de les classifier par lui-même. Les algorithmes non supervisés, comme le *clustering*, regroupent par exemple les documents en *clusters* similaires, mais sans savoir si ces clusters représentent des genres spécifiques. Ce type d'apprentissage est utile pour explorer les données ou découvrir des catégories nouvelles ou cachées, mais il n'est pas aussi précis que l'apprentissage supervisé pour des tâches où les catégories sont définies à l'avance.

Il existe aussi deux types de classification pour l'apprentissage automatique : la classification à classe unique ou *single-label* et la classification multi-étiquette ou *multi-label*. Avec les modèles *single-label*, chaque texte n'est associé qu'à une seule étiquette et le modèle doit retrouver la bonne étiquette pour chacun des textes parmi toutes les étiquettes possibles.

À l'inverse, la classification *multi-label* permet à chaque document d'appartenir éventuellement à plusieurs classes en même temps. Cette approche est plus flexible et s'applique souvent dans des contextes où un exemple peut présenter plusieurs caractéristiques pertinentes.

Un panorama des principaux algorithmes d'apprentissage automatique peut être trouvé dans des manuels tels que *Python Data Science Handbook* (VanderPlas, 2016) ou *Introduction to Machine Learning with Python* (Müller & Guido, 2016), ainsi que dans des articles de synthèse tels que ceux de Osisanwo et al. (2017) et Mahesh (2019) :

Les machines à vecteurs de support ou Support Vector Machines (SVM) sont des algorithmes de classification supervisée qui visent à séparer les classes en traçant un hyperplan optimal entre elles dans un espace à plusieurs dimensions. Les SVM recherchent la "marge maximale", c'est-à-dire la plus grande distance possible entre l'hyperplan et les points de chaque classe. Cela permet de minimiser les erreurs de classification. Lorsqu'une séparation linéaire n'est pas suffisante, les SVM utilisent le *kernel trick*, une méthode qui projette les données dans un espace de dimension plus élevée où elles deviennent séparables (VanderPlas, 2016). Mahesh (2019) décrit les SVM comme un algorithme très puissant pour la classification de texte, en particulier pour des problèmes de classification binaire, car il est très performant pour distinguer des catégories opposées. Les SVM sont également flexibles grâce aux noyaux (kernels) comme le noyau linéaire, le noyau polynomial ou le noyau gaussien (RBF), qui permettent de traiter des relations complexes entre les données. Cependant, d'après Osisanwo et al. (2017), les SVM peuvent devenir coûteux en calcul et en mémoire pour les très grands ensembles de données et pour les tâches multi-classes, car ils ont besoin de multiples étapes d'entraînement pour chaque classe.

La régression logistique est un modèle de classification qui utilise une fonction sigmoïde pour modéliser la probabilité qu'un document appartienne à une classe donnée. Contrairement à une régression linéaire classique, qui produit des valeurs continues, la régression logistique génère une probabilité de classification entre 0 et 1 qui permet de prédire la classe la plus probable pour chaque document. L'algorithme ajuste les poids des caractéristiques pour maximiser l'exactitude des prédictions, en fonction des données d'entraînement. D'après Mahesh (2019) et Osisanwo et al. (2017), la régression logistique est simple et facilement interprétable, ce qui en fait un bon choix pour les applications où l'interprétation des coefficients est cruciale. Chaque coefficient reflète l'influence d'un mot ou d'une caractéristique spécifique sur la probabilité d'appartenance à une classe. Ce modèle est aussi efficace pour les problèmes de classification multi-classe. Bien qu'il soit généralement moins performant que des modèles plus sophistiqués pour des classifications complexes, la régression logistique est rapide, robuste, et fonctionne bien avec des données de texte modérément volumineuses.

L'algorithme Naïve Bayes est un modèle probabiliste basé sur le théorème de Bayes, qui permet de calculer la probabilité qu'un document appartienne à une classe donnée en se basant sur la probabilité conditionnelle des caractéristiques observées dans les données d'entraînement. Ce modèle repose sur une hypothèse d'indépendance des caractéristiques : chaque mot d'un texte est supposé indépendamment lié à la classe, ce qui signifie que la présence d'un mot particulier n'affecte pas la présence d'un autre mot. Bien que cette hypothèse soit rarement vraie en pratique, les résultats de l'algorithme sont étonnamment bons pour la classification de texte. D'après Mahesh (2019), l'un des avantages du Naïve Bayes est sa rapidité de calcul et sa simplicité d'implémentation. Il est particulièrement adapté aux problèmes de classification de texte et est souvent utilisé dans des contextes tels que l'analyse de sentiments et la classification de documents. Ce modèle est robuste même avec des données de petite taille, car il a besoin de peu d'exemples pour estimer les probabilités de chaque classe. Cependant, son hypothèse d'indépendance peut parfois limiter la précision de la classification dans des tâches complexes où les mots ou caractéristiques sont fortement liées entre elles.

Les arbres de décision sont des algorithmes de classification qui fonctionnent en divisant les données en sous-ensembles basés sur des règles de décision successives. Chaque noeud de l'arbre représente une question ou un test sur une caractéristique des données, tandis que chaque branche correspond à un résultat possible de ce test (VanderPlas, 2016). À chaque étape, l'algorithme choisit la caractéristique qui divise le mieux les données selon un critère, comme l'entropie, et cherche à créer des sous-groupes homogènes. Selon Osisanwo et al. (2017) et Mahesh (2019), l'intérêt principal des

arbres de décision réside dans leur interprétabilité : l’arborescence des décisions permet de visualiser clairement comment un modèle arrive à une prédiction, chaque étape de la classification étant transparente et facilement compréhensible. De plus, les arbres de décision sont adaptés à des données mixtes, où des caractéristiques qualitatives et quantitatives peuvent coexister. Cependant, ces modèles peuvent facilement surapprendre sur les données d’entraînement, ce qu’on appelle le surapprentissage ou *overfitting*, en se spécialisant excessivement sur les données d’entraînement, ce qui les rend moins robustes pour des données nouvelles. Les arbres de décision sont souvent utilisés en ensemble, dans des modèles comme les forêts d’arbres de décisions ou Random Forests, qui créent plusieurs arbres à partir de sous-ensembles aléatoires des données pour améliorer la précision et la généralisation. Cette méthode permet de réduire la variance et d’accroître la performance en combinant les prédictions de multiples arbres indépendants, offrant ainsi un modèle plus stable et résistant au surapprentissage.

Les forêts d’arbres décisionnels combinent plusieurs arbres de décision pour améliorer la précision de la classification et réduire le surapprentissage. En créant des arbres de décision multiples basés sur des sous-échantillons aléatoires des données d’entraînement, chaque arbre détermine la classe d’un document, et la classe finale est celle déterminée par le plus grand nombre d’arbres. Cette approche augmente la robustesse des prédictions, car elle réduit l’influence des erreurs potentielles d’un arbre unique. Osisanwo et al. (2017) et Mahesh (2019) soulignent que les forêts aléatoires sont efficaces pour la classification de texte, car elles capturent les relations non linéaires et les interactions entre les mots ou caractéristiques des documents. De plus, les forêts aléatoires offrent une mesure de l’importance des caractéristiques, permettant d’identifier les mots les plus significatifs pour chaque classe. Bien que le modèle soit plus coûteux en calcul que les arbres de décision simples, il est généralement plus précis et moins susceptible de surapprentissage grâce à la diversité des arbres qu’il combine.

L’algorithme des  $k$  plus proches voisins ou *k-Nearest Neighbors* (*k*-NN) est une méthode de classification qui fonctionne en trouvant les  $k$  exemples les plus proches dans les données d’entraînement par rapport à un document donné, selon une mesure de distance, comme la distance euclidienne par exemple (Müller & Guido, 2016). Une fois les  $k$  voisins les plus proches identifiés, le document est classé dans la catégorie majoritaire de ses voisins. D’après Osisanwo et al. (2017) et Mahesh (2019), *k*-NN est un algorithme simple et intuitif, particulièrement efficace pour des données où les catégories forment des groupes bien séparés dans l’espace des caractéristiques. Dans le contexte de la classification de texte, *k*-NN peut être utile lorsque chaque genre littéraire présente des mots ou des expressions caractéristiques qui le différencient des autres. Cependant, *k*-NN a certaines limitations : il est sensible aux données redondantes ou bruitées et peut devenir lent et coûteux en termes de calcul pour les grands ensembles de données, car il nécessite de comparer chaque nouveau document avec tous les exemples d’entraînement. Pour pallier cette limitation, on peut utiliser des représentations vectorielles denses, comme les plongements lexicaux, qui permettent de mieux capturer la similarité sémantique entre textes, tout en étant moins sensibles à la dimensionnalité élevée d’approches telles que les sacs de mots. Par ailleurs, des techniques spécifiques de réduction de dimensionnalité, comme l’analyse en composantes principales (PCA), peuvent également être appliquées sur des représentations vectorielles pour accélérer les calculs.

Ces différents algorithmes peuvent être appliqués à l’aide de la bibliothèque Python scikit-learn (Pedregosa et al., 2010/2011).

Dans le cadre de la classification textuelle, ces algorithmes d’apprentissage automatique s’appuient sur un ensemble de caractéristiques ou *features* extraites des données pour identifier des motifs propres à chaque classe. Ces caractéristiques sont souvent des représentations numériques des

contenus textuels. Elles permettent aux algorithmes de détecter les similarités et différences entre documents. Le choix des caractéristiques est essentiel, car il influence directement les performances des modèles de classification. Brigadoi (2021) donne une liste de différents types de ces caractéristiques :

- Sac de mots (*Bag-of-words* ou BoW) qui permettent notamment d'analyser les mots les plus fréquents d'un texte, sans tenir compte de leur ordre.
- Bigrammes (*bigrams*) qui permettent notamment d'analyser les occurrences de couples de mots consécutifs dans un texte.
- N-grammes (*n-grams*) syntaxiques qui permettent notamment d'analyser les mots consécutifs dans un texte en tenant compte de l'ordre syntaxique d'une phrase.
- Lexique d'émotion (*Emotional Features*) qui permettent notamment d'associer certains mots à certains sentiments pour déterminer le ton émotionnel d'un texte.

La construction d'un modèle d'apprentissage automatique de classification suit un processus en plusieurs étapes. Tout d'abord, il faut collecter les données ; ces données doivent ensuite être labélisées, c'est-à-dire que chaque document se voit attribuer une (ou plusieurs) classes. Une fois les données collectées et annotées, elles doivent être converties en vecteurs de mots ou autre représentation numérique. Ces vecteurs peuvent être créés à l'aide de méthodes simples comme les sacs de mots, qui représentent les données selon la fréquence des mots, ou des méthodes plus complexes comme les plongements lexicaux ou *embeddings* qui capturent les relations sémantiques entre les mots. Durant cette étape, un tri des caractéristiques est souvent réalisé pour ne garder que les plus significatives. Brigadoi mentionne trois techniques courantes pour cette sélection : la méthode *wrapper*, qui teste différentes combinaisons de caractéristiques pour trouver les meilleures ; la méthode *filtering*, qui applique des critères statistiques indépendamment de l'algorithme pour filtrer les caractéristiques ; la méthode *embedded*, qui sélectionne les caractéristiques en parallèle avec l'entraînement de l'algorithme. Après la sélection des caractéristiques, l'étape suivante est l'entraînement du modèle à l'aide des données d'entraînement. Enfin, la dernière étape consiste à évaluer le modèle sur des données d'évaluation pour tester sa robustesse et précision.

Enfin, il convient de mentionner qu'il existe aujourd'hui des algorithmes de classification plus récents et puissants, comme les réseaux de neurones profonds ou les modèles de type transformers, tels que BERT (mentionné notamment par Grootendorst (2022)) qui sont désormais largement utilisés dans les tâches de TAL. Ces algorithmes sont capables d'atteindre des niveaux de performance impressionnantes grâce à leur capacité à capturer des relations complexes dans les données textuelles. Cependant, ces modèles fonctionnent souvent comme des "boîtes noires", c'est-à-dire que leurs processus internes sont difficilement interprétables, rendant complexe l'identification des caractéristiques textuelles spécifiques utilisées pour leurs prédictions. Or, dans le cadre de cette étude, l'objectif principal n'est pas seulement de classifier les fanfictions par sous-genre, mais également de comprendre les caractéristiques qui définissent ces sous-genres. C'est pourquoi les algorithmes plus simples mais plus interprétables sont privilégiés, dans le but de fournir des résultats exploitables pour une analyse qualitative des tags.

### 3.3.4.2. Exemples d'application

Brigadoi (2021) a utilisé un corpus de 1003 livres composés de huit genres différents pour analyser quels algorithmes couplés avec quelles caractéristiques sont les plus efficaces dans la classification en différents genres.

Sa manière de procéder est la suivante. Il a d'abord collecté le corpus de 1003 livres d'intérêt. Ensuite, il a prétraité les données de façon à optimiser leur utilisation par les algorithmes. Il a pour cela

mis tout le texte en minuscules, a supprimé certaines ponctuations non pertinentes, a tokenisé le texte et a supprimé les mots non porteurs de sens. La tokenisation a été effectuée à l'aide de l'outil Stanza. La suppression de la ponctuation inutile et des mots non porteurs de sens (*stopwords*) a été effectuée à l'aide de la librairie Python NLTK. Après cela, les livres du corpus ont été convertis en vecteurs. La valeur des vecteurs représente la fréquence des caractéristiques de chaque livre. Les différents types de caractéristiques utilisés pour cette étude sont : sac de mots, bigrammes, n-grammes, lexique d'émotion.

Les résultats obtenus montrent que les meilleures performances sont obtenues par les algorithmes de forêts d'arbres décisionnels et de machines à vecteurs de supports. Le meilleur résultat est obtenu en combinant un algorithme de forêts d'arbres décisionnels à une approche basée sur les types de caractéristiques *bag-of-words* et bigrammes. C'est donc ce modèle qui est retenu dans la suite de l'étude qui consiste à l'appliquer au corpus de livres pour les classer selon différents genres.

Au final, le modèle retenu parvient à bien classer les textes de type Poésie et les textes de type Dramatique, il est cependant moins performant quand il s'agit de texte de type Fiction ou de type Romance. Cependant, Brigadoi note que la confusion semble venir des textes de type Nouvelle qui sont souvent assignés à une mauvaise classe, car ces textes peuvent contenir des caractéristiques d'autres genres. En effet, la nouvelle est un genre littéraire défini principalement par des critères formels plutôt que par le contenu du texte. Une raison derrière les mauvais résultats obtenus peut donc venir du fait que le corpus n'était pas optimal et que le genre de type Nouvelle a faussé ces résultats.

Un autre exemple est mentionné par Dudar (2023) qui parle des travaux de Hettinger et al. qui ont testé différents algorithmes d'apprentissage et différents types de caractéristiques. Ils ont cherché à évaluer comment les différents types de caractéristiques peuvent affecter les performances de classification à l'aide d'un corpus de romans allemands appartenant aux genres social et d'apprentissage, corpus qu'ils ont par la suite élargi avec des romans d'aventure.

Leur conclusion explique que les meilleurs résultats sont obtenus en combinant une approche basée sur les caractéristiques liées aux thèmes (*topic based features*) – qu'ils ont extrait à l'aide d'un algorithme LDA – et un algorithme SVM.

Enfin, Calvo Tello (2021) a essayé de classer différents textes espagnols appartenant à plusieurs sous-genres de romans à l'aide d'algorithmes de régression logistique. Les résultats n'étaient pas concluants. Pour lui, cela était dû au fait qu'une œuvre ne se réduit pas à un seul genre, mais constitue souvent un mélange de plusieurs genres. Il a donc pensé à appliquer un système de classification multi-classes. Cette méthode permettait d'associer chaque roman à plusieurs étiquettes de sous-genre, plutôt que de le limiter à une seule catégorie. Son approche repose sur l'idée que de nombreux textes littéraires manifestent des caractéristiques de plusieurs genres, ce qui complique les méthodes traditionnelles de catégorisation en genres uniques.

Il a utilisé la régression logistique comme algorithme principal en raison de son efficacité à gérer des caractéristiques complexes et chevauchantes entre les genres. Pour entraîner le modèle, Calvo Tello s'est appuyé sur diverses représentations des caractéristiques, incluant des caractéristiques linguistiques (telles que des motifs lexicaux et syntaxiques). Il a testé plusieurs transformations, comme les scores z et les TF-IDF, afin d'optimiser les résultats. Le modèle a obtenu un score F1 médian de 0,84 pour tous les sous-genres, et ce score a même atteint 0,86 en combinant des caractéristiques linguistiques et littéraires. Ces résultats montrent que la classification des catégories littéraires peut être réalisée avec une qualité de résultats raisonnable lorsque les genres sont traités dans un cadre multi-étiquette plutôt qu'avec une seule étiquette rigide.

En comparaison avec les méthodes de classification à classe unique, qui atteignaient des scores F1 autour de 0,57, Calvo Tello remarque que l'approche multi-étiquette saisit mieux les caractéristiques croisées des sous-genres. Elle permet également une application plus flexible des étiquettes de genre, avec la possibilité pour certaines œuvres d'être classées dans des combinaisons de catégories plutôt que d'être assignées à une seule. L'étude de Calvo Tello montre ainsi l'intérêt de la classification multi-étiquette pour l'analyse littéraire, car elle correspond à la nature hybride des genres et permet une exploration plus en profondeur des multiples influences qui peuvent coexister dans un même texte.

### **3.4. Précédentes applications du TAL dans l'analyse de fanfiction**

#### **3.4.1. Étude sur les paratextes de fanfiction**

Black (2020) concentre son étude sur différents paratextes de fanfictions. Les paratextes sont tous les éléments que l'on trouve autour d'un texte, tel que le titre, le résumé, l'auteur, et dans le cas des fanfictions, les tags de sous-genre, les notes d'auteurs, les commentaires, etc. Black souhaite analyser différents paratextes de fanfictions, à savoir le titre, le résumé, les notes de l'auteur, ainsi que le texte lui-même, à l'aide de plusieurs outils de TAL que sont le topic modeling, l'apprentissage automatique et plus précisément des algorithmes d'arbres de décision, et la textométrie.

Pour réaliser cette étude, elle a premièrement constitué un corpus de fanfictions. Afin de pouvoir mieux constater et contraster les différences dans les paratextes son corpus doit être composé de textes assez semblables. Pour cela elle a choisi de ne collecter que des fanfictions venant d'une même fandom, formée autour du *ship* M/M Captain America/Bucky, issue de l'univers Marvel. Pour collecter les données qui l'intéressent elle a eu recours à l'utilisation d'un web scraper basé sur le code Python de Li & Sterman (2016), qu'elle a modifié afin de récupérer le texte des fanfictions et tous ses paratextes en excluant les commentaires et les *kudos*, c'est-à-dire les réactions laissées par les fans pour exprimer leur satisfaction ou encouragement (l'équivalent d'un *like* sur d'autres réseaux sociaux).

Une fois le corpus collecté, Black a tout d'abord décidé d'analyser les titres de ces fanfictions à l'aide de méthodes de textométrie. En général, un titre d'œuvre, et plus précisément d'un livre ou d'un quelconque texte de fiction, est unique à cette dernière, c'est ce qui en fait une œuvre à part entière et ce qui permet de la séparer et différencier des autres. Il possède aussi généralement une fonction interprétative, car il représente la première chose que les lecteurs voient, il doit donc immédiatement leur donner des indices quant au contenu du texte. Black souhaite vérifier si ces affirmations restent vraies pour les titres de fanfictions. Pour cela elle récupère une liste de titre de livres qu'elle va comparer au titre des fanfictions de son corpus à l'aide de l'outil de textométrie AntConc. Cet outil lui permet de récupérer les mots les plus fréquents dans ces différents titres et d'analyser leurs différences. Les résultats qu'elle observe montrent que les fanfictions utilisent plus de mots grammaticaux, alors que les livres utilisent plus de mots lexicaux, qui permettent de définir le contenu de l'histoire. Les titres de fanfictions manquent donc de mots descriptifs et ne permettent pas vraiment de distinguer une fanfiction d'une autre. Quelle est la raison de cette différence ? Pourquoi les fanfictions contiennent peu de mots lexicaux ? Cela s'explique en partie en raison du fait que, sur les sites de fanfictions, les lecteurs peuvent affiner le contenu de la fanfiction (thème, personnages, type de relation entre les personnages, lieu, époque, etc.) en fonction de leur choix à l'aide des tags de sous-genres. Ces informations n'ont donc pas besoin d'être précisées dans les titres des fanfictions qui importent finalement peu aux lecteurs dans ce cas. Ces résultats confirment donc l'importance des tags de sous-genres dans la classification des fanfictions sur les différents sites de fanfictions.

Dans un deuxième temps, Black a choisi d'appliquer des méthodes de topic modeling sur les textes, résumés et notes de l'auteur des fanfictions de son corpus. Comme le principe du topic modeling est d'extraire le sens d'un texte, les mots non porteurs de sens du corpus peuvent être supprimés avant l'analyse. Black a donc décidé de ne garder que les verbes, adverbes, adjectifs et noms dans son corpus. Elle a aussi supprimé les noms des personnages à l'aide d'une *stop-word-list* des noms de personnages qu'elle a récupéré dans les données du site AO3. Elle a ensuite utilisé un algorithme LDA tel qu'implémenté dans la bibliothèque MALLET (McCallum, 2002) pour générer 25 *topics* à partir de son corpus. En observant les résultats obtenus par le topic modeling, il est possible de conclure que les textes contiennent l'histoire, car l'algorithme en extrait des *topics* faisant référence à des sujets de fiction, narration. Quant aux notes d'auteur et résumés, les résultats montrent qu'ils contiennent une trace de l'évolution et contexte de la création de la fanfiction ainsi que certaines informations sur le texte. Cela montre bien l'interconnexion qu'il y a entre les fans auteurs, lecteurs et leurs œuvres sur les sites de fanfictions.

Enfin, Black applique des algorithmes d'arbres de décisions sur les textes, notes d'auteur et résumés du corpus qui sont aussi les trois classes qu'elle définit pour l'algorithme. Les résultats montrent que l'algorithme est capable de déterminer si le document est un texte, un résumé ou une note d'auteur à plus de 92 %.

En résumé, l'étude de Black constitue une référence utile sur l'analyse computationnelle des paratextes en fanfiction, en montrant comment les titres, résumés et notes d'auteurs diffèrent structurellement et fonctionnellement du texte principal, tout en confirmant l'importance des *tags* pour décrire les œuvres de fanfiction. Son choix d'un corpus homogène (*fandom* spécifique au couple Captain America/Bucky) permet une analyse plus fine, mais limite aussi la généralisation à d'autres *fandoms* et fanfictions.

Dans ce mémoire aucun pré-tri n'est effectué sur les fanfictions collectées pour les tags à étudier, ce qui permet de tirer des conclusions plus générales. De plus ce mémoire se distingue en se concentrant sur l'impact linguistique et stylistique que l'utilisation d'un tag implique dans le texte d'une fanfiction, alors que Black s'intéresse avant tout à la fonction des paratextes et à leur rôle dans la navigation et la classification des œuvres.

### **3.4.2. Analyse du succès et de l'impact des fanfictions auprès des fans**

#### **3.4.2.1. Prédire le succès d'un personnage à l'aide d'un modèle d'apprentissage automatique**

Milli & Bamman (2016) souhaitent aussi entraîner un modèle d'apprentissage automatique capable de prédire le succès et les réactions des fans à un personnage dans une fanfiction en se basant uniquement sur le texte de la fanfiction. De manière similaire, ce mémoire utilise également les textes des fanfictions pour prédire des éléments spécifiques, non pas les réactions des fans à un personnage, mais les tags associés aux œuvres. Les deux approches partagent donc une méthodologie commune, bien que les objectifs diffèrent : Milli & Bamman se concentrent sur les personnages, tandis que ce mémoire explore les caractéristiques linguistiques et stylistiques des tags.

Après avoir collecté le corpus de fanfictions qui servira à entraîner le modèle, la première étape de leur étude consiste à sélectionner les commentaires laissés sur ces fanfictions qui parlent d'un personnage et à analyser ces données d'un point de vue émotionnel, afin de connaître quels seront les résultats attendus. Pour cela, Milli & Bamman (2016) ont demandé à des annotateurs de noter le sentiment exprimé dans chacun des commentaires sélectionnés. Ils remarquent que certains commentaires semblent moins explicites ou plus ambigus que d'autres, ce qui crée une marge d'erreur

dans les annotations. Milli & Bamman (2016) choisissent donc de ne garder que les commentaires qui ont été annotés unanimement de la même manière.

Ils entraînent ensuite leur modèle qui ne se basera que sur le texte et non les commentaires pour analyser les probables réactions des fans face au personnage concerné en se basant sur des algorithmes de régression logistique. Ils utilisent différentes caractéristiques pour représenter les personnages :

- S'il est agent, patient, prédictor ou possessif
- Ses paroles (sous forme d'unigrams)
- Son genre
- Sa fréquence d'apparition
- Skip-gram (prédiction des mots du contexte selon un mot donné)

Les résultats obtenus sont non satisfaisants. Milli & Bamman en concluent que la représentation qu'on se fait d'un personnage dépasse les caractéristiques syntaxiques que le texte lui attribue. Cette conclusion contraste avec les résultats obtenus dans ce mémoire qui, bien qu'ils ne soient pas optimaux, montrent que les caractéristiques linguistiques et stylistiques des textes permettent, dans une certaine mesure, de les distinguer de manière automatique et d'identifier des spécificités pour chaque tag. Ainsi, contrairement aux perceptions très subjectives des personnages, les tags semblent reposer sur une certaine logique de classification plus objective et utilitaire, servant à structurer et organiser les fanfictions pour les auteurs et les lecteurs. Cependant, les limites observées dans les deux études soulignent les défis communs liés à l'analyse des fanfictions : les nuances émotionnelles et thématiques complexes restent difficiles à capturer pleinement dans ces œuvres assez peu codifiées.

### **3.4.2.2. Rapport entre style et succès d'une fanfiction**

Mattei et al. (2020) souhaitent observer si le style d'écriture d'une fanfiction a un impact sur le possible succès d'une fanfiction. Pour cela, ils appliquent des méthodes de stylométrie sur un corpus italien de 55 000 fanfictions basées sur l'univers d'Harry Potter. Ce corpus est collecté par *web-scraping* à l'aide du framework Python Scrapy.

Le corpus collecté est tokenisé et annoté selon les étiquettes disponibles sur Universal Dependencies. La méthode de stylométrie utilisée se base sur le *linguistic profiling*, une approche qui extrait des ensembles de caractéristiques linguistiques d'un texte qui servent à créer une représentation vectorielle de ce dernier. Le corpus est aussi séparé en « fanfictions à succès » et « fanfictions sans succès ».

L'étude s'appuie sur un large éventail de caractéristiques linguistiques couvrant plusieurs niveaux d'annotation. Elles incluent :

- Longueur du texte : Nombre total de mots et de phrases, longueur moyenne des phrases et des mots.
- Richesse lexicale : Variété des mots utilisés, fréquence des mots du vocabulaire de base italien.
- Informations morpho-syntaxiques : Distribution des catégories grammaticales (noms, verbes, adjectifs, etc.) et des formes verbales (temps, mode, personne).
- Structure des prédictors verbaux : Complexité des phrases, nombre de dépendances syntaxiques, longueur des chaînes de dépendance.
- Relations syntaxiques : Types de relations entre les mots dans les phrases.
- Utilisation de la subordination : Fréquence des propositions subordonnées et leur longueur.

Pour chaque caractéristique, les auteurs ont calculé la moyenne et l'écart-type dans les deux groupes. Ils ont ensuite utilisé le test de Wilcoxon pour déterminer si les variations entre les moyennes étaient significatives. Ils ont trouvé que 57 % des caractéristiques (126 sur 219) étaient distribuées de manière significativement différente entre les fanfictions à succès et celles sans succès.

Les résultats de l'analyse montrent que les caractéristiques les plus prédictives du succès ou de l'absence de succès des fanfictions sont les suivantes :

- Plus une fanfiction est longue plus elle a du succès ;
- Les fanfictions avec des phrases courtes ont plus de succès que les fanfictions avec des phrases longues ;
- Les fanfictions qui contiennent une majorité de verbes conjugués à la deuxième personne, c'est-à-dire du discours direct, ont plus de succès ;
- Les fanfictions qui contiennent une majorité de verbes conjugués à la troisième personne, c'est-à-dire du discours indirect, ont moins de succès ;
- Les fanfictions qui contiennent de longues dépendances syntaxiques, c'est-à-dire des phrases complexes, ont moins de succès ;
- Les fanfictions qui contiennent des phrases simples ont plus de succès.

Mattei et al. notent que les deux derniers points contredisent les conclusions que Ganjigunte et al. (2013) avaient tiré en étudiant des œuvres de littérature « classique ».

Contrairement à Mattei et al., ce mémoire ne se concentre pas sur le succès des fanfictions, mais sur l'analyse des caractéristiques linguistiques et stylistiques des tags. Cependant, les résultats de Mattei et al. montrent que certaines caractéristiques linguistiques, comme la longueur des phrases et la complexité syntaxique, peuvent influencer la perception des lecteurs. Ces éléments pourraient également jouer un rôle dans la manière dont les tags sont perçus et utilisés par les auteurs et les lecteurs. Par exemple, des fanfictions taggées *Angst* pourraient contenir des phrases plus longues et complexes, reflétant la tension et la profondeur émotionnelle du récit, tandis que des fanfictions taggées *Fluff* pourraient contenir des phrases plus courtes et simples, reflétant un ton plus léger et réconfortant.

Les observations de cette étude renforcent aussi l'hypothèse selon laquelle les caractéristiques linguistiques peuvent traduire des éléments émotionnels ou narratifs implicites, ce que ce mémoire explore à travers une classification par tags, qui servent d'indicateurs collectifs d'ambiance, de dynamique ou de ton.

Enfin, bien que cette étude offre des informations intéressantes sur les caractéristiques linguistiques associées au succès des fanfictions, elle présente certaines limites, notamment le fait que le corpus est limité à des fanfictions italiennes basées sur l'univers d'Harry Potter, ce qui peut ne pas être représentatif de l'ensemble des fanfictions.

### **3.4.2.3. Impact des fanfictions sur les compétences de rédaction et d'analyse littéraire des fans**

Rouse (2021) note que peu d'études ont été effectuées sur le lien entre l'éducation littéraire et rédactionnelle qu'on acquiert à l'école et les compétences que l'on mobilise dans l'écriture d'une fanfiction. Elle décide donc de réaliser une étude sur l'impact que peuvent avoir les fanfictions sur les compétences littéraires des fans lecteurs et auteurs à l'aide d'une analyse computationnelle des commentaires laissés sur des sites de fanfictions. Elle réalise aussi l'interview d'une fan autrice.

Pour l'analyse computationnelle, elle utilise des méthodes de textométrie sous la forme de nuage de mots qui permettent d'obtenir un aperçu visuel des groupes de mots les plus fréquents dans

les textes des commentaires. Puis, elle applique des outils de topic modeling qui offrent une représentation des commentaires sous la forme des sujets qu'ils traitent.

Les résultats des nuages de mots montrent que les lecteurs établissent beaucoup de comparaisons entre les éléments de la fanfiction et des éléments intérieurs ou extérieurs à cette dernière ou à la *fandom* concernée et usent de nombreuses références. Autrement dit, les lecteurs mobilisent leurs propres expériences personnelles avec différentes *fandoms* ou avec la société dans leur lecture et dans leurs commentaires. Cela confirme que de nombreuses connexions s'établissent entre et au sein des *fandoms*. De plus, ces méthodes de comparaison sont aussi quelque chose qui est abordé dans les cours de littérature.

Les résultats du topic modeling reflètent les connexions intertextuelles et méta textuelles que les fans font quand ils lisent une fanfiction et démontrent en même temps les compétences littéraires d'analyse qu'ils déploient lors de leur lecture.

Rouse conclue de son étude que les fans appliquent des stratégies d'analyse littéraire quand ils lisent pour le plaisir. Les interactions entre les lecteurs de fanfictions possèdent un potentiel éducatif et les commentaires qu'ils laissent montrent comment les communautés de fans peuvent servir d'espaces au développement des compétences en analyse littéraire. Ce papier montre que les auteurs et lecteurs de fanfictions usent des codes littéraires appris en cours lorsqu'ils écrivent ou lisent des fanfictions. L'interview de la fan autrice a de son côté mis en avant l'épanouissement personnel que peut apporter l'appartenance à une *fandom* et les échanges que cela engendre, dont la lecture et l'écriture de fanfiction font partie. Ainsi, les fanfictions, qui sont parfois décriées par certains, sont en fait un véritable moteur d'épanouissement à la fois personnel et académique pour les fans.

Ainsi, comme le souligne Rouse, les communautés de fanfiction sont des espaces où les fans développent et exercent des compétences littéraires avancées, notamment l'analyse, la comparaison intertextuelle et la réflexion critique sur les œuvres. Cette créativité et cette expertise se manifestent non seulement dans les textes et les commentaires, mais aussi dans la création et l'usage des tags. Les tags, loin d'être de simples outils de classement, sont le fruit de l'imagination collective : ils synthétisent des motifs narratifs, des dynamiques relationnelles et des émotions, et témoignent de la capacité des fans à inventer de nouveaux sous-genres et à renouveler sans cesse les conventions littéraires. L'analyse des tags permet ainsi d'observer, à travers le prisme du TAL, cette créativité et compétence littéraire des fans et auteurs de fanfictions.

### **3.4.3. Déceler les différences avec les œuvres originales**

#### **3.4.3.1. Reconnaissance de personnages et leurs mentions dans le texte**

Yoder et al. (2021) ont développé une pipeline d'apprentissage automatique pour l'analyse des personnages d'une fanfiction. Cette pipeline a trois tâches principales à réaliser : résolution de la corréférence des personnages ; attribution de dialogue ; extraction d'affirmations. Concrètement, elle extrait tous les personnages mentionnés dans un texte et leur attribue chaque mention qui les concerne, ce qu'ils disent et ce qu'ils font. C'est la première pipeline de ce genre conçue spécialement pour les fanfictions.

Pour la résolution de corréférences, la pipeline se base sur le modèle SpanBERT (Joshi et al., 2020), affiné sur LitBank (décrit dans *An Annotated Dataset of Coreference in English Literature* (Bamman et al., 2020)), un dataset contenant des annotations de corréférence pour des œuvres littéraires en anglais, un domaine plus similaire à celui des fanfictions. Tous les mots qui ne sont pas utilisés pour mentionner un seul personnage, tels que les pronoms pluriels, pronoms démonstratifs, syntagmes nominaux, etc. sont supprimés de l'analyse.

Pour l'attribution des dialogues, ces derniers sont extraits en récupérant les phrases entre guillemets – cela ne marcherait pas dans tous les cas et toutes les langues, les dialogues français sous formes de tirets ne seraient pas compatibles avec cette méthode par exemple. L'attribution de la phrase à un personnage se fait à l'aide de l'approche déterministe de Muzny et al. et de l'approche de Sims & Bamman.

Pour la résolution des coréférences de pronoms, la pipeline s'inspire de spanBERT. Elle améliore cette approche à l'aide une étape supplémentaire qui récupère les résultats de l'attribution de dialogue pour résoudre l'attribution des pronoms personnels singuliers de première et deuxième personne au sein des dialogues.

Pour l'attribution des affirmations, Yoder et al. utilisent TextTiling (Hearst, 1997) pour segmenter le texte. Chaque segment (sans dialogue) est attribué à tous les personnages qui y sont mentionnés.

Cette pipeline servira par la suite pour déceler des différences entre les fanfictions et les œuvres originales, et plus particulièrement des différences dans les relations entre différents personnages.

### 3.4.3.2. Différences dans les couples de personnages

Yoder et al. (2021) ont ensuite développé un modèle d'apprentissage automatique pour pouvoir déterminer les différences dans la caractérisation des personnages et leurs relations dans les fanfictions et les œuvres originales. Concrètement, un but de cet outil est de reconnaître quand la relation entre deux personnages diffère dans une fanfiction et dans l'œuvre originale, plus précisément d'identifier quand une relation passe de platonique à romantique ou inversement.

Le modèle doit être capable d'identifier trois éléments associés à trois classes différentes :

- La classe « Canon » (la classe principale) : le modèle renvoie *True* si la relation entre les personnages reste la même entre la fanfiction et l'œuvre originale, *False* sinon
- La classe « Romantic » : le modèle renvoie *True* si la relation entre les personnages est romantique, *False* sinon
- La classe « M/M » : le modèle renvoie *True* si les deux personnages étudiés sont tous les deux de genre masculin, *False* sinon.

Pour chaque paire de personnages, les segments de textes qui leur correspondent sont extrait à l'aide de la pipeline développée précédemment. L'analyse de la relation entre les personnages se fait à l'aide de méthodes de *text embedding*, qui consistent à créer des représentations vectorielles (ou plongements) du texte. Ces plongements tentent de saisir le sens d'un texte en transformant les mots en des vecteurs de nombres réels. Dans ce contexte, le texte est modélisé comme des plongements au niveau des mots, en utilisant des approches telles que TF-IDF et des *embeddings* pré-entraînés comme FastText.

Chaque instance de prédiction est une représentation globale d'une paire spécifique de personnages dans une histoire. Les représentations *d'embeddings* pour les paires de personnages sont construites à partir de segments de texte extraits, qui incluent des assertions, des citations ou l'ensemble de l'histoire. Ces *embeddings* sont ensuite utilisés comme entrées pour un modèle de régression logistique. L'objectif est de tester l'efficacité de ces *embeddings*, plutôt que de se concentrer sur la complexité du modèle.

Pour créer ces *embeddings*, Yoder et al. utilisent TF-IDF, pour pondérer les mots en fonction de leur importance dans le texte. Ils testent différentes tailles de fenêtres contextuelles autour des

mentions des personnages et choisissent une fenêtre de 10 mots avant et après les noms des personnages, car elle donne les meilleurs résultats.

Les *embeddings* des paires de personnages dans l'œuvre originale sont également créés de la même manière, en utilisant des fenêtres de 10 mots autour des mentions des personnages.

Pour comparer les relations entre les personnages dans la fanfiction et dans le canon, les différences entre leurs *embeddings* sont mesurées à l'aide de la distance cosinus et de la différence vectorielle. Ces mesures servent de caractéristiques supplémentaires pour le modèle.

Les résultats observés en utilisant le modèle sur un corpus de fanfictions basées sur l'univers d'Harry Potter montrent qu'il n'est pas optimal. Il reconnaît une relation romantique quand elle n'est que platonique et inversement. Cela entraîne aussi forcément des erreurs quand il s'agit de détecter s'il y a eu un changement de la nature de la relation entre les deux personnages en fonction de l'œuvre originale. Néanmoins, le modèle parvient à détecter un changement de relation entre la fanfiction et l'œuvre originale lorsqu'un vocabulaire émotionnel très intense est utilisé, cela semble être un des seuls cas où il ne commet pas d'erreur.

### 3.4.3.3. Différences dans la caractérisation des personnages

Un autre objectif de Yoder et al. (2021) est d'étudier des changements dans la façon de présenter certains personnages entre une fanfiction et l'œuvre originale. La représentation d'un personnage diffère entre l'œuvre originale et les fanfictions quand la place du personnage change dans l'histoire, notamment quand la vision qu'ont les fans d'un personnage n'est pas en accord avec la façon dont il est décrit dans l'œuvre originale. Par exemple, un personnage plutôt associé à un vilain dans l'œuvre originale, qui passe du côté des héros dans la fanfiction.

Pour analyser la façon dont chaque texte présente ses personnages, des techniques de *word embeddings* sont utilisées. Les résultats sont ensuite présentés sous forme visuelle, ce qui permet alors de comparer les représentations de certains personnages dans chaque texte et de repérer des différences ou similitudes.

Le corpus utilisé est à nouveau des fanfictions inspirées de l'univers d'Harry Potter. Les résultats observés sont les suivants. Le personnage de Draco, apprécié des fans, est présenté comme un vilain dans les livres. La représentation visuelle des résultats de l'analyse montre un écart entre la représentation qui lui est donnée dans les livres et dans les fanfictions. La représentation du personnage d'Harry Potter, le héros du livre, ne varie quant à elle que très peu, on ne remarque pas d'écart significatif au niveau de la représentation visuelle des résultats.

Yoder et al. comparent les changements observés dans la caractérisation des personnages aux changements précédemment observés au niveau des relations et couples formés par ces personnages. Ils remarquent une corrélation entre le fait qu'un personnage soit présenté sous un meilleur jour dans une fanfiction et le fait qu'il soit mis en couple avec un autre personnage alors qu'il ne l'était pas dans l'œuvre originale, et inversement le fait qu'un personnage soit déprécié dans la fanfiction et qu'il se retrouve célibataire alors qu'il était en couple dans l'œuvre originale. Dans le corpus choisi, le personnage de Draco dont la représentation varie vers quelque chose de plus positif dans les fanfictions et aussi souvent mis en couple avec un autre personnage apprécié dans les fanfictions, alors qu'il ne l'est pas dans le livre. A l'inverse, un personnage comme Ron qui semble être décrit plus négativement dans les fanfictions et donc moins apprécié, voit souvent la relation romantique qu'il entretient dans les livres avec le personnage apprécié d'Hermione devenir platonique dans les fanfictions.

En conclusion, les différentes expériences de Yoder et al. montrent à quel point les fanfictions constituent un terrain fertile pour l'expérimentation en TAL. Elles mettent en lumière la difficulté de modéliser des dynamiques aussi fines que les relations ou les changements de caractérisation, même avec des outils puissants comme les *embeddings* ou les modèles de langue. L'approche proposée dans ce mémoire se concentre sur des objets un peu plus stables : des tags de fanfiction populaires et donc déjà bien établis en tant que marqueurs de tonalité, de genre ou de type de récit, dans une perspective stylométrique plus directement interprétable.

### **3.4.4. Point de vue social : l'influence de la société sur les fanfictions**

Plusieurs études récentes ont montré que les fanfictions ne sont pas seulement des objets narratifs, mais aussi des espaces où se rejouent, se renversent ou se reproduisent des dynamiques sociales, notamment autour du genre et des sexualités. Ces aspects ont une incidence directe sur le choix des personnages, des relations, et des thématiques, souvent explicités à travers les tags. Cette section présente quelques travaux mettant en évidence ces liens entre contexte social et contenu des fanfictions, pour mieux situer les tags comme marqueurs à la fois stylistiques et culturels.

#### **3.4.4.1. Pourcentage des personnages en fonction de leur genre**

Milli & Bamman (2016) se sont aussi penchés que la question du genre des personnages. Ils ont comparé les pourcentages de personnages féminins et masculins dans les fanfictions et les ont comparés aux pourcentages correspondants dans les œuvres canoniques à nouveau à l'aide de BookNLP.

Ils remarquent alors que le pourcentage associé aux personnages féminins est plus élevé dans les œuvres de fanfictions que les œuvres originales. Ces résultats pourraient confirmer que les fanfictions sont un lieu principalement féminin et un lieu où les fans mettent en scène une représentation qui leur manque dans le canon.

Fast et al. (2016) se sont aussi intéressés aux genres des personnages dans les fanfictions. Ils ont noté que bien que le pourcentage de personnage féminins soit effectivement plus élevé que dans œuvres canoniques il reste toujours inférieur aux pourcentages de personnages masculins.

Ils vont ensuite plus loin en se penchant sur la présence et reproduction de stéréotypes de genre dans les fanfictions. Pour réaliser cette étude, ils utilisent notamment une approche d'apprentissage automatique basée sur la régression logistique. Pour déceler des stéréotypes de genre au sein des textes d'un corpus de fanfictions, ils analysent les actions et les descriptions faites des personnages. Pour l'analyse des actions, ils extraient les données concernées à l'aide d'une approche basée sur la relation sujet-verbe, plus précisément, chaque occurrence de « she » ou « he » suivie d'un verbe est récupérée puis classée selon deux classes correspondant soit à un personnage masculin ou un personnage féminin. Pour l'analyse des descriptions, ils extraient les données concernées à l'aide d'une approche basée sur des relations adjectivales de deux types : les occurrences d'adjectifs associés à des sujets masculins ou féminins par le verbe être (« to be » en anglais) et les occurrences d'adjectifs associés à des noms masculins ou féminins (par exemple : « upset mother » ou « strong brother ») sont récupérées et classées dans l'une des deux classes mentionnées ci-dessus.

Ils se basent ensuite sur de précédents travaux pour déterminer et construire différentes catégories de stéréotypes (par exemple, « violent » pour les hommes et « soumise » pour les femmes). Une fois ces catégories définies, au nombre de 16, ils évaluent le niveau d'appartenance à l'une ou l'autre de ces catégories des mots les plus fréquents collectés précédemment en faisant appel à plusieurs annotateurs. Ils décident de ne retenir que les mots unanimement annotés comme « liés »

ou « fortement liés » à une catégorie pour créer des lexiques représentatifs des différentes catégories définies antérieurement.

Pour étudier la présence de stéréotype de genre dans les fanfictions en se basant sur les actes et descriptions des personnages, ils ont calculé combien de fois en moyenne chaque stéréotype est attribué à des personnages masculins et féminins dans un corpus de fanfictions. Les résultats sont normalisés pour tenir compte du fait que certaines fanfictions peuvent mettre en scène plus de personnages masculins que féminins ou l'inverse. Pour obtenir ces résultats, ils ont appliqué le test t de Welch.

Pour analyser comment les stéréotypes de genre étaient répandus dans les fanfictions d'auteurs masculins et féminins et pour prédire le succès d'une fanfiction en fonction des stéréotypes de genres qu'elle présente ils utilisent une approche de régression linguistique.

Les résultats obtenus servent à répondre à quatre questions posées par Fast et al. (2016) :

1. « Quels verbes et donc quelles actions sont associés aux personnages féminins et masculins ? »<sup>9</sup>
2. « Quels adjectifs et donc quelles descriptions sont associées aux personnages féminins et masculins ? »<sup>10</sup>
3. « Les fanfictions reproduisant des stéréotypes sont-elles plus populaires ? »<sup>11</sup>
4. « Comment le genre de l'auteur impacte les stéréotypes présents dans leurs fanfictions ? »<sup>12</sup>

Les réponses offertes après l'examen des résultats sont les suivantes :

1. Les personnages masculins sont plus « actifs » et plus violents que les personnages féminins.
2. Les personnages féminins sont décrits comme soumis ou immatures.
3. Les fanfictions présentant des stéréotypes de genres n'ont pas plus ni moins de succès que les autres.
4. Les auteurs féminins et masculins reproduisent tous les deux les mêmes stéréotypes dans leurs fanfictions.

Ainsi, la conclusion faite est que même si les fanfictions sont un espace où les fans peuvent laisser cours à leur imagination et réécrire l'histoire comme bon leur semble avec la possibilité de renverser les normes sociales, ils semblent pourtant avoir tendance à reproduire les stéréotypes de la société. Les auteurs féminins reproduisent ces stéréotypes autant que les auteurs masculins, même s'ils leur portent souvent préjudice. Aussi, les auteurs féminins écrivent plus de personnages masculins que les auteurs masculins qui écrivent eux-mêmes plus de personnages masculins que de personnages féminins. Quant aux lecteurs, ils ne semblent pas se soucier de la présence ou non de stéréotypes dans les fanfictions qui lisent, car la présence de stéréotypes n'a pas d'impact sur le succès d'une fanfiction.

---

<sup>9</sup> Traduction personnelle de « What role does gender play in how characters act? Are male and female characters associated with different kinds of verbs? »

<sup>10</sup> Traduction personnelle de « What rôle does gender play in how characters are described? Are male and female characters associated with different kinds of adjectives? »

<sup>11</sup> Traduction personnelle « How do patterns in character action and description impact a story's rating? Are stories that deploy stereotypes more highly rated? »

<sup>12</sup> Traduction personnelle « How does *author* gender impact the the [sic] stereotypes we discover? Do male authors write women differently than female authors, and vice versa? »

Une limite de cette étude à prendre en compte est que l'analyse se fait au niveau des phrases prises une par une, ce qui ne reflète pas toujours le contexte d'une action ou d'un fait ce qui peut fausser les résultats.

### **3.4.4.2. Impact des évènements LGBTQ+**

En plus de s'être intéressé aux personnages et leurs relations, Yoder (2021) s'est aussi penché sur les potentielles répercussions d'évènements LGBTQ+ sur les fanfictions. Il voulait voir s'il y avait un lien entre l'évolution de la représentation LGBTQ+ dans les fanfictions et les évènements de la société autour des questions LGBTQ+.

Pour ce faire, il a réuni un corpus mélangeant des fanfictions de *fandoms* associées à une forte représentation LGBTQ et de *fandoms* associées à une faible représentation LGBTQ+. Les fanfictions retenues sont donc définies par un mélange de tags associés à des relations LGBTQ+ et de tags associés à des relations hétérosexuelles.

Pour l'analyse des *tags*, il applique une technique de régression logistique afin de déterminer si la présence d'un tag LGBTQ+ dépend d'un évènement qui se produit environ au même moment que l'écriture de la fanfiction.

Pour l'analyse des personnages et des couples, il utilise la pipeline qu'il a construit précédemment pour extraire les différents personnages d'une fanfiction. Pour déterminer ensuite si ces derniers sont queer ou non, il compare les noms des personnages récupérés par la pipeline aux noms de personnages associés à des tags de *ships*. Si deux personnages associés à un même tag de *ship* ont été identifiés comme deux personnages de même genre ou non-binaires par la pipeline, alors ils sont considérés comme queer, sinon ils sont considérés comme hétérosexuels. Il utilise ensuite des outils de topic modeling pour identifier les motifs récurrents dans la caractérisation de personnages au fil du temps.

Les résultats montrent qu'après la légalisation du mariage homosexuel, on peut remarquer une augmentation du nombre de fanfictions avec des tags LGBTQ+, ainsi qu'une augmentation des couples LGBTQ+ et de thèmes liés au mariage dans les fanfictions.

Yoder récupère également un corpus d'articles d'actualités sur des sujets LGBTQ+ et utilise des outils de topic modeling sur ce corpus pour identifier des sujets d'intérêt dans la communauté LGBTQ+. Il les compare avec le corpus de fanfictions pour étudier s'il y a un lien avec les sujets identifiés précédemment dans ces dernières. Il remarque alors qu'il y a effectivement une corrélation quand il s'agit de sujets et évènements qu'on pourrait qualifier de culturels, mais moins quand il s'agit d'évènements et de sujets portant sur la législation.

Enfin, il compare l'impact des évènements LGBTQ+ sur les fanfictions de *fandoms* LGBTQ+ et de *fandoms* non-LGBTQ+ et observe que ces évènements ont un plus grand impact sur les fanfictions LGBTQ+ que sur les fanfictions non-LGBTQ+.

Cette approche met en lumière le rôle des tags comme indicateurs sensibles des dynamiques sociales : ils évoluent en réponse aux grands mouvements culturels et servent à structurer la visibilité des identités et des thématiques queer dans la production littéraire amateur. Ces résultats rejoignent la perspective de ce mémoire, qui considère les tags non seulement comme des outils de classification, mais aussi comme des marqueurs de l'évolution des pratiques et des représentations au sein des communautés de fans. Cette dimension sociale, bien que non directement mesurée ici, constitue un arrière-plan essentiel pour comprendre comment certains tags se stabilisent comme des sous-genres

à part entière, tandis que d'autres émergent en lien avec des évolutions culturelles ou communautaires.

### 3.5. Conclusion et contribution personnelle

Cet état de l'art a montré que les fanfictions sont des récits de fans qui forment un véritable réseau entre elles, elles s'inspirent les unes des autres, en s'inspirant des avis et souhaits de différentes *fandoms* qui se construisent elles-mêmes en réponse aux différentes opinions qui naissent autour de différentes œuvres canoniques. Les fanfictions s'écrivent dans le temps en s'inspirant de ce qui les entoure. Les fans peuvent communiquer entre eux et s'entraider dans le processus créatif en temps réel, sur les réseaux sociaux ou directement sur les sites de fanfictions dans les sections commentaires. Des études ont d'ailleurs montré en étudiant les discussions présentes dans ces commentaires qu'en écrivant, lisant et critiquant des fanfictions, les fans développent et usent de plusieurs compétences littéraires. Le monde extérieur et les différents événements sociaux et culturels ont aussi un impact sur le processus de création de fanfictions.

La façon de définir et de classer les fanfictions est aussi particulière, ce sont les auteurs eux-mêmes qui définissent le genre de leurs récits et les classifient à l'aide de *tags*. Ces tags sont souvent le reflet des idées d'une *fandom* particulière, sauf pour certains qui sont si populaires qu'ils sont devenus des genres et sous-genres à part entière réutilisés par tous. C'est à l'aide de ces tags que les lecteurs sont capables de reconnaître si une fanfiction est susceptible de leur plaire ou non. Aussi, la plupart de ces tags et sous-genres reposent sur les personnages et les relations entre eux, qui sont le point central, voire même l'idée de base d'un grand nombre de fanfictions. Les personnages et les couples sont donc un élément capital à prendre en compte lors d'une étude basée sur des fanfictions. Plusieurs chercheurs en TAL ont par ailleurs essayé de développer différents outils capables de reconnaître et d'analyser les personnages et les relations entre eux dans ces textes.

Les fanfictions sont un espace complètement libre où les fans écrivent pour leur propre plaisir. Cet espace se développe en parallèle d'œuvres originales pour répondre aux demandes dues à un manque de représentation ressenti par certains fans. Ainsi, beaucoup d'études se sont intéressées aux différences entre fanfictions et œuvres originales. De plus, les fans continuent pour la plupart de reproduire les stéréotypes de genre présents dans la société et les œuvres populaires.

Cependant, on remarque que peu d'études se sont intéressées aux sous-genres en fanfictions. Pourtant les tags de fanfiction constituent un élément central de la fanfiction, en jouant un rôle crucial dans la classification, la propagation, et même l'inspiration des œuvres auprès des communautés de fans. Ils servent à la fois d'indicateurs de contenu, de genre et de ton et permettent aux fans de sélectionner rapidement les œuvres qui correspondent à leurs attentes et préférences. De plus, ces tags forment de véritables sous-genres littéraires différents des catégories « classiques » comme le policier, le thriller, ou la science-fiction, et ne suivent pas les mêmes conventions établies par la littérature traditionnelle.

Étudier les caractéristiques de ces tags est donc particulièrement intéressant, d'une part pour mieux comprendre le monde de la fanfiction, les attentes et habitudes des fans, et d'autre part pour explorer ces nouveaux genres littéraires, encore peu documentés. En se concentrant sur la spécificité des tags en fanfiction, cette étude cherche à combler un manque dans les recherches en TAL tout en contribuant à une meilleure compréhension des pratiques littéraires et sociales des communautés de fans, où les frontières entre les genres sont continuellement redéfinies.

Ce mémoire s'intéresse plus particulièrement à l'étude des caractéristiques de cinq tags populaires en fanfiction : *Fluff*, *Angst*, *Hurt/Comfort*, *Friends to lovers* et *Enemies to lovers*. En utilisant des méthodes de stylométrie et des algorithmes d'apprentissage automatique, l'objectif est

d'identifier les particularités textuelles propres à chacun de ces tags et de mieux comprendre les spécificités linguistiques et les thématiques qui les distinguent. Cette approche combinée permettra non seulement de classifier les œuvres selon ces catégories, mais aussi d'explorer les motifs récurrents et les choix stylistiques qui définissent ces sous-genres modernes.

## 4. Méthodologie

---

Certaines portions de code, ainsi que des clarifications méthodologiques ponctuelles, ont été réalisées à l'aide d'outils d'intelligence artificielle, notamment ChatGPT (OpenAI). L'ensemble des scripts, notebooks et autres ressources liées à l'analyse présentée dans ce mémoire est disponible sur le dépôt GitLab suivant : [https://gitlab.unistra.fr/paulinemoreau/memoire\\_fanfiction/](https://gitlab.unistra.fr/paulinemoreau/memoire_fanfiction/)

### 4.1. Introduction

Cette partie méthodologique décrit les étapes et les outils utilisés pour collecter, préparer, et analyser les données de fanfictions issues de cinq tags spécifiques. En s'appuyant sur des techniques de stylométrie et d'apprentissage automatique, ce mémoire vise à identifier les caractéristiques textuelles distinctives de chaque tag et à explorer leurs éventuelles similitudes ou différences. Plusieurs étapes constituent cette méthodologie : le choix des données, leur collecte, leur prétraitement, leur vectorisation, puis leur analyse à l'aide de l'outil stylométrique pydistinto et de l'entraînement de modèles d'apprentissage automatique supervisés.

### 4.2. Choix des données

#### 4.2.1. Généralités sur le corpus

La première étape de la méthodologie est d'établir le type de corpus de fanfiction à constituer. J'ai tout d'abord choisi de travailler sur un corpus de fanfiction françaises, car c'est ma langue maternelle mais surtout, car il y a peu de travaux faits sur des fanfictions en français, beaucoup moins que des travaux sur des fanfictions en anglais par exemple.

Aussi, j'ai choisi de collecter mon corpus de fanfiction depuis le site AO3. J'ai fait ce choix, car il s'agit probablement du site d'hébergement et publication de fanfictions le plus populaire, ce qui facilite l'accès à un large éventail de textes. AO3 est également le site le mieux documenté en termes d'outils de TAL comme les scrapers. De plus, c'est un site avec lequel j'étais déjà familiarisée et je considère que sa structure est parmi les plus agréables et intuitives des sites de fanfictions.

#### 4.2.2. Choix des tags

Pour cette étude j'ai sélectionné cinq tags populaires de fanfiction : *Fluff*, *Angst*, *Hurt/Comfort*, *Friends to lovers* et *Enemies to lovers*. Pour faire ce choix j'ai suivi plusieurs critères. Tout d'abord je voulais des tags populaires pour avoir un volume de données suffisant, surtout en sachant que je travaille sur des fanfictions francophones. J'ai donc consulté le nuage des tags populaires disponible sur le site d'AO3.

Je souhaitais aussi travailler sur des tags de relation (entre personnages), car on a vu dans l'état de l'art que c'est un élément central aux fanfictions. Les trois tags de relation les plus populaires sont *Fluff*, *Angst* et *Hurt/Comfort*.

A ces trois tags, j'ai décidé d'en rajouter un ou deux. Certains tags à caractère érotique, bien que très populaires, ont été écartés, car leur analyse aurait surtout révélé des termes à caractère explicitement sexuel sans nécessairement apporter de profondeur thématique supplémentaire. J'ai donc choisi les tags *Friendships* et *Enemies* qui ne sont finalement pas des tags à part entière mais des tags « parapluie ». Ces deux tags contiennent des fanfictions annotées avec différents tags relevant de plus ou moins près d'une relation amicale ou d'une rivalité. En les explorant un peu plus en profondeur, on remarque qu'une écrasante majorité des fanfictions du sous-genre *Friendships* sont en fait des fanfictions portant le tag *Friends to lovers*. De même, une écrasante majorité des

fanfictions *Enemies* sont en fait des fanfictions *Enemies to lovers*. J'ai donc décidé d'ajouter les deux tags *Friends to lovers* et *Enemies to lovers* à mon étude.

Les cinq sous-genres étudiés correspondent à des tags relationnels couramment employés sur AO3. Ils doivent être compris comme des catégories thématiques de contenu communautaires plutôt que comme des genres littéraires traditionnels.

#### 4.2.3. Hypothèses sur les données

Concernant les trois grands tags *Fluff*, *Angst* et *Hurt/Comfort*, une première hypothèse est que les tags *Fluff* et *Angst* sont quelque peu opposés. Je m'attends à trouver un vocabulaire majoritairement léger et réconfortant pour les fanfictions *Fluff*, contre un vocabulaire majoritairement violent et sombre pour les fanfictions *Angst*. Cette hypothèse pourra notamment être vérifiée en appliquant pydistinto à un corpus de fanfictions *Fluff* contre un corpus de fanfictions *Angst*. Quant aux fanfictions *Hurt/Comfort*, je suppose qu'elles sont à mi-chemin entre les fanfictions *Fluff* et *Angst*. En effet, on peut s'attendre à retrouver à la fois un ton sombre et violent pour la partie *Hurt* et un ton plus réconfortant pour la partie *Comfort*, ce qu'on pourra aussi vérifier à l'aide de pydistinto. Il serait aussi intéressant d'essayer de déterminer si le sous-genre *Hurt/Comfort* est plus proche du tag *Fluff* ou du tag *Angst*. Une idée pour déterminer cela serait notamment d'observer les matrices de confusion des algorithmes d'apprentissage automatique, si les résultats sont assez concluants pour le permettre, et voir éventuellement si les fanfictions *Hurt/Comfort* sont plus souvent confondues avec l'un ou l'autre tag.

Concernant les deux tags *Friends to lovers* et *Enemies to lovers*, ils représentent des schémas narratifs très précis, parfois appelés tropes, alors que les trois autres tags sont des tags un peu plus vastes qui permettent de donner le ton général de la fanfiction. On pourrait donc s'imaginer qu'il existe une sorte de hiérarchie entre les tags, où *Friends to lovers* et *Enemies to lovers* peuvent être vus comme des sous-catégories de *Fluff*, *Angst* et *Hurt/Comfort*. En effet, rien n'empêche pour une fanfiction *Fluff*, *Angst* ou *Hurt/Comfort* de mettre en scène une relation de type *Friends to lovers* ou *Enemies to lovers*. Si cette hypothèse s'avère vraie, cela pourrait éventuellement créer des confusions lors de l'apprentissage automatique.

Pour minimiser les confusions, j'ai décidé de ne collecter que des fanfictions qui ne sont taggées qu'avec un seul des cinq tags choisis (des fanfictions *Fluff* qui ne sont pas taggées *Angst*, *Hurt/Comfort*, *Friends to lovers* ni *Enemies to lovers* ; des fanfictions *Angst* qui ne sont pas taggées *Fluff*, *Hurt/Comfort*, *Friends to lovers* ni *Enemies to lovers* ; etc.). J'envisage aussi de commencer par entraîner les modèles d'apprentissage automatique avec uniquement les tags *Fluff*, *Angst* et *Hurt/Comfort*, avant de rajouter les deux autres tags ultérieurement.

#### 4.3. Collecte des données

Pour collecter mon corpus de fanfictions je reprends le même scraper que Black a utilisé dans son étude : *Radiolarian/AO3Scraper* par Li & Sterman (2016), disponible sur GitHub. Ce scraper fonctionne en deux temps. Une première partie permet de récupérer les identifiants de toutes les fanfictions associées à un lien AO3 dans un fichier CSV. Une deuxième partie permet de récupérer les fanfictions correspondant à ces identifiants avec leurs métadonnées dans un fichier CSV.

Dès la collecte des données, je fais un tri des fanfictions selon leurs tags. Le site d'AO3 permet de sélectionner les tags voulus et d'exclure les tags de notre choix, il permet aussi de choisir la langue des fanfictions, comme on peut le voir sur la Figure 4, p.37. Je collecte donc les fanfictions en langue française par groupes, chaque groupe est associé à un lien AO3 qui correspond à l'un des cinq tags

avec les quatre autres exclus. Ces liens sont utilisés comme entrée pour le premier script de collecte des identifiants.

Figure 4 - Tri de la langue et des tags à exclure sur le site d'AO3 lors de la pré-collecte des fanfictions *Fluff*

The screenshot shows the 'Exclude' section of the AO3 search interface. It includes a list of categories to exclude (Ratings, Warnings, Categories, Fandoms, Characters, Relationships, Additional Tags) and a field for 'Other tags to exclude' containing 'Angst', 'Hurt/Comfort', 'Friends to Lovers', and 'Enemies to Lovers', each with a red 'X' button. Below this is the 'More Options' section with links for Crossovers, Completion Status, Word Count, and Date Updated. A 'Search within results' input field is also present. The 'Language' dropdown is set to 'Français'. At the bottom are 'Sort and Filter' and 'Clear Filters' buttons.

**Exclude** [?](#)

- ▶ Ratings
- ▶ Warnings
- ▶ Categories
- ▶ Fandoms
- ▶ Characters
- ▶ Relationships
- ▶ Additional Tags

Other tags to exclude

Angst X

Hurt/Comfort X

Friends to Lovers X

Enemies to Lovers X

**More Options**

- ▶ Crossovers
- ▶ Completion Status
- ▶ Word Count
- ▶ Date Updated

Search within results [?](#)

Language

Français

Sort and Filter

[Clear Filters](#)

Les fanfictions disponibles en français sont très variables selon les tags :

- *Fluff, Angst et Hurt/Comfort* : plusieurs milliers de fanfictions disponibles.
- *Friends to Lovers et Enemies to Lovers* : seulement 250 à 500 fanfictions.

J'ai donc opté pour une collecte approximativement équilibrée, en visant environ 1000 fanfictions pour *Fluff, Angst* et *Hurt/Comfort*, et en récupérant l'intégralité des fanfictions *Friends to Lovers* et *Enemies to Lovers* disponibles.

Le code original du scraper AO3Scraper a été modifié pour améliorer le suivi de la collecte et optimiser la récupération des fanfictions malgré les blocages du site. Pour le script de collecte des identifiants, j'ai effectué les changements et ajouts suivants (voir Annexe 1, p.82) :

- Instructions d'affichage :
  1. Ajout d'un message indiquant quelle page est en cours de traitement.
  2. Ajout d'un message indiquant quelle fanfiction est en cours de collecte sur cette page.
  3. Affichage du nombre d'identifiants récupérés par page.
- Gestion des erreurs dues aux blocages :
  1. Mise en place de plusieurs essais de collecte quand une page semble vide.
- Optimisation des requêtes pour limiter les blocages :
  1. Utilisation d'un temps d'attente (`time.sleep`) entre chaque essai et requête.

Pour le script de collecte des fanfictions et leur métadonnées, j'ai modifié le code ainsi (voir Annexe 2, p.83) :

- Instructions d'affichage :
  1. Indication du nombre total de fanfictions à collecter et de la progression en cours.
  2. Affichage des échecs et succès dans la collecte.
- Système d'essais multiples pour gérer les erreurs
  1. Trois essais en cas d'erreur avant de passer à la fanfiction suivante.
  2. Utilisation de temps d'attente croissant entre les essais.
- Utilisation d'user-agents.

J'ai rencontré plusieurs difficultés lors de la collecte. Il était assez simple de collecter les 500 premières fanfictions de chaque tag, mais ensuite les échecs, sûrement dus à des blocages du site commençaient à s'accumuler. Lors de la collecte des identifiants, le script m'affichait souvent que la page semblait vide et lors de la collecte des fanfictions, le script me renvoyait souvent des erreurs 403, 520 ou 525 trois fois de suite, soit plus que le nombre d'essais fixés. Étrangement, ce sont les fanfictions *Fluff* qui m'ont posé le plus de problèmes avec plusieurs échecs successifs. En revanche, la collecte des quatre autres tags a été plus fluide. C'est pourquoi le nombre final de fanfictions *Fluff* collectées est légèrement inférieur au nombre de fanfictions *Angst* et *Hurt/Comfort*. Quant aux fanfictions *Enemies to lovers* et *Friends to lovers*, la collecte a été limitée non pas par des erreurs, mais simplement par le faible nombre de fanfictions disponibles sur AO3. Le Tableau 1 donne un récapitulatif des fanfictions collectées.

Tableau 1 - Nombre de fanfictions par tag après la collecte

Tag	Nombre de fanfictions
Fluff	822
Angst	1006
Hurt/Comfort	1032
Enemies to lovers	287
Friends to lovers	506
<b>Total</b>	<b>3653</b>

#### 4.4. Prétraitement des données

##### 4.4.1. Fusion et premiers nettoyages des fanfictions

Un premier prétraitement des données est effectué à l'aide de la bibliothèque Python pandas.

La collecte des fanfictions ayant été effectuée par groupes correspondants aux cinq tags choisis, je possède cinq fichiers CSV contenant chacun toutes les fanfictions collectées pour un tag donné.

Je commence par supprimer les éventuels doublons dans chacun des cinq fichiers CSV.

Extrait de code 1 – Exemple de suppression des doublons pour les fanfictions *Fluff*

```
#suppression des doublons Fluff
fluff_tab_no_db = fluff_tab.drop_duplicates(subset='work_id',
keep='first')
```

Comme chaque fichier contient uniquement des fanfictions d'un même tag, j'ajoute une colonne "tag" qui attribue à chaque ligne le tag correspondant : « Fluff », « Angst », « Hurt/Comfort », « Enemies to lovers » ou « Friends to lovers ». Cette colonne sera essentielle lors de la classification automatique pour définir les classes à prédire.

Extrait de code 2 - Exemple d'ajout de la colonne "tag" pour les fanfictions *Fluff*

```
#colonne tag avec la valeur "Fluff" pour les fanfictions Fluff
fluff_tab_no_db.insert(1, 'tag', 'Fluff')
```

Enfin, je fusionne les cinq fichiers ensemble que j'enregistre sous la forme d'un seul fichier CSV qui contient maintenant toutes les fanfictions et leurs métadonnées. C'est ce fichier qui sera utilisé dans la suite des analyses.

Extrait de code 3 - Fusion en un seul fichier

```
# Fusionner les DataFrames
fanfics_complet = pd.concat([fluff_tab_no_db, angst_tab_no_db,
hc_tab_no_db, etl_tab_no_db, ftl_tab_no_db], ignore_index=True)
```

#### 4.4.2. Nettoyages des textes et limites maximales de longueur

Une fois toutes les fanfictions réunies dans un seul fichier, je supprime les lignes où les textes des fanfictions sont vides. En effet, les textes des fanfictions constituent l'élément central de cette étude et les lignes dont le texte est vide ne sont pas exploitables et pourraient même générer des erreurs lors des traitements à suivre.

Extrait de code 4 - Suppression des lignes vides

```
# Supprimer les lignes avec NaN
df_fanfic_no_vide = df_fanfic.dropna(subset=[ 'body' ])
# Supprimer les lignes avec des espaces uniquement
df_fanfic_no_vide =
df_fanfic_no_vide[df_fanfic_no_vide['body'].str.strip().ne('')]
```

Je nettoie ensuite tous les textes en supprimant tous les éventuels éléments parasites qui pourraient s'y trouver (URLs, balises HTML, mauvais encodage ou écriture de caractères spéciaux) à l'aide des modules Python unicodedata, html ou re et bibliothèque BeautifulSoup.

Extrait de code 5 - Fonctions de nettoyage

```
def supprimer_urls(texte):
    return re.sub(r'http[s]?:\/\/\S+|www\.\S+', '', texte)
def supprimer_html(texte):
    return BeautifulSoup(texte, "html.parser").get_text()
def normaliser_caracteres(texte):
    return unicodedata.normalize("NFKC", texte)
def remplacer_entites_html(texte):
    return html.unescape(texte)
def nettoyer_retours_ligne(texte):
    return re.sub(r'\n+', '\n', texte).strip()
```

Aussi, je décide de fixer une limite de longueur maximale de caractères pour les textes de fanfiction pour supprimer les textes trop volumineux qui causent des problèmes lors des prochaines étapes de l'analyse, notamment des problèmes de mémoire RAM. En effet une minorité de fanfictions atteignent des longueurs aberrantes de plusieurs millions de caractères (la plus longue faisant plus de 7 500 000 caractères). Avant de choisir les limites à fixer, j'explore les tailles des fanfictions des différents tags. Je m'intéresse aux longueurs moyennes et écart-types des tailles des fanfictions dans chaque tag. Je vérifie aussi le nombre de fanfictions par tag qui possèdent plus de 200 000 à 500 000 caractères.

Extrait de code 6 - Exploration des longueurs des fanfictions

```
# Colonne contenant le nombre de caractères des fanfictions
df_fanfic_no_vide["nb_caracteres"] =
df_fanfic_no_vide["body_clean"].apply(len)
# Calculer le nombre de fanfictions avec plus de 300,000 caractères
long_fanfics =
df_fanfic_no_vide[df_fanfic_no_vide['body_clean'].apply(len) > 300000]
# Compter le nombre de fanfictions par tag
tag_counts = long_fanfics['tag'].value_counts()
```

```
# Calcul des moyennes et écart-types des longueurs
moyennes = df_fanfic_no_vide.groupby("tag") ["nb_caracteres"].mean()
ecarts_types = df_fanfic_no_vide.groupby("tag") ["nb_caracteres"].std()
```

Au final je décide de fixer une longueur maximale de 200 000 caractères pour *Fluff*, *Angst* et *Hurt/Comfort* qui me semblent être un bon compromis entre une longueur raisonnable, pas trop volumineuse et une perte de données raisonnable. En effet, cela entraîne la suppression de 98 fanfictions *Hurt/Comfort* qui étaient les plus nombreuses, 68 fanfictions *Angst* et seulement 15 fanfictions *Fluff*. Pour *Enemies to lovers* et *Friends to lovers* j'augmente la limite à 500 000 caractères, car elles sont déjà moins nombreuses que les autres et une limite à 200 000 caractères me ferait perdre un nombre de données trop important (environ 25 % du total des fanfictions *Enemies to lovers*). On peut voir la répartition des fanfictions restantes dans le Tableau 2.

Tableau 2 - Nombre de fanfictions par tag après mise en place d'une limite maximale de longueur

Tag	Nombre de fanfictions
Fluff	805
Angst	935
Hurt/Comfort	934
Enemies to lovers	269
Friends to lovers	494

#### 4.4.3. Stop-words

Je décide de garder une version des textes où les noms des personnages de chaque fanfiction sont conservés et une version où ils sont supprimés, stockés dans deux colonnes différentes du fichier CSV. Pour ce faire j'utilise la colonne « character » créée lors de la collecte. Elle contient les noms des personnages impliqués dans chaque fanfiction qui sont indiqués par les auteurs dans une section dédiée sur le site AO3. En explorant rapidement le corpus, j'ai remarqué qu'il existe différentes façons de noter les noms des personnages qu'il faut donc prendre en compte (voir Annexe 3, p.87).

Pour éliminer les mots-vides ou mots-outils des textes de fanfiction, j'utilise la bibliothèque SpaCy pour le français.

#### 4.4.4. Tokenisation et lemmatisation

Les textes sont tokenisés, c'est-à-dire séparés en unités minimales, et sont aussi lemmatisés, c'est-à-dire réduits à des mots dans leur forme de base, à l'aide de SpaCy. Je tokenise et lemmatise à la fois les textes où les noms des personnages ont été supprimés et ceux où ils ont été conservés.

Extrait de code 7 - Fonction de tokenisation

```
def split_into_tokens_wosw_spacy(text):
    doc = spacy_pipeline(text)
    return ' '.join([t.text for t in doc if not t.is_stop])
```

Extrait de code 8 - Fonction de lemmatisation

```
def split_into_lemmas(text):
    doc = spacy_pipeline(text)
    lemmatised_text = [w.lemma_ for w in doc]
```

```
return ' '.join(lemmatised_text)
```

Au final j'obtiens quatre versions des textes, stockés dans quatre colonnes différentes :

- Textes tokenisés avec noms des personnages
- Textes tokenisés sans noms des personnages
- Textes lemmatisés avec noms des personnages
- Textes lemmatisés sans noms des personnages

#### 4.4.5. Longueur minimale

Certaines fanfictions sont très courtes (moins de 100 mots) et peu informatives pour une analyse basée sur le texte.

Pour garantir la pertinence des données, je choisis de supprimer les fanfictions de moins de 500 caractères.

Extrait de code 9 - Suppression des fanfictions de moins de 500 caractères

```
df_limit_min = df_fanfics[df_fanfics["nb_caracteres"] >= 500]
```

Cette suppression a un faible impact sur le corpus (seulement 12 textes). De plus, une analyse de ces fanfictions montre qu'elles ne correspondent pas toujours à de véritables fanfictions, mais plutôt à des extraits, annotations ou textes hors sujet.

Leur suppression permet ainsi de réduire le bruit dans les analyses à venir et d'améliorer la performance des algorithmes de classification. Le Tableau 3 suivant montre le nombre total de fanfictions par tag à la fin des prétraitements, avant la classification automatique.

Tableau 3 - Nombre de fanfictions par tag après limite minimale

Tag	Nombre de fanfictions
Fluff	800
Angst	930
Hurt/Comfort	932
Enemies to lovers	269
Friends to lovers	494

#### 4.5. Vectorisation des données

Pour que les textes soient exploitables par les modèles d'apprentissage automatique, ils sont convertis en vecteurs numériques. J'utilise une vectorisation TF-IDF. C'est la vectorisation qui me semble la plus adaptée à cette étude, car elle est transparente, facilement interprétable et compatible avec les différents algorithmes de classification dits classiques qui seront utilisés ultérieurement tels que Naïve Bayes, arbres de décision ou encore régression logistique.

Extrait de code 10 - Vectorisation des textes

```
# Découpage des textes en listes de mots
def split_text(text) :
    # Les textes ayant déjà été pré-tokénisés, il suffit de les découper
    # au niveau des espaces
```

```

    return text.split(' ')
# Objet TfidfVectorizer
text_vectorizer = TfidfVectorizer(tokenizer=split_text, min_df=0.01)

```

#### 4.6. Construction des corpus de développement et de test

Pour la construction des modèles d'apprentissage automatique, les données sont divisées en sous-corpus de développement et de test. Afin de garantir que les mêmes auteurs ne soient pas présents dans différents corpus, les fanfictions ont été regroupées par auteur avant la division. Cette précaution évite que le modèle ne se base sur des spécificités stylistiques propres aux auteurs, ce qui pourrait fausser les résultats en introduisant un biais. En effet, si un même auteur est présent dans plusieurs corpus, le modèle pourrait « reconnaître » le style de cet auteur au lieu de se concentrer uniquement sur les caractéristiques des tags.

La séparation des auteurs en amont cause un déséquilibre lors de la division en sous-corpus de développement et de test. En effet, les proportions de chaque tag n'étaient pas les mêmes entre le corpus de développement et le corpus de test obtenus, de plus elles ne représentaient plus forcément les proportions globales observées dans l'ensemble des données. Ce déséquilibre pouvait fausser les résultats en rendant certains tags sur-représentés ou sous-représentés dans l'un des sous-corpus, influençant ainsi la performance des modèles.

Pour limiter ce problème, j'ai utilisé une division stratifiée qui prend en compte les proportions des tags au moment de la séparation des auteurs. L'objectif était d'attribuer les auteurs aux corpus de développement et de test de manière à conserver une distribution des tags la plus proche possible de la distribution initiale.

Extrait de code 11 - Division stratifiée des données

```

# Récupérer les proportions globales des tags
proportions = df_fanfics["tag"].value_counts(normalize=True)

# Séparer les auteurs en fonction de ces proportions
auteurs_uniques = df_fanfics["author"].unique()

# On crée un DataFrame pour stocker chaque auteur et les tags qu'il a écrit
author_tag_counts =
df_fanfics.groupby("author") ["tag"].value_counts().unstack(fill_value=0)

# Normaliser les tags pour chaque auteur
author_tag_counts =
author_tag_counts.div(author_tag_counts.sum(axis=1), axis=0)

# Séparer les auteurs en gardant un équilibre des tags
auteurs_dev, auteurs_test = train_test_split(
    author_tag_counts.index, test_size=0.23, random_state=42)

# Assigner les fanfictions aux sous-corpus
df_dev = df_fanfics[df_fanfics["author"].isin(auteurs_dev)]

```

```
df_test = df_fanfics[df_fanfics["author"].isin(auteurs_test)]
```

Cette division n'a pas abouti à une répartition parfaitement équilibrée. Néanmoins, le déséquilibre final restait suffisamment faible pour que les modèles puissent être entraînés sans risque majeur de biais. Le Tableau 4 suivant montre la répartition des fanfictions et des auteurs dans les différents jeux de données.

Tableau 4 - Répartition des données suite à la division en corpus de développement et de test

	Corpus de développement		Corpus de test		Ensemble des données	
	Nombre	Pourcentage	Nombre	Pourcentage	Nombre	Pourcentage
Nombre de fanfictions	2556	74.63 %	869	25.37 %	3425	100 %
Nombre d'auteurs	797	76.93 %	239	23.07 %	1036	100 %
Répartition des tags						
Fluff	617	24.14 %	183	21.06 %	800	23.36 %
Angst	727	28.44 %	203	23.36 %	930	27.06 %
Hurt/Comfort	716	28.01 %	216	24.86 %	932	27.19 %
Enemies to lovers	175	6.85 %	94	10.82 %	269	8.22 %
Friends to lovers	321	12.56 %	173	19.91 %	494	14.17 %

Un autre biais possible pourrait venir des *fandoms*. En effet, une surreprésentation de fanfictions issues d'une même *fandom* au sein d'un sous-corpus pourrait amener les algorithmes à se baser sur des caractéristiques propres à cet univers pour classifier les textes. Cependant, compte tenu du volume limité de fanfictions en français et de la complexité de la tâche, il n'était pas possible de garantir que les mêmes *fandoms* ne soient pas représentées dans plusieurs ensembles. Aussi, comme nous le verrons dans les résultats, les différences stylistiques propres aux sous-genres restent suffisamment marquées pour être interprétables au-delà de ces effets.

## 4.7. Entraînement des algorithmes de classification

### 4.7.1. Objectifs de la classification

L'objectif de cette classification est de déterminer si les fanfictions possèdent des caractéristiques linguistiques distinctes et spécifiques à chaque tag, qui permettraient de les classer automatiquement à l'aide de modèles d'apprentissage automatique. Ensuite, les coefficients des meilleurs modèles seront examinés pour identifier les caractéristiques les plus déterminantes de chaque tag.

#### **4.7.2. Préparation des données**

Cette analyse se découpe en deux étapes. Un premier entraînement est effectué uniquement sur les trois tags principaux (*Fluff, Angst, Hurt/Comfort*) qui sont les plus abondants et équilibrés en termes de données. Dans un second temps, les deux autres tags restants (*Enemies to lovers* et *Friends to lovers*) sont rajoutés à l'étude pour analyser si la confusion augmente avec l'ajout de tags de *tropes* relationnels et éventuellement déceler une hiérarchie entre les tags.

Les modèles sont entraînés sur le corpus de développement et évalués sur le corpus de test créés précédemment (voir Tableau 4, p.44). La colonne cible utilisée pour déterminer les classes est la colonne « tag ». Les textes des fanfictions à classer sont présents dans quatre colonnes différentes représentant les différentes versions prétraitées (tokenisation sans suppression des noms de personnages, tokenisation avec suppression des noms de personnages, lemmatisation sans suppression des noms de personnages, lemmatisation avec suppression des noms de personnages). Ainsi, quatre entraînements distincts sont réalisés pour évaluer l'impact des prétraitements sur la classification et pour choisir la meilleure configuration avant d'effectuer l'évaluation sur les données de test.

#### **4.7.3. Classification sur les trois tags principaux**

##### **4.7.3.1. Entraînement et validation croisée**

L'apprentissage des modèles est effectué à l'aide de validation croisée. Plusieurs algorithmes d'apprentissage automatique supervisés sont testés (voir Annexe 4, p.88) :

- Baseline, un classifieur naïf assignant toujours la classe majoritaire
- Naïve Bayes multinomial (MultinomialNB)
- Arbres de décision (CART)
- Régression logistique (LR)
- k plus proches voisins (KNN)
- Forêt d'arbres décisionnels (Random Forest)
- Support Vector Machines linéaire (LinearSVC)

Les résultats sont explicités dans la partie 5.2.1.1 p.50 et montrent que LinearSVC est le meilleur modèle.

##### **4.7.3.2. Première analyse des caractéristiques discriminantes**

Les performances obtenues avec les textes tokenisés sans personnages et avec personnages pour LinearSVC étant extrêmement proches, j'ai extrait les 10 mots les plus discriminants pour chaque tag en utilisant les coefficients de LinearSVC pour tenter d'identifier quel prétraitement retenir pour la suite des analyses.

Extrait de code 12 - Récupération des mots discriminants à l'aide des coefficients de LinearSVC

```
# Entraînement du modèle avec pipeline
model_pipeline = make_pipeline(text_vectorizer, LinearSVC())
model_pipeline.fit(X_dev, y_dev)

# Récupérer le vecteur TF-IDF
feature_names = text_vectorizer.get_feature_names_out()

# Récupérer les coefficients du modèle
```

```

coefs = model_pipeline.named_steps['linearsvc'].coef_

# Associer les coefficients aux mots du vocabulaire
coef_df = pd.DataFrame(coefs, columns=feature_names,
index=model_pipeline.named_steps['linearsvc'].classes_)

# Afficher les 10 mots les plus discriminants pour chaque classe
for tag in coef_df.index:
    print(f"\nTop 10 mots pour {tag} :")
    print(coef_df.loc[tag].nlargest(10))

```

Les résultats de cette comparaison sont présentés et discutés dans la partie 5.2.1.2 p.51 et permettent de déterminer la version de texte retenue pour la suite des analyses.

#### **4.7.3.3. Impact de la suppression de la ponctuation**

J'ai aussi réentraîné les modèles en supprimant la ponctuation des textes pour voir si cette dernière avait un impact sur les résultats.

#### **4.7.3.4. Evaluation sur le corpus de test**

Le test final a été réalisé avec LinearSVC sur les textes tokenisés sans les noms de personnages, qui s'est révélée être la meilleure configuration.

Une recherche des meilleurs hyperparamètres a été effectuée à l'aide de GridSearchCV (voir Annexe 5, p.89), mais les paramètres optimaux qui en sont ressortis ont obtenu les mêmes performances que les valeurs par défaut utilisées lors de l'entraînement en validation croisée précédent.

Les résultats finaux sur le corpus de test sont mesurés avec une matrice de confusion et les métriques de classification (exactitude, précision, rappel, F1-score).

#### **4.7.4. Classification sur les cinq tags**

Je reprends la configuration qui a le mieux marché pour les trois tags principaux, à savoir LinearSVC appliqué aux textes de fanfictions tokenisés sans les noms des personnages.

##### **4.7.4.1. Déséquilibre des classes**

L'ajout des tags *Enemies to Lovers* et *Friends to Lovers* a entraîné un déséquilibre des classes, ceux-ci étant nettement sous-représentés par rapport aux trois premiers. Une classification sans correction aurait favorisé les classes majoritaires, rendant le modèle inefficace sur les classes rares.

##### **4.7.4.2. Stratégies d'équilibrage des classes**

Trois approches ont été testées pour atténuer ce déséquilibre :

- Pondération des classes : augmentation du poids des classes minoritaires dans la fonction de coût du modèle (voir Annexe 6, p.89).
- Rééchantillonnage avec SMOTE : génération de nouvelles instances synthétiques des classes minoritaires (voir Annexe 7, p.90).

- Combinaison des deux méthodes : application conjointe de pondération et de SMOTE (voir Annexe 8, p.91).

#### 4.7.4.3. Évaluation sur les données de test

Je décide d'évaluer sur les trois stratégies de rééquilibrage. En effet, même si la combinaison de pondération des classes et de rééchantillonnage offrent les meilleures performances lors de l'entraînement comme mentionné dans la partie 5.2.2 p. 52, il est possible que cela soit dû à un sur-apprentissage causé par le rééchantillonnage par SMOTE.

### 4.8. Analyse stylométrique avec pydistinto

#### 4.8.1. Préparation des données

Avant d'utiliser Pydistinto, il a été nécessaire d'extraire les textes des fanfictions sous forme de fichiers .txt pour chaque tag. J'ai utilisé la colonne « body\_clean », qui contenait le texte nettoyé sans prétraitement spécifique, car Pydistinto intègre déjà des étapes de nettoyage.

Cependant, en raison de la taille des fichiers générés, l'exécution de Pydistinto entraînait des problèmes de mémoire. Pour éviter cela, j'ai dû limiter la taille des fichiers en ne conservant que les fanfictions dont la longueur était inférieure à un nombre fixé de caractères, avec une valeur différente selon les tags afin d'obtenir des fichiers d'environ 2 000 000 caractères chacun (voir Tableau 5, p.47) :

Extrait de code 13 - Récupération des fichiers .txt nécessaires à l'analyse par pydistinto

```
# Parcourir chaque tag et enregistrer les textes dans des fichiers séparés
for tag in tags:
    # Filtrer les textes pour le tag actuel
    texts = df[df['tag'] == tag]['body_clean']

    # Obtenir la limite de longueur pour le tag actuel
    max_length = length_limits[tag]

    # Compter et enregistrer les textes qui respectent la limite de longueur
    with open(f'data/{safe_tag}.txt', 'w', encoding='utf-8') as file:
        for text in texts:
            if len(text) <= max_length:
                file.write(text + '\n')
                fanfiction_counts[tag] += 1
                total_characters[tag] += len(text)
```

Tableau 5 - Taille des corpus collectés dans des fichiers txt pour chaque tag suite à la limitation sur les tailles des fanfictions

Tag	Nombre de caractères du corpus
Fluff	2 434 469
Angst	2 178 907
Hurt/Comfort	2 023 713
Enemies to lovers	2 460 070
Friends to lovers	2 393 565

#### **4.8.2. Installation et modifications du code**

Pydistinto étant un outil clé en main, j'ai principalement suivi les instructions du dépôt Git pour l'utiliser. J'ai cloné le dépôt et installé les dépendances nécessaires. J'ai créé le fichier de métadonnées (un exemple d'un des fichiers metadata.csv utilisé est disponible en Annexe 9, p.91), en l'adaptant à mes propres données puis le fichier stopwords.txt (que j'ai laissé vide) nécessaires pour l'exécution de pydistinto.

Aussi, le code contenait quelques éléments obsolètes mineurs que j'ai dû corriger :

- Remplacement des occurrences de « get\_features\_names » par « get\_feature\_names\_out » dans le fichier prepare.py.
- Remplacement d'une méthode « append » par « concat » dans le fichier visualize.py :

```
zetadata = zetadata.head(numfeatures).append(zetadata.tail(numfeatures))
```

est remplacé par :

```
zetadata = pd.concat([zetadata.head(numfeatures),  
zetadata.tail(numfeatures)])
```

#### **4.8.3. Paramétrage de pydistinto**

Pydistinto possède un fichier parameters.txt servant à configurer l'analyse. J'ai modifié ce fichier pour l'adapter aux analyses que je souhaitais effectuer (un exemple de fichier parameters.txt utilisé lors de l'analyse est disponible en Annexe 10, p.92).

Concernant la mesure statistique utilisée, j'ai choisi zeta\_sd0, qui est la plus adaptée pour mon objectif d'identifier les spécificités lexicales d'un tag par rapport aux autres. Parmi les différentes variantes de la mesure Zeta, j'ai choisi d'utiliser zeta\_sd0, car elle reste la plus simple à interpréter (Schöch & Schlör, 2018). En effet, bien que certaines améliorations aient été proposées, notamment avec la variante log-transformée (zeta\_sd2), ces dernières peuvent parfois conduire à des effets indésirables, comme une mise en avant excessive des noms propres (noms de personnes, de lieux) dans la liste des mots distinctifs. Ce phénomène est observé notamment dans les résultats présentés en annexe de leur article (Schöch & Schlör, 2018). Or, dans le cadre de cette étude centrée sur l'identification des caractéristiques lexicales des sous-genres, il est important que les noms de personnages ne biaissent pas l'analyse en devenant artificiellement prédominants. Le choix de zeta\_sd0 permet ainsi de mieux cibler les caractéristiques lexicales générales propres aux tags étudiés.

J'ai spécifié que la langue utilisée était le français (pour que pydistinto utilise la bonne version de spaCy).

J'ai laissé une segmentation des textes par tranches de 5 000 caractères.

J'ai choisi d'analyser et obtenir les mots spécifiques de chaque corpus comparé sous forme de lemmes.

#### **4.8.4. Analyses effectuées**

J'ai lancé plusieurs comparaisons :

- Comparaisons de chaque tag en contraste avec les quatre autres pour identifier les spécificités lexicales globales de chaque catégorie qui pourraient être comparé aux mots discriminants obtenus lors de la classification automatique. Cela résulte en cinq analyses :
  1. *Fluff* comparé aux 4 autres tags
  2. *Angst* comparé aux quatre autres tags
  3. *Hurt/Comfort* comparé aux quatre autres tags
  4. *Enemies to lovers* comparé aux quatre autres tags
  5. *Friends to lovers* comparé aux quatre autres
- Comparaisons en un contre un pour observer plus précisément les différences de vocabulaires entre deux tags spécifiques. Cela résulte en 10 analyses :
  1. *Angst* comparé à *Fluff*
  2. *Angst* comparé à *Hurt/Comfort*
  3. *Angst* comparé à *Enemies to lovers*
  4. *Angst* comparé à *Friends to lovers*
  5. *Fluff* comparé à *Hurt/Comfort*
  6. *Fluff* comparé à *Enemies to lovers*
  7. *Fluff* comparé à *Friends to lovers*
  8. *Hurt/Comfort* comparé à *Enemies to lovers*
  9. *Hurt/Comfort* comparé à *Friends to lovers*
  10. *Enemies to lovers* comparé à *Friends to lovers*

À chaque exécution, pydistinto génère des diagrammes illustrant les termes spécifiques à chaque tag, facilitant l'interprétation des différences lexicales entre catégories.

#### **4.9. Conclusion**

Cette méthodologie permet d'analyser les caractéristiques textuelles de cinq tags de fanfictions, en utilisant une approche combinant apprentissage automatique et stylométrie. En testant les modèles sur deux jeux de données, cette étude vise à explorer non seulement les distinctions entre tags, mais aussi les éventuels recoulements, en vérifiant si certains tags se comportent comme des sous-catégories de genres plus larges. Les résultats qui seront présentés dans la section suivante permettront d'approfondir la compréhension des genres en fanfiction et d'identifier des éléments distinctifs au sein de ce corpus littéraire unique.

## 5. Résultats et analyse

### 5.1. Introduction

Cette section vise à examiner les performances des modèles de classification automatique et les résultats obtenus à l'aide de pydistinto pour déceler des tendances lexicales associées aux différents tags. Une attention sera aussi portée à la taille des différentes fanfictions. L'objectif est de déterminer des caractéristiques et thématiques propres à chaque tag et d'observer d'éventuelles différences ou ressemblances entre les différents tags.

Cette section commence par évaluer les performances des modèles de classification automatique et les conclusions qu'elles permettent de tirer sur les différents tags. Ensuite, elle examine les caractéristiques linguistiques de chaque tag et les relations entre ces tags. Enfin une dernière partie s'intéresse aux longueurs des fanfictions de chaque tag.

### 5.2. Performances de la classification automatique

#### 5.2.1. Résultats sur les trois tags principaux

##### 5.2.1.1. Impact des prétraitements et des différents algorithmes

Lors de l'entraînement et validation croisée sur les trois principaux tags, LinearSVC obtient les meilleurs résultats pour tous les différents prétraitements appliqués aux textes de fanfiction et les textes tokenisés obtiennent des résultats légèrement supérieurs aux textes lemmatisés pour quasiment tous les algorithmes sauf KNN et Random Forest, comme on peut le voir sur la Figure 5 et le Tableau 6 ci-dessous.

Figure 5 - Performances (f1-score macro) en validation croisée des algorithmes de classification selon les prétraitements appliqués aux textes

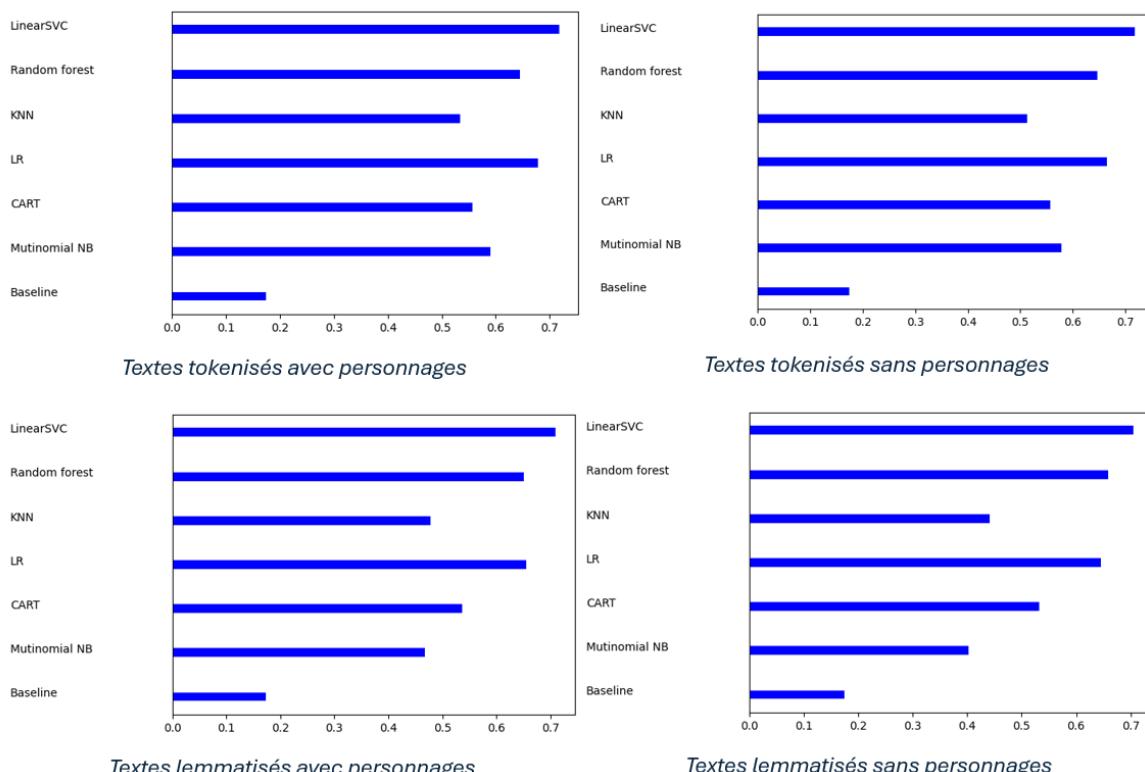


Tableau 6 - F1-scores macro de l'entraînement en validation croisée selon les prétraitements et les algorithmes

Algorithmme	F1-score macro			
	Tokenisé avec personnages	Tokenisé sans personnages	Lemmatisé avec personnages	Lemmatisé sans personnages
Baseline	0.17	0.17	0.17	0.17
Multinomial NB	0.59	0.58	0.48	0.46
CART	0.57	0.56	0.54	0.55
LR	0.68	0.66	0.67	0.66
KNN	0.53	0.51	0.54	0.53
Random Forest	0.64	0.65	0.66	0.66
LinearSVC	0.72	0.72	0.71	0.71

Si LinearSVC fonctionne bien, c'est que les différences entre les tags sont en partie capturables par une séparation linéaire. Cela suppose que certains mots ou groupes de mots sont suffisamment caractéristiques pour distinguer les classes et donc que les tags de fanfictions sont principalement reconnaissables par leur vocabulaire, avec des différences lexicales assez marquées entre certaines catégories.

Le fait que les textes tokenisés permettent d'obtenir de meilleures performances que les textes lemmatisés sous-entend que le choix précis des mots joue un rôle dans la classification et que certaines nuances grammaticales peuvent être utiles pour distinguer les tags entre eux.

### 5.2.1.2. Analyse des mots discriminants

Aussi, après avoir extrait les 10 mots les plus discriminants pour chaque tag, j'ai constaté que dans la version avec personnages, de nombreux noms propres ressortaient parmi les traits distinctifs, en particulier pour le tag *Hurt/Comfort*. Dans ce cas, la majorité des mots discriminants correspondaient à des noms de personnages plutôt qu'à des choix lexicaux liés au style ou aux thématiques. Comme l'objectif de ce mémoire est d'identifier des caractéristiques linguistiques reflétant les tons et thèmes des fanfictions, cette observation a confirmé l'intérêt de conserver la version sans personnages pour les analyses principales.

### 5.2.1.3. Impact de la ponctuation

Bien que le vocabulaire et la forme des mots présents dans les fanfictions de chaque tag semblent être la caractéristique principale permettant de distinguer ces tags, la légère baisse des performances de chaque algorithme lorsqu'on les entraîne sur des données où la ponctuation a été effacée (voir Tableau 7) suggère que la syntaxe peut aussi jouer un rôle dans la distinction des différents tags.

Tableau 7 - Performances en validation croisée des différents algorithmes sur les différentes données sans ponctuation en comparaison avec les performances obtenues sur les données avec ponctuation

Algorithmme	F1-score macro			
	Tokenisé avec personnages	Tokenisé sans personnages	Lemmatisé avec personnages	Lemmatisé sans personnages
Baseline	0.17 (=)	0.17 (=)	0.17 (=)	0.17 (=)
Multinomial NB	0.57 (-0.02)	0.53 (-0.05)	0.47 (-0.01)	0.40 (-0.06)
CART	0.54 (-0.03)	0.54 (-0.02)	0.54 (=)	0.53 (-0.02)
LR	0.63 (-0.05)	0.62 (-0.04)	0.66 (-0.01)	0.65 (-0.01)
KNN	0.42 (-0.11)	0.34 (-0.17)	0.48 (-0.06)	0.44 (-0.09)
Random Forest	0.64 (=)	0.64 (-0.01)	0.65 (-0.01)	0.66 (=)
LinearSVC	0.69 (-0.03)	0.70 (-0.02)	0.71 (=)	0.70 (-0.01)

#### 5.2.1.4. Performances sur les données de test

L'évaluation finale du meilleur modèle (LinearSVC couplé avec les données tokenisées sans les noms des personnages) atteint un f1-score macro de 0.60 sur les trois tags principaux et un score plus faible de 0.45 sur les cinq tags, comme on peut le voir sur le Tableau 8. Ces performances montrent que le modèle peine à capturer des distinctions nettes entre les catégories, ce qui suggère que les algorithmes de classification classiques ont des difficultés à identifier des caractéristiques réellement discriminantes propres à chaque tag.

Tableau 8 - Résultats de la classification automatique sur le corpus de test avec LinearSVC

Algorithmme	Prétraitements du corpus	Précision macro	Rappel macro	F1-score macro	Exactitude
LinearSVC	Tokenisé sans personnages	0.62	0.60	0.60	0.60

#### 5.2.2. Résultats sur les cinq tags

Les résultats en validation croisée de la classification automatique sur les cinq tags ont montré que SMOTE semblait produire des performances trop élevées, suggérant un surajustement. Les résultats avec pondération des classes ont obtenu un F1-score macro assez peu élevé de 0.53. Enfin, la combinaison des deux méthodes a permis d'obtenir un F1-score macro de 0.73.

Lors des tests finaux sur le corpus de test, les trois approches ont donné des résultats similaires, avec une exactitude autour de 50 %, confirmant la difficulté de la tâche.

### 5.3. Analyse des mots discriminants et vocabulaire des tags

Les mots discriminants récupérés à l'aide des coefficients de LinearSVC après la classification ainsi que les listes de vocabulaires obtenus à l'aide de pydistinto pour chaque tag coïncident et confirment plusieurs des hypothèses émises sur les tons employés et thèmes évoqués dans les différents tags.

Pour *Angst*, la violence et la douleur ressortent, à la fois suite à la classification et l'analyse par pydistinto (Figure 6 et Figure 7) avec des mots tels que « sang », « mort », « mourir », « douleur »,

« horreur », « tuer », « cadavre » ou « monstre ». On remarque que ces mots expriment plutôt une douleur physique.

Figure 6 - Résultats pydistinto : Angst comparé aux quatre autres tags

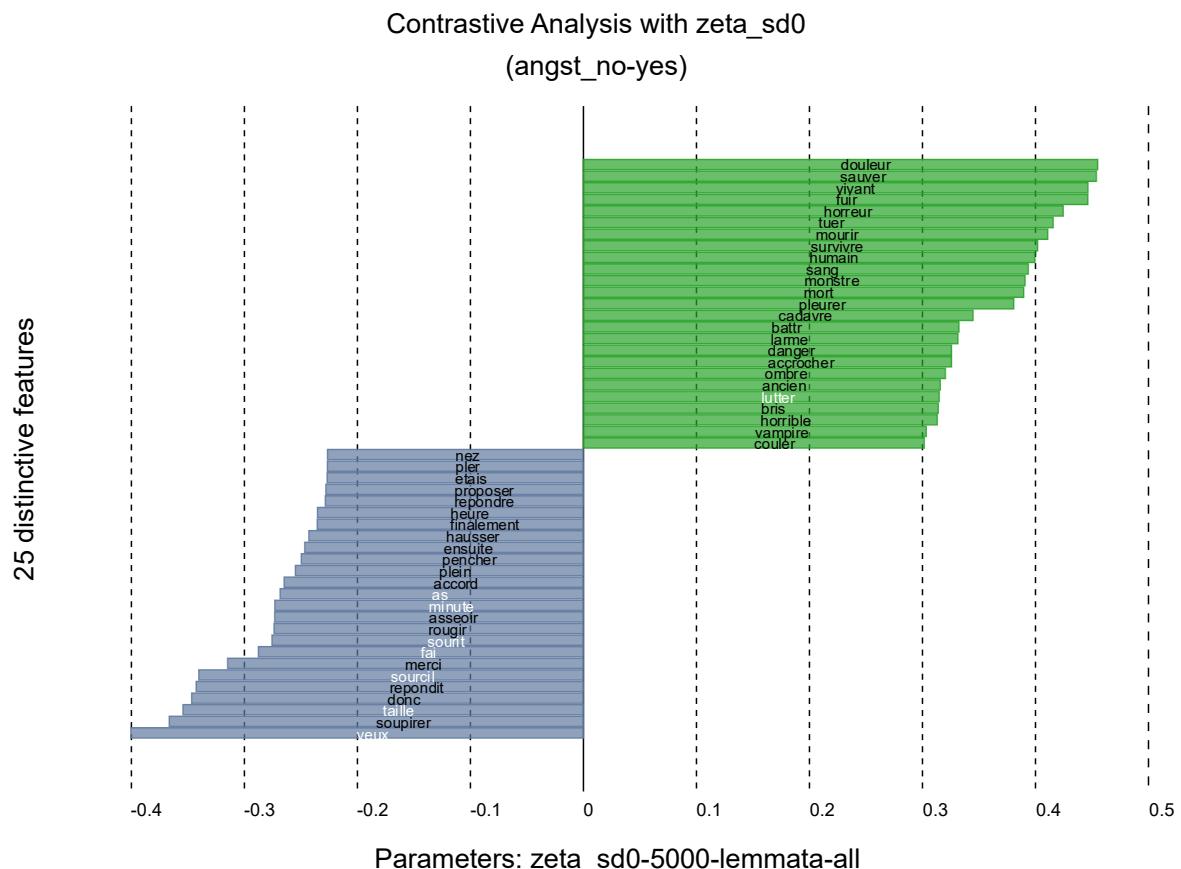
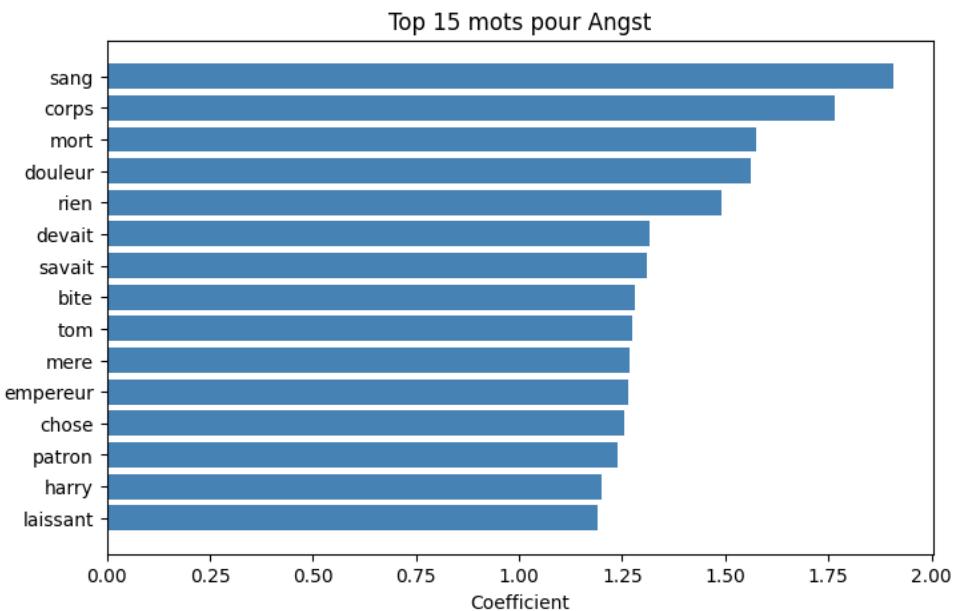


Figure 7 - 15 mots discriminants de *Angst* (classification sur trois tags)



Pour *Fluff* (voir Figure 9 et Figure 8), on retrouve la légèreté et la joie avec des mots tels que « sourire », « rire » ou « heureuse », et l’aspect romantique dans « compagnon », « compagne », « chuchote » ou « embrasser ». On peut aussi remarquer un vocabulaire festif, axé sur les fêtes de Noël avec « noel » ou « neige ». Dans la culture populaire, Noël est souvent associé à un temps heureux, romantique et léger, ce n'est donc pas surprenant de retrouver ce vocabulaire qui suggère que de nombreuses fanfictions *Fluff* doivent se dérouler autour des fêtes de Noël. La présence de tokens non-standards comme « d » ou « -et » parmi les caractéristiques identifiées par LinearSVC est expliquée plus loin dans cette section, p.60.

Figure 9 - Résultats pydistinto : *Fluff* comparé aux quatre autres tags

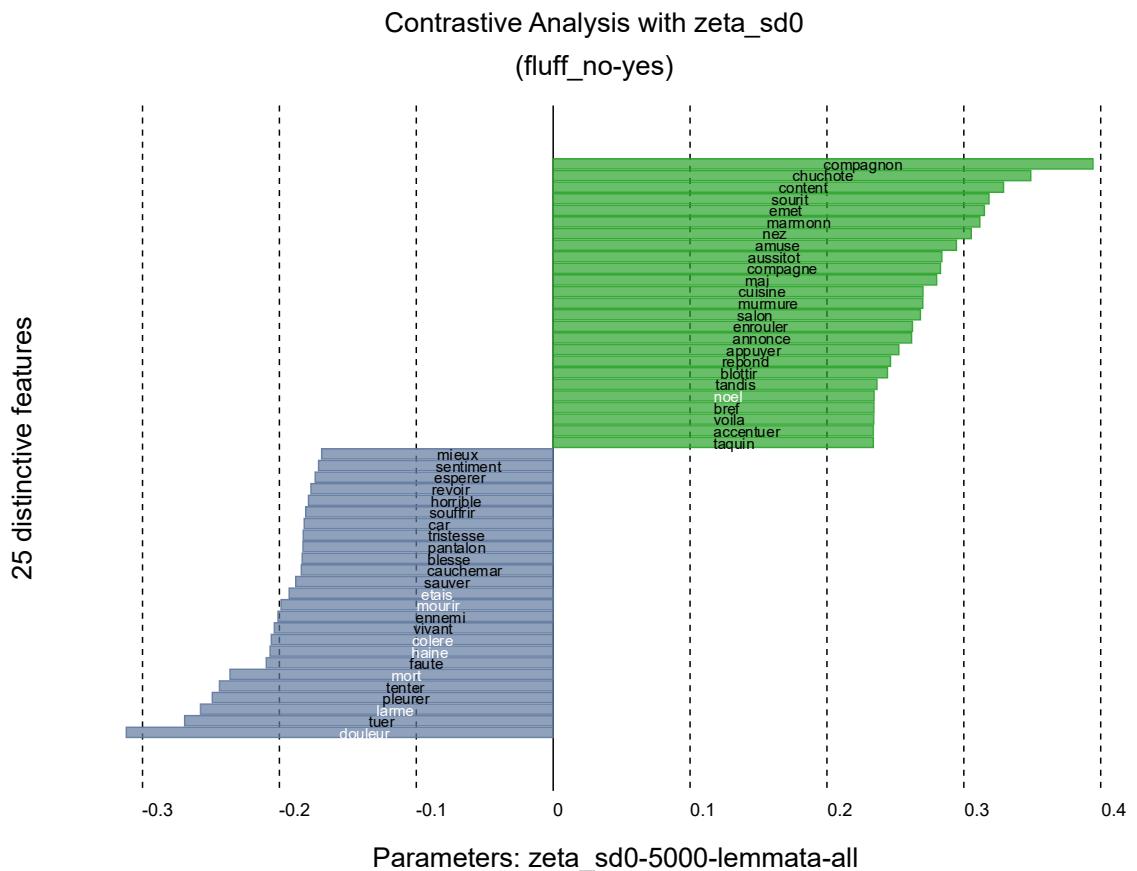
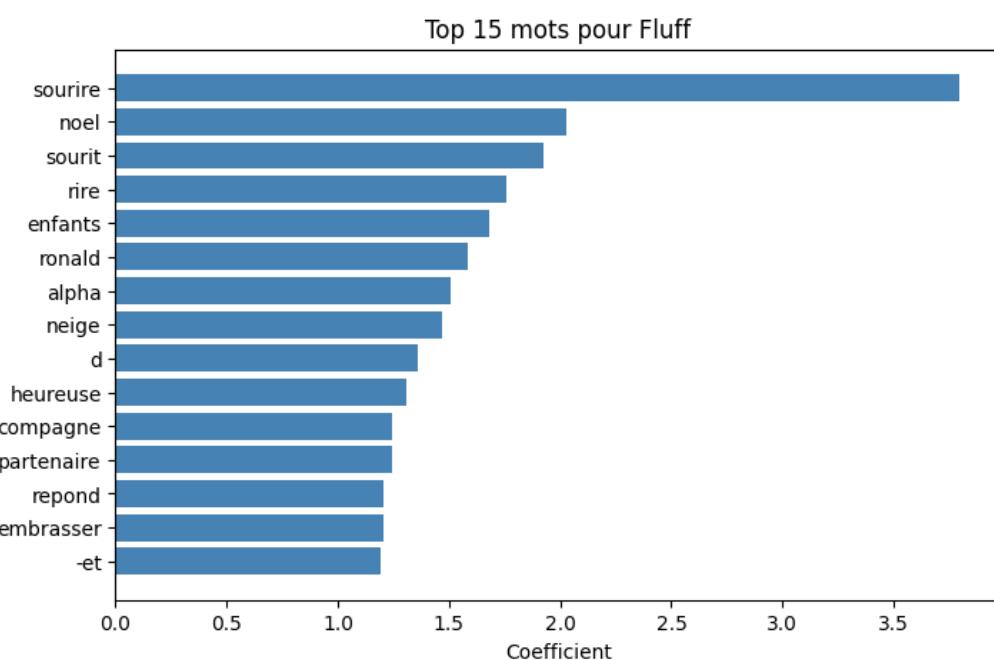


Figure 8 - 15 mots discriminants de *Fluff* (classification sur cinq tags)



Pour *Hurt/Comfort*, l'aspect douloureux de Hurt se retrouve dans des mots tels que « larmes », « démons », « pleurer », « sanglot », « mourir », « triste », qui expriment plutôt une douleur psychologique (voir Figure 10 et Figure 11). Cela diffère notamment de *Angst* (Figure 6, p.53 et Figure 7, p.54) où les mots observés précédemment relevaient plutôt de violences et douleurs physiques, comme en témoignent des termes tels que « sang », « couler », « cadavre » ou « battre ».

L'aspect réconfortant et romantique de *Comfort* s'observe dans les mots « murmura », « supporter », « apaiser », « levre », ou encore « bras » qui peut suggérer des gestes romantiques et réconfortants où un personnage prend un autre personnage dans ses bras par exemple. Le mot « passe » pourrait également évoquer l'idée d'un soulagement émotionnel (un mal ou une douleur qui « passe »). Le terme « soupira » pourrait, selon le contexte, être associé à un personnage qui soupire quelque chose à un autre ou à un soupir de soulagement, en cohérence avec la dynamique de réconfort propre à ce tag.

Figure 10 - Résultats pydistinto : *Hurt/Comfort* comparé aux quatre autres tags

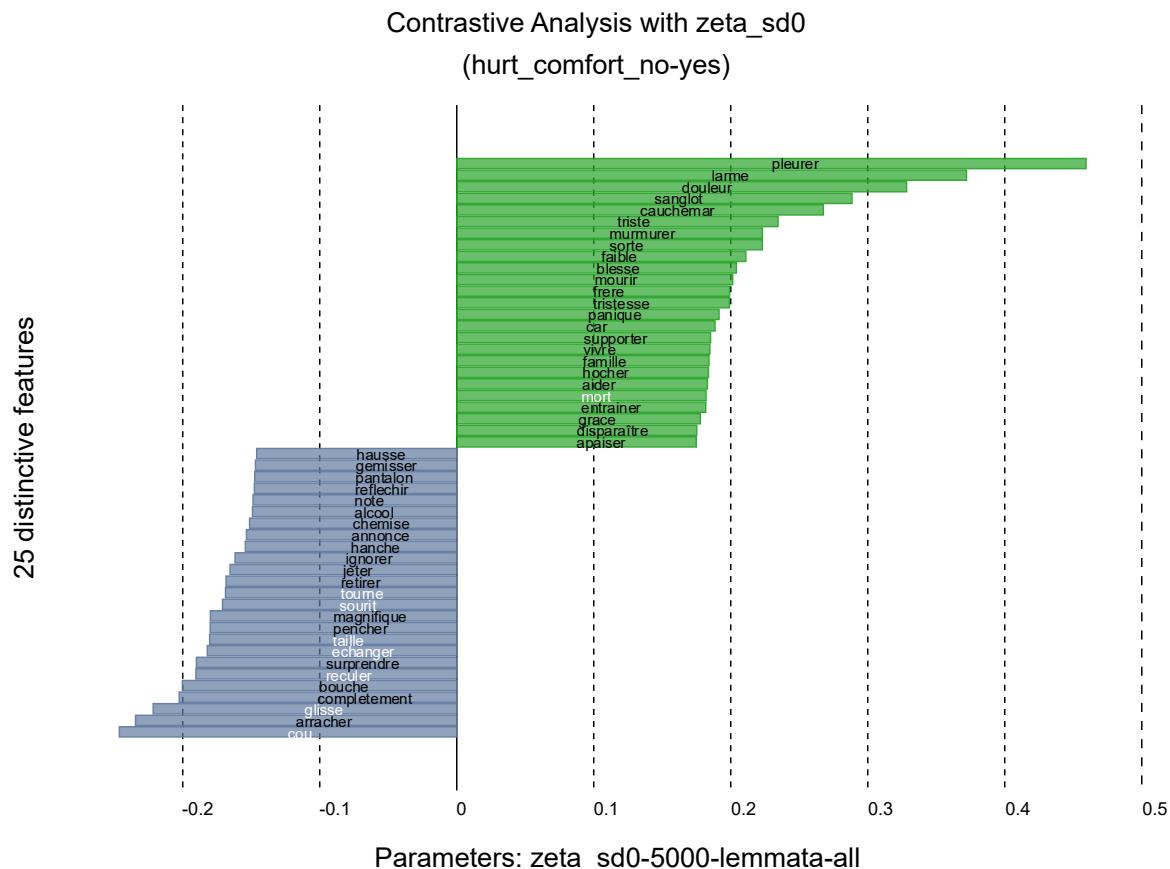
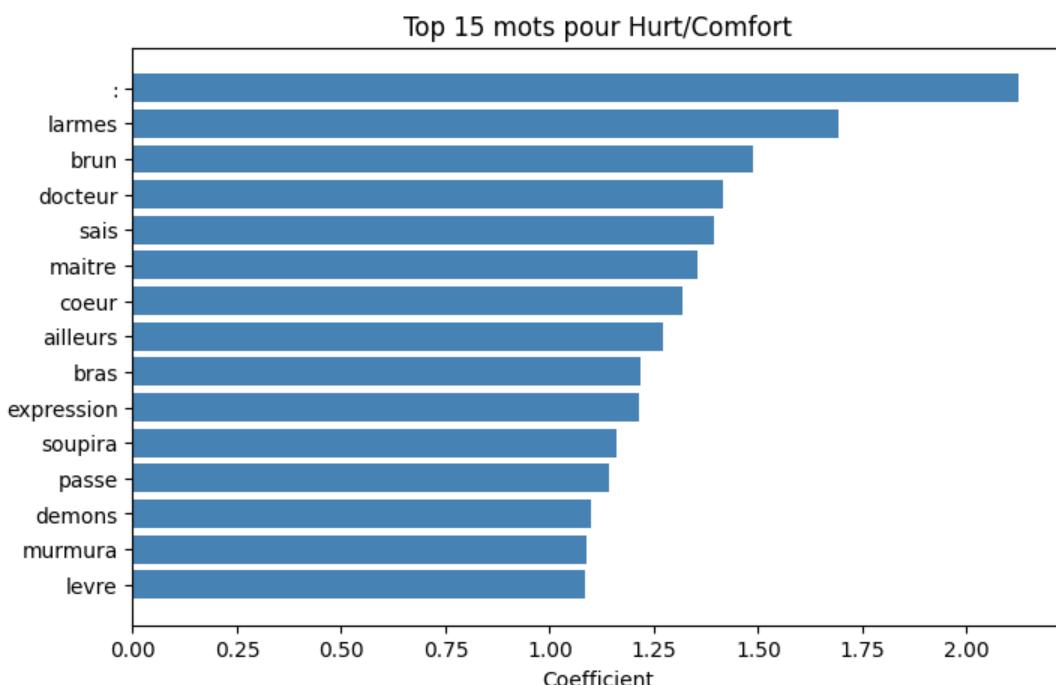


Figure 11 - 15 mots discriminants de *Hurt/Comfort* (classification sur trois tags)



Pour *Friends to lovers* (voir Figure 12 et Figure 13), on retrouve le vocabulaire prédictible de l'amitié avec « amitié », « ami », « amie », puis l'aspect romantique avec « amoureux », « sentiments », « embrassant », « gemit » ou « levres ». Aussi la présence des mots « gêne » et « rougir » traduit que l'évolution d'une relation amicale vers une relation romantique est souvent exprimée par une sorte d'embarras dans ces fanfictions.

Figure 12 - Résultats pydistinto : *Friends to lovers* comparé aux quatre autres tags

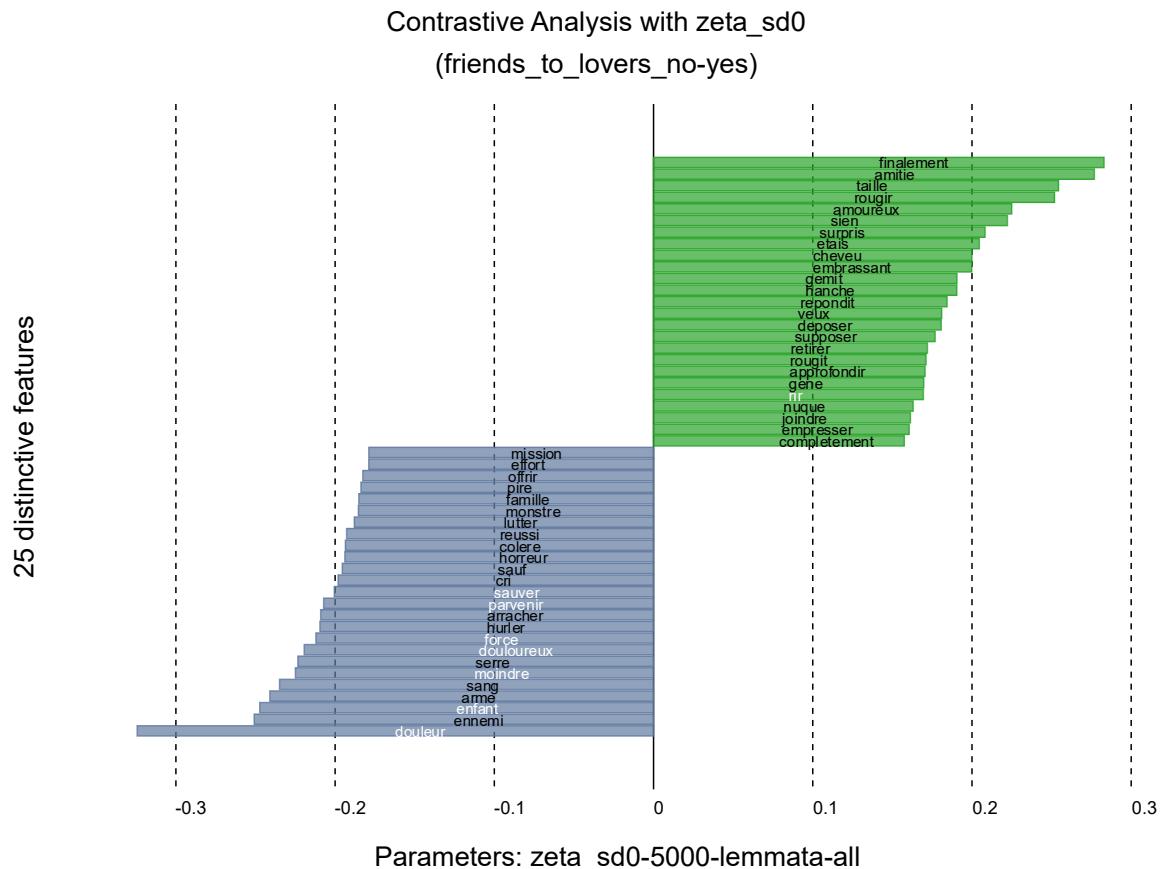
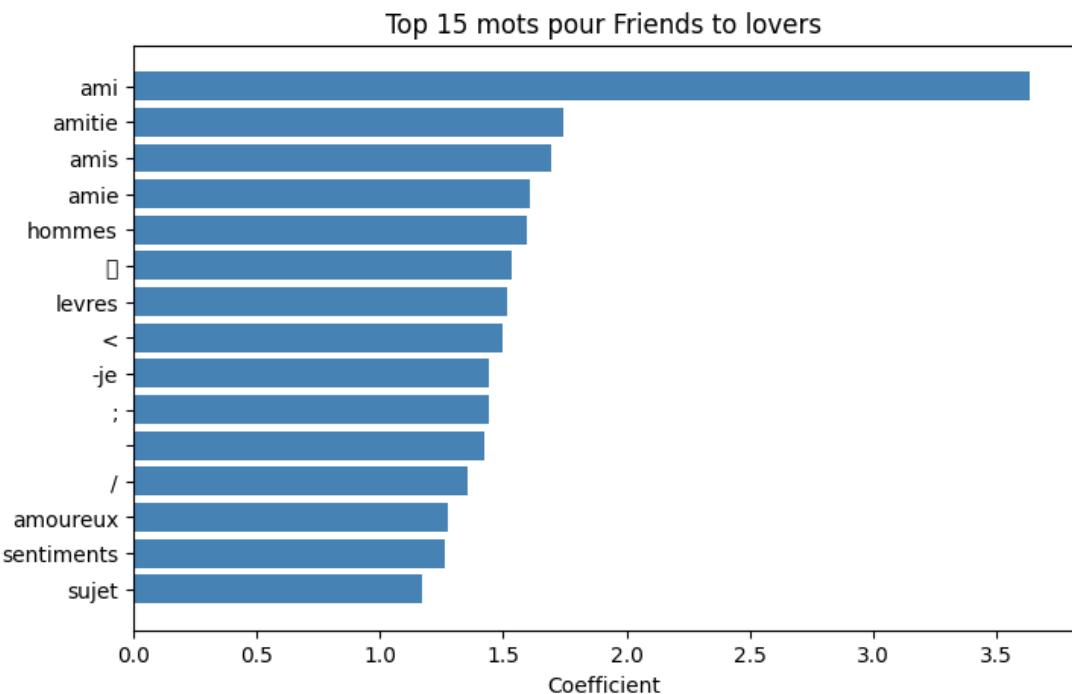


Figure 13 - 15 mots discriminants de *Friends to lovers* (classification sur cinq tags)



Pour *Enemies to lovers* (voir Figure 14 p. 60 et Figure 15 p.61), on retrouve sans surprise un vocabulaire de rivalité avec « ennemi » ou « rival ». Pour le côté *lovers*, il est exprimé avec un vocabulaire cru avec des mots comme « erection », « foute », « sexe » ou « jouir ». On peut sentir une certaine violence dans ce vocabulaire cru et dans l'utilisation d'injures comme « putain », reflétant la rivalité voire haine entre les protagonistes, qui reste sous-jacente quand leur relation prend un tournant romantique.

On peut aussi remarquer un vocabulaire qui se rapporte à l'univers d'*Harry Potter*, qui semble très présent dans ce tag : « serpentard », gryffondor », « potter », « ginny », « malefoy ». Cela suggère que de nombreuses fanfictions *Enemies to lovers* s'inspirent de l'univers de cette saga.

Par ailleurs, l'univers d'*Harry Potter* semble être populaire dans les fanfictions en général, car on retrouve aussi « harry » et « ronald », qui se réfèrent sûrement aux noms de personnages de cette saga, dans les mots discriminants de *Angst* (voir Figure 7, p.54) et *Fluff* (voir Figure 8, p.55). Cette observation illustre également une limite du corpus déjà mentionnée dans l'état de l'art (partie 4.6, p.43) : en raison du nombre restreint de textes, il n'était pas possible d'éviter que certains *fandoms* très populaires, comme Harry Potter, soient répartis dans plusieurs divisions du corpus (corpus d'entraînement et corpus de test). Toutefois, malgré cette influence possible, les caractéristiques lexicales liées aux sous-genres (émotions, dynamiques relationnelles) demeurent dominantes dans les résultats obtenus.

Aussi, la présence de ces noms propres dans les résultats, alors que j'avais appliquée une procédure de suppression des noms de personnages lors du prétraitement, suggère que la procédure mise en place pour récupérer puis supprimer les noms de personnages dans les textes de fanfictions (voir Annexe 3, p.87) a quelques limitations. Cela pourrait s'expliquer par le fait de ne pas avoir tenu compte de certaines notations moins fréquentes utilisées pour renseigner les noms de personnages

lors de l'implémentation de la fonction de récupération et suppression de ces noms, qui compare la liste de personnage fournie par les auteurs parmi les métadonnées avec le texte des fanfictions. Une autre source à cette limitation pourrait venir du fait que les auteurs ne renseignent pas toujours tous les noms des personnages et leurs variantes présentes dans la fanfiction de manière exhaustive. En effet, certains auteurs ne renseignent que les personnages principaux, voire ne renseignent parfois aucun personnage, ce qui pourrait donc expliquer que des noms de personnages secondaires ou autres soient toujours présents dans les textes.

Enfin, on observe dans les caractéristiques discriminantes extraites par l'apprentissage automatique la présence de certains éléments qui pourraient être considérés comme des « parasites » (par exemple, « ~ », « -et », « d », « / », « -je », « ; », etc.). Ces formes peuvent être expliquées par le fait que, dans la version du corpus utilisée pour l'entraînement, la ponctuation a été volontairement conservée, car les performances du modèle étaient légèrement meilleures en conservant la ponctuation. Aussi, certaines constructions linguistiques particulières dans les fanfictions, comme des contractions (ex : « d' ») ou des marques de dialogues (ex : « -je » dans « dis-je » ou « répondis-je ») auraient pu générer des tokens partiels lors de la tokenisation ; et quelques formes mal segmentées ou liées à l'encodage (caractères spéciaux, ponctuation collée) peuvent avoir échappé au nettoyage. Ces cas restent rares dans les résultats et n'empêchent pas une interprétation cohérente des tendances globales. Leur présence peut même parfois refléter des marques stylistiques spécifiques à certains sous-genres.

Figure 14 - Résultats pydistinto : *Enemies to lovers* comparé aux quatre autres tags

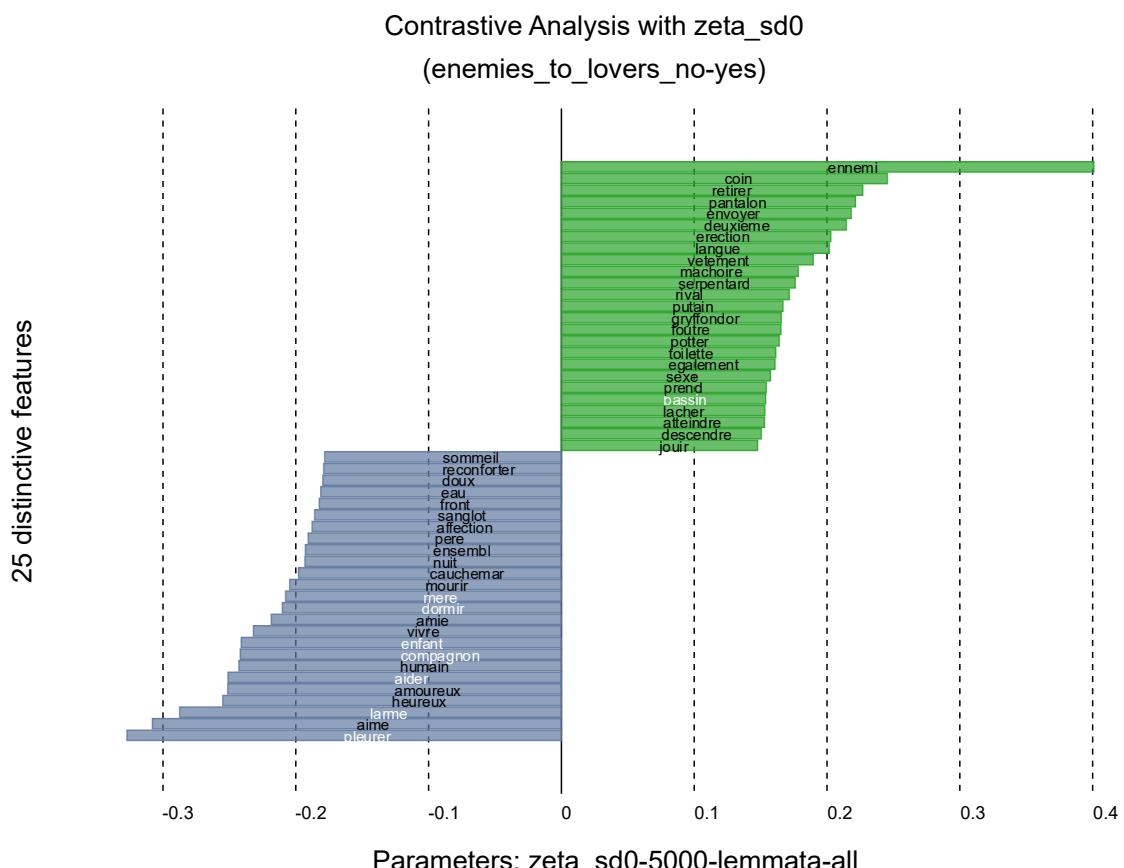
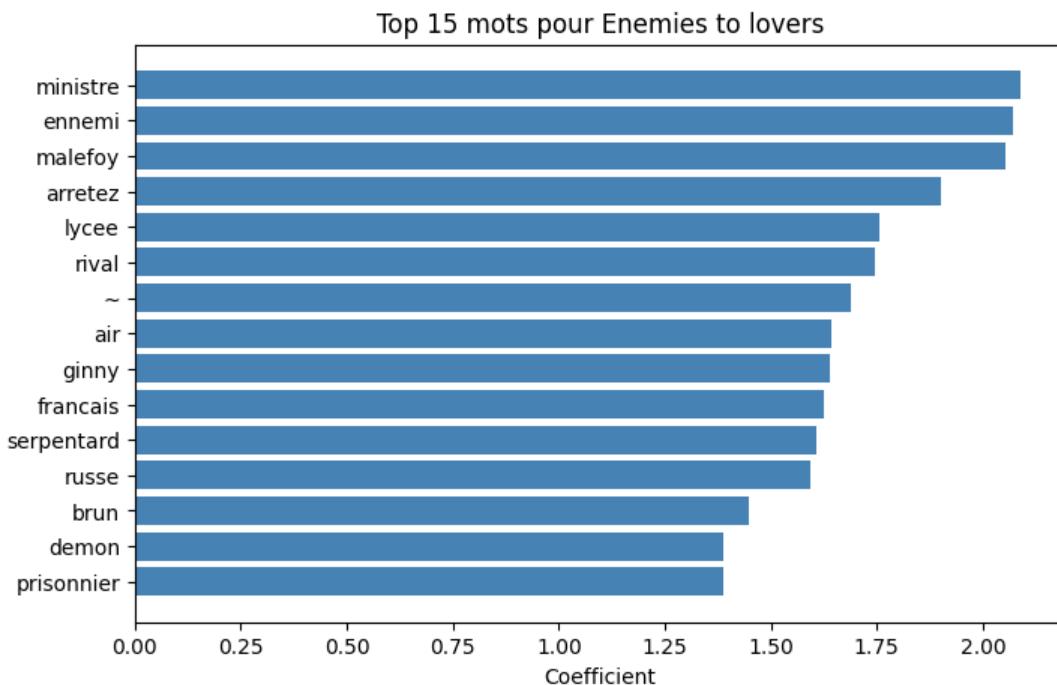


Figure 15 -15 mots discriminants de *Enemies to lovers* (classification sur cinq tags)



Le reste des graphes des mots discriminants de chaque tag, issus des résultats de la classification automatique sont disponibles en Annexe 11, p.93.

### 5.3.2. Similarités et différences entre tags

#### 5.3.2.1. Analyse des matrices de confusion

En regardant la matrice de confusion issue de la classification automatique sur les trois tags principaux (Figure 16, p.62), on peut voir que les fanfictions *Hurt/Comfort* sont plutôt bien classées avec assez peu de fanfictions mal classées en tant que *Fluff* ou *Angst*. Cela suggère que ce tag possède des caractéristiques spécifiques qui le distingue des autres tags.

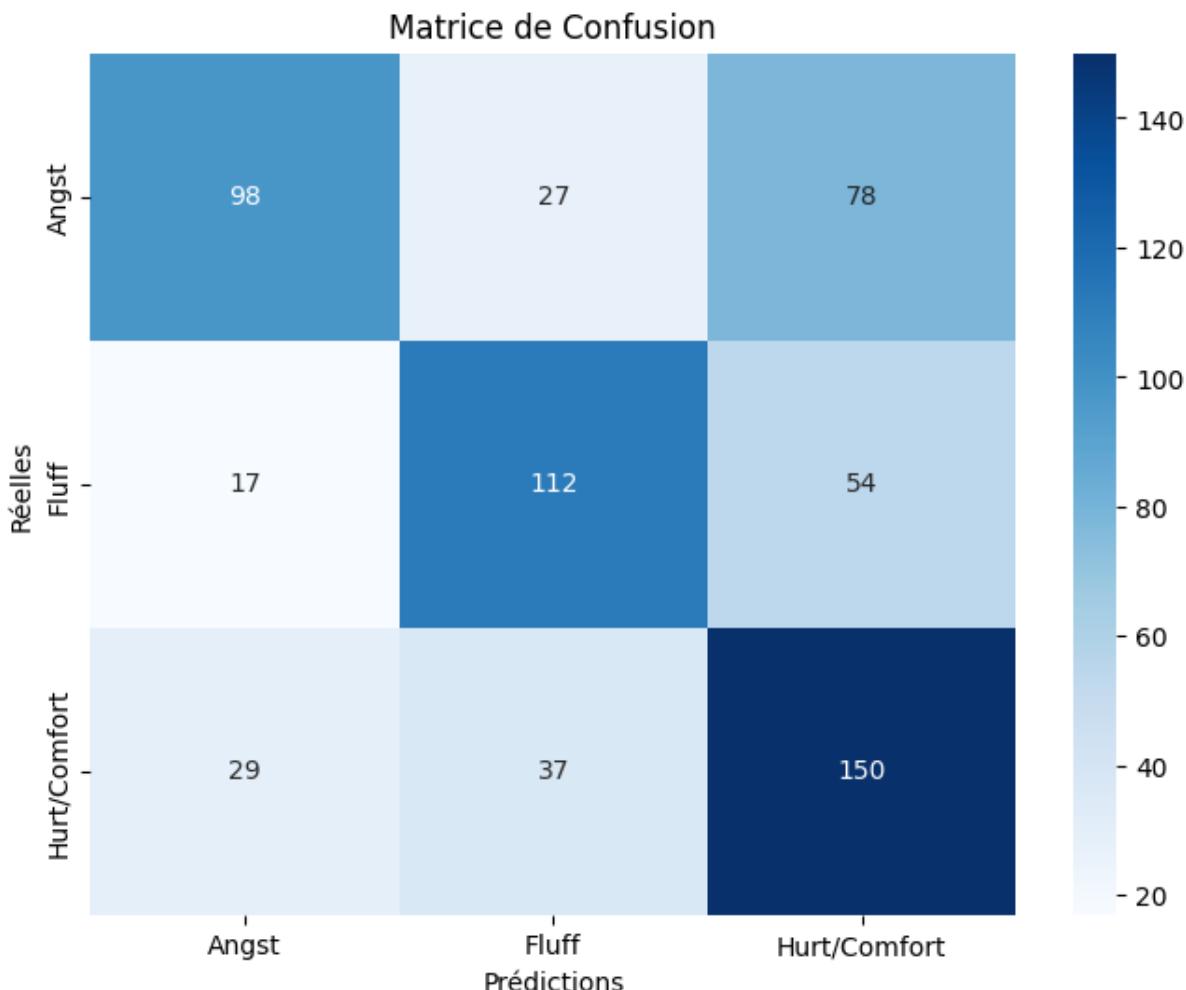
En revanche, une grande partie des fanfictions *Angst* et *Fluff* sont mal classées en tant que *Hurt/Comfort*. Cela laisse penser que *Hurt/Comfort* comporte des caractéristiques qu'on retrouve dans *Fluff* et des caractéristiques qu'on retrouve dans *Angst*, ce qui conforte l'idée que *Hurt/Comfort* pourrait se situer à mi-chemin entre les deux autres tags.

Aussi, le fait que ce soit des fanfictions *Angst* et *Fluff* qui sont mal prédites en tant que *Hurt/Comfort* et non l'inverse sous-entend que *Fluff* et *Angst* possèdent des caractéristiques plus générales qui ne leur permettent pas de se distinguer suffisamment de *Hurt/Comfort*. Cela suggère que *Fluff* et *Angst* sont des tags plutôt généraux qui regroupent respectivement des histoires légères et romantiques en tout genre et des histoires violentes en tout genre. Quant aux fanfictions *Hurt/Comfort* elles représentent déjà un autre niveau de classification, elles sont définies plus clairement : une histoire avec un personnage qui souffre et un qui le console.

Ainsi, dans la hiérarchisation des tags, on peut dire que *Angst* et *Fluff* sont des tags généraux et que *Hurt/Comfort* est un sous-tag. Aussi, à l'intérieur de *Angst* et *Fluff* il peut y avoir des fanfictions *Hurt/Comfort*.

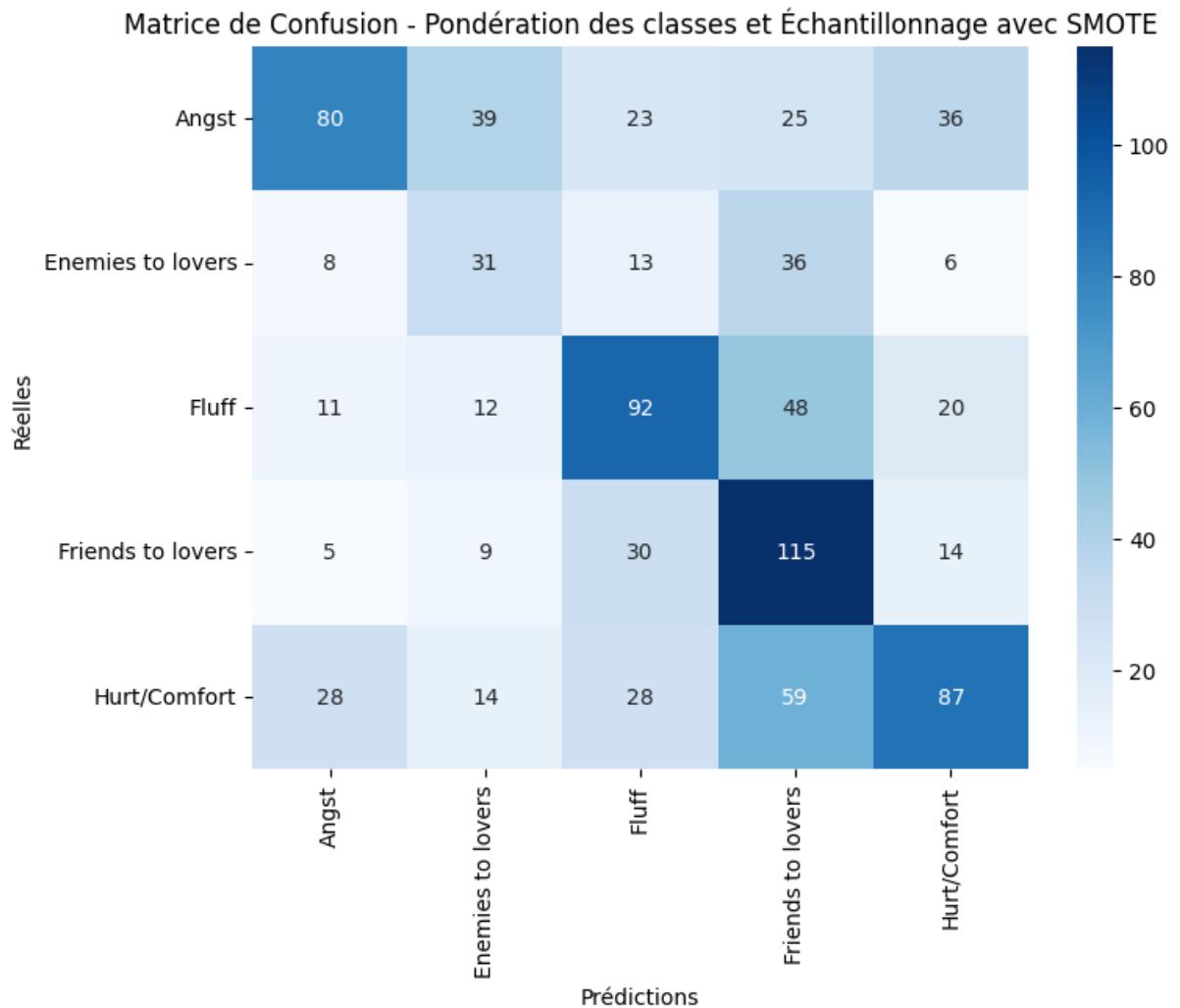
Enfin, il y a assez peu de confusion entre *Angst* et *Fluff*. Cela confirme que ces deux tags sont à l'opposé l'un de l'autre.

Figure 16 - Matrice de confusion de la classification automatique par LinearSVC sur trois tags



Si on regarde la matrice de confusion issue de la classification automatique sur les cinq tags (Figure 17, p.63), on peut voir que les fanfictions *Angst* sont parfois mal prédites et confondues avec les quatre autres tags, même si les scores de confusion les plus élevés sont vis-à-vis de *Enemies to lovers* et de *Hurt/Comfort*. Cela n'est pas étonnant, car ce sont les trois tags avec des thèmes et vocabulaires violents ou exprimants la douleur. Ici encore, ce sont les fanfictions *Angst* qui ont plus tendance à être mal identifiées en tant que *Hurt/Comfort* ou *Enemies to lovers* plutôt que l'inverse, ce qui suggère que *Angst* est un tag plus général alors que *Hurt/Comfort* et *Enemies to lovers* sont des sous-tags plus définis. On peut même en conclure qu'à l'intérieur des fanfictions *Angst*, il y a des fanfictions *Enemies to lovers* et des fanfictions *Hurt/Comfort*.

Figure 17 - Matrice de confusion de la classification automatique par LinearSVC sur les cinq tags (avec pondération et rééchantillonnage)



Concernant *Enemies to lovers*, les résultats ne sont pas vraiment concluant, sûrement que le nombre trop faible de fanfiction de ce tag n'a pas permis à l'algorithme de le distinguer clairement des autres tags. Cependant, on peut voir qu'une majorité des fanfictions *Enemies to lovers* sont mal prédites en tant que *Friends to lovers*. Cela peut être dû au fait que ces deux tags reposent sur une formule similaire où la relation entre deux personnages évolue progressivement vers une relation romantique.

Par rapport à *Fluff*, on peut voir qu'un nombre assez important des fanfictions de ce tag sont confondues avec *Friends to lovers*. Ces deux tags partagent des caractéristiques communes, notamment des thèmes et vocabulaires légers et romantiques. Cependant, la proportion de fanfictions *Friends to lovers* qui sont mal identifiées en tant que *Fluff* est bien inférieure à l'inverse, ce qui veut dire que *Fluff* est un tag plus général tandis que *Friends to lovers* est un sous-tag plus défini. De plus, on peut supposer qu'au sein des fanfictions *Fluff*, il y a des fanfictions *Friends to lovers*.

Quand on observe les confusions au niveau du tag *Hurt/Comfort*, on voit que ces fanfictions sont souvent mal prédites en tant que *Friends to lovers*. A l'inverse, *Friends to lovers* est assez peu

confondu avec *Hurt/Comfort*, ainsi on peut en déduire que *Hurt/Comfort* est un tag plus général que *Friends to lovers* et qu'à l'intérieur de *Hurt/Comfort*, il y a des fanfictions *Friends to lovers*.

En conclusion, on peut dire que *Angst* et *Fluff* sont deux tags généraux qui regroupent des fanfictions sombres et violentes d'un côté et des fanfictions légères et romantiques de l'autre. *Hurt/Comfort* est un sous-tag qui recoupe des aspects de *Angst* et de *Fluff*. *Friends to lovers* est un sous-tag encore plus défini que *Hurt/Comfort*, et contient des fanfictions qu'on peut retrouver à l'intérieur de *Fluff* et de *Hurt/Comfort*. Concernant la hiérarchie des tags, on peut mettre *Angst* et *Fluff* à un premier niveau général de classification, *Hurt/Comfort* à un second niveau un peu plus précis et *Friends to lovers* à un troisième niveau encore plus défini. Pour *Enemies to lovers*, les résultats ne sont pas assez flagrants pour déduire sa place dans ces niveaux de classification.

### 5.3.2.2. Analyse des résultats de pydistinto en un contre un

Concernant *Angst*, les résultats des analyses tag contre tag de pydistinto montrent que c'est le tag le plus violent. En effet, peu importe avec quel tag on le compare, un vocabulaire de la violence et souffrance physique ressort du côté *Angst* : « sang », « douleur », « mort », « cadavre », « tuer ». Cette souffrance semble être la conséquence d'un milieu hostile. En effet, on peut remarquer plusieurs mots comme « danger », « effondrer », « monstre » ou « ombre » qui indiquent que les fanfictions *Angst* prennent pour décor des environnements dangereux qui imposent aux personnages des actions physiques : « fuir », « lutter », « accrocher », « saut ». L'enjeu des fanfictions *Angst* semble être la survie ou non des personnages : « survivre », « vivre », « mourir », « sauf », « sauver » (voir Figure 18, Figure 19, Figure 20, Figure 21, p.65-67).

La comparaison avec *Hurt/Comfort* a permis de mettre en avant un point intéressant des fanfictions *Angst*. Au-delà de mettre en scène un environnement hostile, les fanfictions *Angst* semblent plus précisément s'inscrire dans des univers s'inspirant de l'horreur et du surnaturel. En effet, le mot « horreur » apparaît dans les vocabulaires de *Angst* quand on le compare avec n'importe quel autre des quatre tags, mais dans les résultats de sa comparaison avec *Hurt/Comfort* (voir Figure 21, p.67), ce mot est accompagné des mots « vampire » et « croc » qui rappellent des éléments du surnaturel, ainsi que des mots « Wesker », « Chris » et « Jonathan » qui sont des personnages de *Resident Evil*, des jeux vidéo d'horreur impliquant des éléments surnaturels.

Aussi, quand on compare *Angst* aux autres tags, notamment *Enemies to lovers* (voir Figure 19, p.66), *Friends to lovers* (voir Figure 20, p.66) ou encore *Hurt/Comfort* (voir Figure 21, p.67), on peut voir que cela permet de faire ressortir chez les autres tags des mots de dialogue et interactions entre personnages (« merci », « repondit », « lancer »). Cela sous-entend que ce genre de mots est bien moins présent dans *Angst*. On peut en conclure que les fanfictions *Angst* sont plus centrées sur l'intrigue et l'action que sur les relations entre les protagonistes.

Figure 18 - Résultats pydistinto : *Angst* (droite) comparé à *Fluff* (gauche)

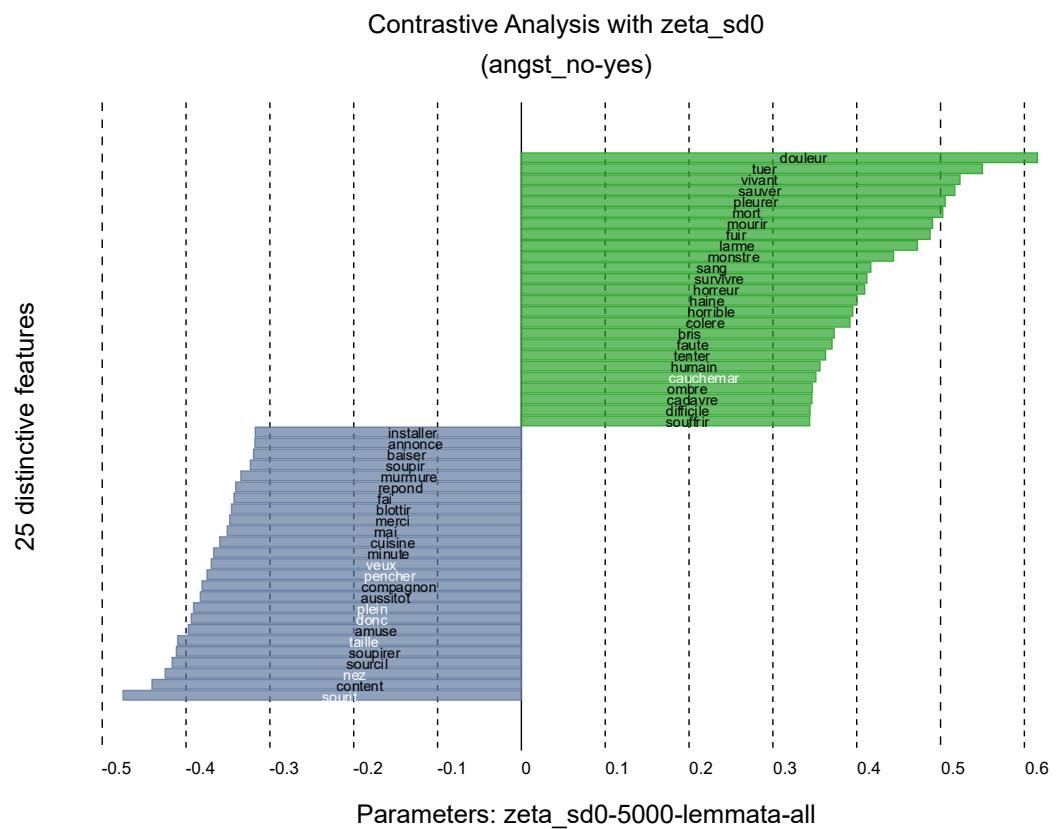


Figure 19 - Résultats pydistinto : *Angst* (droite) comparé à *Enemies to lovers* (gauche)

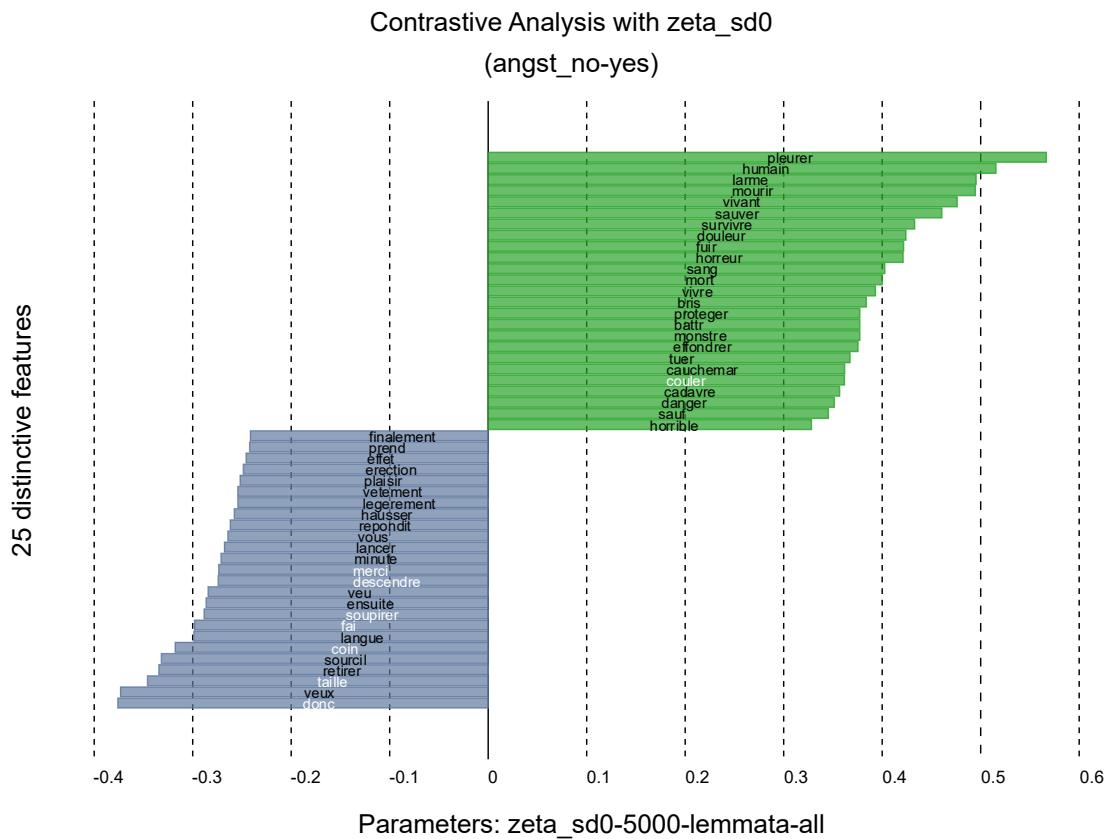


Figure 20 - Résultats pydistinto : *Angst* (droite) comparé à *Friends to lovers* (gauche)

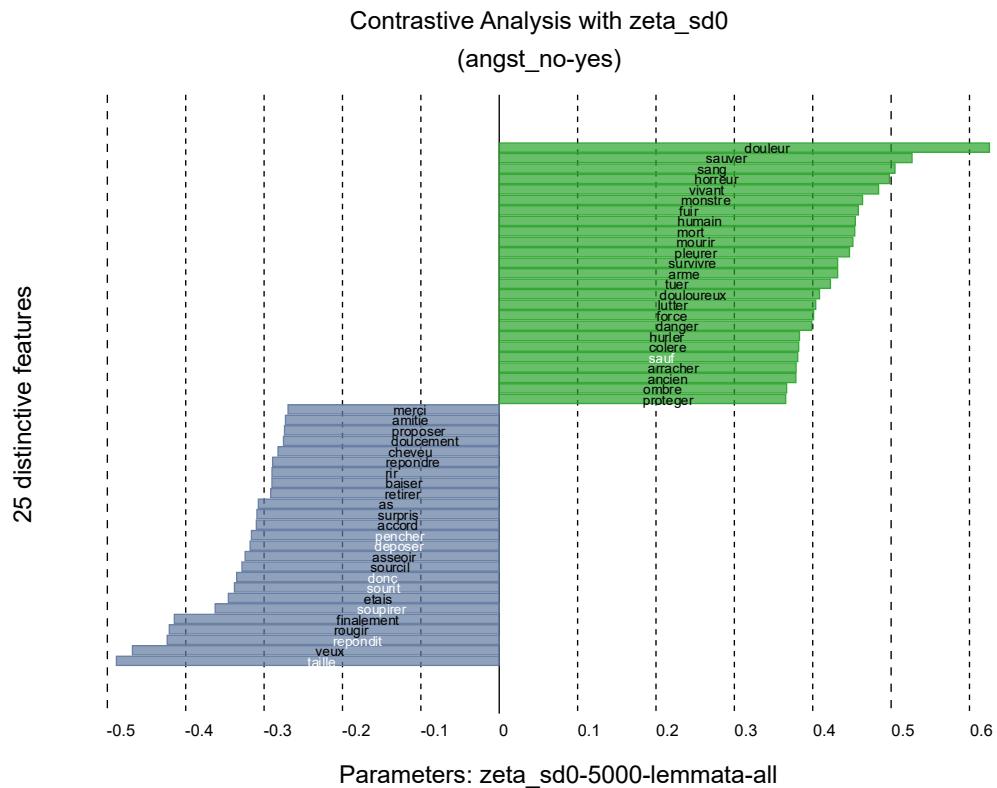
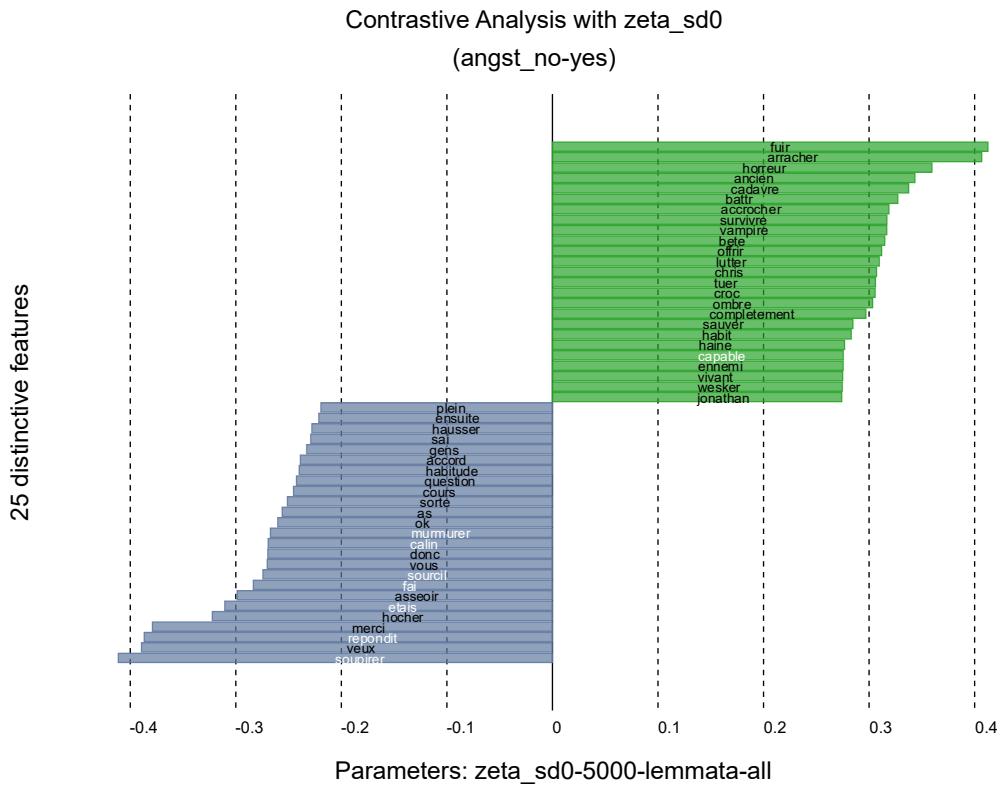


Figure 21 - Résultats pydistinto : *Angst* (droite) comparé à *Hurt/Comfort* (gauche)



Par rapport à *Enemies to lovers*, on remarque que c'est un tag de relation. L'enjeu de ces fanfictions réside dans la relation entre les protagonistes. Cela implique de nombreuses interactions verbales qui sont traduites dans des verbes tels que « lancer » ou « repondit » ou bien des mots de l'oral et du dialogue tels que « merci » ou encore « vous » qui sont mis en avant dans la comparaison avec *Angst* (voir Figure 19, p.66).

Ce tag met en scène des relations de tensions sexuelles entre les personnages. En effet, la tension se traduit dans des relations conflictuelles : « ennemi », « detester », « rival », « adversaire », « provoquer », « stupide ». L'aspect sexuel est lui très présent et ressort lors de chaque comparaison avec les autres tags, il est sensuel (« plaisir », « soupirer »), charnel (« langue », « bouche », « retirer », « pantalon », « vetement ») et exprimé dans des termes crus et explicites (« erection », « sexe », « jouir »). Cette tension sexuelle est exacerbée par un ton intense, les personnages ressentent des sensations et émotions puissantes : « colere », « haine », « putain », « jouir » (voir Figure 19, p.66, Figure 20, p.68, Figure 23, p.69 et Figure 24, p.69).

Figure 22 - Résultats pydistinto : *Enemies to lovers* (droite) comparé à *Friends to lovers* (gauche)

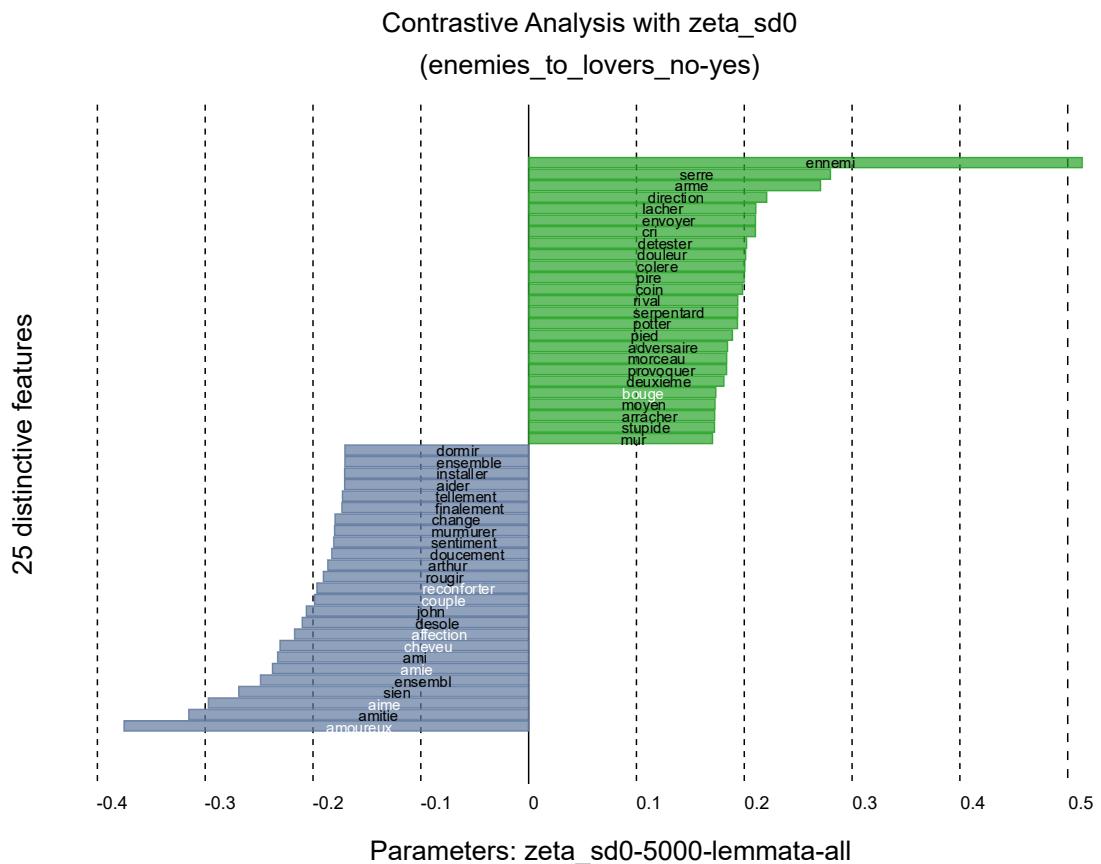


Figure 23 - Résultats pydistinto : *Fluff* (droite) comparé à *Enemies to lovers* (gauche)

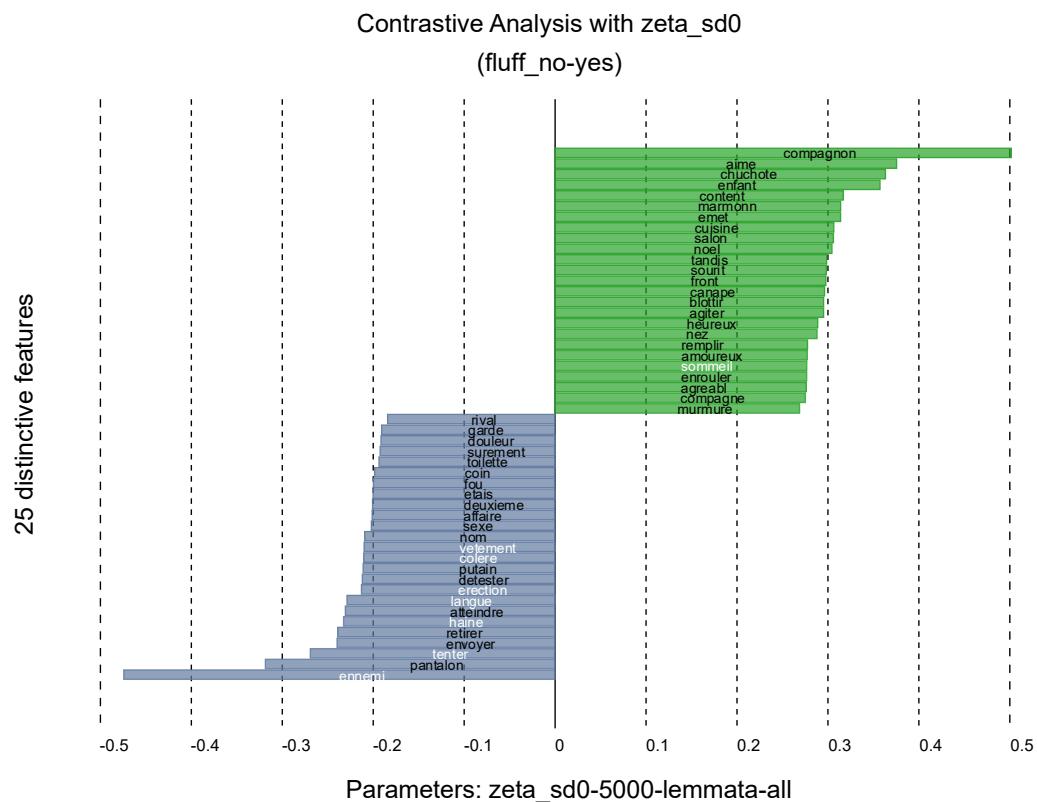
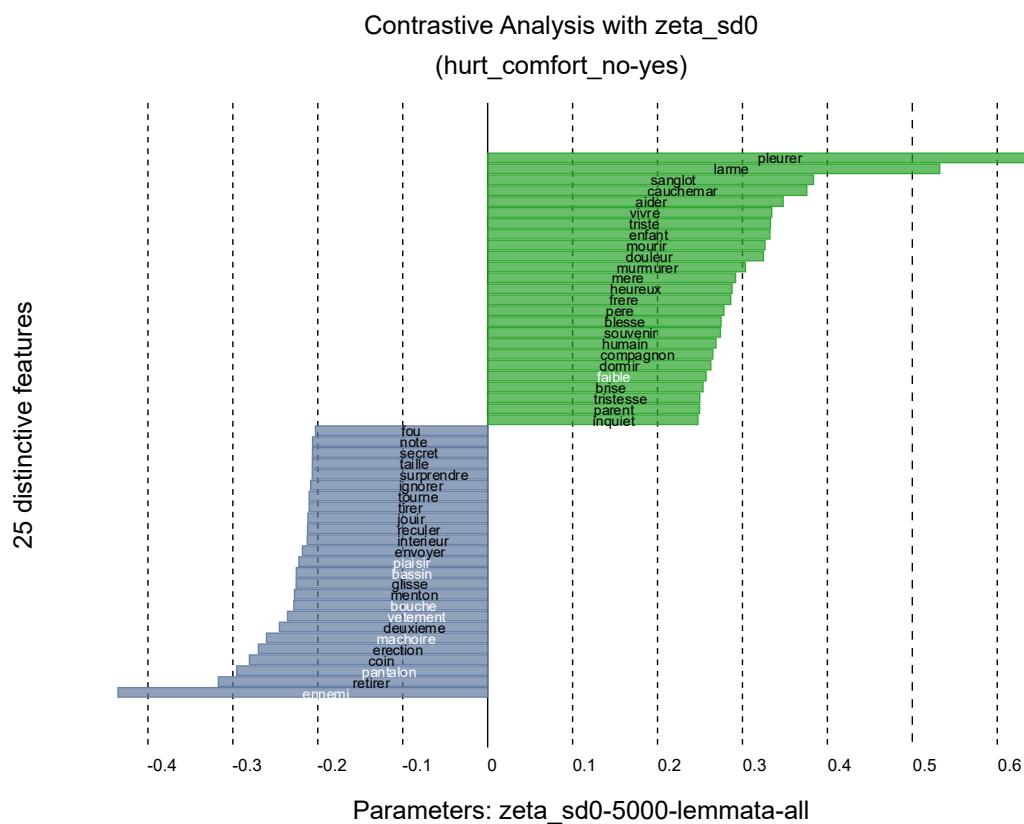


Figure 24 - Résultats pydistinto : *Hurt/Comfort* (droite) comparé à *Enemies to lovers* (gauche)



Les résultats de pydistinto montrent que *Friends to lovers* est aussi un tag de relation. Comme pour *Enemies to lovers*, on retrouve plusieurs mots impliquant des dialogues et interactions comme « merci », « repondit », « proposer » dans la comparaison avec *Angst* (voir Figure 20, p.66).

Comme dans *Enemies to lovers*, les fanfictions *Friends to lovers* mettent en scène une évolution vers une relation romantique qui implique des relations sexuelles entre les personnages. Cependant les scènes intimes sont décrites de manière moins crue et plus douce : « desir », « gemit », « embrasser », « plaisir », « doucement », « amoureux », « murmurer ». Cela est peut-être dû au fait que le point de départ de la relation n'est pas la haine ni le conflit mais l'amour et l'amitié : « amitié », « ami », « amie », « aime », « sentiment », « affection ». Et contrairement à *Enemies to lovers*, l'attraction des personnage n'est pas consommée de manière intense et violente, mais avec de l'hésitation et de l'appréhension : « surpris », « fievreusement », « tenter », « reculer », « rougir », « accord », « arreta » (voir Figure 20, p.66, Figure 22, p.68, Figure 25, p.70 et Figure 26, p.71).

Figure 25 - Résultats pydistinto : *Fluff* (droite) comparé à *Friends to lovers* (gauche)

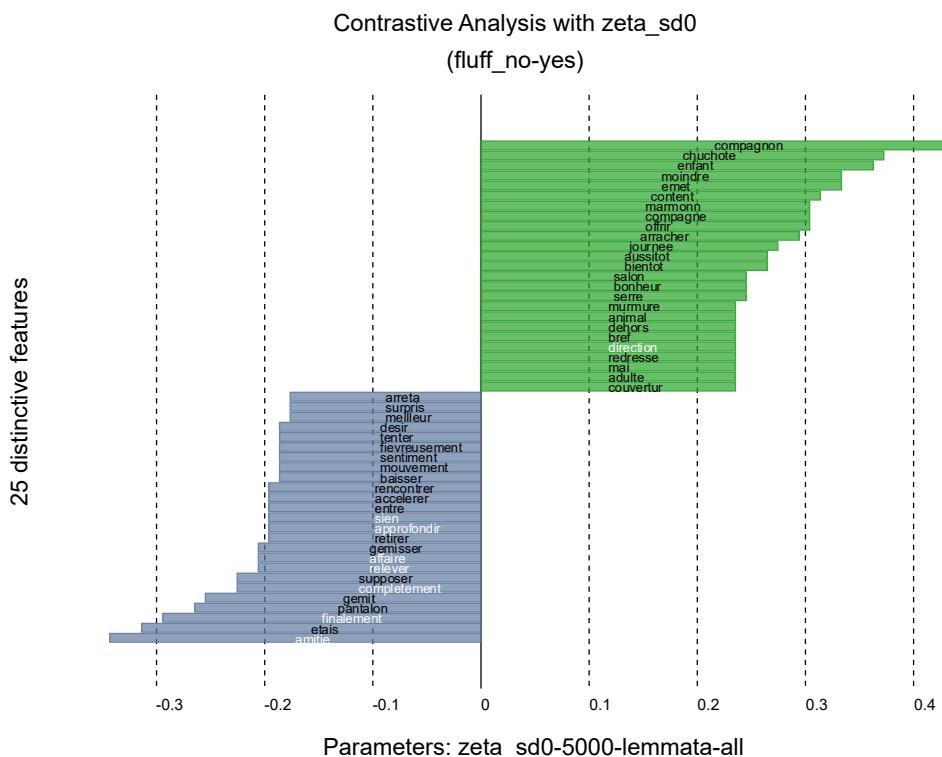
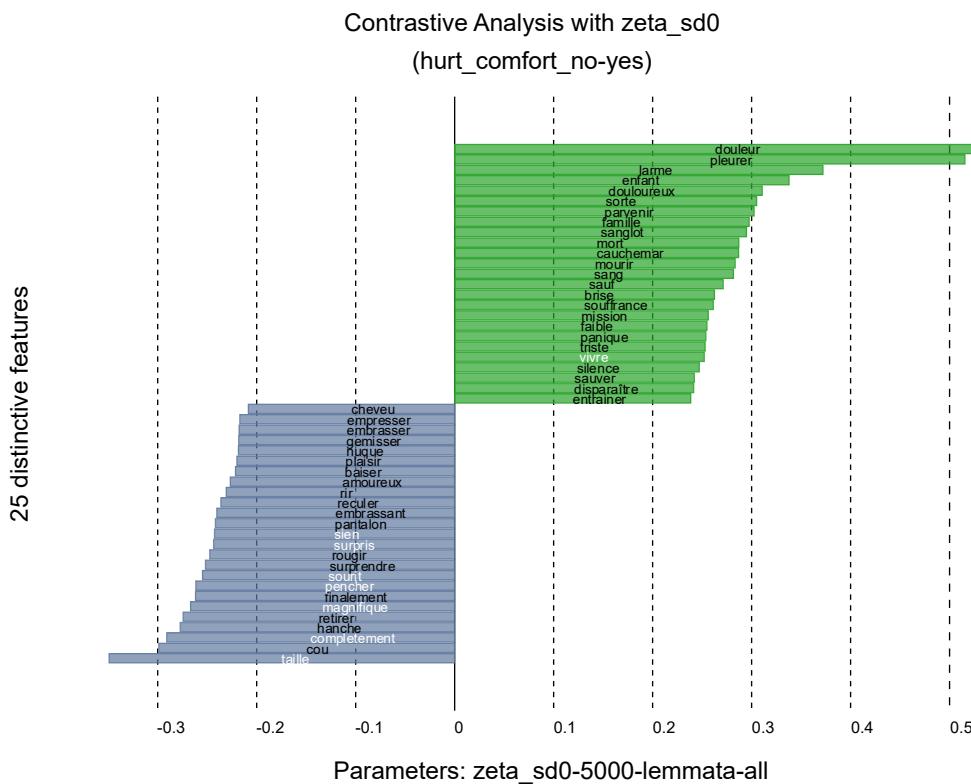


Figure 26 - Résultats pydistinto : *Hurt/Comfort* (droite) comparé à *Friends to lovers* (gauche)

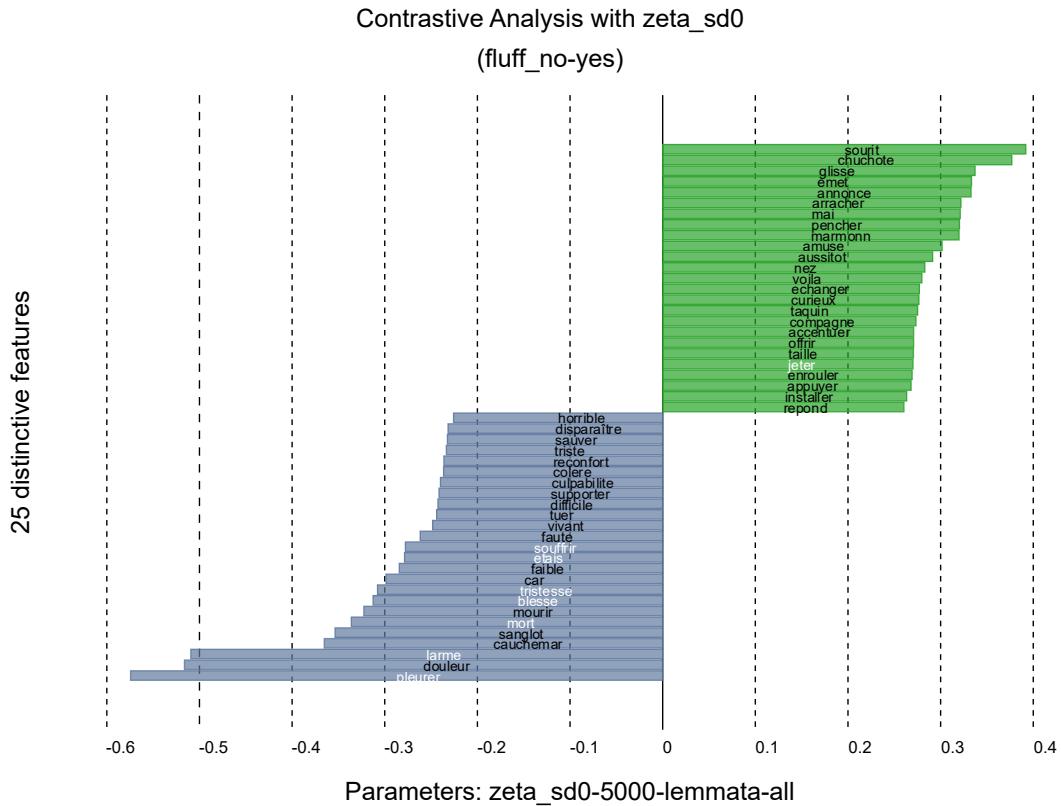


*Hurt/Comfort* est aussi un tag de relation. De nouveau on le voit quand on compare ce tag à *Angst* avec des mots comme « répondit », « merci » ou « vous » (voir Figure 21, p.67).

Les relations qu'entretiennent les personnages s'expriment au travers de gestes rassurants et réconfortants : « aider », « murmurer », « sauver », « reconfort », « supporter », « calin ». Ces gestes sont là pour apaiser des personnages qui souffrent psychologiquement : « pleurer », « douleur », « larme », « cauchemar », « sanglot », « tristesse », « souffrir », « culpabilité », « inquiet », « panique » (voir Figure 21, p.67, Figure 24, p.69, Figure 26, p.71 et Figure 27, p.72). Néanmoins, on ne remarque pas de vocabulaire qui sous-entend des relations sexuelles comme dans *Enemies to lovers* et *Friends to lovers*, ce qui veut dire que la finalité de *Hurt/Comfort* n'est pas forcément une romance.

Aussi, contrairement à *Enemies to lovers* et *Friends to lovers* qui ne mettent en scène supposément que des relations romantiques, *Hurt/Comfort* peut aussi impliquer des relations familiales reflétées dans des mots comme « frère », « mère », « enfant », « parent », « père », « famille » qui apparaissent quand on compare ce tag à *Enemies to lovers* (voir Figure 24, p.69) et *Friends to lovers* (voir Figure 26, p.71) justement.

Figure 27 - Résultats pydistinto : *Fluff* (droite) comparé à *Hurt/Comfort* (gauche)



Enfin, ce qui caractérise *Fluff*, c'est un ton léger et joyeux : « sourit », « amuse », « content », « bonheur », « heureux ». Ce tag semble présenter des scènes domestiques dans des lieux de vie quotidien comme la « cuisine », le « salon », le « canapé » ou « dehors ». Les actions mises en scène peuvent être aussi banales que d'aller simplement se coucher : « sommeil », « blottir », « couvertur » (voir Figure 18, p.65, Figure 23, p.69, Figure 25, p.70, Figure 27, p.72).

Il y a assez peu de mots qui représenteraient des interactions tant physiques que verbales entre les personnages. Cela sous-entend qu'il y a peu d'évolution dans les relations entre les personnages, celles-ci semblent par ailleurs être prédéfinies sous la forme de familles ou ne peut plus normales et traditionnelles : « compagne », « compagnon », « enfant ».

Aucun mot de vocabulaire résultant des différentes analyses ne semblent indiquer un quelconque enjeu ou finalité associée aux fanfictions *Fluff*. Cela confirme la définition donnée dans l'état de l'art qui explique que *Fluff* contient des récits très légers sans réelle intrigue.

## 5.4. Longueurs des fanfictions

Dans cette section, je m'intéresse à la longueur des différentes fanfictions associées à chaque tag. Cela permet de mettre en évidence des différences concernant la forme et structures des récits des différents tags, qui sont souvent liées au contenu de ces récits.

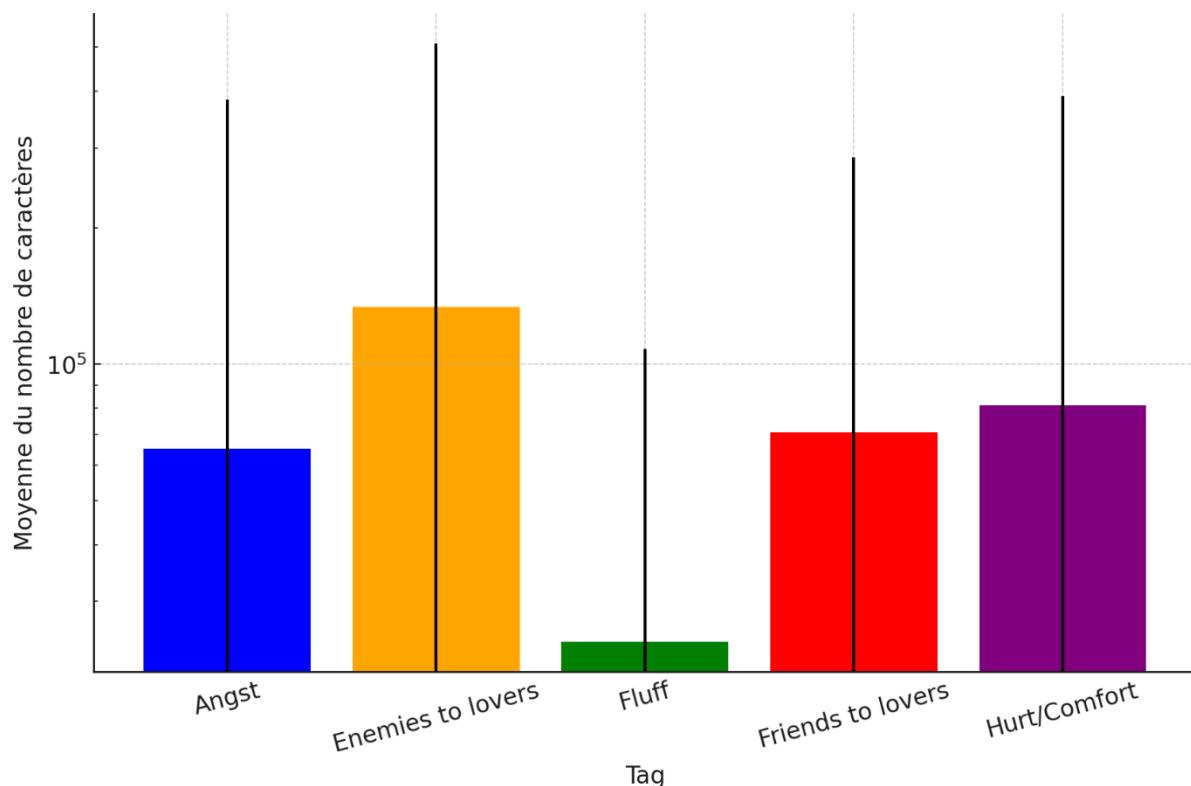
### 5.4.1. Moyennes et écart-types

Le calcul des moyennes et écart-types des longueurs des fanfictions de chaque tag (avant suppression des fanfictions trop courtes ou trop longues) donne les résultats suivants :

Tableau 9 - Moyenne et écart-type des longueurs par tag

Tag	Moyenne (nombre de caractères)	Écart-type (nombre de caractères)
Fluff	24 444	83 475
Angst	65 154	318 654
Hurt/Comfort	81 091	309 595
Enemies to lovers	133 682	376 102
Friends to lovers	70 676	215 482

Figure 28 - Graphique en barres des moyennes (barres de couleurs) et écart-types (traits noirs) des longueurs de fanfictions par tag



On voit que les longueurs moyennes de *Angst*, *Friends to lovers* et *Hurt/Comfort* sont assez proches et varient entre 65 000 et 80 000 caractères environ. Les fanfictions *Enemies to lovers* sont les plus longues en moyenne avec une moyenne de 133 682 caractères. Enfin, les fanfictions *Fluff* sont beaucoup plus courtes que les autres avec une moyenne de seulement 24 444 caractères.

Concernant les écart-types, celui de *Fluff* est le plus faible (83 475 caractères). L'écart-type de *Friends to lovers* est plus élevé avec une valeur de 215 482 caractères. Enfin, les écart-types des trois autres tags sont très élevés et dépassent les 300 000 caractères, avec un maximum de presque 380 000 caractères pour *Enemies to lovers*.

On voit aussi que les tags relationnels que sont *Enemies to lovers*, *Hurt/Comfort*, *Friends to lovers* sont les plus longs en moyenne, sans doute parce qu'établir et faire évoluer une relation entre

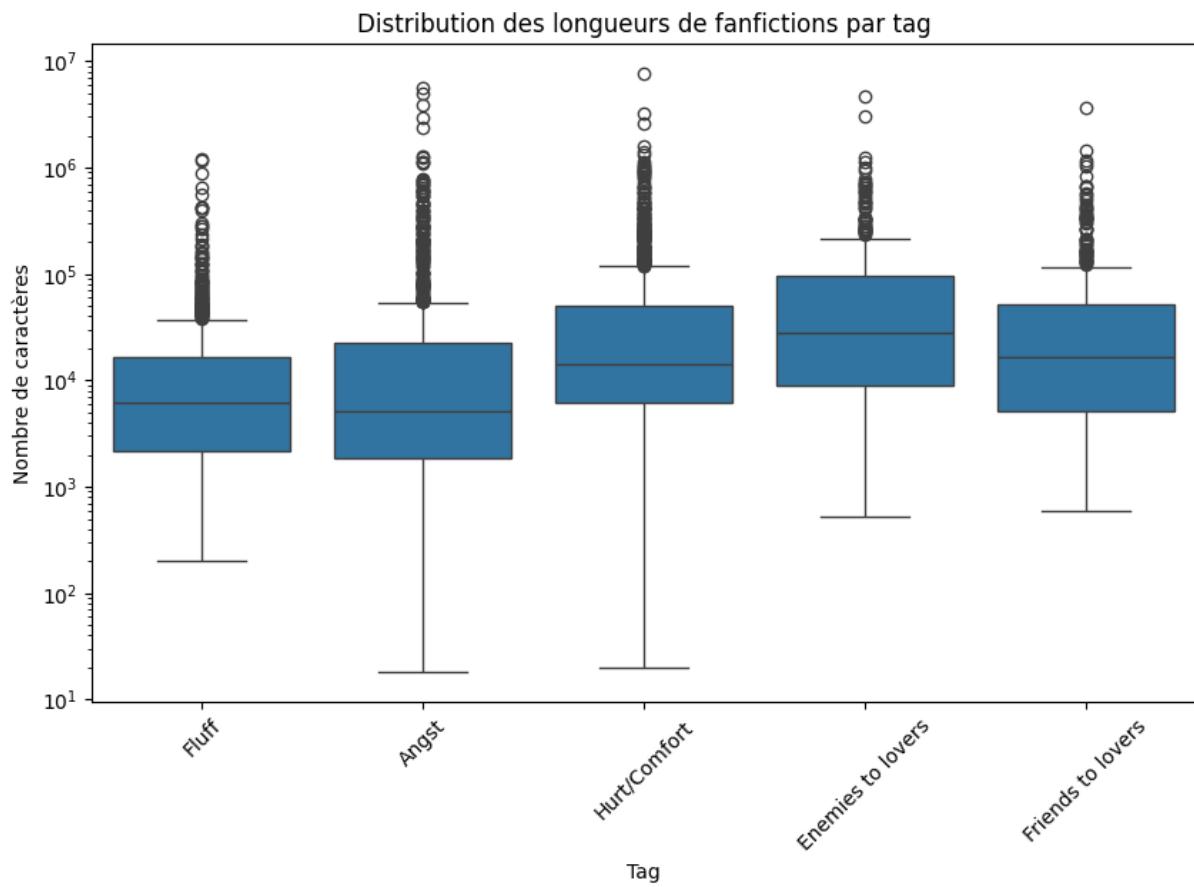
des personnages demande plus de temps et de développement que des fanfictions où les personnages interagissent moins.

Les fanfictions *Fluff* sont les plus courtes, ce qui coïncide avec le fait qu'il s'agit de fanfictions sans intrigue, ce qui ne permet donc pas de développer le récit. Elles sont de plus les plus homogènes en termes de longueur avec l'écart-type le plus faible, même s'il reste élevé.

En effet, les valeurs des écart-types sont très élevées, cela suggère qu'elles sont biaisées par des valeurs aberrantes, sûrement des fanfictions anormalement longues dans chaque tag. Ces valeurs aberrantes pourraient aussi causer des moyennes très élevées qui ne reflètent pas la majorité des fanfictions.

#### 5.4.2. Diagramme à moustaches

Figure 29 - Diagramme à moustaches des longueurs de fanfictions par tag



Grâce à ce diagramme à moustaches, on voit que les moyennes précédentes semblent en effet trop élevées, car elles dépassent le haut des boîtes pour chaque tag, ce qui veut dire que 75 % des fanfictions de chaque tag sont moins longues que les moyennes calculées plus haut.

Chaque tag possède plusieurs *outliers* et donc plusieurs fanfictions anormalement longues, qui ont effectivement pu causer des valeurs d'écart-types et de moyennes trop grandes.

Même si ces valeurs étaient trop élevées, ce diagramme à moustaches confirme néanmoins les tendances observées. *Enemies to lovers* contient les fanfictions les plus longues avec la boîte la plus

haute, suivi de *Hurt/Comfort* et *Friends to lovers*, puis de *Angst* et *Fluff* qui sont les fanfictions les plus courtes.

Sans compter les *outliers*, les fanfictions *Fluff* ont les moustaches les moins étendues, ce qui veut dire que leurs longueurs sont assez homogènes. Les fanfictions *Fluff* sont donc en grande majorité des histoires courtes. *Enemies to lovers* et *Friends to lovers* ont aussi des moustaches assez peu étendues et présentent donc des longueurs plutôt homogènes, mais les moustaches sont placées plus haut, ce sont donc en grande majorité des histoires assez longues.

*Hurt/Comfort* et *Angst* ont quant à elles des moustaches étendues et donc des longueurs apparemment assez variables.

### 5.5. Conclusion

Ces résultats ont permis de montrer comment les auteurs utilisent un lexique spécifique pour transmettre des émotions et ambiances propres à chaque tag, tel que la légèreté pour Fluff, l'intensité pour Enemies to lovers, le danger et l'hostilité pour Angst, la tendresse pour Friends to lovers et la détresse psychologique pour Hurt/Comfort.

L'étude a aussi permis de mettre en avant des différences et similitudes entre les tags et même d'établir une hiérarchie entre eux. *Angst* et *Fluff* sont deux tags généraux à l'opposé l'un de l'autre. *Hurt/Comfort*, *Friends to lovers* et *Enemies to lovers* sont des sous-tags mieux définis et sont tous trois des tags relationnels, avec *Friends to lovers* et *Enemies to lovers* qui sont particulièrement portés sur la romance.

Enfin, on a pu mettre en évidence l'importance du développement relationnel dans la longueur des récits.

## 6. Conclusions et perspectives

---

Cette étude a permis de mieux comprendre comment les tags qu'utilisent les auteurs de fanfiction coïncident avec les structures et contenus de ces fanfictions en se concentrant sur cinq tags populaires. L'analyse lexicale a révélé que chacun de ces tags est associé à un champ lexical spécifique qui souligne des thématiques diverses : *Angst* se caractérise par un vocabulaire marqué par la violence, *Fluff* par des termes légers et des décors de la vie quotidienne, tandis que *Hurt/Comfort*, *Enemies to lovers* et *Friends to lovers* sont des tags relationnels qui mettent en avant des interactions et dynamiques émotionnelles complexes. L'étude des longueurs a également montré que les fanfictions Fluff qui n'ont pas d'intrigue particulière sont généralement courtes, à l'inverse des tags relationnels qui donnent lieu à des récits plus longs qui nécessitent des développements narratifs plus poussés.

En ce qui concerne la classification automatique des tags, les résultats ont été mitigés, notamment en raison de la complexité de certains tags et du format de la fanfiction en lui-même mais surtout en raison du manque de données pour certains tags comme *Enemies to lovers*. L'idéal serait d'augmenter le nombre de données, cependant il n'y a pas assez de données existantes en français sur le site AO3. Pour y remédier, il serait possible d'élargir le corpus, soit en collectant des fanfictions sur d'autres plateformes ou bien en étudiant des textes en anglais où les fanfictions sont plus abondantes. Cependant, bien que les scores F1-macro sur la partie test n'aient pas été très élevés, l'étude des matrices de confusion a permis de proposer une hiérarchie intéressante entre certains tags, offrant ainsi des pistes pour améliorer la classification future.

Une autre limite de cette étude vient du fait que les fanfictions utilisées ne sont annotées qu'avec un seul des tags sur lesquels se penchent les analyses. Or dans les faits, les auteurs de fanfictions multiplient souvent les tags, il est donc commun de trouver des fanfictions annotées avec plusieurs des cinq tags sélectionnés. Ainsi, un autre axe d'exploration serait d'analyser des fanfictions comportant plusieurs tags à l'aide de classification automatique multi-label notamment, en s'inspirant par exemple des travaux de Calvo Tello (2021) qui s'est intéressé à différents sous-genres de romans espagnols produits entre le XIXe siècle et la guerre civile espagnole en les classant grâce à une classification automatique multi-label. Cela permettrait d'observer si certains tags dominent ou si les textes mêlent réellement les caractéristiques lexicales de plusieurs catégories. Cela permettrait également d'étudier d'éventuels abus de tags, où certains auteurs pourraient les employer de manière opportuniste sans que leurs récits ne reflètent réellement les caractéristiques qu'on leur associe. Enfin, l'ajout d'autres tags moins courants pourrait enrichir la compréhension des tendances de narration et de classification en fanfiction.

Ainsi, ce travail constitue une première approche quantitative dans la catégorisation des fanfictions par tags. Il ouvre la voie à des analyses plus approfondies qui pourraient non seulement affiner la compréhension des dynamiques narratives spécifiques à ce genre, mais aussi contribuer à l'amélioration des systèmes de recommandation et de classification automatique des textes littéraires amateurs.

## 7. Bibliographie

---

- Bacon-Smith, C. (1992). *Enterprising women: Television fandom and the creation of popular myth*. University of Pennsylvania Press.
- Bamman, D., Lewke, O., & Mansoor, A. (2020). An Annotated Dataset of Coreference in English Literature. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis (Éds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference* (p. 44-54). European Language Resources Association. <https://aclanthology.org/2020.lrec-1.6/>
- Black, S. R. (2020). Adding a digital dimension to fan studies methodologies. *Transformative Works and Cultures*, 33. <https://journal.transformativeworks.org/index.php/twc/article/view/1725>
- Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84. <https://doi.org/10.1145/2133806.2133826>
- Brigadói, I. (2021). *Genre classification using syntactic features* [Mémoire de master, Université d'Uppsala]. <https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-454667>
- Busse, K. (2017). *Framing Fan Fiction: Literary and Social Practices in Fan Fiction Communities*. University of Iowa Press. <https://doi.org/10.2307/j.ctt20q22s2>
- Calvo Tello, J. (2021). *The Novel in the Spanish Silver Age*. <https://www.bielefeld-university-press.org/978-3-8376-5925-2/the-novel-in-the-spanish-silver-age/>
- Chang, J., & Blei, D. (2009). Relational Topic Models for Document Networks. *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 81-88. <https://proceedings.mlr.press/v5/chang09a.html>
- Du, K., Dudar, J., & Schöch, C. (2021). *pydistinto—A Python implementation of different measures of distinctiveness for contrastive text analysis* [Python]. <https://doi.org/10.5281/zenodo.5245096>

Dudar, J. (2023). Data Analysis for Genre. In C. Schöch, J. Dudar, & E. Fileva (Éds.), *Survey of Methods*

*in Computational Literary Studies (= D3.2 : Series of Five Short Survey Papers on Methodological Issues)*. CLS INFRA. <https://doi.org/10.5281/zenodo.7892112>

Fabien, M. (2019, octobre 28). *Traitemet Automatique du Langage Naturel en Français (TAL)*.

<https://maelfabien.github.io/machinelearning/NLPfr/>

Fast, E., Vachovsky, T., & Bernstein, M. S. (2016). *Shirtless and Dangerous : Quantifying Linguistic*

*Signals of Gender Bias in an Online Fiction Writing Community* (arXiv:1603.08832). arXiv.

<https://doi.org/10.48550/arXiv.1603.08832>

Ganjigunte Ashok, V., Feng, S., & Choi, Y. (2013). Success with Style : Using Writing Style to Predict the

Success of Novels. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Éds.),

*Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (p.

1753-1764). Association for Computational Linguistics. <https://aclanthology.org/D13-1181/>

Goldmann, J. E. (2022). *Fan Fiction Genres Gender, Sexuality, Relationships and Family in the*

*Fandoms »Star Trek« and »Supernatural«*. transcript publishing.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of*

*Sciences, 101(suppl\_1)*, 5228-5235. <https://doi.org/10.1073/pnas.0307752101>

Grootendorst, M. (2022). BERTopic : Neural topic modeling with a class-based TF-IDF procedure. *arXiv*

*preprint, arXiv:2203.05794*. <https://doi.org/10.48550/ARXIV.2203.05794>

Hearst, M. A. (1997). Text Tiling : Segmenting Text into Multi-paragraph Subtopic Passages.

*Computational Linguistics, 23(1)*, 33-64.

Hellekson, K., & Busse, K. (Éds.). (2014). *The Fan Fiction Studies Reader*. University of Iowa Press.

<https://doi.org/10.2307/j.ctt20p58d6>

Herrmann, J. B., FRONTINI, F., & Rebora, S. (2022). *Anatomy of tools : A closer look at "textual DH"*

*methodologies*. <https://osf.io/preprints/osf/6dx4c>

Jähnichen, P., Oesterling, P., Heyer, G., Liebmann, T., Scheuermann, G., & Kuras, C. (2017). Exploratory Search Through Visual Analysis of Topic Models. *Digital Humanities Quarterly*, 011(2).  
<https://www.digitalhumanities.org/dhq/vol/11/2/000296/000296.html>

Jamison, A. E. (2013). *Fic : Why Fanfiction Is Taking Over the World*. BenBella Books, Inc.

Jenkins, H. (2013). *Textual poachers : Television fans and participatory culture* (Updated 20th anniversary edition). Routledge.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., & Levy, O. (2020). SpanBERT : Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8, 64-77. [https://doi.org/10.1162/tacl\\_a\\_00300](https://doi.org/10.1162/tacl_a_00300)

Kestemont, M., Luyckx, K., Daelemans, W., & Crombez, T. (2012). Cross-Genre Authorship Verification Using Unmasking. *English Studies*, 93(3), 340-356.  
<https://doi.org/10.1080/0013838X.2012.668793>

Li, J., & Sterman, S. (2016). *radiolarian/AO3Scraper : A Python scraper for getting fan fiction content and metadata from Archive of Our Own*. GitHub. <https://github.com/radiolarian/AO3Scraper>

Mahesh, B. (2019). Machine Learning Algorithms—A Review. *International Journal of Computer Trends and Technology*. [https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762\\_Machine\\_Learning\\_Algorithms\\_-A\\_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096t](https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096t)

Mattei, A., Brunato, D., & Dell'Orletta, F. (2020). The Style of a Successful Story : A Computational Study on the Fanfiction Genre. In F. Dell'Orletta, J. Monti, & F. Tamburini (Éds.), *Proceedings of the Seventh Italian Conference on Computational Linguistics CLiC-it 2020 : Bologna, Italy, March 1-3, 2021* (p. 284-289). Accademia University Press.  
<https://books.openedition.org/aaccademia/8718>

McCallum, A. K. (2002). *MALLET: A Machine Learning for Language Toolkit* [Logiciel].

<https://mimno.github.io/Mallet/>

Milli, S., & Bamman, D. (2016). Beyond Canonical Texts : A Computational Analysis of Fanfiction.

*Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*,

2048-2053. <https://doi.org/10.18653/v1/D16-1218>

Müller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python : A Guide for Data*

*Scientists*. O'Reilly Media. <http://archive.org/details/andreas-c.-muller-sarah-guido-introduction-to-machine-learning-with-python-a-gui>

OpenWalnut Development Team. (2010). *OpenWalnut –An Open-Source Tool for Multi-Modal Medical*

*and Brain Data Visualization* [Logiciel]. Université de Leipzig / Forschungszentrum Jülich.

<https://openwalnut.org/>

Osisanwo, F. Y., Akinsola, J. E. T., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017).

Supervised Machine Learning Algorithms : Classification and Comparison. *International Journal of Computer Trends and Technology*. <https://dev.ijcttjournal.org//archives/ijctt-v48p126>

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer,

P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M.,

& Duchesnay, É. (2011). *Scikit-learn : Machine Learning in Python* [Python].

<https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html> (Édition originale 2010)

Pincemin, B. (2020, décembre 7). *La textométrie, une approche quantitative et qualitative des données*

*textuelles*. Semaine Data SHS Traiter et analyser des données en sciences humaines et sociales.

<https://shs.hal.science/halshs-03089085>

Řehůřek, R., & Sojka, P. (2010). *Software Framework for Topic Modelling with Large Corpora*.

<https://doi.org/10.13140/2.1.2393.1847>

Rouse, L. (2021). Fan fiction comments and their relationship to classroom learning. *Transformative Works and Cultures*, 35. <https://doi.org/10.3983/twc.2021.1911>

Rybicki, J., & Eder, M. (2011). Deeper Delta across genres and languages : Do we really need the most frequent words? *Literary and Linguistic Computing*, 26(3), 315-321. <https://doi.org/10.1093/lrc/fqr031>

Schöch, C. (2017). Topic Modeling Genre : An Exploration of French Classical and Enlightenment Drama. *Digital Humanities Quarterly*, 011(2).

Schöch, C., & Schlör, D. (2018). *Burrows' Zeta : Exploring and Evaluating Variants and Parameters*. <https://dh2018.adho.org/en/burrows-zeta-exploring-and-evaluating-variants-and-parameters/>

VanderPlas, J. (2016). *Python Data Science Handbook : Essential Tools for Working with Data*. O'Reilly Media. <https://jakevdp.github.io/PythonDataScienceHandbook/>

Xanthoudakis, A. (2021). *Mushroom for improvement : A model for the circulation of fanfiction sub-genres* [(Project) M.Pub.]. <https://summit.sfu.ca/item/34582>

Yoder, M., Khosla, S., Shen, Q., Naik, A., Jin, H., Muralidharan, H., & Rosé, C. (2021). FanfictionNLP : A Text Processing Pipeline for Fanfiction. In N. Akoury, F. Brahman, S. Chaturvedi, E. Clark, M. Iyyer, & L. J. Martin (Éds.), *Proceedings of the Third Workshop on Narrative Understanding* (p. 13-23). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.nuse-1.2>

Yoder, M. M. (2021). *Computational Models of Identity Presentation in Language [Proposition de thèse de doctorat]* [Carnegie Mellon University]. <https://michaelmilleryoder.github.io/>

## 8. Annexes

---

### 8.1. Annexe 1 : Portions de code modifiées dans le script pour la collecte d'identifiants d'AO3scraper

Extrait de code 14 - Fonction get\_ids modifiée

```
def get_ids(header_info='', max_retries=3, delay=2):
    global page_empty
    global seen_ids
    global consecutive_failures

    page_number = 1
    page_key = "page="
    start = url.find(page_key)
    if start != -1:
        page_start_index = start + len(page_key)
        page_end_index = url.find("&", page_start_index)
        if page_end_index != -1:
            page_number = int(url[page_start_index:page_end_index])
        else:
            page_number = int(url[page_start_index:])

    print(f"\n[INFO] Récupération des fanfictions à partir de la page
{page_number} de l'URL : {url}")

    retries = 0
    works = []

    # Essayer plusieurs fois en cas d'échec
    while retries < max_retries:
        try:
            headers = {'user-agent': get_random_user_agent()}
            req = requests.get(url, headers=headers)

            # Si le serveur répond par le code 429 (limite de requêtes
            atteinte), on attend et on réessaie
            while req.status_code == 429:
                print("Réponse 429 reçue, réessayer après un délai...")
                time.sleep(10) # Attente prolongée pour la réponse 429
                req = requests.get(url, headers=headers)

            soup = BeautifulSoup(req.text, "lxml")

            # Recherche des fanfictions sur la page
            works = soup.select("li.work.blurb.group")

            print(f"[INFO] Page actuelle : {url} - Nombre de fanfictions
récupérées : {len(works)}")

            if len(works) > 0:
                break # Sort de la boucle si on a trouvé des fanfictions

        else:
            print(f"[INFO] Aucune fanfiction trouvée sur cette page.
Essai {retries + 1}/{max_retries}.")
            retries += 1
            time.sleep(delay) # Attente avant de réessayer

    except requests.exceptions.RequestException as e:
        print(f"[ERROR] Erreur de requête sur la page {url}: {e}")
```

```

        retries += 1
        time.sleep(delay) # Attente avant de réessayer

    if retries == max_retries and len(works) == 0:
        print(f"[INFO] Échec après {max_retries} tentatives. La page {url} semble vide.")
        consecutive_failures += 1 # Incrémentation du compteur d'échecs
    else:
        consecutive_failures = 0 # reset du compteur d'échecs si des fics sont trouvés

    ids = []
    for idx, tag in enumerate(works, start=1):
        t = tag.get('id')[5:]

        print(f"[INFO] Récupération de l'ID {t} (Fanfiction {idx} sur la page {page_number})")

        if t not in seen_ids:
            ids.append(t)
            seen_ids.add(t)
            print(f"[SUCCÈS] L'ID {t} a été ajouté à la liste.")
        else:
            print(f"[DUPLICATA] L'ID {t} a déjà été collecté, ignoré.")

    return ids

```

Extrait de code 15 - Fonction main modifiée

```

def main():
    header_info = get_args()
    make_readme()

    print("Chargement des IDs existants si présents...\\n")
    load_existing_ids() # Charge les IDs déjà collectés si le CSV existe

    print("Démarrage de la collecte des fanfictions...\\n")

    # Si des tags sont fournis, récupérer les fanfictions pour chaque tag
    if len(tags):
        for t in tags:
            print(f"[INFO] Récupération des fanfictions pour le tag : {t}")
            reset() # Réinitialise les variables de collecte pour chaque tag
            add_tag_to_url(t) # Modifie l'URL pour inclure le tag
            process_for_ids(header_info) # Collecte des fanfictions avec la nouvelle URL
        else:
            process_for_ids(header_info) # Si aucun tag n'est donné, collecte sans tag

    print("Collecte terminée.")

```

## 8.2. Annexe 2 : Portions de code modifiées dans le script pour la collecte des fanfictions et leurs métadonnées d'AO3scraper

Extrait de code 16 - Ajout de user-agents et de la fonction random\_user\_agent

```

user_agents = [
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,

```

```

like Gecko) Chrome/91.0.4472.124 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/14.0 Safari/605.1.15",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101
Firefox/89.0",
    "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like
Gecko",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/47.0.2526.111 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; AS_8MSR; rv:11.0)
like Gecko",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KKHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:89.0) Gecko/20100101
Firefox/89.0",
    "Mozilla/5.0 (Linux; Android 10; SM-G960F) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/91.0.4472.124 Mobile Safari/537.36",
    "Mozilla/5.0 (Linux; Android 11; Pixel 5) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/90.0.4430.210 Mobile Safari/537.36",
    "Mozilla/5.0 (iPhone; CPU iPhone OS 14_4 like Mac OS X)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/15E148
Safari/604.1",
    "Mozilla/5.0 (Android 14; Mobile; rv:138.0) Gecko/138.0 Firefox/138.0"
]

def get_random_user_agent():
    return random.choice(user_agents)

```

#### Extrait de code 17 - Fonction write\_fic\_to\_csv modifiée

```

def write_fic_to_csv(fic_id, only_first_chap, lang, include_bookmarks,
metadata_only, writer, errorwriter, header_info=''):
    """
    fic_id is the AO3 ID of a fic, found every URL /works/[id].
    writer is a csv writer object
    the output of this program is a row in the CSV file containing all
    metadata
    and the fic content itself (excludes content if metadata_only=True).
    header_info should be the header info to encourage ethical scraping.
    """
    print(f"Scraping {fic_id}...")
    url = f'http://archiveofourown.org/works/{fic_id}?view_adult=true'
    if not (only_first_chap or metadata_only):
        url += '&view_full_work=true'

    max_retries = 3
    for attempt in range(1, max_retries + 1):
        headers = {'user-agent': header_info}
        req = requests.get(url, headers=headers)
        status = req.status_code

        if status == 200:
            break

        if status == 429:
            print(f"Erreur 429 : tentative {attempt}/{max_retries}, attente
60 sec...")
            time.sleep(60)
        else:
            print(f"Erreur {status} : tentative {attempt}/{max_retries},
attente 30 sec...")

```

```

        time.sleep(30)

    if status >= 400:
        print(f"✗ Échec après {max_retries} tentatives. Fic {fic_id} ignorée.")
        errorwriter.writerow([fic_id, status])
        return False # Signale un échec

    soup = BeautifulSoup(req.text, 'html.parser')
    if access_denied(soup):
        print('Access Denied')
        errorwriter.writerow([fic_id, 'Access Denied'])
        return False

    meta = soup.find("dl", class_="work meta group")
    author = get_authors(soup.find("h3", class_="byline heading"))
    tags = get_tags(meta)
    stats = get_stats(meta)
    title = unidecode(soup.find("h2", class_="title heading").string).strip()
    visible_kudos = get_kudos(soup.find('p', class_='kudos'))
    hidden_kudos = get_kudos(soup.find('span', class_='kudos_expanded_hidden'))
    all_kudos = visible_kudos + hidden_kudos

    if lang and lang != stats[0]:
        print(f"Fic non en {lang}, ignorée.")
        return False

    all_bookmarks =
    get_bookmarks(f'http://archiveofourown.org/works/{fic_id}/bookmarks',
    header_info) if include_bookmarks else []

    if not metadata_only:
        content = soup.find("div", id="chapters")
        chapters = content.select('p')
        chapter_text = '\n\n'.join([unidecode(chapter.text) for chapter in chapters])
    else:
        chapter_text = ""

    row = [fic_id, title, author] + [', '.join(tag) for tag in tags] +
    stats + [all_kudos, all_bookmarks, chapter_text]

    try:
        writer.writerow(row)
        print("✓ Fic collectée avec succès.")
        return True
    except Exception as e:
        print(f"✗ Erreur d'écriture pour {fic_id}: {e}")
        errorwriter.writerow([fic_id, str(e)])
        return False

```

#### Extrait de code 18 - Fonction main modifiée

```

def main():
    fic_ids, csv_out, headers, restart, is_csv, only_first_chap, lang,
    include_bookmarks, metadata_only = get_args()
    os.chdir(os.getcwd())

```

```

output_directory = os.path.dirname(csv_out)
if output_directory and not os.path.isdir(output_directory):
    print("Creating output directory " + output_directory)
    os.mkdir(output_directory)

with open(csv_out, 'a', newline="") as f_out:
    writer = csv.writer(f_out)
    with open(os.path.join(os.path.dirname(csv_out), "errors_" +
os.path.basename(csv_out)), 'a', newline="") as e_out:
        errorwriter = csv.writer(e_out)

        # Vérification si le fichier CSV a une en-tête
        if os.stat(csv_out).st_size == 0:
            print('Writing a header row for the csv.')
            header = ['work_id', 'title', 'author', 'rating',
'category', 'fandom', 'relationship', 'character',
                'additional tags', 'language', 'published',
'status', 'status date', 'words', 'chapters',
                'comments', 'kudos', 'bookmarks', 'hits',
'all_kudos', 'all_bookmarks', 'body']
            writer.writerow(header)

        # Compteur pour afficher la progression
        total_fics = 0
        if is_csv:
            with open(fic_ids[0], 'r', newline="") as f_in:
                total_fics = sum(1 for _ in csv.reader(f_in))  #
Compter le nombre total de lignes

        elif fic_ids:
            total_fics = len(fic_ids)  # Si on donne une liste d'IDs
directement

        processed_fics = 0  # Fanfics traitées
        failed_fics = 0  # Nombre d'échecs

        if is_csv:
            csv_fname = fic_ids[0]
            with open(csv_fname, 'r+', newline="") as f_in:
                reader = csv.reader(f_in)
                found_restart = False if restart else True

                for row in reader:
                    if not row:
                        continue

                    found_restart = process_id(row[0], restart,
found_restart)
                    if found_restart:
                        processed_fics += 1
                        print(f"Fanfiction
{processed_fics}/{total_fics} en cours...")
                        success = write_fic_to_csv(row[0],
only_first_chap, lang, include_bookmarks, metadata_only, writer,
errorwriter, headers)
                        if not success:
                            failed_fics += 1

                        time.sleep(delay)
                    else:
                        print('Skipping already processed fic')

```

```

        else:
            for fic_id in fic_ids:
                processed_fics += 1
                print(f"Fanfiction {processed_fics}/{total_fics} en
cours...")
                success = write_fic_to_csv(fic_id, only_first_chap,
lang, include_bookmarks, metadata_only, writer, errorwriter, headers)
                if not success:
                    failed_fics += 1

                time.sleep(delay)

        print(f"\n\n\N{checkmark} Collecte terminée : {processed_fics} fanfictions
traitées.")
        print(f"\N{cross mark} Nombre de fanfictions échouées : {failed_fics}")

```

### 8.3. Annexe 3 : Code suppression des noms de personnages

Extrait de code 19 - Fonction d'extraction des noms de personnages

```

def clean_character_list(characters):
    if pd.isna(characters):
        return []

    # Remplacer les "|" par ","
    characters = characters.replace('||', ',', ',')

    # Extraire les surnoms entre guillemets
    surnoms = re.findall(r'"(.*)"', characters)

    # Supprimer les surnoms des noms principaux
    characters = re.sub(r'\s*\s*.*?\s*\s*', ' ', characters)

    # Supprimer les précisions entre parenthèses
    characters = re.sub(r'\s*\(\.*?\)\s*', ' ', characters)

    # Transformer en liste propre
    character_list = [char.strip() for char in characters.split(',') if
char.strip()]

    # Séparer prénom et nom pour chaque personnage
    character_list_sep_complet = []
    for char in character_list:
        # Ajouter nom et prénom séparément à la liste
        character_list_sep_complet.extend(char.split())

    # Ajouter les surnoms à la liste des noms à supprimer
    character_list_sep_complet.extend(surnoms)

    # Supprimer les doublons
    return list(set(character_list_sep_complet))

```

Extrait de code 20 - Fonction pour supprimer noms de personnages d'un texte

```
def remove_characters_from_text(texte, characters):
    if pd.isna(texte) or not characters:
        return texte

    for character in characters:
        # \b pour éviter de supprimer des bouts de mots
        texte = re.sub(r'\b' + re.escape(character) + r'\b', '', texte,
flags=re.IGNORECASE)

    return texte
```

#### 8.4. Annexe 4 : Entraînement et validation croisée de la classification automatique

Extrait de code 21 - Entraînement en validation croisée des différents algorithmes de classification

```
# Modèles à comparer
models = [
    ('Baseline', DummyClassifier(strategy='most_frequent')),
    ('Mutinomial NB', MultinomialNB()),
    ('CART', DecisionTreeClassifier()),
    ('LR', LogisticRegression()),
    ('KNN', KNeighborsClassifier()),
    ('Random forest', RandomForestClassifier()),
    ('LinearSVC', LinearSVC()),
]

# Evaluation de chaque résultat l'un après l'autre
scores = []
names = []
scoring = 'macro F1'
# Validation croisée à 5 plis
kfold = model_selection.StratifiedKFold(n_splits=5, shuffle=True,
random_state=12)
# Itération sur les modèles
for name, model in models:
    # Ajout du nom du modèle à la liste name
    names.append(name)
    # Création de la pipeline pour le modèle
    model_pipeline = make_pipeline(text_vectorizer, model)
    # Validation croisée
    y_pred = model_selection.cross_val_predict(model_pipeline,
                                                X_dev, y_dev,
                                                cv=kfold)
    print(name)
    print(classification_report(y_dev, y_pred))
    f1 = metrics.f1_score(y_dev, y_pred, average='macro')
    scores.append(f1)

# Représentation graphique des résultats
```

```

indices = np.arange(len(scores))
fig = plt.figure()
plt.barh(indices, scores, .2, label="score", color='b')
plt.yticks(())
for i, c in zip(indices, names):
    plt.text(-.3, i, c)
plt.show()

```

## 8.5. Annexe 5 : Paramètres optimaux avec GridSearchCV

Extrait de code 22 - Recherche des meilleurs hyperparamètres pour LinearSVC à l'aide de GridSearchCV

```

# Définir les paramètres à tester
param_grid = {
    'linearsvc_C': [0.01, 0.1, 1, 10, 100],
    'linearsvc_loss': ['hinge', 'squared_hinge'],
    'linearsvc_penalty': ['l2'],
    'linearsvc_tol': [1e-4, 1e-3, 1e-2]
}

# Créer la pipeline avec TF-IDF et LinearSVC
pipeline = make_pipeline(TfidfVectorizer(tokenizer=split_text),
LinearSVC())

# GridSearchCV avec validation croisée 5-fold
grid_search = GridSearchCV(pipeline, param_grid, cv=5,
scoring='f1_macro', n_jobs=-1)

# Lancer la recherche des meilleurs hyperparamètres
grid_search.fit(X_dev, y_dev)

# Afficher les meilleurs paramètres
print("Meilleurs paramètres:", grid_search.best_params_)

# Meilleur score obtenu
print("Meilleur F1-score:", grid_search.best_score_)

```

## 8.6. Annexe 6 : Entraînement sur les cinq tags avec pondération des classes

Extrait de code 23 - Entraînement et validation croisée de LinearSVC sur les cinq tags avec pondération des classes

```

# Calculer les poids des classes en utilisant les classes de
y_dev_complet
class_weights = compute_class_weight(class_weight='balanced',
                                      classes=np.unique(y_dev),
                                      y=y_dev)

# Créer un dictionnaire de poids de classes avec toutes les classes de
y_dev_complet

```

```

class_weights_dict = dict(zip(np.unique(y_dev), class_weights))

# Validation croisée à 5 plis
kfold = model_selection.StratifiedKFold(n_splits=5, shuffle=True,
random_state=12)

# Modèle à entraîner avec pondération des classes
model = LinearSVC(C=1, loss='squared_hinge', penalty='l2', tol=0.01,
                    class_weight=class_weights_dict)

# Création de la pipeline pour le modèle
model_pipeline = make_pipeline(text_vectorizer, model)

# Validation croisée
y_pred = model_selection.cross_val_predict(model_pipeline,
                                             X_dev,
                                             y_dev,
                                             cv=kfold)

# Affichage des résultats
print("LinearSVC")
print(classification_report(y_dev, y_pred))

# Calcul et affichage du F1-score macro
f1 = metrics.f1_score(y_dev, y_pred, average='macro')
print(f"Macro F1-score: {f1:.4f}")

```

## 8.7. Annexe 7 : Entraînement sur les cinq tags avec rééchantillonnage

Extrait de code 24 - Entraînement de LinearSVC sur les cinq tags avec rééchantillonnage (SMOTE)

```

# Vectorisation TF-IDF des textes
X_vectorized = text_vectorizer.fit_transform(X_dev)

# Appliquer SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_vectorized, y_dev)

# Refaire l'entraînement avec LinearSVC
model_smote = LinearSVC(C=10, loss='squared_hinge', penalty='l2',
tol=0.01)
model_smote.fit(X_resampled, y_resampled)

# Prédictions
y_pred_smote = model_smote.predict(text_vectorizer.transform(X_dev))
print(classification_report(y_dev, y_pred_smote))

```

## 8.8. Annexe 8 : Entraînement sur les cinq tags avec pondération et rééchantillonnage

Extrait de code 25 - Entraînement et validation croisée de LinearSVC sur les cinq tags avec pondération et rééchantillonnage (SMOTE)

```
# Calculer les poids des classes
class_weights = compute_class_weight(class_weight='balanced',
                                      classes=np.unique(y_dev),
                                      y=y_dev)
class_weights_dict = dict(zip(np.unique(y_dev), class_weights))

# Vectorisation TF-IDF des textes
X_vectorized = text_vectorizer.fit_transform(X_dev)

# Appliquer SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_vectorized, y_dev)

# Modèle à entraîner avec pondération des classes et données
rééchantillonnées
model = LinearSVC(C=10, loss='squared_hinge', penalty='l2', tol=0.01,
                    class_weight=class_weights_dict)

# Création de la pipeline pour le modèle
model_pipeline = make_pipeline(model)

# Validation croisée
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=12)
y_pred = cross_val_predict(model_pipeline, X_resampled, y_resampled,
                           cv=kfold)

# Affichage des résultats
print("LinearSVC avec SMOTE et pondération des classes")
print(classification_report(y_resampled, y_pred))

# Calcul et affichage du F1-score macro
f1 = f1_score(y_resampled, y_pred, average='macro')
print(f"Macro F1-score: {f1:.4f}")
```

## 8.9. Annexe 9 : Exemple du contenu d'un fichier metadata.csv utilisé pour l'analyse par pydistinto

Extrait de code 26 – Contenu du fichier metadata utilisé pour comparer chaque tag aux quatre autres

idno	angst	fluff	hurt_comfort	enemies_to_lovers	friends_to_lovers
angst_fanfic	yes	no	no	no	no
fluff_fanfic	no	yes	no	no	no
hurt_comfort_fanfic	no	no	yes	no	no
enemies_to_lovers_fanfic		no	no	no	yes
					no

friends_to_lovers_fanfic	no	no	no	no	yes
--------------------------	----	----	----	----	-----

#### **8.10. Annexe 10 : Exemple du contenu d'un fichier parameters.txt pour l'analyse par pydistinto**

Extrait de code 27 – Contenu du fichier parameters utilisé lors de la comparaison du corpus fluff aux quatre autres tags

```
corpus=C:\Users\pauli\OneDrive\Documents\TDL\M2\Mémoire\pydi\ana_pydistinto\corpus
workdir=C:\Users\pauli\OneDrive\Documents\TDL\M2\Mémoire\pydi\ana_pydistinto\result
language=French
segmentlength=5000
randomizing=yes
use_randomized_texts=no
random_segment_size=50
forms=lemmata
pos=all
contrast=fluff
target_corpus=yes
comparison_corpus=no
no_of_features=25
measures=zeta_sd0
```

**8.11. Annexe 11 : Graphes des 15 mots les plus discriminants de chaque tag suite à la classification automatique à l'aide de LinearSVC**

Figure 30 - 15 mots discriminants de *Angst* (classification sur cinq tags)

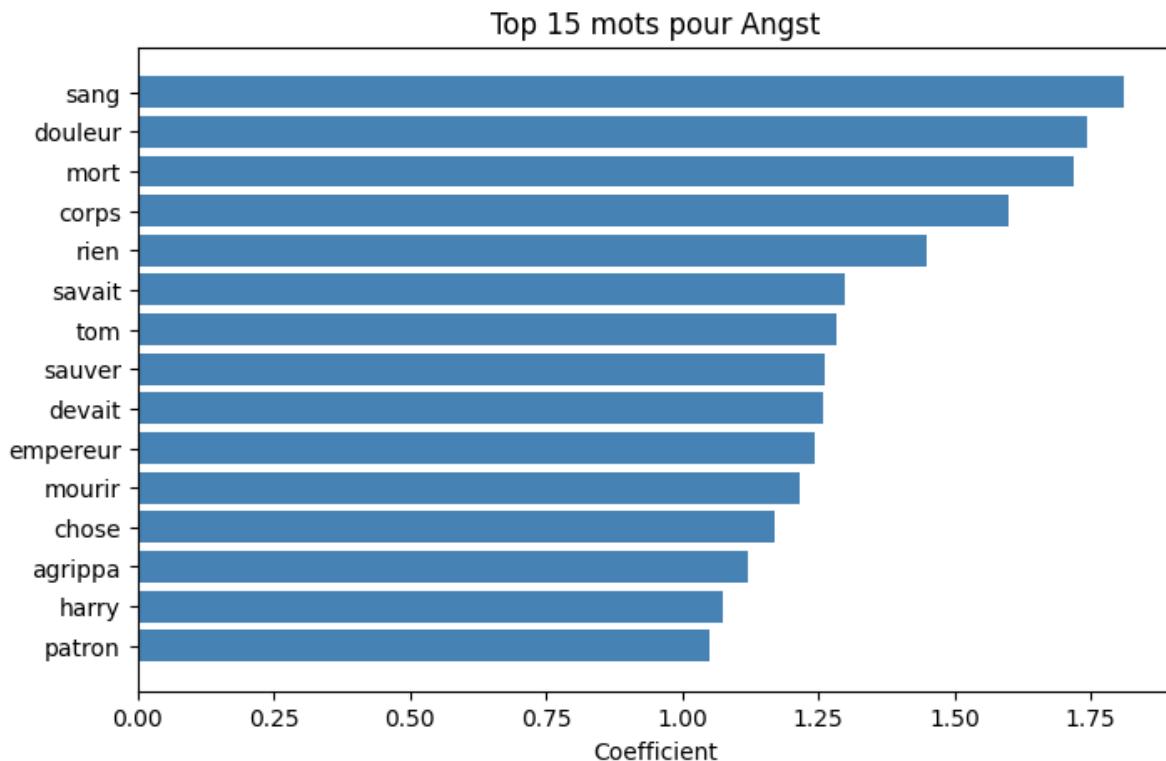


Figure 31 - 15 mots discriminants de *Fluff* (classification sur trois tags)

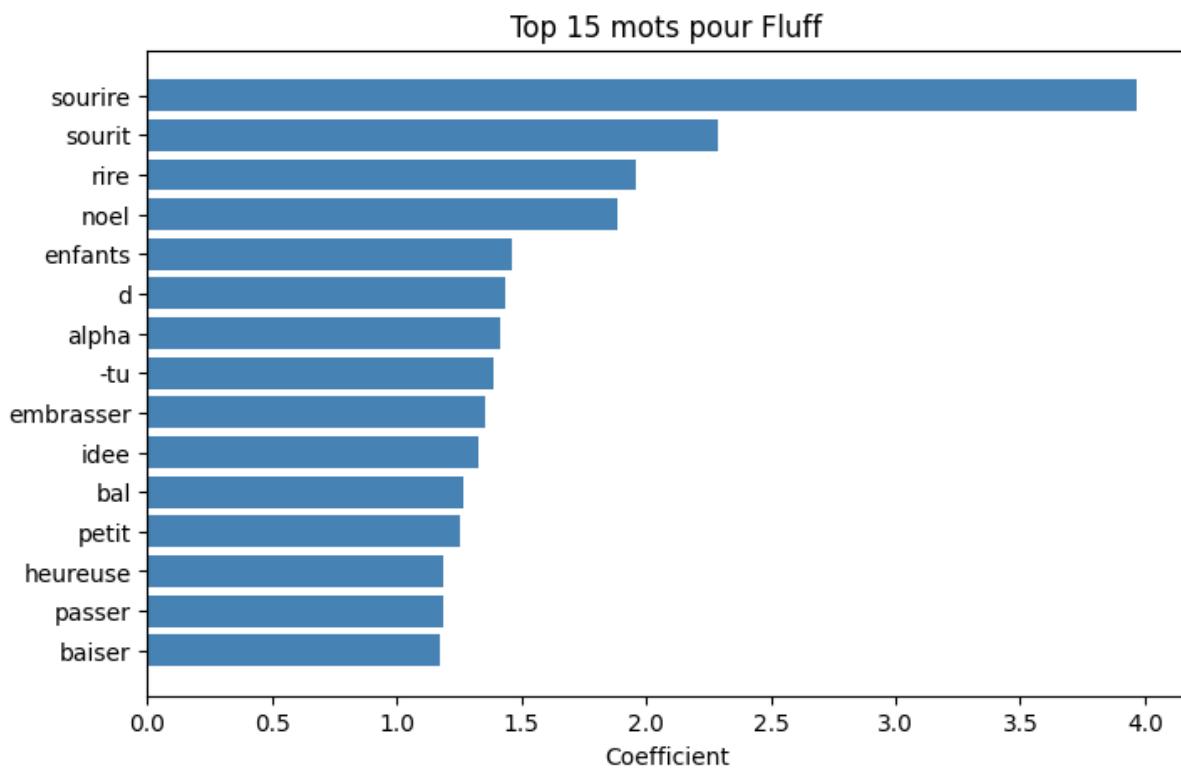


Figure 32 - 15 mots discriminants de *Hurt/Comfort* (classification sur cinq tags)

