

Faciliter la conception d'un assistant conversationnel avec le clustering interactif

THÈSE

présentée et soutenue publiquement le 01 décembre 2023

pour l'obtention du

Doctorat de l'Université de Lorraine
(mention informatique)

par

Erwan SCHILD

Composition du jury

Présidents : Dr. ??

Rapporteurs : Dr. Pascale KUNTZ-COSPEREC
Dr. Thomas LAMPERT

Examinateur : Dr. Adrien COULET (à demander)

Encadrants : Dr. Jean-Charles LAMIREL
Dr. Florian MICONI

Invités : Dr. Gautier DURANTIN
Dr. Mathieu POWALKA

M i s e n p a g e a v e c l a c l a s s e t h e s u l .

Résumé

Le résumé à faire.

Mots-clés: chat, chien, puces.

Abstract

The abstract to do

Keywords: cat, dog, flees.

Remerciements

Par la présente, je souhaitez remercier :

...

*Je dédie cette thèse
à quelqu'un de bien.*

Table des matières

Résumé	i
Abstract	i
Remerciements	iii
1	
Introduction	
1.1 (<i>Asset centrality</i>)	1
1.2 (<i>Establishing a Niche</i>)	1
1.3 (<i>Gap</i>)	2
1.4 (<i>Occupying the Niche</i>)	2
2	
Revue de littérature sur la tâche d'annotation en intelligence artificielle	
2.1 Présentation théorique de l'annotation	4
2.1.1 Définition et objectifs de l'annotation de données	4
2.1.2 Exemples de tâches d'annotations	5
2.1.3 Bilan concernant la présentation de l'annotation	10
2.2 Organisation usuelle d'un projet d'annotation	10
2.2.1 Étapes clés du cycle d'annotation	10
2.2.2 Portraits des acteurs intervenant sur un projet d'annotation	15
2.2.3 Choix du logiciel d'annotation	17
2.2.4 Bilan concernant l'organisation d'un projet d'annotation	18
2.3 Aperçu des nombreux défis de l'annotation	19
2.3.1 Défis concernant le besoin de qualité des données	20
2.3.2 Défis concernant la complexité inhérente à la tâche d'annotation	27
2.3.3 Défis concernant les différences de comportements d'annotation	32
2.3.4 Bilan concernant les nombreux défis de l'annotation	35

2.4 Contexte du doctorat : comment assister la conception d'une base d'apprentissage pour un agent conversationnel bancaire en français ?	35
---	----

3

Proposition d'un *Clustering Interactif* pour assister la modélisation d'un jeu de données textuelles

3.1 Intuitions à l'origine d'un <i>clustering</i> interactif	40
3.1.1 Utiliser une approche non-supervisées pour créer une modélisation	40
3.1.2 Corriger l'approche non-supervisée avec une annotation de contraintes	41
3.1.3 Tirer parti des avantages de l'apprentissage actif pour optimiser les interactions Homme/machine	43
3.2 Description de notre <i>clustering</i> interactif.	43
3.2.1 Description générale	44
3.2.2 Description détaillée	44
3.2.3 Description techniques et implémentation	47
3.3 Espoirs portés sur la méthode proposée.	48

4

Étude de six hypothèses sur le *Clustering Interactif*

4.1 Évaluation de l'hypothèse d'efficacité	53
4.1.1 Étude de convergence vers une vérité terrain pré-établissement en simulant l'annotation d'une base d'apprentissage et mesurant la vitesse de sa création	53
4.2 Évaluation de l'hypothèse d'efficience	60
4.2.1 Étude d'optimisation des paramètres d'implémentation en analysant leurs tailles d'effets sur la vitesse de création d'une base d'apprentissage	60
4.3 Évaluation de l'hypothèse sur les coûts	71
4.3.1 Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle	72
4.3.2 Étude du temps de calcul nécessaire aux algorithmes implémentés en chronométrant des exécutions dans différentes situations	81
4.3.3 Étude du nombre de contraintes nécessaires à la convergence vers une vérité terrain pré-établissement en fonction de la taille du jeu de données	91
4.3.4 Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts	95
4.3.5 Ouverture vers une annotation en parallèle du <i>clustering</i>	97
4.4 Évaluation de l'hypothèse de pertinence	100

4.4.1	Étude d'une validation manuelle et non assistée de la valeur métier d'une base d'apprentissage par un expert	101
4.4.2	Étude des patterns linguistiques pertinents à l'aide de la Maximisation des Traits pour assister la validation d'une base d'apprentissage	105
4.4.3	Étude d'un résumé automatique des <i>clusters</i> à l'aide d'un large modèle de langage	111
4.4.4	Mise en commun des stratégies d'évaluation de la pertinence métier d'un résultat de <i>clustering</i> interactif	118
4.5	Évaluation de l'hypothèse de rentabilité	119
4.5.1	Étude de l'évolution d'accord entre l'annotation et le <i>clustering</i>	120
4.5.2	Étude de l'évolution de la différence entre deux <i>clusterings</i> consécutifs	124
4.5.3	Mise en commun des stratégies d'évaluation de la rentabilité d'une itération de la méthode et définition d'un cas d'arrêt indépendant d'une vérité terrain.	128
4.6	Évaluation de l'hypothèse de robustesse	129
4.6.1	Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle	130
4.6.2	Étude de l'impact d'une erreur d'annotation et l'intérêt de la corriger . . .	134
4.6.3	Étude de l'impact de la subjectivité de l'annotation sur la divergence des résultats obtenus	139
4.6.4	Bilan concernant la robustesse du <i>clustering</i> interactif	147
4.7	Autres hypothèses non vérifiées	148
4.7.1	Étude du nombre de clusters optimal	148
4.7.2	Étude d'autres méthodes de vectorisation	149
4.7.3	Étude d'autres méthodes d'échantillonnage	149
4.7.4	Étude de techniques de transfert d'apprentissage	149
4.7.5	Étude ergonomique de l'interface d'annotation	149

5	Bilan et Guide d'utilisation du <i>Clustering Interactif</i>
---	---

5.1	Présentation rapide du clustering interactif et de ses avantages	151
5.2	Conseils pour organiser l'annotation	151
5.3	Conseils pour analyser les résultats	152
5.4	Conseils pour paramétrier la méthode	152
5.5	Estimation des coûts du projet d'annotation	152

6

Conclusion

6.1	(<i>Occupying the Niche</i>)	153
6.2	(<i>Gap</i>)	153
6.3	(<i>Establishing a Niche</i>)	153
6.4	(<i>Asset centrality</i>)	153

Annexes

A

Annexe des jeux de données utilisés dans nos études

A.1	Jeu de données Bank Cards	155
A.2	Jeu de données MLSUM	156

B

Annexe des architectures d'assistants conversationnels (*chatbot*)

C

Annexe des implémentations de notre *clustering* interactif

C.1	Table des nomenclatures utilisées	162
C.2	Implémentation de la librairie cognitivefactory-interactive-clustering	163
C.2.1	Gestion des données	163
C.2.2	Gestion des contraintes	165
C.2.3	Algorithme de <i>clustering</i> sous contraintes	167
C.2.4	Algorithme d'échantillonnage de contraintes	169
C.3	Implémentation de la librairie cognitivefactory-features-maximization-metric	171
C.4	Implémentation de l'application web cognitivefactory-interactive-clustering-gui	172

D

Annexe technique

D.1	v-measure : métrique d'évaluation de <i>clustering</i>	181
-----	--	-----

Bibliographie

183

Liste des TODOs	193
Liste des figures	197
Liste des tableaux	203
Liste des algorithmes	205
Liste de codes informatiques	207

LISTE DE CODES INFORMATIQUES

Chapitre 1

Introduction

CHAPITRE : TITRE À TROUVER : "Introduction"

CHAPITRE : INTRODUCTION À RÉDIGER

CHAPITRE : INTRODUCTION À RÉDIGER

Sommaire

1.1	<i>(Asset centrality)</i>	1
1.2	<i>(Establishing a Niche)</i>	1
1.3	<i>(Gap)</i>	2
1.4	<i>(Occupying the Niche)</i>	2

1.1 *(Asset centrality)*

SECTION : TITRE À TROUVER : "Asset centrality"

SECTION : À RÉDIGER :

- Utilisation intensive de l'IA / des chatbots, et description des nombreuses opportunités liées ;
- Mais aussi, mystification de l'IA : le grand public ne se sait pas comment ça marche, comme ça s'entraîne, ont des craintes sur les capacités des modèles, ...

1.2 *(Establishing a Niche)*

SECTION : TITRE À TROUVER : "Establishing a Niche"

SECTION : À RÉDIGER :

- Pour démystifier :
- 1. Chat-oriented ou Task-oriented ;
- 2. Conception par approche symbolique ou par approche générative ;
- 3. Besoin d'annotation pour avoir des données de qualité.

1.3 (*Gap*)

SECTION : TITRE À TROUVER : "Gap"

SECTION : À RÉDIGER :

- Peu de travaux sur la conception d'un jeu de données : en recherche les données sont publiques, en entreprises les données sont privées ;
- Nombreuses pistes d'amélioration, mais peu sont exploitées en pratique ;
- Défis d'organisation, de gestion de coûts, de complexité, de qualité, ...
- Conception trop manuelle, experts pas à leur place, ...

1.4 (*Occupying the Niche*)

SECTION : TITRE À TROUVER : "Occupying the Niche"

SECTION : À RÉDIGER :

- Besoin de recentrer l'activité des experts métiers ;
- Besoin d'assister la conception d'un jeu de données ;
- Nous proposons donc une méthode itérative et semi-supervisée.

TRANSITION : Annonce du plan ?

Chapitre 2

Revue de littérature sur la tâche d'annotation en intelligence artificielle

L'annotation (ou labellisation) de données est une tâche essentielle de l'apprentissage automatique, permettant de décrire des jeux de données nécessaires à l'entraînement et à l'évaluation de modèles d'intelligence artificielle. Dans ce chapitre, nous allons traiter les points suivants :

- Définir à quoi correspond une tâche de labellisation, et détailler la vaste organisation technique et méthodologique autour d'un projet d'annotation de données ;
- Présenter un aperçu des difficultés principales que peut rencontrer un projet d'annotation, ainsi que certaines techniques conçues pour s'en prévenir ;
- Expliquer le contexte industriel dans lequel ce doctorat s'inscrit et montrer l'intérêt d'assister la conception des jeux de données d'entraînement.

Sommaire

2.1	Présentation théorique de l'annotation	4
2.1.1	Définition et objectifs de l'annotation de données	4
2.1.2	Exemples de tâches d'annotations	5
2.1.3	Bilan concernant la présentation de l'annotation	10
2.2	Organisation usuelle d'un projet d'annotation	10
2.2.1	Étapes clés du cycle d'annotation	10
2.2.2	Portraits des acteurs intervenant sur un projet d'annotation	15
2.2.3	Choix du logiciel d'annotation	17
2.2.4	Bilan concernant l'organisation d'un projet d'annotation	18
2.3	Aperçu des nombreux défis de l'annotation	19
2.3.1	Défis concernant le besoin de qualité des données	20
2.3.2	Défis concernant la complexité inhérente à la tâche d'annotation	27
2.3.3	Défis concernant les différences de comportements d'annotation	32
2.3.4	Bilan concernant les nombreux défis de l'annotation	35
2.4	Contexte du doctorat : comment assister la conception d'une base d'apprentissage pour un agent conversationnel bancaire en français ?	35

2.1 Présentation théorique de l'annotation

Tout d'abord, introduisons quelques définitions pour appréhender le concept d'« annotation » et donnons quelques exemples pour comprendre les enjeux qui y sont associés.

2.1.1 Définition et objectifs de l'annotation de données

2.1.1.a Qu'est que l'« apprentissage automatique » ?

Nous proposons la définition suivante inspirée de l'ACM (*Association for Computing Machinery*) : l'« apprentissage automatique » (ou « Machine Learning ») est une branche de l'intelligence artificielle dédiée au développement de méthodes permettant à l'ordinateur de **reproduire une tâche par l'exemple** : il n'est donc pas explicitement programmé pour réaliser cette tâche, mais il l'« apprend » à l'aide d'un modèle mathématique. Cet apprentissage peut être *supervisé* (l'interprétation des exemples est fournie par un humain), *non-supervisé* (la machine déduit l'interprétation des données sans intervention humaine) ou *semi-supervisé* (mélange des deux précédentes approches).

Le *Machine Learning* permet ainsi d'automatiser l'analyse et la manipulation de certains phénomènes complexes tels que le langage, l'observation visuelle, la détection d'anomalies, le traitement acoustique, ...

i Pour information : Si vous voulez revoir les bases de l'apprentissage automatique, des livres comme ZHOU, 2021 ou RASCHKA et MIRJALILI, 2019 traitent des notions principales et de leur mise en application.

2.1.1.b Qu'est qu'un « corpus d'entraînement » ?

Pour concevoir un modèle en apprentissage automatique, il nous faut un ensemble d'exemples (textes, images, sons, vidéos, ou tout autre relevé d'informations) permettant de capturer le phénomène à appréhender : cela nous aide à la fois à le décrire et à mieux le comprendre. Nous utilisons alors les termes « **corpus d'entraînement** », « **jeu de d'entraînement** » ou « **base d'apprentissage** » pour désigner cet ensemble de données.

Il est important de noter qu'un corpus n'est qu'un échantillon de taille finie d'un phénomène pouvant être infini ou indénombrable. Il est donc d'usage de valoriser cet échantillon s'il est « **représentatif** » du phénomène qu'il décrit, c'est-à-dire s'il capture bien le large panel de variations que peuvent prendre les données (BIBER, 1993).

i Pour information : Nous discuterons davantage de cette notion de représentativité dans la SECTION 2.3.1.A. D'autre part, si vous voulez mieux comprendre cette notion de corpus, vous pouvez vous référer à SINCLAIR, 2004 issu du livre *Developing Linguistic Corpora* (WYNNE, 2004).

2.1.1.c Qu'est que l'« annotation » ?

Les données d'un corpus manquent parfois d'information pour bien cerner un phénomène, il est alors nécessaire de faire intervenir un humain pour introduire des connaissances supplémentaires qui ne sont pas explicitement présentes dans ces données. Nous appelons « **annotation** »

(ou « **étiquetage** », « **labellisation** ») cette tâche consistant à décrire les données d'un corpus, et nous distinguons ainsi les données dites « **brutes** » (utilisées par les approches non-supervisées) des données dites « **annotées** » (utilisées par les approches supervisées) en fonction de l'absence ou de la présence d'un complément d'informations.

Les informations renseignées peuvent porter sur la donnée entière ou seulement sur une partie seulement, peuvent concerner des variables catégorielles (ensemble fini) ou numériques (ensemble infini), et peuvent aussi être cumulatives ou mutuellement exclusives. Dans la littérature, GARSIDE et al., 1997 présente l'annotation comme la tâche permettant de donner une « **valeur ajoutée** » aux données ; de son côté, LEECH, 2004 précise que l'annotation est une action d'« **interprétation** » qui aide à la compréhension et à la reproduction d'un phénomène, mais aussi au contrôle du comportement des modèles d'apprentissage automatique.

2.1.2 Exemples de tâches d'annotations

Les définitions données dans la section précédente peuvent paraître abstraites car il est difficile de dépeindre la vaste diversité d'applications nécessitant des données labellisées. En effet, une tâche d'annotation répond toujours à un besoin précis, mais il y a une telle multiplicité de types de données (*données tabulaires, textuelles, visuelles, auditives, ...*) et de cas d'usages (*prédition d'une valeur numérique (tâche de régression), prédition d'une catégorie (tâche de classification), détection d'objets (tâche d'extraction), création de nouvelles données (tâche de génération), ...*) qu'une unique définition ne peut être que purement théorique.

Ainsi, nous estimons qu'il est préférable de compléter ces définitions par quelques exemples concrets. Nous pourrons ainsi mieux dresser le portrait d'une tâche d'annotation, avec ses intérêts et ses complications. Pour cela, nous allons prendre le thème de la bande dessinée et explorer ensemble différents cas d'usage qui pourraient intéresser un auteur, un libraire ou un lecteur.

2.1.2.a Estimation du prix d'une bande dessinée.

Les acheteurs et les vendeurs de bandes dessinées s'interrogent forcément sur le juste prix de l'oeuvre qu'ils veulent acquérir ou céder. Répondre à cette question avec précision nécessite diverses informations, à la fois sur l'oeuvre (comme son identification ou l'avis de ses lecteurs), sur le document en tant que tel (comme son état de conservation), mais aussi sur le prestige de son édition (éditions originales ou de collection). Sans un regard d'expert, il est possible de trouver certaines oeuvres rares vendues pour presque rien sur le marché d'occasion, ou à l'inverse voir certaines BD être achetées à prix d'or alors que le document est en piteux état.

Afin d'aiguiller les acquéreurs, il est possible d'utiliser un modèle de **régression**¹ permettant de prédire le prix d'une BD à partir des différentes métadonnées à disposition. Mais pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples de transactions avec leur prix de vente. Nous pouvons structurer l'ensemble des informations nécessaire dans un tableau, et la tâche d'annotation consiste alors à renseigner pour chaque transaction :

- l'identification complète de la BD (titre, auteur, édition, ...),
- l'état du document grâce à un regard d'expert (l'état peut par exemple être défini par une variable catégorielle dont les valeurs seraient "Mauvais état", "Bon état", "Très bon état", "Neuf") ;
- le prix de la BD, estimé ou réel (défini par une variable numérique).

1. Pour plus de détails sur la régression : voir la revue de MAALOUF, 2011 ; voir un exemple basé sur la méthode des moindres carrés dans ZDANIUK, 2014.

Un exemple de résultat d'annotation de ces données est disponible dans la TABLE 2.1.

 Exemples :

Collection	N° : Titre	Édition	Note	État	Prix (€)
Lucky Luke	01 : La mine d'or de Dick Digger	1949	3.2/5	Très bon	5 000,00
Lucky Luke	12 : Les cousins Dalton	1958	4.3/5	Bon	40,00
Lucky Luke	12 : Les cousins Dalton	1962	4.3/5	Très bon	65,00
Lucky Luke	12 : Les cousins Dalton	1985	4.3/5	Très bon	6,00
Lucky Luke	15 : L'évasion des Dalton	1960	4.1/5	Mauvais	3,00
...					

TABLE 2.1 – Exemple d'annotation du prix de vente de bandes dessinées en fonction de leur édition, de la note de leur lecteurs et de leur état (source : <https://www.bedetheque.com/serie-213-BD-Lucky-Luke.html>).

Ainsi, si quelqu'un s'intéresse au prix d'une nouvelle bande dessinée pour lequel il n'y a pas de référence tarifaire, il peut interroger le modèle de régression qui proposera un prix en accord avec les exemples dont il dispose dans sa base d'apprentissage.

 **Pour information :** De manière équivalente, il est possible de faire de la régression dans d'autres domaines, notamment pour prédire un volume, une surface, une quantité, ... La tâche d'annotation consistera à chaque fois à renseigner la valeur numérique à prédire en fonction des différentes données à disposition.

2.1.2.b Classification de l'état d'une bande dessinée à partir d'une photo.

Il est d'usage d'adapter le prix de vente d'un produit en fonction de son état, et nous avons intégré ce facteur dans l'estimation du prix d'une bande dessinée (voir exemple précédent). Cependant, l'état de conservation n'est pas une notion objective et chacun peut avoir des références différentes. Au final, c'est souvent un libraire qui détermine si l'oeuvre est en bon ou en mauvais état, et, sans un regard d'expert, nous pouvons omettre un détail ou nous tromper lors de notre appréciation.

Afin de nous aider à estimer l'état d'une bande dessinée, il est possible d'utiliser un modèle de classification² permettant, à partir d'une image, d'affecter à chaque BD une catégorie pré-définie (par exemple : "Mauvais état", "Bon état", "Très bon état", "Neuf"). Pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples d'images de BD associées avec leur catégorie d'état. La tâche d'annotation peut alors consister à renseigner pour chaque couverture de bande dessinée la catégorie d'état qui lui correspond le plus.

Un exemple d'annotation de la classification de l'image est disponible dans la FIGURE 2.1.

2. Pour plus de détails sur la classification : voir les revues de AIZED AMIN SOOFI et ARSHAD AWAN, 2017 ou de KOTSIANTIS et al., 2006 ; voir un exemple basé sur les machine à vecteurs de support (SVM) dans CORTES et VAPNIK, 1995.



FIGURE 2.1 – Exemple d'annotation de l'état d'une BD (ici : MORRIS et GOSCINNY, 1950 et MORRIS et GOSCINNY, 1952). La première est en très bon état (couverture comme neuve, tranches légèrement usées, pages intactes) tandis que la seconde est en mauvais état (couverture usée, dos abimée, traces sur les pages, ...).

Ainsi, si quelqu'un s'interroge sur l'état d'une bande dessinée en sa possession, ce modèle peut identifier l'état le plus probable d'après les exemples disponibles dans sa base d'apprentissage.

i Pour information : De manière équivalente, il est possible de faire de la classification sur d'autres données, comme par exemple la classification de textes pour identifier la langue de l'ouvrage. Dans l'exemple ci-dessous, les catégories proposées sont "Français", "Anglais" et "Allemand", et la tâche d'annotation consiste ici à associer à chaque texte une catégorie de langue.

- | | |
|---|-------------------|
| « <i>Les cousins Dalton ont dévalisé la diligence.</i> » | ⇒ Français |
| « <i>The Dalton cousins robbed the stagecoach.</i> » | ⇒ Anglais |
| « <i>Die Dalton-Cousins haben die Postkutsche ausgeraubt.</i> » | ⇒ Allemand |

2.1.2.c Identification d'une bande dessinée à partir de sa couverture.

Identifier une bande dessinée n'est pas toujours facile, et recopier l'ensemble des informations l'identifiant peut prendre du temps. Les libraires ou les collectionneurs désirant faire l'inventaire des ouvrages en leur possession peuvent ainsi y passer de nombreuses heures, avec le risque de faire des erreurs lors de l'inscription des bande dessinée dans leur registre.

Afin d'aider les collectionneurs, il est possible d'utiliser un modèle de **reconnaissance optique des caractères (OCR)**³ pour extraire automatiquement les informations importantes présentes sur les couvertures d'une BD à identifier. Pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples de pages de couverture avec la position et la valeur des informations pertinentes à extraire. La tâche d'annotation peut alors consister à renseigner pour chaque couverture de bande dessinée :

- la position des informations en l'encadrant sur l'image (avec un rectangle par exemple) ;
- la valeur écrite dans l'encadré sur l'image.

Un exemple d'annotation de textes dans une image est disponible dans la FIGURE 2.2.



Ainsi, si quelqu'un veut identifier une nouvelle bande dessinée, il peut interroger le modèle d'extraction de caractères pour récupérer les informations textuelles présentes dans la couverture, à l'image des exemples disponibles dans sa base d'apprentissage.

i Pour information : De manière équivalente, il est possible de réaliser une reconnaissance d'entités nommées (NER)⁴pour extraire les informations citées dans un texte. Dans l'exemple ci-dessous, les types d'entités présentes sont "personnage", "métier", "argent", "lieu" et "date". La tâche d'annotation consiste ici à identifier la position et le type de chaque entité présente.

« *Lucky Luke* (**personnage**), le *cow-boy* (**métier**) solitaire, a attrapé les *Dalton* (**personnage**) à *Coyote Gulch* (**lieu**) et a touché **50.000\$** (**argent**) en les livrant au pénitencier (**lieu**). Ils se sont évadés le *jeudi suivant* (**date**). »

3. Pour plus de détails sur l'OCR : voir la revue de BERCHMANS et KUMAR, 2014 ou de AWEL et ABIDI, 2019.

2.1.2.d Interprétation audio d'une bande dessinée.

Il est de plus en plus commun de trouver des livres disponibles avec une lecture audio. Ces audio-livres, réalisés par une personne ou synthétisés par l'ordinateur, peuvent être à visée éducative ou simplement disponibles pour le loisir. Dans le cadre de notre exemple sur le thème des bandes dessinées, peu d'entre elles disposent d'une lecture audio. Une idée serait donc d'interpréter une lecture audio de ces bandes dessinées en synthétisant la voix des doubleurs de leurs adaptations télévisées (ou simplement d'un narrateur si l'oeuvre n'a pas été portée à l'écran).

Dans le but de créer ces audio-BD, nous pourrions envisager d'utiliser des modèles de **synthèse vocale** (TTS)⁵ pour générer automatiquement la lecture des bulles d'une bande dessinée. Pour entraîner de tels modèles, il est nécessaire d'avoir une base d'apprentissage contenant des exemples d'audios prononcés par chacun des personnages (ici : les doubleurs de l'adaptation télévisée) avec la transcription de leur paroles pour chaque audio. La tâche d'annotation peut alors consister à renseigner le personnage et les paroles qu'il a prononcées.

Un exemple d'annotation phonétique est illustré dans la FIGURE 2.3.

Exemples :

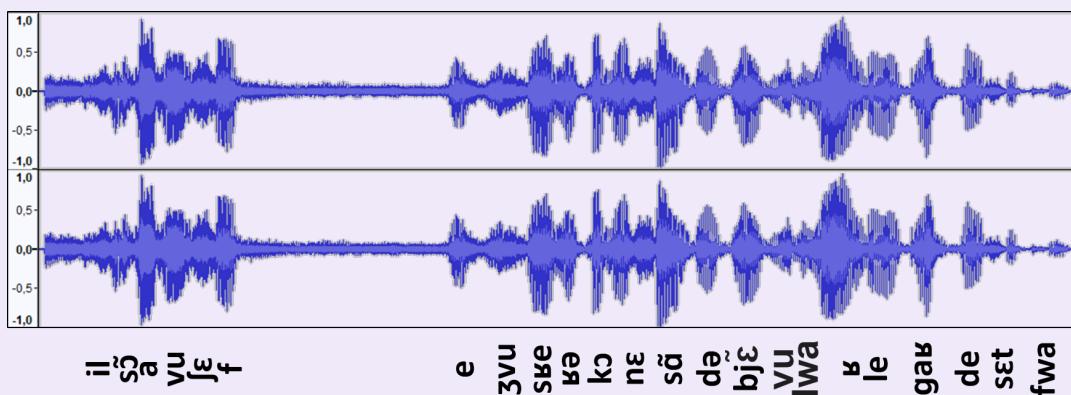


FIGURE 2.3 – Exemple de paroles prononcées dans un audio. Ici, la voix de Lucky Luke est interprétée par Jacques THEBAULT. Le texte annoté, c'est-à-dire celui prononcé dans l'audio, est « Ils sont à vous chef, et j'veous s'rurai reconnaissant de bien les garder cette fois. ». Les phonèmes en alphabet phonétique international associés à chaque séquence de l'audio sont disponibles si besoin.

Ainsi, nous pourrions consulter le modèle de synthèse vocale d'un personnage (ici : celui de **Lucky Luke**) avec un nouveau texte à prononcer pour en obtenir une lecture audio dont la voix se rapproche des enregistrements de la base d'apprentissage (ici : celle de Jacques THEBAULT).

4. Pour plus de détails sur la reconnaissance d'entités nommées (NER) : voir les revues de GOYAL et al., 2018 ou de LI et al., 2022.

5. Pour plus de détails sur la synthèse vocale : voir la revue de KOTHADIYA et al., 2020 ; voir un exemple d'architecture neuronale end-to-end dans MU et al., 2021.

i Pour information : Nous pourrions compléter le cas d'usage (1) en extrayant automatiquement le texte d'une planche de BD par OCR, (2) en détectant automatiquement le personnage prononçant la bulle de BD par classification, puis (3) en générant du texte à prononcer par le personnage par synthèse vocale. Bien entendu, la conception et l'enchaînement de ces différents modèles sont plutôt complexes, et chaque tâche de *Machine Learning* demande ses propres données annotées pour construire une base d'apprentissage.

2.1.3 Bilan concernant la présentation de l'annotation

Points à retenir :

- « Annoter » une donnée consiste à **ajouter un complément d'information** pour pouvoir mieux interpréter puis reproduire un phénomène.
- Le type d'annotation à réaliser **dépend du problème à traiter** : régression, classification, extraction d'information, génération ou synthèse de données, ...
- L'ensemble des données annotées peut être utilisé pour concevoir un modèle d'« **apprentissage automatique** » : il est alors appelé « **corpus d'entraînement** ».

2.2 Organisation usuelle d'un projet d'annotation

Dans la section précédente, nous avons présenté l'importance d'avoir des données annotées pour entraîner un modèle de *Machine Learning*. Nous allons maintenant détailler l'organisation de cette tâche d'annotation, identifier les compétences nécessaires aux intervenants du projet ainsi que les fonctionnalités essentielles des outils de labellisation.

2.2.1 Étapes clés du cycle d'annotation

L'organisation d'un projet d'annotation, établie aujourd'hui comme une référence, est proposée par PUSTEJOVSKY et STUBBS, 2012 et est complétée dans STUBBS, 2013. Les auteurs y formalisent la conception et l'amélioration **cyclique** d'un modèle de *Machine Learning*. Ce cycle est appelé cycle MATTER en référence aux six étapes de conception qui le composent : **Modelize, Annotate, Train, Test, Evaluate et Revise**. Ces étapes sont schématisées en FIGURE 2.4 et nous détaillons chacune d'entre elles ci-dessous.

Notes de l'auteur : Nous conseillons la lecture de FINLAYSON et ERJAVEC, 2016 pour son excellente revue de littérature qui détaille pas à pas le cycle MATTER tout en dressant la liste des points importants de chacune des étapes.

2.2.1.a Concevoir la base d'apprentissage (**Modelize, Annotate**).

Pour obtenir un modèle de *Machine Learning*, il faut avoir une base d'apprentissage de qualité. Comme nous l'avons dit précédemment, cela commence par disposer d'un ensemble de données d'exemples qui représente fidèlement les différentes facettes du problème à modéliser

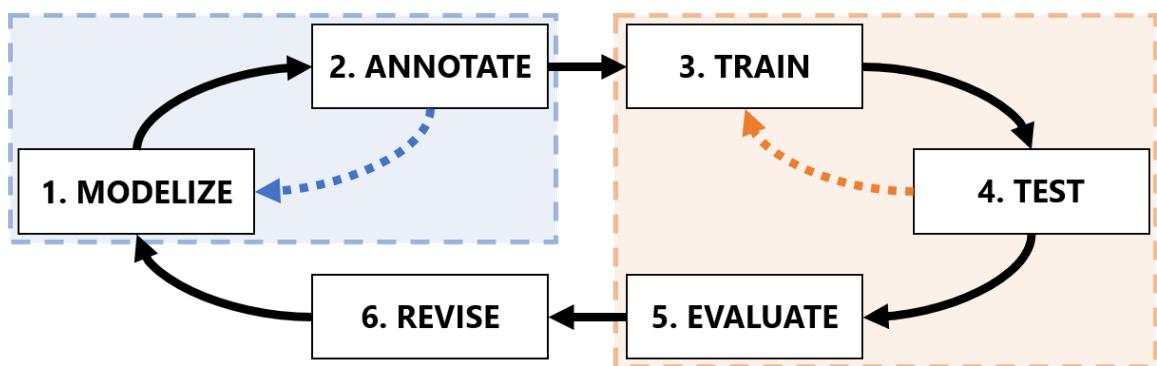


FIGURE 2.4 – Cycle MATTER structurant un projet d'annotation en six étapes principales : Modelize, Annotate, Train, Test, Evaluate et Revise.

Le carré bleu identifie le mini-cycle *MAMA* durant lequel la modélisation est adaptée en cours d'annotation, et le carré orange identifie le mini-cycle *Train-Test* lors de la conception du modèle.

(voir SECTION 2.3.1.A). Une phase de **collecte** de données est alors organisée : cette collecte peut se baser sur des extractions de bases de données ou de sites internet à disposition, sur des enquêtes réalisées auprès d'utilisateurs finaux, ou encore sur les avis éclairés d'experts du problème. Certaines données peuvent aussi être artificiellement créées afin de compléter la collecte pour les aspects du problème difficile à observer. Une fois la collecte terminée, ces données brutes ont besoin d'être annotées pour pouvoir être exploitées.

Afin de garantir la qualité de cette labellisation, **il est important de ne pas précipiter la tâche d'annotation**. En effet, l'objectif de cette tâche peut considérablement changer en fonction du phénomène à décrire, des données à disposition et de la finalité du modèle de *Machine Learning* à entraîner. Il est donc fortement conseillé de bien **modéliser le problème**, c'est-à-dire de définir l'objectif de l'annotation, de clarifier en amont les modalités et les attendus de cette tâche, et de préciser les règles que devront suivre les opérateurs.

PUSTEJOVSKY et STUBBS, 2012 précisent notamment deux concepts importants de cette phase :

- la **modélisation** du problème, représentant de manière abstraite l'objectif à atteindre et décrivant ainsi la logique générale de l'annotation dans un *schéma d'annotation* ;
- les **spécifications**, compilant dans un *guide d'annotation* l'ensemble des règles concrètes à respecter pour mettre en application la modélisation.

Pour résumer cette distinction, la modélisation représente *quoi* annoter (*objectif, définition, valeurs possibles, ...*) alors que les spécifications décrivent *comment* annoter (*règles d'attribution, exemples et contre-exemple, règles de format, ...*).

Exemples : PERROTIN et al., 2018, s'intéressant à la classification des conversations d'assistance en ligne en actes de dialogues, décrit son guide d'annotation dans ASHER et al., 2017. On y retrouve (1) la modélisation avec la présentation des étiquettes possibles à annoter, et (2) les spécifications avec les définitions concrètes, des exemples, des restrictions d'attribution, et la gestion des données non pertinentes.

Dans nos exemples précédents (cf. SECTION 2.1.2.B), nous avions modélisé le problème

de classification de l'état d'une bande dessinée en quatre classes : "Mauvais état", "Bon état", "Très bon état", "Neuf". Il faudrait désormais rédiger les spécifications avec des définitions concrètes et quelques exemples pour guider un annotateur, notamment pour l'aider à distinguer "Bon état" de "Très bon état".

Bien entendu, il n'est pas toujours facile de modéliser un problème ni de rédiger un guide d'annotation adéquat. Nous reviendrons plus tard sur les caractéristiques de cette tâche pouvant introduire de la complexité (voir SECTION 2.3), mais il est important de souligner d'emblée les points élémentaires suivants :

- le besoin d'*inter-opérabilité* et de *ré-utilisabilité* : un projet d'annotation est toujours un investissement coûteux, il serait donc regrettable de perdre ou de ne pas pourvoir ré-utiliser ces données après ce projet. Par conséquent, il faut réfléchir au format des données ainsi qu'aux types de détails à fournir pour être sûr de pouvoir toujours exploiter les données si la modélisation évolue légèrement ou si un futur projet désire en bénéficier ;
- la balance entre *généralité* et *spécificité* : le niveau de détail requis dépend sans conteste du problème à modéliser : annoter trop peu de détails ne permet pas d'exploiter les données, mais en annoter trop peut rapidement complexifier la tâche et introduire des erreurs. Il faut donc trouver le juste milieu pour réaliser un travail de qualité.

Q Exemples : Dans la classification de langues exposée en SECTION 2.1.2.B, nous avons annoté chaque texte grâce à trois classes : "Français", "Anglais" et "Allemand".

- par souci d'*inter-opérabilité*, nous pourrions plutôt utiliser la norme ISO 639-3 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2007), soit les codes "**fra**", "**eng**" et "**deu**", afin de standardiser l'annotation et ainsi pouvoir partager plus facilement les données labellisées avec d'autres projets ;
- afin de présenter un cas simple, nous avions proposé un modèle avec trois langues communes pour une bande dessinée d'origine belge. Toutefois, nous aurions pu *spécialiser* davantage notre modèle en fonction des variations régionales en prenant en compte le Corse ("**cos**") ou le Wallon ("**wln**"). Cette distinction peut être essentielle pour certaines saga publiées dans ces langues (comme *Astérix & Obélix*), mais peut simplement être une source de confusion pour les autres (comme *Lucky Luke*).

i Pour information : Pour aider à concevoir le guide d'annotation et afin de se poser les bonnes questions, DIPPER et al., 2004 dresse une liste de définitions et de recommandations à prendre en considération. Bien que ces conseils soient issus du traitement de données linguistiques, ils permettent d'identifier les sections importantes d'un guide d'annotation en fonction des attentes des différents acteurs de l'annotation (*l'auteur*, *l'annotateur*, *l'explorateur de données*, ...) et de les rédiger en suivant certaines règles simples (*introduire les objectifs*, *ordonner les règles par complexité*, *traiter en premier les cas par défaut*, *trier les valeurs des variables catégorielles par ordre alphabétiques*, ...). Des exemples reconnus pour leur bonne conception y sont notamment cités.

Lorsque le guide d'annotation est rédigé, la **phase de labellisation** peut commencer. Cette tâche est traditionnellement réalisée par un groupe d'experts choisi en fonction de leurs connaissances sur le problème à caractériser (*dans nos exemples sur les bandes dessinées, ce serait plutôt des libraires ou des collectionneurs*). Après leur avoir expliqué l'objectif de leur travail et partagé les règles de labellisation contenues dans le guide, les annotateurs se partagent les données et réalisent chacun une partie du corpus d'apprentissage.

i Pour information : C'est généralement à ce stade que la théorie rencontre la pratique : certaines règles d'annotation peuvent difficilement être applicables, certains données peuvent être ambiguës ou hors-sujet, deux annotateurs peuvent aussi avoir des avis différents sur l'annotation la plus adéquate, ... Il est aussi important de rappeler que l'annotation est un acte d'interprétation, et que les données sont donc labellisées par un humain dont l'avis n'est pas infaillible. PUSTEJOVSKY et STUBBS, 2012 introduisent donc le premier sous-cycle MAMA en référence à la boucle entre *Modelize* et *Annotate* qui peut avoir lieu tant que le guide d'annotation n'est pas adapté aux données manipulées ou que différents points de vues opposent les annotateurs.

Par exemple, lors de l'annotation de la transcription audio en SECTION 2.1.2.D, il peut y avoir une voix principale accompagnée de plusieurs voix en arrière plan : une première adaptation du guide serait de clarifier si ces voix secondaires doivent être transcrrites ou ignorées, voire si l'audio entier doit être considéré comme inexploitable. La réponse à cette question dépend bien entendu du phénomène à décrire et de l'objectif du modèle de *Machine Learning* à entraîner : dans notre cas, nous pourrions probablement annoter uniquement la voix principale et ignorer l'audio si le bruit gêne la compréhension.

À la fin de l'annotation (ou du cycle MAMA), le corpus d'entraînement est disponible pour concevoir un modèle de *Machine Learning*.

2.2.1.b Concevoir le modèle (*Train, Test, Evaluate*).

La phase d'entraînement du modèle est l'étape centrale de l'apprentissage automatique. Toutefois, comme l'apprentissage se base sur des méthodes statistiques, il est important d'introduire une phase de test et d'évaluation pour s'assurer des performances du modèle obtenu. Il est donc courant de considérer une boucle de raffinement du modèle tant que les performances n'ont pas atteint un seuil acceptable.

En pratique, il est d'usage de créer trois jeux de données à partir de la base d'apprentissage qui vient d'être annotée :

- le jeu d'**entraînement** : c'est sur cette partie des données que le modèle de *Machine Learning* est conçu ;
- le jeu de **développement** (ou de validation) : le modèle entraîné est évalué sur ce jeu de donnée pour étudier son comportement, identifier ses forces et ses faiblesses, et ainsi permettre de le comparer à d'autres modèles entraînés pour cette même tâche ;
- le jeu de **test** : le modèle retenu est évalué sur ce jeu de test pour déterminer ses performances réelles.

Ainsi, le modèle représente la connaissance présente dans le jeu **entraînement**, il est étudié puis affiné grâce au jeu de **développement**, et est finalement évalué en fonction de ses performances sur le jeu de **test**. Il est encore une fois difficile d'être exhaustif sur les analyses et les

métriques à considérer car elles dépendent fortement du type de problème que le modèle tente de résoudre. Une métrique basique est l'**Accuracy** (ou taux de bonnes prédictions), décrivant simplement le nombre de fois que le modèle a fait une bonne proposition sur l'ensemble du test. Suivant le problème et le type de données, d'autres métriques usuelles peuvent être utilisées comme le **MSE** (*Mean Squared Error*) pour la prédiction de variables numériques (voir WALLACH et GOFFINET, 1987), le **f1-score** pour les variables catégorielles (voir SASAKI, 2007) ou le **WER** (*Word Error Rate*) pour la transcription de textes (voir MCCOWAN et al., 2005). Dans tous les cas, une règle d'or est de ne pas utiliser le jeu de test lors de la phase de développement pour éviter tout biais de sur-apprentissage⁶.

À la fin de ce cycle, le modèle de *Machine Learning* est à disposition, et ses performances théoriques sont celles obtenues sur le jeu de test.

2.2.1.c Revoir la base d'apprentissage (*Revise*).

Pour terminer cette boucle, il est parfois nécessaire d'envisager de corriger son modèle en remettant en cause la modélisation du problème et l'annotation des données. VOORMANN et GUT, 2008 formalisait en effet ce besoin de réviser la conception d'une base d'apprentissage en observant les lacunes du modèle obtenu, et PUSTEJOVSKY et STUBBS, 2012 évoque certaines révisions nécessaires de la modélisation dès la phase d'annotation (voir sous-cycle MAMA dans la FIGURE 2.4).

Divers pistes peuvent mener à une évolution de la base d'apprentissage :

- le modèle de *Machine Learning* peut avoir de mauvaises performances, malgré son affinage lors de la phase de développement, ou peut manquer d'adaptabilité sur des données réelles ;
- la modélisation ou l'annotation peuvent devenir obsolètes car le phénomène modélisé évolue dans le temps ;
- un cas d'usage non identifié jusqu'à présent nécessite de nouvelles données pour être pris en compte ;
- le modèle peut ne pas convenir aux utilisateurs finaux par manque d'ergonomie ou à cause d'une utilisation non prévue initialement ;
- un nouvel algorithme de *Machine Learning* a priori plus performant requiert une modélisation différente pour traiter le problème.

Q Exemples : Pour illustrer nos propos, prenons la tâche d'estimation du prix d'une bande dessinée (cf. SECTION 2.1.2.A) : il se peut que les prix annoté sur les transactions ne soient plus d'actualité à cause de l'inflation, et que les données doivent être ré-annotées pour prendre en compte les nouvelles valeurs du marché.

D'autre part, la modélisation en tant que telle peut aussi être impactée : par exemple, dans le cadre de la classification de l'état d'une bande dessinée à partir d'une photo (cf. SECTION 2.1.2.B), on pourrait constater à l'usage qu'il manque une catégorie "Très mauvais état" nécessaire pour trier d'emblée toute BD indigne à la vente.

Enfin, il est possible que le modèle se comporte mal sur certaines données. Par exemple lors de l'identification d'une bande dessinée à partir de sa couverture (cf. SECTION 2.1.2.C), certains textes du décors pourraient être extraits à tort (comme le texte de la pancarte

6. Pour plus de détails sur le sur-apprentissage : voir COLLINS, 2017

« Saloon » dans la FIGURE 2.2). Il faudra peut-être adapter l'annotation pour identifier les textes à ne pas extraire (avec une classe de rebus par exemple).

Nous bouclons ainsi le cycle MATTER qui préfigure le besoin d'une amélioration continue d'un modèle de *Machine Learning* pour que celui-ci soit le plus adapté à son environnement d'utilisation.

2.2.2 Portraits des acteurs intervenant sur un projet d'annotation

Au cours du cycle MATTER, nous pouvons constater que divers acteurs interviennent pour concevoir la base d'apprentissage et entraîner un modèle de *Machine Learning*. Cette diversité de métiers qui gravitent autour du traitement automatique des données semble difficile à détailler, tant à cause de leur grand nombre que de leurs subtiles différences. Pour avoir un aperçu, vous pouvez consulter les offres d'emplois du marché actuel (voir TEAM DATASCIENTEST, 2022 ou DATA BIRD, 2023) ou certaines formations professionnelles (voir ISOZ, 2017) pour pouvoir faire la distinction entre *data scientist*, *data analyst*, *data librarian*, *data journalist*, *data architect*, *data engineer*, *data steward*, *data archivist*, ou encore *machine learning engineer*...

Afin d'avoir une approche moins commerciale de ces métiers, nous proposons plutôt de dresser les compétences requises aux diverses phases du cycle, à l'image de RADOVILSKY et al., 2018 qui présente les acteurs de la science des données grâce à quatre groupes de compétences :

1. les compétences **métiers** : elles sont liées aux connaissances et à l'expertise sur le phénomène à modéliser ou le problème à résoudre. Ce sont grâce à ces compétences qu'un acteur peut être apte à annoter une donnée ou à qualifier la pertinence de la prédiction d'un modèle de *Machine Learning*. Les métier(s) associé(s) sont : l'**expert métier** (*business expert*) ;
2. les compétences **analytiques** : elles concernent entre autres la modélisation du problème, la gestion des données, et les analyses statistiques sur les biais et les performances. Ce sont grâce à ces compétences qu'un acteur peut concevoir le guide d'annotation, estimer le taux d'accords inter-annotateurs, ou encore réaliser l'évaluation statistique d'un modèle de *Machine Learning*. Les métier(s) associé(s) sont : l'**analyste des données** (*data analyst*) ou le **scientifique des données** (*data scientist*) ;
3. les compétences **techniques** : elles portent sur l'ingénierie autour du modèle de *Machine Learning*, comme le choix du meilleur algorithme d'entraînement et réglage fin des hyper-paramètres, l'archivage des différentes versions du modèle ainsi que son déploiement dans un environnement de production. Les métier(s) associé(s) sont : le **scientifique des données** (*data scientist*), l'**ingénieur en Machine Learning** (*Machine Learning Engineer*) ou l'**architecte des données** (*data architecte*) ;
4. les compétences en **gestion** ou en **communication** : elles permettent d'aborder le cadrage du projet et la définition des objectifs, ainsi que diverses aptitudes transverses comme l'établissement de rapports, la gestion de projet, la vérification des normes, ... Les métier(s) associé(s) sont : le **chef de projet** (*project leader*) ou le **responsable de la protection des données** (*data protection officer*) .

i Pour information : On peut compléter cette vision par compétences avec la vision donnée par FORT, 2017, selon laquelle il y a trois types d'experts lors d'un projet d'annotation :

- les **experts du corpus** de données, ayant par exemple les connaissances sur les bandes dessinées, s'approchant donc de compétences **métiers** ;
- les **experts de l'annotation**, ayant par exemple les connaissances sur l'annotation de textes dans une image, s'approchant donc de compétences **analytiques** ;
- et les **experts de la tâche** de *Machine Learning*, ayant par exemple les connaissances sur les techniques d'**OCR**, s'approchant donc des compétences **techniques**.

Ainsi, durant le cycle MATTER, nous pouvons voir les compétences ci-dessus se compléter :

1. la conception de la **base d'apprentissage** (étapes *Modelize* et *Annotate*) nécessite :
 - des compétences de **gestion** pour cadrer l'objectif du modèle à entraîner, et ainsi définir l'objectif auquel doit répondre l'annotation de données ;
 - des compétences **analytiques** pour proposer une modélisation stable du phénomène et un guide d'annotation précis pour limiter les biais de conception ;
 - des compétences **métiers** pour vérifier que la proposition de modélisation est pertinente vis-à-vis du cas d'usage, mais aussi pour réaliser l'annotation des données.
2. la conception du **modèle de Machine Learning** (étapes *Train*, *Test*, *Evaluate*) nécessite :
 - des compétences **analytiques** pour gérer les jeux de données (*entraînement*, *développement*, *test*) et évaluer les performances statistiques du modèle ;
 - des compétences **techniques** pour manipuler l'écosystème de développement du modèle, régler les hyper-paramètres, versionner les changements, et planifier la distribution du modèle sur un environnement de production ;
 - des compétences de **gestion** pour s'assurer du respect des normes et des caractères privée ou confidentielle de certaines données.
3. la **révision** de la base d'apprentissage (étape *Revise*) nécessite :
 - des compétences **métiers** pour identifier le manque de pertinence du modèle vis-à-vis de certains cas d'usage ;
 - des compétences **analytiques** pour disserter des performances réelles du modèle face à des données de production et remettre en question les précédents choix de modélisation pour espérer améliorer le modèle.

On notera que les compétences transverses de **gestion** ou **communication** ne sont pas spécifiques à une étape du cycle MATTER (*le cadrage, la gestion de projet et l'établissement de rapports étant réalisés tout au long du projet*), alors que les compétences **métier** et **techniques** n'interviennent généralement pas au même moment du cycle : autrement dit, des experts métiers

croisent rarement des experts techniques et ne partagent donc que très rarement leurs connaissances.

2.2.3 Choix du logiciel d'annotation

Pour terminer la description de l'organisation d'un projet d'annotation, attardons nous sur le choix du logiciel à utiliser pour labelliser les données. S'il est vrai qu'une diversité d'applications existent pour répondre aux besoins des annotateurs, il est important de noter que l'absence de certaines fonctionnalités essentielles peuvent gêner le projet d'annotation.

Nous faisons référence à FINLAYSON et ERJAVEC, 2016 pour dresser ci-dessous une liste des fonctionnalités principales (voire essentielles) d'un logiciel d'annotation. Pour simplifier la lecture, nous proposons de regrouper ces fonctionnalités dans les catégories suivantes :

- répondre au **besoin d'annotation** : cette fonctionnalité est bien entendu obligatoire, car un logiciel ne permettant pas d'annoter vos données ne vous sera daucune utilité. Cette remarque semble être une évidence, mais nous nous permettons aussi d'étendre l'avertissement aux logiciels n'étant pas destinés à votre besoin d'annotation mais qui peuvent être détournés pour y répondre indirectement : de telles contournements peuvent introduire des biais et offrent généralement une expérience utilisateur assez médiocre ;
- intégrer le **guide d'annotation** : ce livrable issu de la phase de modélisation du cycle MATTER doit être facilement accessible par les annotateurs car il contient la documentation et les instructions à appliquer lors de la labellisation. Les logiciels permettant d'intégrer directement ces définitions (*avec exemples et contre-exemples*) ainsi que les règles d'annotation (*comme les labels mutuellement exclusifs, les détails obligatoires, ...*) ont donc un net avantage ergonomique pour respecter la modélisation définie et ainsi garantir la qualité de la base d'apprentissage ;
- autoriser l'**annotation multiple** et l'**annotation multi-modale** : il est fréquent de devoir annoter une même donnée suivant des modélisations ou des paradigmes différents pour répondre à plusieurs cas d'usage (*en prenant l'exemple de l'annotation d'images, on peut détourner les objets présents, identifier les textes inscrits, proposer une ou plusieurs catégories générales, proposer une description textuelle, ...*) ou encore de devoir annoter des données de différents types (*en combinant texte, image et voix comme dans l'annotation des sous-titres d'une vidéo*). Ainsi, les logiciels offrant la possibilité de labelliser plusieurs informations et de manipuler plusieurs types de données sont donc pertinents afin de centraliser les annotations et de pouvoir facilement les réutiliser ;
- évaluer la **qualité de l'annotation** : les erreurs d'annotation et les divergences d'opinions sur la modélisation sont inévitables. Il est donc appréciable de pouvoir les identifier, soit sur la base d'une comparaison directe entre deux annotateurs, soit en comparant avec l'annotation la plus probable issue d'une base de référence. Il peut aussi être intéressant de pouvoir calculer les scores d'accord inter-annotateurs sur un même échantillon de données pour estimer la qualité de la base d'apprentissage, de pouvoir corriger les erreurs lors de revues ou encore de trancher les cas de conflits apparents lors d'avis discordants ;
- permettre l'**inter-opérabilité** technique : il peut être frustrant de ne pas pouvoir réutiliser des annotations d'un projet à l'autre car le format de stockage n'est pas compatible. Les logiciels prenant donc en considération plusieurs format de données (*PNG/JPG, MP3/WAV, XLSX/XML/JSON, ...*) et respectant les standards de la tâche d'annotation lors des imports

- et exports sont donc à privilégier. De plus, il est conseillé de ne pas écrire directement les annotations dans les données (« *[Lucky Luke]/(personnage)*, le *[cow-boy]/(métier)* solitaire, a ... »), mais de les stocker dans des fichiers séparés pour garder une meilleure gestion et permettre les annotations multiples ;
- gérer le **flux de travail** et le **suivi de projet** : certaines fonctionnalités simples sont nécessaires à l'organisation de l'équipe d'annotation. Cela peut comprendre la répartition de la charge de travail, l'historisation des changements pour permettre les retours arrières, la possibilité d'émettre des appels d'aide ou d'écrire des commentaires sur les choix d'annotation, ou encore l'accompagnement des nouveaux annotateurs lors de leur monté en compétence ;
 - favoriser le **confort de l'annotateur** : le logiciel choisi sera utilisé au quotidien par l'équipe d'annotation, il semble donc essentiel de leur offrir une expérience utilisateur agréable pour réaliser leur tâche. Cela peut passer par une customisation de l'interface utilisateur afin d'être adapter à l'objectif d'annotation et par le paramétrage de raccourcis claviers. Simplifier l'accès et l'installation du logiciel peut aussi s'avérer utile pour favoriser son adoption, en favorisant par exemple les applications web permettant plus facilement le travail collaboratif ;
 - permettre des **annotations et d'analyses avancées** : la littérature scientifique regorge de techniques permettant d'assister un annotateur dans son travail (*pré-annotation, apprentissage actif, visualisation, interaction, ...*). Nous détaillons plusieurs de ces techniques dans la SECTION 2.3.

Considérant la diversité de cas d'usage d'annotation, une liste exhaustive des outils de labellisation n'est bien entendu pas possible. Nous tenons toutefois à présenter quelques exemples illustrés dans la FIGURE 2.5.

 **Notes de l'auteur :** De part notre expérience, nous constatons malheureusement que peu de projets industriels utilisent un outil d'annotation dédié, souvent au profit d'outils rudimentaires comme des traitements de textes ou des tableurs tels que Microsoft Excel (MICROSOFT CORPORATION, 2018). Une étude serait à mener pour étudier cette tendance et expliquer le manque d'intérêt porté aux outils spécialement conçus pour les tâches d'annotations. Peut-être que ces derniers s'adaptent mal aux particularités des divers projets industriels, expliquant ainsi l'utilisation d'outils simplistes mais flexibles. Ou alors est-ce par méconnaissance des difficultés et des avantages potentiels de l'annotation que ces outils aux fonctionnalités avancées ne sont pas employés ?

2.2.4 Bilan concernant l'organisation d'un projet d'annotation

Points à retenir :

- ✓ En général, un projet d'annotation est **organisé en cycle** (MATTER) au cours duquel nous réalisons une modélisation abstraite des données que nous formalisons dans un guide (*Modelize*), nous appliquons ce guide pour labelliser notre base d'apprentissage (*Annotate*), puis nous entraînons et testons un modèle de *Machine Learning* (*Train*,

2.3. Aperçu des nombreux défis de l'annotation

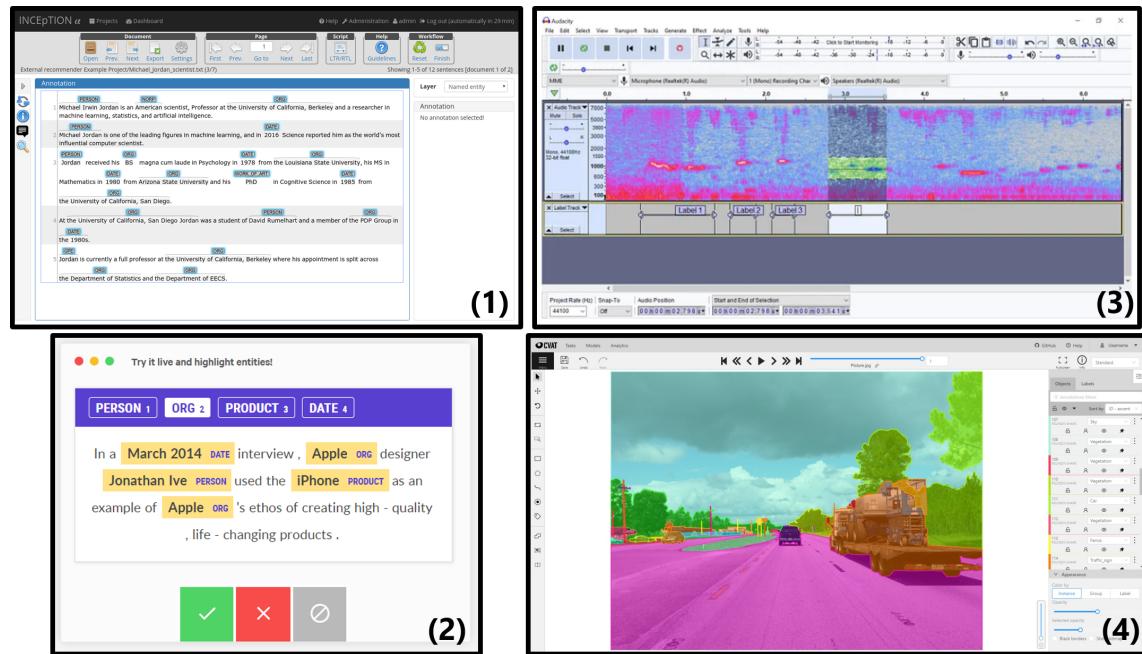


FIGURE 2.5 – Quatre exemples d’outils d’annotation : (1) INCEPTION pour le texte (KLIE et al., 2018), (2) prodigy pour le texte ou l’image (MONTANI et HONNIBAL, 2017), (3) Audacity pour l’audio (AUDACITY TEAM, 2000 et (4) CVAT pour l’image (CVAT.AI CORPORATION, 2019).

Test et Evaluate). Ensuite, l’évaluation du modèle peut mener à une révision de la modélisation des données en fonction des performances obtenues (**Revise**) ;

- Un tel projet d’annotation nécessite une **diversité de connaissances et de compétences** qui peuvent être réparties en quatre catégories : métier, analytique, technique et gestion/communication ;
- Un tel projet nécessite aussi un **outil d’annotation dédié** possédant certaines fonctionnalités essentielles comme la possibilité d’intégrer le guide d’annotation, le besoin de contrôler la qualité des annotations, la capacité à réaliser des annotation multiples ou multimodales, ou encore l’assimilation d’éléments de gestion de projet.

2.3 Aperçu des nombreux défis de l’annotation

Comme nous avons pu l’apercevoir dans les sections précédentes, le cycle d’annotation recèle de zones d’ombres pouvant introduire des complications dans la conception d’une base d’apprentissage (BALEDENT, 2023). Pour aborder cette partie, nous allons voir :

- qu’il y a une forte pression sur la **qualité des données** devant constituer le corpus d’entraînement (cf. SECTION 2.3.1) ;
- que ce standard de qualité entretient une **complexité inhérente** aux étapes de modélisation et d’annotation (cf. SECTION 2.3.2) ;

- et que cette complexité provoque des **différences de comportements** chez les annotateurs (cf. SECTION 2.3.3).

En détaillant chacun de ces trois points, nous discuterons de l'ensemble de techniques et bonnes pratiques mises en avant dans la littérature pour limiter les désagréments d'un projet d'annotation. Nous identifierons aussi les freins récurrents pouvant intervenir dans les mises en application industrielles.

2.3.1 Défis concernant le besoin de qualité des données

Comme nous l'avons défini en SECTION 2.1.1.A, l'« **apprentissage automatique** » regroupe un ensemble de techniques dont l'objectif est de reproduire une tâche **par l'exemple** : il est donc normal de porter une attention particulière aux données utilisées, car la qualité du modèle de *Machine Learning* va fortement dépendre de la qualité de sa base d'apprentissage. Nous allons ici détailler trois défis actuels concernant cette création d'un jeu de données.

2.3.1.a Problèmes de représentativité

La phase de collecte de données est une étape importante du projet d'annotation. Malheureusement, la littérature scientifique associée à cette tâche est assez légère alors que c'est précisément à ce moment que ce joue une caractéristique cruciale de la future base d'apprentissage : la **représentativité** du phénomène à modéliser.

Cette notion est assez ambiguë, notamment car le terme technique « **représentatif** » fait écho à un mot de la vie courante qui peut avoir plusieurs sens. Dans KRUSKAL et MOSTELLER, 1979a et CLEMMENSEN et KJAERSGAARD, 2022, plusieurs usages communs de ce terme sont recensés :

- « *assertive claim* » : l'opérateur déclare que ses données sont représentatives du problème sans apporter d'arguments. Bien entendu, cette option est à bannir car elle n'apporte aucune information et peut cacher des vices de conception du modèle ;
- « *absence or presence of selective force* » : la représentativité du phénomène est supposée en sélectionnant des données de **manière aléatoire** et en limitant le nombre de critères de sélection à ceux nécessaires pour l'étude réalisée ;
- « *miniature* » : aussi appelée **sélection stratifiée**, cette approche consiste à dire qu'un échantillon est représentatif d'un phénomène si la proportion de chacune de ses parties y est respectée (*par exemple, un sondage peut respecter la répartition des tranches d'âge d'une population*) ;
- « *typical/ideal case* » : cette définition consiste à représenter chaque partie du phénomène par un **exemple emblématique** ou un **exemple moyen** (*par exemple, on peut illustrer l'univers de la bande dessinée française par un numéro de la saga Astérix & Obélix*) ;
- « *population coverage* » : ici, la représentativité est associée à la présence **exhaustive** de l'ensemble des caractéristiques importantes du phénomène avec au moins un exemple par caractéristique (*par exemple, dans l'arche de Noé, il y devait y avoir au moins un couple de chaque animaux*).

Pour compléter ces définitions, KRUSKAL et MOSTELLER, 1979b incitent sur le besoin de **définir avec précision la méthode d'échantillonnage** plutôt que l'échantillon lui-même : en effet, il est important de caractériser le phénomène à modéliser, de définir l'objectif de la collecte de données et de détailler comment cette collecte va être réalisée. CLEMMENSEN et

KJAERSGAARD, 2022 introduisent à leur tour trois mesures pour aider à caractériser une collecte : la *réflexion* (est-ce l'échantillon respecte la distribution de la population ?), la *couverture* (est-ce que l'échantillon illustre la diversité de la population ?) et la *présence de représentants* (est-ce que l'échantillon contient les exemples emblématiques de la population ?). De telles informations sont essentielles pour pouvoir juger de la valeur d'un échantillon par rapport à un cas d'usage et déterminer s'il est réutilisable dans pour une autre application.

Q Exemples : Illustrons nos propos avec la classification de l'état d'une bande dessinée à partir d'une photo (voir SECTION 2.1.2.B). Afin de représenter correctement le cas d'usage, nous pourrions collecter des exemples de BD couvrant l'ensemble des dégradations fréquemment identifiées par les libraires (couvertures froissées, pages déchirées, couleurs délavées, ...) et les intégrer de manière proportionnelle dans la base d'apprentissage. Nous pourrions aussi nous assurer de la présence de cas emblématiques permettant de catégoriser les BD en "Mauvais état", "Bon état", "Très bon état", "Neuf".

Toutefois, cette base d'apprentissage ne sera plus représentative si nous voulons détecter la langue de la bande dessinée (auquel cas, une répartition par langue sera a priori plus adéquate).

La description d'un phénomène reste cependant une **tâche difficile**, notamment lorsque que celui-ci possède un ensemble vaste et abstrait de caractéristiques à décrire. Nous comptons généralement sur la loi des grands nombres pour espérer dresser un portrait fidèle du phénomène, mais cela impose parfois de traiter des **immenses quantités de données**.

Q Exemples : Considérons le traitement du langage : le vocabulaire employé peut concerner des dizaines de milliers de mots, il existe des variantes régionales et divers jargons techniques, certains termes peuvent avoir plusieurs sens et des expressions peuvent dépendre de leur contexte (comme l'humour ou les critiques). Pour représenter ces spécificités (listées de manière non exhaustive), une base d'apprentissage devra contenir de nombreux exemples afin de capturer les différents aspects du langage à traiter. On peut citer par exemple **MLSUM: The Multilingual Summarization Corpus** (SCIALOM et al., 2020), une base de 1.5 millions d'articles de journaux sur 5 langues pour entraîner un modèle de résumé automatique, ou encore **The Multilingual Amazon Reviews Corpus** (KEUNG et al., 2020), une base de 1.26 millions de commentaires de produits sur 6 langues pour entraîner un modèle de classification de la note d'un commentaire sur 5 étoiles.

Toutefois, la masse de données ne résout pas toujours tous les problèmes de représentativité. Une des difficultés récurrentes concerne les aspects peu fréquents d'un phénomène qui se retrouvent ainsi **sous-représentés** : si l'enjeu du modèle à concevoir consiste justement à détecter ou reproduire ces aspects, il peut être intéressant de volontairement biaiser les proportions du corpus d'entraînement pour mieux les illustrer. À l'inverse, des cas communs ou fréquents peuvent être **sur-représentées** : il est parfois nécessaire de limiter leur occurrence dans le corpus d'apprentissage pour ne pas concevoir un modèle véhiculant des généralités ou des stéréotypes. Dans les deux cas, **toute intervention va introduire un biais** : l'opération doit donc être réfléchie et judicieusement réalisée pour contribuer à la finalité du modèle, d'où l'intérêt de bien la documenter pour faire entendre ce que vous voulez signifier par « échantillon représentatif ».

Q Exemples : D'une part, considérons le besoin de détecter la langue d'une bande dessinée (voir SECTION 2.1.2.B). Il est fort probable que la base d'apprentissage contiennent peu de données sur les parutions en langues régionales (en Corse, en Wallon, en Alsacien, ...). Nous pouvons donc être amenés à ajouter des exemples supplémentaires pour espérer mieux les détecter et ainsi augmenter la *couverture* de notre jeu de données.

D'autre part, regardons l'analyse du modèle de **Stable Diffusion** réalisée par NICOLETTI et BASS, 2023 sur la génération de portraits de personnes fictives à partir d'une description textuelle de leur métier. L'étude montre que le modèle tant générer des portraits d'hommes à la peau blanche pour le métier d'architecte ou d'ingénieur, des femmes pour le rôle de concierge ou encore des personnes à la peau noire pour illustrer la classe ouvrière (voir la FIGURE 2.6).

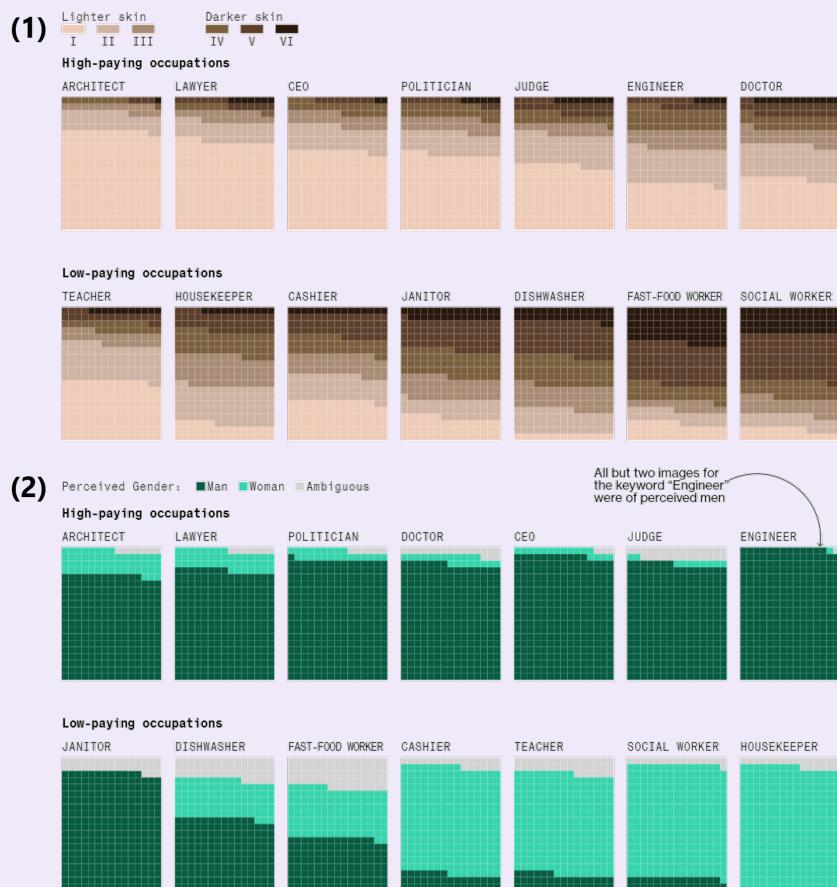


FIGURE 2.6 – Répartitions des teintes de peaux (1) et des genres (2) par métier lors de génération de portrait avec *Stable Diffusion* (étude menée par NICOLETTI et BASS, 2023).

Ici, ce modèle dépeint les inégalités de notre société en perpétuant des stéréotypes, et cela ouvre la question suivante : veut-on vraiment reproduire à l'identique cette représentation ?

💡 Idées : Une piste pour équilibrer efficacement les corpus d'entraînement et permettre de corriger leurs biais consiste à utiliser des **données synthétiques** (JAIPURIA et al., 2020). Ces données peuvent être créées manuellement ou être générées automatiquement (voir SHORTEN et al., 2021 pour une revue de génération de textes et SHORTEN et KHOSHGOFTAAR, 2019 pour la génération d'image). Bien entendue, une telle approche doit restée réfléchie pour ne pas introduire davantage de biais et pour répondre à un objectif précis d'équilibrage du jeu de données.

Une dernière difficulté concerne l'**obsolescence** des données au cours du temps. En effet, peu de phénomènes sont immuables, et il est courant de devoir questionner la représentativité d'un problème pour prendre en compte des nouveaux aspects ou corriger des caractéristiques devenues inexactes.

🔍 Exemples : Ce problème est particulièrement impactant si nous voulons estimer le prix d'une bande dessinée (voir SECTION 2.1.2.A) car les œuvres vont gagner ou perdre de la valeur avec le temps. Par exemple, en 1949, le premier album de *Lucky Luke* devait s'acheter pour quelques francs, alors qu'aujourd'hui, cette édition est estimée à plusieurs milliers d'euros.

De manière similaire, le traitement du langage constate aussi des évolutions au cours du temps (*nouveaux mots de vocabulaire, nouvelles expressions, influence de langues étrangères, ...*), imposant ainsi la mise à jour des jeux de données.

2.3.1.b Problèmes de bruits

La qualité d'une base d'apprentissage dépend fortement du bruit qu'elle contient. Ce bruit est inévitablement inséré lors de la collecte : d'une part, la méthode de collecte elle-même peut en introduire (*instrument de mesure faillible, erreur humaine, ...*) ; d'autre part, par soucis de représentativité, le bruit intrinsèque du phénomène va être capturé (*forte variabilité, présence d'irrégularité, ...*). Un échantillon de données va donc forcément devoir se confronter

Nous nous inspirons de MAHARANA et al., 2022 et de ALASADI et BHAYA, 2017 pour dresser une liste de problèmes récurrents sur les données suite à une collecte :

- présence de **données non pertinentes** par rapport au cas d'usage : une collecte automatique ou aléatoire peut sélectionner des données n'ayant pas ou peu de rapport avec le phénomène à modéliser. Garder de telles données peut introduire de la confusion dans le modèle à entraîner ;
- dégradation des données par des **variations parasites** : comme décrit en introduction de cette partie, les bruits peuvent être intrinsèques au phénomène ou être introduits par la méthode de collecte. Il convient de lisser ou limiter ces bruits pour ne pas perturber le modèle ;
- **absence de valeurs** descriptives essentielles : cette absence peut venir d'une erreur de mesure, d'une méconnaissance du phénomène à caractériser, ou simplement d'un oubli. Cependant, un trou de description peut rendre inutile une donnée si cela concerne une caractéristique importante du phénomène ;

- présence d'**incohérences** ou d'**ambiguïté** entre les données : les données sont rarement catégoriques et plusieurs interprétations sont parfois possibles (voir la discussion sur la subjectivité en SECTION 2.3.3.A). Toutefois, des contradictions entre les données peuvent pénaliser le modèle à entraîner.

Q Exemples : Pour illustrer ces problèmes, considérons la tâche d'estimation du prix d'une bande dessinée (voir SECTION 2.1.2.A) :

- une donnée concernant le prix d'un roman ou d'une encyclopédie avoir été inséré par mégarde et pourrait être considéré comme données non pertinentes pour ce cas d'usage ;
- un changement de typographie (*majuscules, minuscules, accents, ponctuation*) dans l'écriture d'un titre pourrait mal identifier une bande dessinée ;
- une information peut ne pas avoir été renseigné lors d'une transaction (l'année d'édition par exemple), alors que c'est une caractéristique importante de la prise de décision ;
- une même bande dessinée (avec les mêmes caractéristiques) peut avoir été vendue à deux prix radicalement différent, introduisant ainsi une légère ambiguïté dans les données.

Des problèmes similaires peuvent impacter le traitement du texte (*fautes de grammaticales ou syntaxiques, ambiguïtés ou sémantiques, omissions, ...*), des images (*flous, mauvais cadrages, colorimétries gênantes, ...*) et de l'audio (*bruits en arrière plan, saturations, coupures inopportunes, mots mâchés, ...*). Quelques exemples sont présentés ci-dessous dans la FIGURE 2.7.

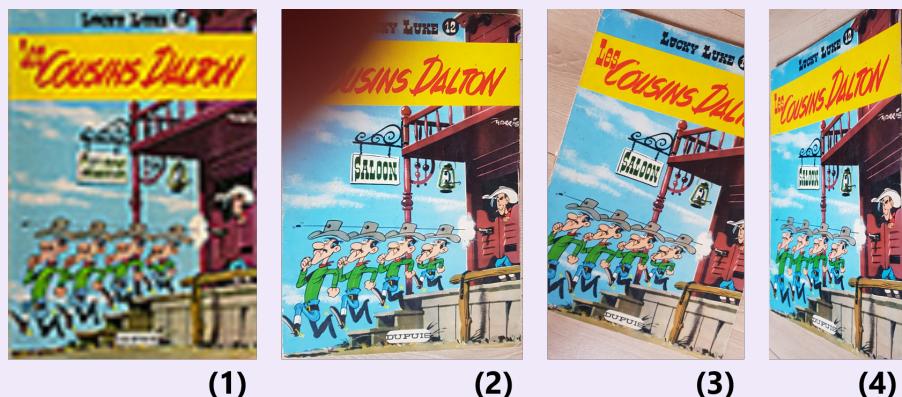


FIGURE 2.7 – Exemple de bruits courant perturbant l'analyse d'une image : (1) le flou, (2) un doigt sur le capteur, (3) un problème de cadrage et (4) un problème d'angle de vue.

Pour limiter l'impact du bruit dans les données, ALASADI et BHAYA, 2017 structurent les étapes de prétraitement de données entre quatre catégories :

- le **nettoyage des données** : cette étape consiste à compléter les données manquantes (*en prenant la valeur moyenne par exemple*), à filtrer les données aberrantes ou inintéressantes,

- et surtout à lisser le bruit dans les données en gommant les variations parasites ;
- **l'intégration des données** : dans certains cas, plusieurs sources de données sont disponibles, il peut être intéressant de croiser ces sources de données pour augmenter la consistance de la base d'apprentissage et identifier les incohérences ;
 - **le formatage des données** : pour exploiter facilement les données, certaines transformations sont parfois nécessaires pour limiter les ambiguïtés dues au leur format (*par exemple : normaliser une valeur entre 0 et 1, limiter les caractères spéciaux dans un texte*) ;
 - **la réduction des données** : en réalisant une analyse approfondie des données, on peut quelques fois constater que certaines caractéristiques présentes sur les données sont peu utiles et peuvent être supprimer pour réduire la complexité de la base d'apprentissage.

BALEDENT, 2023 rappelle néanmoins que le document source (ou ici : la donnée brute) doit rester accessible pour la phase d'annotation pour ne pas manquer d'information potentiellement intéressante.

 **Exemples :** Reprenons les problèmes évoqués précédemment sur la tâche d'estimation du prix d'une bande dessinée :

- une donnée inintéressante peut simplement être supprimée ;
- normaliser les champs décrivant une bande dessinée en passant tout en minuscules limiterait les chances de mal l'identifier ;
- une année d'édition manquante pourrait être identifiée par l'étiquette **inconnue** et un prix manquant pourrait être complété par la moyenne du prix des BD ayant les mêmes caractéristiques ;
- un prix faussé pourrait être identifié comme incohérent en analysant les prix des BD ayant les mêmes caractéristiques ;

2.3.1.c Problèmes d'exploitation et de diffusion

En plus des difficultés techniques sur la réalisation d'une collecte de données, il y a aussi contraintes législatives et stratégiques à prendre en compte.

D'une part, il faut considérer le fait que certaines données sont protégées et ne peuvent pas être collectées ou exploitées librement. C'est le cas de données soumises aux droits de **propriétés intellectuelles** qui empêchent cet usage : on peut citer par exemple LOIGNON, 2023 qui évoque le levé de bouclier des médias français contre l'utilisation de leur article pour entraîner des modèles de langues, mais aussi LES ECHOS, 2023 qui questionne la violation du **droit d'auteur** lorsque qu'un modèle est entraîné sur l'œuvre d'un artiste et qu'il est capable de la reproduire.

Ces limites concernent aussi la Réglementation Générale européenne sur la Protection des Données (General Data Protection Regulation, 2016) restreignant les **collectes et usages non consenties** de données personnelles. Ainsi, il n'est pas possible d'entraîner n'importe quel modèle sur n'importe quelles données, et une telle contrainte impose de manipuler les données en garantissant l'anonymat et la confidentialité des personnes consentantes.

Q Exemples : Considérons la conception d'un modèle de synthèse vocale pour réaliser une audio-BD (voir SECTION 2.1.2.D). D'une part, un tel projet nécessiterait déjà de demander les droits d'adaptation pour entraîner un modèle de synthèse vocale avec les doublage de l'adaptation télévisée de *Lucky Luke*. D'autre part, l'utilisation de la voix d'une personne se confrontera probablement à plusieurs restrictions pour éviter que ce modèle ne soit détourné pour des usages non consentis par le doubleur.

Pour aller plus loin, cette notion de confidentialité touche les données personnelles, mais aussi le **caractère stratégique** d'une organisation. En effet, dans le monde académique, les données manipulées sont le plus souvent publiques et peuvent être employées pour contribuer à la recherche scientifique. Mais dans le secteur industriel, les jeux de données sont liés au domaine d'activité de l'entreprise : ils ont généralement requis un investissement conséquent en temps et en moyens, et ils représentent donc son avantage concurrentiel (*par leur spécificité, leur caractère secret ou novateur, leur qualité compétitive, ...*). Il est donc rare de voir une entreprise partager ses jeux de données car elle pourrait perdre un de ses atouts stratégiques.

Une solution est de ce tourner les jeu de données **accessibles en Open Source**. Plusieurs plateformes mettent en effet à disposition des données ou des modèles, comme *Hugging Face* (HUGGING FACE, 2016) ou *Zenodo* (RE3DATA.ORG, 2013). Toutefois, deux limites subsistent à l'usage de ces données :

- les données mises à disposition publiquement sont souvent assez générales et ne reflète pas la spécificité des cas d'usage de l'entreprise, limitant ainsi leur intérêt ;
- les données publiques ne sont pas forcément ouvertes à un usage commercial (*elles peuvent par exemple employé la licence CC BY-NC 4.0, CREATIVE COMMONS, 2013*), restreignant ainsi les seules applications au domaine de la recherche et veille scientifique.

Pour ne pas faire de faux pas juridique, RAJBAHADUR et al., 2022 propose une approche pour vérifier si une licence permet d'exploiter un jeu de données.

i Pour information : Pour terminer nous mentionnons aussi une proposition de législation européenne concernant la futures réglementation des modèles d'intelligence artificielle (The Artificial Intelligence Act, 2021). Cette loi concerne les quatre objectifs suivants :

- « veiller à ce que les systèmes d'IA mis sur le marché de l'Union et utilisés soient sûrs et respectent la législation en vigueur en matière de droits fondamentaux et les valeurs de l'Union » ;
- « garantir la sécurité juridique pour faciliter les investissements et l'innovation dans le domaine de l'IA » ;
- « renforcer la gouvernance et l'application effective de la législation existante en matière de droits fondamentaux et des exigences de sécurité applicables aux systèmes d'IA » ;
- « faciliter le développement d'un marché unique pour des applications d'IA légales, sûres et dignes de confiance, et empêcher la fragmentation du marché » .

Un besoin de traçabilité des données et des modèles se fait donc sentir, renforçant les recommandations à documenter et détailler les traitement et choix pour garantir la représentativité et la qualité des données des bases d'apprentissage.

2.3.2 Défis concernant la complexité inhérente à la tâche d'annotation

Selon BALEDENT, 2023, deux types de complexités sont à différencier :

- la **complexité propre au phénomène** que l'on veut modéliser : nous avons pu apercevoir celle-ci dans la SECTION 2.3.1.A, notamment en considérant la nécessité de collecter une grande quantité de données pour représenter la diversité du phénomène et le besoin de contrôler les bruits pour en assurer la qualité, ...
- et la **complexité propre à la procédure d'annotation** mise en place : cet aspect est abordée brièvement dans la SECTION 2.2 en exposant le besoin d'établir une modélisation du problème avec son guide d'annotation, le recours à un processus itératif pour affiner les problèmes de conception et différences d'interprétation (cycles MATTER et MAMA), ou encore l'intervention de différents utilisateurs ayant des compétences différentes.

Dans cette section, nous allons voir comment analyser et mesurer cette complexité, puis nous nous intéresserons aux coûts qu'elle peut engendrer lors de la conception du jeu de données.

2.3.2.a Modélisation difficile du phénomène

Comme nous l'avons énoncé lors de la présentation au cours du cycle MATTER en SECTION 2.2.1.A, la modélisation est une étape importante du processus de conception car elle permet de clarifier ce qu'il faut annoter et avec quel(s) objectif(s). Toutefois, plusieurs aspects rendent cette tâche difficile à réaliser.

Tout d'abord, le projet d'annotation concerne généralement un cas d'usage précis dont la spécificité requiert l'intervention d'experts ayant des **connaissances métiers**. Cependant, de tels experts n'ont pas forcément les compétences analytiques requises pour établir une représentation abstraite de leurs connaissances. En effet, diverses questions sont à discuter concernant la modélisation à choisir :

- quel standard existe-t-il qui soit réutilisable dans ce cas d'usage ?
- sera-t-elle stable et explicite à l'annotation ?
- sera-t-elle traduite en un seul ou en plusieurs modèle(s) de *Machine Learning* ?
- sera-t-elle maintenable avec de grands volumes de données ?

Pour y répondre, ateliers de conception sont généralement organisés avec les experts, permettant ainsi d'identifier les notions pertinentes à intégrer dans la modélisation finale du problème. Si le phénomène est intrinsèquement complexe, il est possible néanmoins que ces ateliers se réalisent en mode essai-erreur (à l'image du mini-cycle MAMA) jusqu'à trouver une modélisation qui puisse convenir, rendant cette tâche particulièrement pénible.

Q Exemples : Pour se rendre compte de cette difficulté, essayons par exemple de détailler l'ensemble des actions que peut entreprendre un robot conversationnel en domotique

(comme Alexa, ALEXA INTERNET, 2018), et modélisons ensuite les diverses instructions verbales pouvant déclencher ces actions.

💡 **Idées :** Quelques pistes approches peuvent être mises en place pour assister la modélisation d'un problème :

- les approches **non supervisées** : elles consistent à laisser la machine extraire et structurer automatiquement la connaissance contenue dans les données collectées. Par exemple pour le traitement du texte, nous pouvons citer les méthodes d'exploration utilisant le regroupement automatique (*clustering*) comme KMeans (MACQUEEN, 1967), ou la modélisation thématique (*topic modeling*) comme LDA (BLEI et al., 2003) ;
- les approches **semi-supervisées** : elles consistent à travailler main dans la main avec la machine pour explorer les données collectées. On peut citer par exemple les méthodes d'apprentissage actif (*active learning*) dont une revue est proposée par SETTLES, 2010 ;
- les approches **itératives** : elles consistent à concevoir d'abord un petit modèle dont le périmètre est limité, puis d'étendre pas à pas son périmètre suite aux retours de son utilisation. Une organisation fait bien entendu écho à la philosophie du cycle MATTER.

Toutefois, plusieurs handicaps pénalisent ces approches, notamment lors de déséquilibrage ou de bruits dans les données, ou lorsque l'interprétation nécessite une forte connaissance métier.

💬 **Notes de l'auteur :** De part notre expérience, nous constatons que peu d'approches non-supervisées ou semi-supervisées sont utilisées, notamment parce que leurs résultats sont jugés peu pertinents sur des phénomènes complexes. Les experts métiers réalisent alors cette étape de modélisation manuellement, absorbant la complexité de cette tâche en organisant de nombreux ateliers de conception en mode essai-erreur.

C'est particulièrement le cas lors de la concevoir de la base d'apprentissage d'un assistant conversationnelle (*task-oriented*) : la tâche de modélisation consiste généralement à identifier l'intention formulée dans une demande utilisateur en fonction du verbe d'action (« *je veux résERVER un bilLET* », « *jouE moi du jazz!* », « *comment éDITER une attestation ?* »). Cependant, la vaste diversité du langage permet rarement d'identifier les thématiques présents dans une collecte de données. L'étape de modélisation est alors réalisée manuellement par les experts du domaine couvert par l'assistant conversationnel.

Lorsqu'une modélisation acceptable du phénomène a été définie, il est nécessaire de la structurer dans le but de **rédiger le guide d'annotation** associé (voir SECTION 2.2.1.A). NÉDELLEC et al., 2006 a pu montrer que la clarté de ce guide impacte directement la qualité des annotations : il est donc important de rédiger celui-ci avec soin pour donner à l'annotateur une vision de son objectif (FORT et al., 2009). Cela requiert toutefois de solides compétences analytiques et pédagogiques, notamment pour définir, organiser et transmettre ce vaste ensemble des règles sans introduire de biais ni de confusion.

i Pour information : Nous rappelons que DIPPER et al., 2004 a dressé une liste de recommandations pour rédiger un guide d'annotation fiable.

Malgré tout, BALEDENT, 2023 rappelle que l'établissement d'une modélisation d'un phénomène **relève d'un choix**, et que celui-ci est par conséquent subjectif. En effet, plusieurs représentations peuvent convenir à un même cas d'usage, et celui retenu devra être celui qui correspondra le plus à la finalité de votre modèle. Ce choix, aussi avisé soit-il, entrera inévitablement en friction avec certaines données collectées qui s'inséreront difficilement au sein de la modélisation choisie. Lors de la rédaction du guide d'annotation, il faut donc trouver l'équilibre entre entre la rigueur (*pour fiabiliser la qualité de la labellisation*) et la flexibilité (*pour pouvoir s'adapter aux données contrariantes*).

? **Exemples :** Dans la tâche de transcription d'un audio pour réalisé un modèle de synthèse vocale (SECTION 2.1.2.D), on constate que *Lucky Luke* mâche certains de ses mots (« [...] j'veux s'rai reconnaissant [...] ») : préférez-vous annoter strictement ce qu'il dit (*au risque de complexifier le modèle*) ou corriger la transcription (*au risque de ne pas reproduire la prononciation exacte du personnage*) ?

2.3.2.b Estimation de la complexité

Dans FORT et al., 2012, une approche analytique est proposée pour mesurer la complexité des tâches de modélisation et d'annotation. Six dimensions sont détaillées, réparties en trois axes :

- la complexité de **localisation** de l'annotation (*discrimination, délimitation*) : y a-t-il plusieurs parties à annoter dans une donnée ? quel est le ratio entre les parties à annoter et les parties potentiellement annotables ? est-ce que ces parties sont clairement délimitées ou ont-elles des bornes floues ?
- la complexité de **caractérisation** de l'annotation (*expressivité, dimensionnalité, ambiguïté*) : doit-on annoter une variable catégorielle ou numérique ? s'il y a plusieurs variables entre elles, y a-t-il des relations entre elles ? leur cardinalité est-elle finie ou infinie ? quelle est la proportion de confusion ou de désaccord d'interprétation de cette modélisation ?
- la complexité de **situation** de l'annotation (*poids du contexte*) : a-t-on besoin d'informations complémentaires pour être capable d'interpréter la donnée ?

Une autre manière de mesurer la complexité d'une tâche d'annotation consiste à **évaluer les différences entre annotateurs**. En effet, GUT et BAYERL, 2004 ont notamment montré qu'un nombre élevé de désaccords de labellisation peut mettre en avant une ambiguïté de modélisation, une différence d'interprétation entre annotateurs, ou encore une difficulté à saisir pleinement la subtilité des données manipulées : dans tous les cas, ces désaccords témoignent de la complexité de la tâche. Pour mesurer cet accord, différents scores ont été développés comme peut en témoigner la revue de ARTSTEIN et POESIO, 2008. Nous reviendrons plus en détails sur ces différences de comportement dans la SECTION 2.3.3.

i Pour information : L'un des plus scores les plus utilisé est α de *Krippendorff* (KRIPPENDORFF, 2004). Le score est normalisé entre 0 (*aucun accord*) et 1 (*accord parfait*). On considère un accord fort lorsque le score est supérieur à 0.8, et l'accord reste acceptable si la valeur est supérieure à 0.667.

Q Exemples : Considérons l'identification d'une bande dessinée à partir de sa couverture, pour laquelle l'annotation de textes présents sur une image sont nécessaires pour entraîner un modèle de reconnaissance optique de caractères (voir SECTION 2.1.2.C). Nous pouvons voir que :

- la localisation des annotations est assez évidente car l'information est visuelle : (1) En général, il y a peu de textes sur une couverture de BD, l'information est donc facilement identifiable (*bonne discrimination*) ; (2) Cependant, il peut y avoir de légères différences sur la position exacte des boxes entourant les textes (*délimitation délicate*) ;
- la caractérisation concerne du texte mais son annotation est explicite : (1) L'annotation concerne du texte et des nombres, il y a donc une grande variabilité parmi les valeurs possibles (*forte expressivité*). (2) Plusieurs informations sont attendues, comme le nom de la collection, l'auteur, le titre, ou la date de parution (*dimensionnalité modérée*) ; (3) Toutefois, il y a assez peu de doute sur les valeurs à indiquer car ces dernières sont explicitement liées aux inscriptions sur l'image (*peu d'ambiguité*) ;
- la situation de l'annotation n'a pas d'impact : en effet, peu d'informations complémentaires sont requises pour interpréter les textes de l'image (*poids du contexte faible*).

Au final, l'analyse révèle que cette tâche est relativement peu complexe. Le calcul d'un score d'accord entre annotateurs permettrait de confirmer cette hypothèse et d'en déduire la confiance que nous pouvons accorder à la qualité des données labellisées.

2.3.2.c Problèmes de coûts

Toute cette complexité a un impact non négligeable sur les coûts à engager dans un projet de conception de jeu de données. Nous allons ici détailler certains de ces coûts et donner quelques exemples notoires.

Tout d'abord, il y a des **charges de main d'œuvre**. En effet, de nombreuses personnes aux compétences diverses vont s'investir dans un tel projet (voir SECTION 2.2.2), à la fois pour modéliser le phénomène mais aussi pour labelliser les données le représentant. L'intervention de ces acteurs va engager des coûts dans l'embauche des experts et opérateurs nécessaires, mais aussi dans leur formation si cela est nécessaire (veille technologique, montée en compétence sur la compréhension du phénomène, ...).

Ensuite, il faut considérer les **facteurs temporels** qui impactent le délais de mise en production du modèle de *Machine Learning*. En plus des délais de monté en compétences des experts, nous pouvons ajouter :

- le temps nécessaire à la *collecte de données*, pouvant prendre plusieurs semaines celle-ci correspond à un sondage ou à une mesure d'un phénomène s'étalant sur la durée ;

- le temps imposé par les *ateliers de conception* de la modélisation, pouvant prendre entre quelques heures et quelques semaines en fonction de sa complexité et des différences d'opinions entre experts ;
- le temps dédié à l'*annotation des données*, dépendant entre autre de la complexité de la modélisation et de la quantité à labelliser ;
- et le temps requis pour entraîner le modèle de *Machine Learning*.

À cela s'ajoute aussi la perspective de révision de la modélisation, et donc à la multiplication des coûts en appliquant plusieurs itérations du cycle MATTER pour obtenir un modèle convenable. Il est possible de réduire certains de ces coûts temporels en ajoutant davantage de personnes dans le projet, mais cela augmentera évidemment les charges de main d'oeuvre, et quelques aspects demeureront toutefois incompressibles (les débats lors de la phase de modélisation par exemple).

Enfin, il y a divers **coût financiers à engager** en plus des recrutements et des conséquences des délais de réalisation. On peut notamment considérer l'achat ou le développement de l'infrastructure technique tel que de l'outil d'annotation et de stockage des données, les potentiels instruments de mesure du phénomène (micro, caméra, sondes, ...) ou encore les acquisitions de droits d'utilisation de données ou de modèles sous licence.

Exemples : Détaillons ici deux projets d'annotation dont le chiffrage est disponible :

- **Prague Dependency Treebank** (BÖHMOVÁ et al., 2003) : L'annotation de 180 000 phrases tchèques (environ 1.8 million de *tokens*) pour l'analyse morpho-syntaxique a duré 5 ans (entre 1996 et 2003), impliquant 22 personnes. La somme totale engagée est estimée à 600 000 dollars ;
- **MS COCO** (LIN et al., 2014) : L'annotation d'objets présent dans 328 000 images (environ 2.5 millions d'objets répartis en 91 types différents) a nécessité environ 70 000 heures d'annotation.

Idées : Pour limiter les coûts, diverses solutions ont déjà été proposées :

- la **pré-annotation** : cette approche consiste à employer un petit modèle pour suggérer une annotation à l'annotateur, celui-ci pouvant alors l'approuver ou corriger. FORT et SAGOT, 2010 démontre le gain de temps et de qualité d'une telle approche. toutefois, certains biais peuvent influencer négativement l'annotateur : DANDAPAT et al., 2009 souligne par exemple le risque de biais de confirmation (influence de la machine et tendance de l'annotateur d'être en accord avec celle-ci, notamment sur des données ambiguës) ;
- l'**apprentissage actif** (*active learning*) : cette approche consiste à interagir avec la machine pour sélectionner les données les plus intéressantes à annoter (SETTLES, 2010). On peut par exemple entraîner un modèle avec les données déjà disponibles, le tester sur les données non annotées et sélectionner les données sur lesquelles ses prédictions sont le moins confiantes ;
- le **transfert d'apprentissage** (*transfert learning*) : cette approche consiste à réutiliser la connaissance contenue dans un modèle déjà existant dans le but d'exploiter

certaines de ces fonctionnalités (ZHUANG et al., 2021, IMAN et al., 2023). Il a été montré qu'adapter un modèle pour un nouveau cas d'usage similaire peut se faire avec peu de données, réduisant ainsi drastiquement la charge d'annotation (PARNAMI et LEE, 2022) ;

- la **myriadisation** (*crowdsourcing*) : cette approche consiste à déléguer l'annotation à des plateformes collaboratives en ligne comme **Amazon Mechanical Turk** (CALLISON-BURCH et DREDZE, 2010) ou **Language ARC** (FIUMARA et al., 2020). N'importe quel internaute peut alors contribuer à la tâche d'annotation, permettant ainsi de labelliser de grands volumes de données rapidement et à moindre coûts. Ces techniques comportent toutefois un inconvénient majeur : les opérateurs ne sont généralement pas des experts et sont rémunérés au volume labellisé. La qualité des annotations risque donc d'être délaissée au profit de la quantité.

2.3.3 Défis concernant les différences de comportements d'annotation

On peut constater deux divergences de comportements :

- ceux entre deux annotateurs (voir SECTION 2.3.3.A) ;
- ceux d'un annotateur avec lui-même (voir SECTION 2.3.3.B).

2.3.3.a Différences inter-annotateurs

Comme nous avons pu le voir dans la SECTION 2.3.2.A, la modélisation d'un phénomène relève d'un choix subjectif. Il est donc normal de constater que cette subjectivité entraîne des différences de comportement d'annotation. Celles-ci se manifestent particulièrement sur des données ambiguës ou bruitées car les limites de la modélisation sont exposées. D'autres éléments comme le caractère abstrait d'une modélisation ou la présence de règles de labellisation trop floues peuvent aussi mener à des divergences d'interprétation entre opérateurs.

BAYERL et PAUL, 2011 ont pu étudier un ensemble de **facteurs source de désaccords**. Nous en dressons une liste résumée ci-dessous :

- la *complexité* du problème : que celle-ci vienne intrinsèquement du phénomène à annoter ou du cas d'usage reflétée dans la modélisation, on observe que si l'annotation est difficile, alors les chances d'interpréter différemment les données sont plus élevées ;
- le *nombre* d'annotateurs : il est normal de constater que plus y a de personnes donnant leurs avis, les avis peuvent diverger ;
- le *niveau d'expertise* du phénomène : on remarque des divergences d'opinion entre les opérateurs n'étant pas spécialistes du phénomène et ceux détenant des connaissances sur celui-ci (un linguiste pour des données textuelles, un libraire pour des BD, ...) ;
- le *niveau de formation* à la tâche de labellisation : on constate des divergences entre les opérateurs habitués à l'outil et au guide d'annotation, et ceux découvrant la tâche de labellisation et sa mise en oeuvre pour la première fois.

i Pour information : BAYERL et PAUL, 2011 montrent aussi qu'il peut y avoir des différences entre deux annotateurs experts non-formés à la tâche de labellisation, alors qu'il y a moins de divergences entre des annotateurs expert et des annotateurs non-expert s'ils ont été formés tous les deux. Cela souligne bien l'**importance de la formation à la tâche d'annotation** et la nécessité de rédiger un guide d'annotation qui détaille l'objectif et les règles de labellisation dans le but d'en limiter la subjectivité.

💡 Idées : Pour aller plus loin, il est recommandé d'organiser des sessions d'**adjudication** durant laquelle les opérateurs peuvent confronter leurs visions en annotant les mêmes données. Cela permet de mesurer un score d'accord entre annotateurs mais aussi de discuter des points de désaccords dans l'application du guide de labellisation. Une règle de bon sens consiste à confronter 2 annotateurs pour être capable d'identifier ces points de désaccords, mais il faut être au moins 3 opérateur pour en discuter et les résoudre. BAYERL et PAUL, 2011 conseille même d'être au moins 5 annotateurs si le cas d'usage est critique.

2.3.3.b Différences intra-annotateur

Malgré la conception d'un guide et la rédaction de règles, malgré les sessions de revues dédiées à affiner ce guide et malgré les diverses pistes mises en oeuvre pour rendre l'annotation moins complexe, nous pouvons tout de même constater qu'un annotateur peut manifester des variations de comportement. Ces différences peuvent s'apparenter à des erreurs d'inattention, à des oubliés de consignes, ou à d'autres fluctuations similaires.

Pour expliquer cela, nous pouvons nous baser sur la théorie de la **régulation de la charge de travail** (SPERANDIO, 1978, SPERANDIO, 1987). Celle-ci s'exprime de la manière suivante :

- D'une part, l'opérateur **estime la charge de travail** à accomplir (*la quantité, la difficulté, les délais, ...*). Concernant la tâche d'annotation, elle sera perçue comme **très élevée** (*complexité intrinsèque, volumes conséquent à traiter, règles strictes de labellisation, ...*) ;
- D'autre part, l'opérateur **estime les ressources** à sa disposition (*ses capacités, ses connaissances, ses outils, ...*). Celles-ci seront plutôt perçues comme **minimes** face à cette montagne (*peu de compétences techniques, manque de formation, méconnaissance de la modélisation utilisée, ...*) ;;
- Enfin, l'opérateur compare les deux estimations (*est-ce que la charge de travail et mes ressources sont à l'équilibre ?*) et **ajuste sa charge de travail** pour la proportionner aux ressources qu'il va engager. Dans notre cas, l'opérateur va probablement **essayer de réduire** (intentionnellement ou non) sa charge de travail pour qu'elle soit perçue comme acceptable vis-à-vis de ses ressources.

Ce principe explique alors certaines variations, notamment lorsque la modélisation est particulièrement complexes ou que l'annotateur est sous pression ou fatigué.

Q Exemples : On peut appliquer cette théorie lorsque l'opérateur n'a pas eu de congés depuis plusieurs semaines ou que la fin de semaine approche : les capacités de celui-ci

sont perçues comme plus faibles à cause de la fatigue, on peut alors constater des erreurs d'annotation même si la complexité de la tâche n'a pas changée.

Idées : Une première approche consiste à **essayer de simplifier la tâche** de labellisation (recommandation de BAYERL et PAUL, 2011), permettant ainsi d'avoir une meilleure qualité sur une modélisation moins sophistiquée. FORT et al., 2012 rappelle que le contexte autour de la donnée peut aussi introduire un surplus d'information à gérer, diminuer ce contexte peut parfois alléger l'annotation mais peu en échange introduire plus d'ambiguïté. BALEDENT, 2023 expose le dilemme entre annoter un phénomène complexe en une seule passe et le découper en plusieurs tâches unitaires distinctes : si la première permet d'avoir une meilleure vue d'ensemble, la seconde peut alléger la charge de travail.

Il est aussi possible d'essayer de **rendre la tâche ludique** (*gamification*, VON AHN, 2006) : plusieurs outils tentent en effet de faire oublier à l'opérateur qu'il est en train de travailler en déguisant sa mission sous la forme d'un jeu (par exemple : GUILLAUME et al., 2016 propose ZombiLingo pour l'annotation morpho-syntaxique de textes). Cette approche requiert toutefois une grande créativité pour réussir à faire cette illusion (FORT, 2017).

2.3.3.c Faible valorisation du rôle d'annotateur

Comme vu en SECTION 2.2.2, l'annotation se repose principalement sur des opérateurs ayant une connaissance **métier** du phénomène à modéliser, ou des opérateurs formés spécifiquement à la tâche de labellisation. Ces personnes sont donc essentielles à la conception d'une base d'apprentissage.

Cependant, on constate que de plus en plus d'entreprises décident de sous-traiter ces tâches d'annotation à des plateformes de myriadisation (*crowdsourcing*, HOWE, 2008) comme **Amazon Mechanical Turk** (CALLISON-BURCH et DREDZE, 2010) ou **Language ARC** (FIUMARA et al., 2020). Ces plateformes collaboratives permettent à n'importe quel internaute de contribuer à une tâche d'annotation, permettant ainsi de labelliser de grands volumes de données rapidement et à moindre coûts. Plusieurs études montrent en effet l'intérêt de faire participer une foule d'opérateurs non-experts (SNOW et al., 2008, FORT, 2017), mais de **sérieuses questions éthiques** sont néanmoins soulevées par l'utilisation de ces plateformes.

Tout d'abord, on peut remarquer que ces opérateurs sont souvent payés un salaire dérisoire proportionnel à la quantité de données qu'ils labellisent. Cela ouvre la porte à des dérives, comme la polémique autour de **ChatGPT** (OPENAI, 2023) où PERRIGO et ZORTHIAN, 2023 avait dénoncé l'emploi de Kényans pour moins de 2\$ de l'heure dans le but de corriger ce modèle. On peut aussi citer DZIEZA, 2023 qui alerte sur l'apparition cette nouvelle classe de travail sous-payée, n'ayant pas de place claire dans le droit du travail, et généralement dévalorisée dans l'organisation des projets d'annotation.

D'autre part, ROWE, 2023 rapporte aussi les impacts émotionnels et psychologiques causés par certaines tâches d'annotation. En effet, une mission récurrente consiste à modérer des contenus à caractères offensants (*insultes, violence, drogue, sexe, armes, ...*) : de nombreux annotateurs sombrent ainsi dans la dépression après avoir labellisé de telles données pendant des jours voire des semaines...

i Pour information : Pour conclure sur ces dérives éthiques, VALETTE, 2016 rédige une critique franche des stratégies d'annotation dans le domaine du traitement automatique du langage naturel. Celle-ci dénonce l'organisation actuelle où les experts linguistes sont devenus de simples sous-traitants n'ayant plus leur mot à dire dans la conception d'une modélisation : ils ont généralement traité avec mépris, ne récoltent pas les lauriers des projets à succès mais sont tenus pour responsables des projets en échecs.

2.3.4 Bilan concernant les nombreux défis de l'annotation

Points à retenir : En dressant la liste des défis autour de la tâche d'annotation, nous avons pu voir que :

- ✓ L'enjeu d'un projet d'annotation consiste à **avoir des données de qualité** : celles-ci doivent être représentatives du problème à traiter, en quantité suffisante, avec un minimum de bruit, et dont les droits d'usage sont disponibles ;
- ✓ Cependant, la tâche de labellisation et son exigence de qualité **engendre de la complexité** : celle-ci peut venir de la difficulté à modéliser le phénomène, de l'annotation elle-même, ou encore des coûts qui y sont associés ;
- ✓ Ainsi, cette complexité **crée des différences de comportements** : ces divergences s'expliquent par la subjectivité de l'annotation et par la régulation de la charge de travail opérée par l'opérateur lorsque celle-ci est trop élevée.

2.4 Contexte du doctorat : comment assister la conception d'une base d'apprentissage pour un agent conversationnel bancaire en français ?

Durant ce doctorat, nous nous sommes intéressés à la **conception d'assistants conversationnels (chatbot)**. En effet, leur utilisation en entreprise est de plus en plus courante (GOASDUFF, 2019, COSTELLO et LODOLCE, 2022), notamment pour l'automatisation de certaines tâches simples et l'accès aux informations de bases documentaires. La popularité de ces assistants vient entre autres de la possibilité de dialoguer directement avec la machine grâce à des requêtes exprimées en langage naturel, offrant ainsi un gain de confort, de disponibilité et de performance.

De part notre expérience, nous avons constaté que plusieurs critères sont nécessaires pour déployer un assistant conversationnel dans un contexte industriel :

- Il faut être capable de gérer le dialogue entre l'utilisateur et l'assistant (*comprendre la requête initiale, demander ou confirmation ou des informations complémentaires, ...*) ;
- Il faut être capable de contrôler le contenu des réponses de l'assistant et de s'assurer de ses performances (*répondre ou agir de manière adaptée, ne pas de fournir de réponses contenant des informations confidentielles, ne pas de répondre de manière indécente, ...*) ;

- Il faut être capable de donner à l'assistant l'accès à certaines ressources (*lire et écrire en base de données, exécuter des programmes tiers, ...*) tout en garantissant la sécurité (*se prémunir contre les requêtes malveillantes et les erreurs de manipulations*).

Pour toute ces raisons, **l'architecture traditionnelle des *chatbot* est plutôt orientée par tâches** (*task-oriented*), c'est-à-dire qu'elle manipule une abstraction du dialogue en intentions⁷ et gère un paramétrage des réponses dépendant des intentions détectées (voir CHEN et al., 2017 et BRABRA et al., 2022).

i Pour information : L'ANNEXE B détaille plus amplement les différences entre les assistants *task-oriented* (approches symboliques) et les assistants *chat-oriented* (approches numériques ou génératives). Cette annexe se base notamment sur une revue des architectures de conceptions publiée par CHEN et al., 2017.

Néanmoins, l'élaboration de tels assistants reste un **défi difficile à relever dans le monde industriel** :

- Le traitement du langage en tant que tel est un problème complexe : il faut traiter une grande variété de bruits et d'ambiguïtés de dialogue en plus d'un vocabulaire souvent spécifique au domaine de l'assistant (voir les problèmes de bruits et de représentativité en SECTION 2.3.1) ;
- La base d'apprentissage ainsi que les réponses de l'assistant peuvent contenir des données privées ou confidentielles : il y a donc peu de données réutilisables des sources publiques, et de fortes pressions sont exercées sur le contrôle du comportement du *chatbot* (voir les problèmes de droits d'utilisation et de confidentialité en SECTION 2.3.1) ;
- L'assistant doit parfois pouvoir manipuler un grand nombre de cas d'usages : sa modélisation peut alors représenter des dizaines d'intentions pour lesquelles des centaines de branches de dialogues peuvent être paramétrées, introduisant ainsi une grande complexité aux tâches de modélisation et d'annotation de sa base d'apprentissage (voir SECTION 2.3.2) ;

Pour surmonter ces difficultés, les entreprises font alors intervenir des experts aux compétences diverses (voir SECTION 2.2.2), notamment des experts analytiques pour concevoir une modélisation stable des textes en intentions, puis des experts métiers pour valider la pertinence de la modélisation proposée et annoter les données suivant cette modélisation.

Or au vu de la complexité d'un tel projet, des différences de comportements entre opérateurs, telles que des erreurs d'annotations ou des divergences d'opinion, sont inévitables (voir SECTION 2.3.3). Il est alors nécessaire de former les experts métiers aux tâches d'annotations et à certaines tâches d'analyses afin d'encadrer les discussions autour de certaines différences de comportements pouvant mener à des remises en cause de la modélisation abstraite de textes en intentions (voir étape *Revise* du cycle MATTER, SECTION 2.2.1.A). Au final, **cette organisation devient très coûteuse** car elle demande des formations analytiques à des experts métiers, nécessite l'organisation d'ateliers de modélisation en mode essai-erreur pour trouver une base

7. Intention de dialogue : en traitement automatique du langage naturel, une intention représente la compréhension de la demande formulée par un utilisateur au cours de la conversation. Elle est généralement définie par le verbe d'action de la demande, et est représentée par une étiquette. Par exemple, les requêtes « *joue moi du jazz s'il te plaît !* » ou « *lance une playlist de musiques de Noël sur l'enceinte du salon !* » peut être modélisée par l'intention *jouer_musique*. Pour plus d'information, consultez l'ANNEXE B.

2.4. Contexte du doctorat : comment assister la conception d'une base d'apprentissage pour un agent conversationnel

d'apprentissage stable et pertinente, et fait intervenir des experts métiers sur une abstraction de leurs connaissances du quotidien.

🔍 Exemples : Au cours de ce doctorat, nous avons entre autres travaillé sur des assistants conversationnels à destinations de conseillers bancaires et de leur clients. Ces assistants doivent traiter une large variété de sujets (banque, assurance, finance, ...) et peuvent donc rapidement contenir une centaine d'intentions pour plus d'un millier de branches de dialogue. La conception de la base d'apprentissage de tels assistants représente ainsi un réel défi d'organisation, sur plusieurs semaines, notamment pour faire intervenir des experts de la banque-assurance dans un projet d'intelligence artificielle, domaine dans lequel ils n'ont pas ou peu de connaissances...

Cependant, il pourrait être intéressant de remettre en question cette organisation des projets d'annotation où les experts métiers sont interrogés sur des compétences qui ne sont pas les leurs. Dans le cadre de ce doctorat, et afin de trouver une solution à cette problématique, **nous nommes alors posé la question suivante** :

Comment assister la phase de modélisation de textes en intentions pour concevoir la base d'apprentissage d'un assistant conversationnel en impliquant des experts métiers sur leur vrai domaine de connaissances et en leur demandant un minimum de bagages analytiques ou techniques ?

💡 Idées : Pour répondre à cette problématique, nous nous sommes intéressés à trois concepts intéressants de la littérature :

- aux techniques de *clustering*, permettant de déléguer à la machine la tâche de modélisation grâce à une segmentation des données sur la base de leurs similarités (XU et TIAN, 2015) ;
- à l'annotation de contraintes binaires sur les données, permettant de corriger le fonctionnement d'un algorithme de *clustering* en y introduisant de la connaissance métier (LAMPERT et al., 2018) ;
- aux techniques d'apprentissage actif, favorisant les interactions entre l'Homme et la machine pour atteindre un objectif (SETTLES, 2010).

Chapitre 3

Proposition d'un *Clustering Interactif* pour assister la modélisation d'un jeu de données textuelles

Dans le chapitre précédent, nous avons vu les points essentiels suivants :

- L'étape de modélisation est nécessaire pour définir les objectifs et les règles d'un projet d'annotation ; Or cette étape rencontre de nombreux défis qui la rendant particulièrement laborieuse (*complexité intrinsèque du phénomène, subjectivité des opérateurs, différences de comportements, ...*). Ainsi, la modélisation est régulièrement révisée (cycle MATTER).
- La modélisation de textes en intentions pour entraîner un assistant conversationnel orienté par tâches n'échappe pas à ce constat, notamment à cause de la complexité du langage naturel, de la diversité d'intentions de dialogue à représenter, et de la pression sur le contrôle du comportement de l'assistant.
- Dans un cadre industriel, des experts métiers sont responsables de la modélisation et de l'annotation les données spécifiques ou confidentielles de l'entreprise ; Or ces interventions requièrent des compétences analytiques et techniques dont les experts métiers ne disposent pas forcément ; De ce fait, la manipulation d'une modélisation abstraite de leurs connaissances est alors vécue comme une tâche pénible, nécessitant un grand nombre de formations et organisée sous la forme d'ateliers en mode essay-erreur.

Dans cette partie, nous cherchons une alternative à cette organisation traditionnelle, et nous proposerons une méthodologie d'annotation basée sur un *clustering* interactif visant à remplir un double objectif :

- Permettre d'assister la modélisation et l'annotation des données pour créer plus efficacement une base d'apprentissage destinée la classification d'intentions d'un assistant conversationnel ;

- Redéfinir les tâches et les objectifs des différents acteurs afin de rester au plus proche de leurs compétences réelles, particulièrement en ce qui concerne l'intervention des experts métiers du projet.

i Pour information : Cette proposition de méthode a été l'objet d'une présentation à la conférence EGC (Extraction et Gestion des Connaissances) (SCHILD et al., 2021), et d'une extension dans le journal IJDWM (International Journal of Data Warehousing and Mining) (SCHILD, DURANTIN et al., 2022). Nous reprenons ici certains des éléments présentés avec quelques détails supplémentaires.

Sommaire

3.1	Intuitions à l'origine d'un clustering interactif	40
3.1.1	Utiliser une approche non-supervisées pour créer une modélisation	40
3.1.2	Corriger l'approche non-supervisée avec une annotation de contraintes	41
3.1.3	Tirer parti des avantages de l'apprentissage actif pour optimiser les interactions Homme/machine	43
3.2	Description de notre clustering interactif.	43
3.2.1	Description générale	44
3.2.2	Description détaillée	44
3.2.3	Description techniques et implémentation	47
3.3	Espoirs portés sur la méthode proposée.	48

3.1 Intuitions à l'origine d'un clustering interactif

Tout d'abord, détaillons trois intuitions qui nous ont permis de concevoir notre méthodologie d'annotation.

3.1.1 Utiliser une approche non-supervisées pour créer une modélisation

Dans le but d'assister la phase de modélisation des données, une piste intéressante revient à déléguer cette tâche à la machine. En effet, grâce à une **classification non-supervisés (clustering)**, un algorithme peut regrouper les données en fonction de leur similarité intrinsèque et ainsi suggérer une modélisation intentions. Plusieurs algorithmes et méthodes connus peuvent être utilisés :

- Le **clustering KMeans** (MACQUEEN, 1967) : Cette méthode se repose sur la minimisation de l'inertie intra-classes en attribuant chaque donnée au barycentre de *cluster* le plus proche. Cette approche est l'une des plus répandues en raison de sa simplicité et de sa rapidité de calcul;
- Le **clustering hiérarchique** (MURTAGH et CONTRERAS, 2012) : Cette méthode revient à fusionner itérativement les données les plus similaires dans un nouveau *cluster*. Plusieurs liens de similarité peuvent être implantés (*le lien single fusionnant les deux clusters ayant les frontières les plus proches, le lien complete fusionnant les deux clusters ayant*

*les frontières opposées les plus proches, le lien *average* fusionnant les deux clusters ayant les barycentres les plus proches et le lien *ward* fusionnant les deux clusters qui donneront le prochain cluster le plus compact) ;*

- Le **clustering spectral** (NG et al., 2002) : Cette méthode consiste à modéliser la matrice de similarité entre les données par ses vecteurs propres puis de regrouper ces derniers à l'aide d'un KMeans. Cette approche permet d'obtenir des *clusters* aux formes complexes ;
- Le **clustering DBScan** (ESTER et al., 1996) : Cette méthode utilise la densité de données dans l'espace pour identifier des regroupements. Cette approche permet de découvrir des *clusters* aux formes complexes si leur densité est suffisante.
- ...

Cependant, ces algorithmes non-supervisés font régulièrement face à un ensemble de difficultés qui pénalisent leur utilisations. En effet, ces méthodes ont souvent du mal à traiter des données de grandes dimensionnalités ou en grand nombre (STEINBACH et al., 2004), la présence de bruits peut rapidement perturber le fonctionnement d'un algorithme (YANG et WANG, 2004), et certains *clusters* aux géométries complexes peuvent être difficiles à identifier (KRIEGEL et al., 2011). De plus, le choix de certains hyperparamètres de ces méthodes n'est pas toujours simple, surtout en ce qui concerne le nombre de clusters, la méthode d'initialisation ou la mesure de distance à utiliser (AGARWAL et al., 2011). Enfin, toutes ses difficultés se retrouvent dans le traitement du langage naturel, notamment à cause de la taille de vocabulaire important, de la présence de nombreux bruits et de la vaste diversité et complexité de thématiques pouvant y être abordés.

i Pour information : La revue de XU et TIAN, 2015 détaille plusieurs algorithmes de *clustering*, en fonction de leurs forces et faiblesses sur les points mentionnés ci-dessus.

Ainsi, toutes ces limites étaient le fait qu'**un résultat brut d'une classification non supervisées est généralement perçu comme peu pertinent par les experts métiers**. Il est donc nécessaire d'introduire une intervention humaine dans le processus pour guider le fonctionnement d'un algorithme de *clustering*.

3.1.2 Corriger l'approche non-supervisée avec une annotation de contraintes

Une variante aux approches non-supervisées consiste à demander à un humain certaines informations nécessaires à leur amélioration nous considérons donc les **approches semi-supervisées**. Une manière efficace d'introduire les connaissances d'un expert dans le processus est l'ajout de contraintes.

LAMPERT et al., 2018 rappelle qu'il peut y avoir deux types de contraintes :

- les **contraintes sur les données** : on parle alors principalement des contraintes binaires MUST-LINK et CANNOT-LINK décrivant si deux données doivent ou ne doivent être dans un même clusters (WAGSTAFF et CARDIE, 2000)
- les **contraintes sur les clusters** : ces contraintes peuvent concerner le nombre de *clusters* à trouver, leur taille minimale ou maximale, la distance minimale de séparation de leurs frontières, ...

Bien que d'autres méthodes permettent d'insérer des contraintes (heuristiques non-supervisées, transferts de connaissances préalables), nous nous concentrerons ici sur l'ajout manuel

par un expert. Comme cet expert n'a a priori pas de connaissances techniques ou analytiques, ce dernier aurait du mal à manipuler des contraintes sur les *clusters*. Toutefois, ses connaissances métiers lui permettent de caractériser facilement la similarité entre deux données, et donc de décrire des contraintes de type MUST-LINK et CANNOT-LINK en répondant à la question : « *est-ce les deux données traité du même cas d'usage ?* ».

Notes de l'auteur : La pierre angulaire de la méthode que nous proposons en SECTION 3.2 repose notamment sur le fait qu'il est difficile pour un expert métier de classer une question suivant une modélisation abstraite prédéfinie : cela l'éloigne de ses compétences métiers initiales, nécessite en contre-partie de nombreuses formations, et introduit de nombreuses erreurs d'annotations. De fait, il semble plus adéquat de demander à l'expert métier de discriminer deux questions sur la base de leurs similarité de cas d'usage métier.

Parmi les algorithmes de *clustering* sous contraintes connus, nous disposons des adaptations suivantes :

- Le **clustering KMeans sous contraintes** comme COP-KMeans (WAGSTAFF et al., 2001) : Dans cette version, l'attribution d'une donnée se fait au *cluster* dont le barycentre est le plus proche et où aucune contraintes n'est violée. Cette adaptation est relativement simple à mettre en oeuvre, mais elle peut mener à des cas de blocage où plus aucun *cluster* n'est accessible à cause de violation de contraintes (il est possible d'adapter l'algorithme en créant un nouveau *cluster*) ;
 - Le **clustering hiérarchique sous contraintes** (DAVIDSON et RAVI, 2005) : Dans cette version, les données liées par des contraintes MUST-LINK sont d'abord fusionnées, puis le processus agglomératif commence en prenant garde de ne pas fusionner des *clusters* ayant des contraintes CANNOT-LINK entre eux. Cette adaptation est très simple à mettre en oeuvre car il suffit d'adapter le calcul de distance entre *clusters* ;
 - Le **clustering spectral sous contraintes** (KAMVAR et al., 2003) : Dans cette version, les coefficients de la matrice de similarité sont forcés à 0 (respectivement 1) si deux données sont liées par une contrainte CANNOT-LINK (respectivement MUST-LINK). Cette adaptation demande peu d'effort pour être mise en oeuvre, mais des modifications aussi drastiques de la matrice de similarité peuvent entraîner des changements imprévisibles de comportements ;
 - Le **clustering DBScan sous contraintes** comme C-DBScan (RUIZ et al., 2010) : Dans cette version, la densité des données ainsi que les contraintes CANNOT-LINK sont utilisées pour identifier des clusters locaux qui seront ensuite fusionnés à l'aide des contraintes MUST-LINK. Cette adaptation est plus compliquée à réaliser car elle change légèrement le fonctionnement interne d'exploration de la densité de l'espace de données.
- ...

Ces algorithmes de clustering sous contraintes sont ainsi plein de potentiels : ils sont en effet capable de tirer parti de la similarité intrinsèque des données et des contraintes judicieusement placées de l'expert pour segmenter les données de manière adéquate et ainsi proposer une modélisation pertinente.

Q Exemples : On peut citer LAMPERT et al., 2019 où l'annotation de contraintes par un expert d'identifier efficacement dans objets dans une image satellite.

Cependant, pour un jeu de données de taille N , il y a N^2 possibilités de contraintes à ajouter. L'estimation du placement des contraintes les plus appropriées et les plus efficaces pour corriger le *clustering* semble donc être un problème NP-complexe. De plus, si ce choix peut être visuel pour des images (comme dans l'exemple cité ci-dessus), il ne l'est pas pour des données textuelles dont la diversité et la complexité sont difficiles à appréhender. Il est donc important d'assister l'expert métier à disposer ses contraintes afin de maximiser son impact dans la correction du *clustering*.

3.1.3 Tirer parti des avantages de l'apprentissage actif pour optimiser les interactions Homme/machine

Une dernière piste intéressante est celle de l'apprentissage actif (*active learning*, voir SETTLES, 2010) : celle-ci prône les interactions Homme/machine comme moyen d'atteindre un objectif qu'aucun en peut atteindre séparément. BAE et al., 2021 liste par exemple un ensemble d'interactions possibles avec un algorithme de *clustering* : celles-ci peuvent concerner la manipulation et l'adaptation des résultats, l'adaptation des hyperparamètres des algorithmes, mais aussi des initiatives de la machine pour préparer la réalisation d'une tâche.

Dans notre cas, nous nous intéressons en particulier aux initiatives de la machines pour identifier la liste des contraintes à annoter pour corriger ou confirmer efficacement un résultat de *clustering*. Cela peut se faire par exemple à l'aide d'une heuristique sélectionnant les parties les moins sûres de la segmentation.

Points à retenir : Dans cette partie, nous avons exposés les intuitions suivantes :

- ✓ La phase de modélisation des données peut être déléguée à la machine en employant des algorithmes de classification non-supervisée (*clustering*) ;
- ✓ Pour corriger le fonctionnement d'un algorithme de *clustering* et ainsi améliorer la pertinence de ses résultats, l'expert peut ajouter des contraintes binaires sur les données (MUST-LINK et CANNOT-LINK, *clustering* sous contraintes) ;
- ✓ Dans le but d'ajouter des contraintes de manière efficace, il est possible d'interagir avec la machine afin de maximiser l'impact de l'intervention de l'expert (*active learning*).

3.2 Description de notre *clustering* interactif.

Sur la base des intuitions que nous venons de détailler, nous proposons la méthode suivante dans le but d'assister la modélisation et l'annotation d'une collecte de données brutes en une base d'apprentissage nécessaire à l'entraînement d'un assistant conversationnel.

3.2.1 Description générale

Notre méthode d'annotation, que nous appelons « **Clustering Interactif** », repose sur l'alternance successive entre deux phases **clefs** (voir FIGURE 3.1) :

- une phase d'**annotation de contraintes** par un expert permettant de caractériser la similarité entre deux données suivant leur cas d'usage métier ;
- une phase de **segmentation automatique** des données par une machine sur la base de la proximité sémantique des données et des contraintes précédemment annotées.

L'objectif recherché en associant ces deux phases est de **créer un cercle vertueux pour améliorer itérativement la qualité de la base d'apprentissage** en cours de construction. En effet, à chaque itération, l'expert métier obtiendra une proposition de segmentation des données qu'il pourra raffiner dans le but de corriger le fonctionnement de la machine et ainsi d'obtenir une segmentation plus pertinente à l'itération suivante.

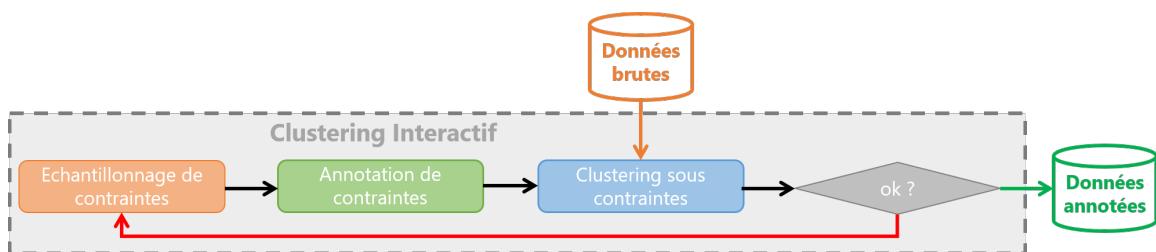


FIGURE 3.1 – Schéma illustrant l'architecture du clustering interactif. La boucle principale enchaîne un échantillonnage de couples de données, une annotation de contraintes, et un clustering sous contraintes.

3.2.2 Description détaillée

L'ALGORITHME 3.1 décrit formellement notre proposition de **Clustering Interactif** que nous détaillons ci-dessous.

Pour l'**initialisation** de la méthode (cf. ALGORITHME 3.1, lignes 1 à 3), nous définissons une liste vide de contraintes : tout au long du processus, nous y ajoutons les contraintes annotées par l'expert grâce à ses connaissances métiers (nous entrerons en détails en décrivant la phase d'annotation). Il faut aussi une première segmentation des données par la machine : celle-ci se réalise par l'exécution d'un algorithme de *clustering*. Nous estimons qu'il n'est pas du ressort de l'expert métier de choisir le réglage de l'algorithme de *clustering* et ses hyper-paramètres. Ces derniers pourront être déterminés par un *data scientist* en fonction du problème à traiter ou laissés par défaut. Il est à noter que cette première segmentation des données est réalisée sans bénéficier de la connaissance de l'expert, il est donc peu probable que le résultat soit pertinent à ce stade.

Nous entrons dans le cœur de la boucle itérative par la phase d'**échantillonnage** (cf. ALGORITHME 3.1, lignes 5 et 6). Comme mentionné au préalable, savoir quelles contraintes ajouter pour corriger efficacement le *clustering* est un problème NP-difficile (le nombre de possibilités croît proportionnellement au carré du nombre de données). De plus, l'intervention d'experts est chiffrée et représente en général une partie des coûts à investir dans un projet (voir SECTION 2.3.2.c). Il est donc inconcevable de laisser un expert métier annoter des contraintes "seul" et "au hasard". Ainsi, pour optimiser ses interventions, il convient de déterminer là où l'expert

Données : données non segmentées

Entrées : budget à disposition

1 initialisation : créer une liste vide de contraintes ;

2 optionnel : évaluer les hyper-paramètres de la segmentation automatique ;

3 segmentation initial : regrouper les données par similarité ;

4 répéter

5 **optionnel** : évaluer les hyper-paramètres de l'échantillonnage ;

6 **échantillonnage** : sélectionner une partie de la segmentation à corriger ;

7 **annotation** : corriger la segmentation en ajoutant des contraintes sur l'échantillon ;

8 **optionnel** : ré-évaluer les hyper-paramètres de la segmentation automatique ;

9 **segmentation** : regrouper les données par similarité avec les contraintes ;

10 **validation** : estimer la pertinence et la stabilité de la segmentation ;

11 **coûts** : estimer le budget restant et les coûts restant à investir ;

12 **jusqu'à segmentation satisfaisante OU budget épuisé**;

13 interprétation : trier et nommer les clusters pour les exploiter ;

Résultat : données segmentées (i.e. base d'apprentissage)

ALGORITHME 3.1 – *Description en pseudo-code de la méthode d'annotation proposée employant le clustering interactif.*

aura le plus d'impact lors de sa transmission de connaissances. C'est pourquoi la phase d'échantillonnage est primordiale dans la méthode proposée : Nous proposons d'y sélectionner des couples de données sur la base de leur similarité, de leur segmentation ou encore de leurs relations avec d'autres données déjà liées par des contraintes.

Sur la base de cet échantillon, l'expert peut entamer son étape d'**annotation de contraintes** (cf. ALGORITHME 3.1, *ligne 7*). Pour alléger la charge d'annotation, nous avons décidé de discriminer les données de l'échantillon par des contraintes binaires simples : **MUST-LINK** et **CANNOT-LINK**. Ces contraintes représentent respectivement la similitude ou la différence entre deux données, et seront utilisées pour regrouper ou séparer certaines données dans la prochaine segmentation. En fonction de l'orientation du projet et afin de rester au plus proche des compétences réelles de l'expert, la formulation de l'énoncer d'annotation doit être judicieusement définie : par exemple, les contraintes peuvent représenter une similitude sur la thématique concernée⁸, sur l'action désirée⁹, ou encore sur le besoin de l'utilisateur¹⁰. On notera que des incohérences peuvent s'introduire, ayant pour conclusions de devoir à la fois considérer comme similaires et différentes deux données : ces incohérences peuvent être détecter grâce à aux propriétés de transitivités des contraintes (voir la gestion des conflits en ANNEXE C.2.2).

Pour finir, la dernière phase de cette boucle est composée d'une nouvelle **segmentation** des données (cf. ALGORITHME 3.1, *lignes 8 et 9*). Cette segmentation devra respecter les contraintes préalablement définies par l'expert, nous nous tournons donc vers l'utilisation d'un *clustering* sous contraintes. Au fur et à mesure des itérations, de plus en plus de contraintes seront ajoutées pour corriger le *clustering*. ainsi, au bout d'un certain nombre d'itérations, la segmentation des données reflétera la vision que l'expert aura voulu transmettre. Comme précédemment, nous estimons qu'il n'est pas du ressort de l'expert métier de choisir de l'algorithme de *clustering* et ses hyper-paramètres. Ces derniers pourront être déterminés par un *data scientist* en fonction

8. Exemples de thématiques : *crédit vs. assurance*; *sport vs. culture*, ...

9. Exemples d'actions : *souscrire vs. résilier*; *activer vs. désactiver*; *s'informer vs. réaliser*, ...

10. Exemple de besoins : *souscrire un crédit vs. souscrire une assurance*; *s'informer en sport vs. s'informer en culture*, ...

du problème à traiter, estimés en fonction de l'itération et des contraintes disponibles, ou laissés par défaut.

Comme la méthode est itérative, il faut pouvoir estimer des **cas d'arrêt** (cf. ALGORITHME 3.1, lignes 10 à 12). Le cas d'arrêt le plus évident n'est pas technique mais relatif aux coûts investis dans l'opération : si le projet n'a plus de budget dédié à l'annotation, il faudra créer la base d'apprentissage avec le résultat à disposition, quel que soit la pertinence de la segmentation obtenue sur les données. Ce cas d'arrêt par défaut peut malheureusement être synonyme d'échec pour le projet si les résultats sont inexploitables. D'autres cas d'arrêts peuvent être envisagés en fonction de la qualité ou de la pertinence de la segmentation. D'une part, nous pouvons comparer l'évolution de la segmentation des données : si les segmentations sont similaires sur plusieurs itérations, il est possible que la modélisation atteint un optimum local ou un palier de performance. D'autre part, nous pouvons aussi comparer l'évolution de l'accord entre la segmentation obtenue et l'annotation de l'expert : en effet, si l'expert ne contredit plus la répartition proposée des données, il est probable que sa vision et la vision de la machine aient convergé. Dans les deux cas, l'analyse de l'expert métier reste nécessaire pour valider si la modélisation des données est pertinente ou si elle comporte encore des incohérences à corriger.

Lorsque la boucle itérative est finie, nous avons à disposition une segmentation des données qui a été corrigé par un expert et qui reflète ses connaissances métier. La dernière étape consiste alors à **interpréter** ces *clusters* pour pouvoir les exploiter (cf. ALGORITHME 3.1, ligne 13). Cela commence par leur attribuer un nom au lieu de leur identifiant technique, de les définir en les rapprochant d'un cas d'usage métier, et éventuellement de les raffiner manuellement en supprimant certaines données aberrantes.

Exemples : La FIGURE 3.2 déroule l'initialisation et la première itération de la méthode sur un exemple fictif. Nous pouvons constater qu'entre les images (2) et (5), la segmentation des données a évolué grâce à l'introduction de contraintes.

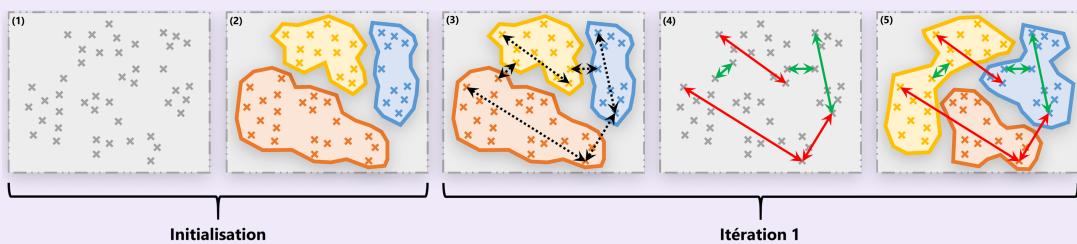


FIGURE 3.2 – Exemple d'une itération de clustering interactif.

Lors de l'initialisation, (1) correspond au jeu de données brut, et (2) correspond à une première segmentation des données en 3 clusters. Lors de l'itération 1 : (3) correspond à un exemple d'échantillonnage de 6 contraintes représentées par les flèches en pointillées, (4) correspond à la caractérisation de ces 6 contraintes par des liens *MUST-LINK* en vert et *CANNOT-LINK* en rouge, et (5) correspond à la nouvelle segmentation des données en 3 clusters respectant les 6 contraintes annotées. La prochaine itération se poursuivra par un nouvel échantillonnage de contraintes.

3.2.3 Description techniques et implémentation

Au cours de ce doctorat, nous avons réalisé un ensemble d'implémentations en **Python** afin de mettre en oeuvre notre méthodologie de *clustering interactif*. Celle-ci est répartie en trois librairies :

1. **cognitivefactory-interactive-clustering**¹¹ (SCHILD, 2022a), regroupant les gestions de données et des contraintes, les algorithmes de *clustering* et d'échantillonnage ;
2. **cognitivefactory-features-maximization-metric**¹² (SCHILD, 2023), disposant d'une méthode de sélection des patterns linguistiques pertinents d'un jeu de données labellisées, permettant ainsi d'analyser la pertinence d'un résultat de *clustering* ;
3. **cognitivefactory-interactive-clustering-gui**¹³ (SCHILD, TREMBLE et MISIAK, 2022), intégrant la logique de la méthodologie dans une application web.

Exemples : La FIGURE 3.3 représente une capture d'écran de la page d'annotation de contraintes de l'application web intégrant notre méthodologie de *clustering interactif*.

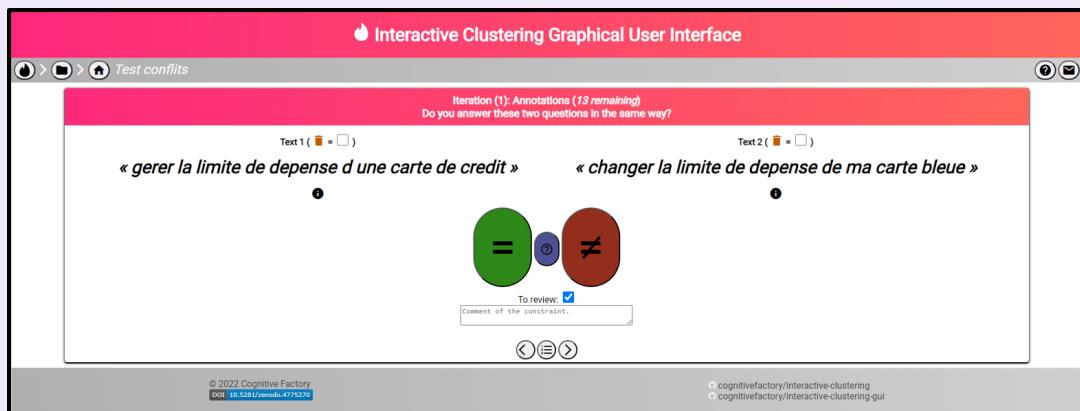


FIGURE 3.3 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page d'annotation d'une contrainte. Parmi les éléments importants, nous retrouvons les deux textes à annoter (disposés à gauche et droite de l'écran) et les boutons d'annotation (bouton vert pour un *MUST-LINK*, bouton rouge pour un *CANNOT-LINK*). Les autres fonctionnalités sont détaillées en ANNEXE C.4.

i Pour information : Ces implémentations sont présentées dans l'ANNEXE C. L'ensemble des détails techniques et des explications sur les choix d'implémentation y sont décrits.

11. <https://pypi.org/project/cognitivefactory-interactive-clustering/>
 12. <https://pypi.org/project/cognitivefactory-features-maximization-metric/>
 13. <https://pypi.org/project/cognitivefactory-interactive-clustering-gui/>

3.3 Espoirs portés sur la méthode proposée.

Nous avons proposé une méthodologie d'annotation basée sur une interaction entre l'Homme et la machine dans le but de soulager l'expert métier dans son intervention.

Grâce à une telle approche, nous pouvons espérer que :

- Les experts métiers n'auront désormais plus besoin de bagages analytiques ou techniques pour intervenir dans un projet d'annotation ;
- Les experts métiers pourront désormais participer à la modélisation d'une base d'apprentissage en ayant des discussions pragmatiques sur les cas d'usages des données manipulées ;
- Une telle méthodologie d'annotation permettra d'obtenir efficacement une base d'apprentissage stable et pertinente pour entraîner une modèle de classification d'intention ;
- Une telle méthodologie d'annotation sera réaliste en terme de délais et d'investissement financier.

Nous allons explorer diverses pistes pour confirmer ou infirmer ces espoirs dans le CHAPITRE 4, et nous détaillerons nos conclusions dans un guide d'utilisation qui sera présenté dans le CHAPITRE 5.

Chapitre 4

Étude de six hypothèses sur le *Clustering Interactif*

Dans le chapitre précédent, nous avons présenté une méthode de création d'un jeu de données d'entraînement pour un assistant conversationnel, que nous appelons « **Clustering Interactif** » :

- ✓ La méthode proposée repose sur la combinaison entre un regroupement automatique des données par la machine et l'annotation de contraintes binaires par un expert métier pour corriger le regroupement proposé ;
- ✓ Une telle approche devrait limiter les pré-requis de compétences analytiques et techniques actuellement exigés d'un expert métier en les déléguant à la machine.
- ✓ En échange, l'expert se concentre d'avantage sur la transmission de ses connaissances avec une annotation caractérisant la similitude métier entre deux données.

Il existe des travaux similaires sur l'annotation de données visuelles (LAMPERT et al., 2019) et des revues sur les interactions possibles avec un algorithme de *clustering* (BAE et al., 2021). Cependant, peu d'études de la littérature scientifique ont exploré les possibilités d'interactions entre un expert métier et un algorithme de *clustering* sous contraintes dans le but de modéliser des données textuelles en intentions. Ainsi, dans cette partie, **nous étudions la faisabilité d'un tel Clustering Interactif pour des données textuelles** en explorant les six questions suivantes :

- Peut-on obtenir une base d'apprentissage à l'aide de notre proposition d'implémentation de la méthodologie du *clustering* interactif ? (cf. hypothèse d'**efficacité** en SECTION 4.1) ;
- Peut-on déterminer un paramétrage optimal de cette implémentation pour obtenir plus rapidement une base d'apprentissage ? (cf. hypothèse d'**efficience** en SECTION 4.2) ;
- D'après les données initiales, peut-on approximer l'investissement nécessaire pour

obtenir une base d'apprentissage exploitable ? (cf. hypothèse sur les **coûts** en SECTION 4.3) ;

- A un instant donné, peut-on estimer la pertinence métier d'une base d'apprentissage en cours de construction ? (cf. hypothèse de **pertinence** en SECTION 4.4) ;
- Au cours du processus de construction de la base d'apprentissage, peut-on aisément estimer les potentiels d'une étape de raffinage supplémentaire ? (cf. hypothèse de **rentabilité** en SECTION 4.5) ;
- Peut-on estimer l'influence d'une différence d'annotation dans la construction de la base d'apprentissage ? (cf. hypothèse de **robustesse** en SECTION 4.6).

Afin de vérifier ces différentes hypothèses, nous organisons un ensemble d'études basées soit sur des approches théoriques (*simulations des comportements et comparaisons à une vérité terrain grâce à un score de v-measure*¹⁴), soit sur des approches empiriques (*expériences en situations réelles et analyses basés sur les compétences d'opérateurs métier*). L'imbrication de ces études est représentée dans la FIGURE 4.1 qui évoluera au cours des sections suivantes pour résumer les hypothèses vérifiées et annoncer l'hypothèse en cours d'étude.



FIGURE 4.1 – Illustration des études réalisées sur le clustering interactif (étape 0/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en *v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations par un expert métier).

14. **v-measure** : pour plus de détails sur cette mesure de qualité d'un *clustering*, consultez l'ANNEXE D.1.

i Pour information : Les jeux de données utilisés pour ces études sont détaillés en ANNEXE A. Une annexe est aussi disponible pour recenser les différentes conventions de nommage destinées à simplifier le partage des résultats voir (ANNEXE C.1).

i Pour information : Pour ces études, l'exécution des différentes expériences a été réalisée sur des CPU Intel(R) Xeon(R) CPU E5-2660 v4 2.00GHz et parallélisée avec la librairie *multiprocessing*¹⁵(utilisant un worker par CPU). Les scripts d'exécution et d'analyse de ces expériences, rédigés au sein de *notebooks Python* (VAN ROSSUM et DRAKE, 2009) ou de scripts R (R CORE TEAM, 2017), sont disponibles dans SCHILD, 2022b.

Sommaire

4.1 Évaluation de l'hypothèse d'efficacité	53
4.1.1 Étude de convergence vers une vérité terrain pré-établie en simulant l'annotation d'une base d'apprentissage et mesurant la vitesse de sa création	53
4.2 Évaluation de l'hypothèse d'efficiency	60
4.2.1 Étude d'optimisation des paramètres d'implémentation en analysant leurs tailles d'effets sur la vitesse de création d'une base d'apprentissage	60
4.3 Évaluation de l'hypothèse sur les coûts	71
4.3.1 Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle	72
4.3.2 Étude du temps de calcul nécessaire aux algorithmes implémentés en chronométrant des exécutions dans différentes situations	81
4.3.3 Étude du nombre de contraintes nécessaires à la convergence vers une vérité terrain pré-établie en fonction de la taille du jeu de données	91
4.3.4 Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts	95
4.3.5 Ouverture vers une annotation en parallèle du <i>clustering</i>	97
4.4 Évaluation de l'hypothèse de pertinence	100
4.4.1 Étude d'une validation manuelle et non assistée de la valeur métier d'une base d'apprentissage par un expert	101
4.4.2 Étude des patterns linguistiques pertinents à l'aide de la Maximisation des Traits pour assister la validation d'une base d'apprentissage	105
4.4.3 Étude d'un résumé automatique des <i>clusters</i> à l'aide d'un large modèle de langage	111
4.4.4 Mise en commun des stratégies d'évaluation de la pertinence métier d'un résultat de <i>clustering</i> interactif	118
4.5 Évaluation de l'hypothèse de rentabilité	119
4.5.1 Étude de l'évolution d'accord entre l'annotation et le <i>clustering</i>	120
4.5.2 Étude de l'évolution de la différence entre deux <i>clusterings</i> consécutifs	124
4.5.3 Mise en commun des stratégies d'évaluation de la rentabilité d'une itération de la méthode et définition d'un cas d'arrêt indépendant d'une vérité terrain.	128
4.6 Évaluation de l'hypothèse de robustesse	129

15. *multiprocessing* : <https://pypi.org/project/multiprocessing/>

4.6.1	Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle	130
4.6.2	Étude de l'impact d'une erreur d'annotation et l'intérêt de la corriger	134
4.6.3	Étude de l'impact de la subjectivité de l'annotation sur la divergence des résultats obtenus	139
4.6.4	Bilan concernant la robustesse du <i>clustering</i> interactif	147
4.7	Autres hypothèses non vérifiées	148
4.7.1	Étude du nombre de clusters optimal	148
4.7.2	Étude d'autres méthodes de vectorisation	149
4.7.3	Étude d'autres méthodes d'échantillonnage	149
4.7.4	Étude de techniques de transfert d'apprentissage	149
4.7.5	Étude ergonomique de l'interface d'annotation	149

4.1 Évaluation de l'hypothèse d'efficacité

En premier lieu et afin de poser les bases de nos études, nous devons nous demander si notre implémentation du *clustering* interactif est fonctionnel et si elle permet d'atteindre son objectif. Nous aimerions donc vérifier l'hypothèse suivante :

⌚ Hypothèse d'efficacité ⌚

« Une méthodologie d'annotation basée sur le *clustering* interactif permet d'obtenir une base d'apprentissage pour un assistant conversationnel qui respecte la vision donnée par l'expert métier au cours de l'annotation. »

La FIGURE 4.2 illustre cette hypothèse et l'espoir de convergence d'une base d'apprentissage en cours de construction vers sa vérité terrain.

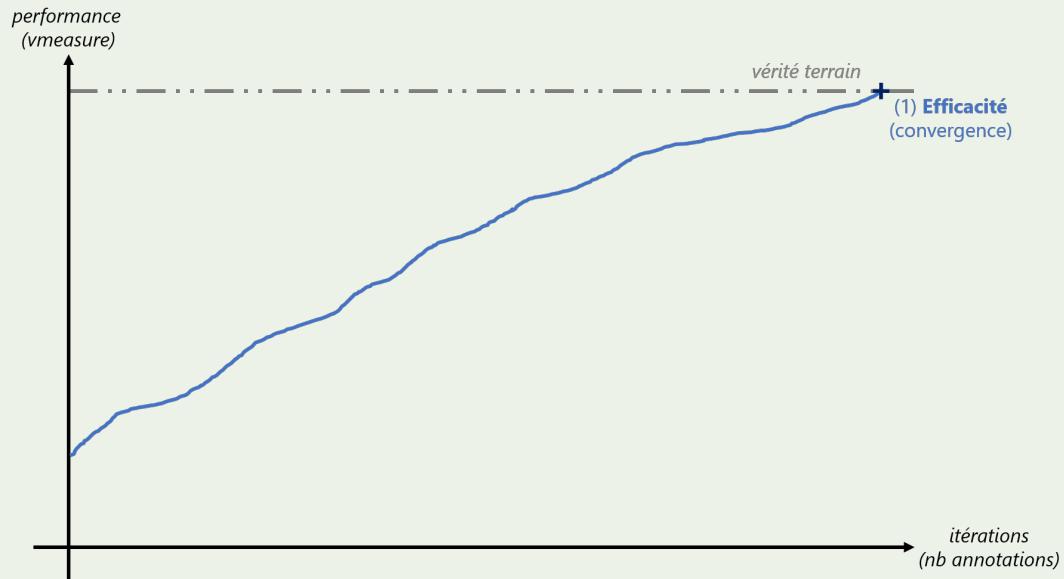


FIGURE 4.2 – Illustration des études réalisées sur le clustering interactif (étape 1/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en *v*-measure) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations par un expert métier).

Afin de vérifier cette hypothèse, nous mettons en place une **expérience de ré-annotation** d'une base d'apprentissage (qui servira ici de vérité terrain) à l'aide de notre méthode, en simulant l'annotation d'un expert, et nous critiquons l'évolution de la nouvelle base d'apprentissage obtenue ainsi que sa similitude avec la base d'apprentissage initiale (cf. SECTION 4.1.1).

4.1.1 Étude de convergence vers une vérité terrain pré-établie en simulant l'annotation d'une base d'apprentissage et mesurant la vitesse de sa création

Nous voulons vérifier qu'une méthodologie d'annotation basée sur notre implémentation du *clustering* interactif permet de créer une base d'apprentissage pour un assistant conversation-

nel. Pour cela, nous prenons une base d'apprentissage employée pour entraîner un modèle de classification de textes, et nous utilisons ce jeu de données comme vérité terrain. L'objectif de cette expérience est de simuler la création de cette base d'apprentissage et de nous assurer que le résultat obtenu correspond à la vérité terrain.

i Pour information : Cette étude a été l'objet d'une présentation à la conférence EGC (Extraction et Gestion des Connaissances) (SCHILD et al., 2021), et d'une extension dans le journal IJDWM (International Journal of Data Warehousing and Mining) (SCHILD, DURANTIN et al., 2022). Les résultats et la discussion de ces articles ont été mis à jour pour mieux s'intégrer au discours ce manuscrit.

4.1.1.a Protocole expérimental

⚠️ Attention : Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Cependant, cette hypothèse forte n'est pas toujours vérifiée en pratique, surtout lorsque l'on manipule des données non structurées. L'impact de ce point sur les résultats obtenus est discuté en fin de partie, et nous nous y intéressons plus en détails dans la SECTION 4.6 (hypothèse de robustesse).

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.1.

Nous utilisons comme vérité terrain le jeu de données **Bank Cards** (v1.0.0) : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte**, **carte avalée**, **commande de carte**, ...). Pour plus de détails, consultez l'annexe A.1.

Lors de cette expérience, chaque tentative de la méthode commencera sur la version non labellisée de la vérité terrain à disposition, sans aucune contrainte connue à l'avance. À chaque itération de la méthode, nous simulons l'annotation de l'expert métier en comparant les labels de la vérité terrain : ainsi, deux données ont une contrainte **MUST-LINK** si elles ont le même label, et une contrainte **CANNOT-LINK** sinon. Cela traduit le prérequis d'avoir un annotateur qui soit capable, dans son domaine d'expertise, de différencier deux données selon leur ressemblance. Une tentative de l'application de notre méthode s'arrête lorsque toutes les contraintes possibles entre les données ont été annotés par l'expert.

Pour cette étude, nous essayons une tentative pour chaque combinaison de paramètres de notre implémentation du *clustering* interactif (cf. ANNEXE C.2). Cela comprend les tâches et leurs paramètres respectifs suivants :

1. le **prétraitement** des données, avec les niveaux suivants : **aucun** (noté `prep.no`), **simple** (supprimant les minuscules, la ponctuation, les accents et les espaces blancs ; noté `prep.simple`), avec **lemmatisation** (noté `prep.lemma`) et avec **filtres** (supprimer les mots trop éloignés de la racine de l'arbre de dépendances syntaxiques ; noté `prep.filter`) ;
2. la **vectorisation** des données, avec les niveaux suivants : **TF-IDF** (noté `vect.tfidf`) et **SpaCy** (noté `vect.frcorenewsmd`) ;

Données : jeu de données annotées (vérité terrain)

Entrées : combinaisons d'algorithmes et de paramètres à tester

```

1 pour chaque algorithmes et de paramètres à tester faire
2   initialisation (données) : récupérer les données et la vérité terrain ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   prétraitement : supprimer le bruit dans les données ;
5   vectorisation : transformer les données en vecteurs ;
6   clustering initial : regrouper les données par similarité des vecteurs ;
7   évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
8   répéter
9     échantillonnage : sélectionner de nouvelles contraintes à annoter ;
10    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
11    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
12    clustering : regrouper les données par similarité avec les contraintes ;
13    évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
14  jusqu'à annotation de toutes les contraintes possibles;
15  évaluation finale : espérer avoir un score d'équivalence de 100% avec la vérité
  terrain ;

```

Résultat : algorithmes et de paramètres ayant un score d'équivalence de 100%

ALGORITHME 4.1 – *Description en pseudo-code du protocole expérimental de l'étude de convergence du clustering interactif vers une vérité terrain pré-établie.*

3. le **clustering sous contraintes** des données, avec les niveaux suivants : **KMeans** (modèle *COP* noté `clust.kmeans.cop`), **Hiérarchique** (lien *single* noté `clust.hier.sing`; lien *complete* noté `clust.hier.comp`; lien *average* noté `clust.hier.avg`; lien *ward* noté `clust.hier.ward`) et **Spectral** (modèle *SPEC* noté `clust.spec`). Le choix du nombre de *clusters* n'est pas étudié ici, et ce nombre est fixé au nombre de classes présentes dans la vérité terrain ;
4. l'**échantillonnage** des contraintes à annoter, avec les niveaux suivants : couples de données **purement aléatoires** (noté `samp.random.full`), couples de données **pseudo-aléatoires** (sélectionnant des données provenant d'un même *cluster* ; noté `samp.random.same`), couples de données **issues d'un même cluster et étant les plus éloignées** (noté `samp.farhest.same`) et couples de données **issues de clusters différents et étant les plus proches** (noté `samp.closest.diff`). Le choix de la taille d'échantillon n'est pas étudié ici, et cette taille est arbitrairement fixée à 50¹⁶.

Notes de l'auteur : Il est difficile de discuter de l'influence du paramétrage du nombre de *clusters* : en effet, dans notre étude, nous désirons évaluer la capacité de notre méthode à atteindre une vérité terrain théorique ; or chercher un nombre de *clusters* différent du nombre de classes reviendrait à remettre en cause la vérité terrain que nous

16. Une taille d'échantillon de 50 contraintes à annoter semble a priori un bon compromis entre (1) ne pas donner trop de travail à un annotateur en une session et (2) donner suffisamment de nouvelles contraintes au clustering pour proposer un partitionnement plus pertinent des données. Ce choix sera discuté en fin de partie, et nous nous y intéresserons davantage dans la SECTION 4.3 (hypothèse sur les coûts).

essayons d'atteindre, nous empêchant donc d'évaluer la capacité de notre méthode... Devant ce cercle vicieux, nous laissons l'étude de ce problème ouvert à de futures études.

Il y a donc 192 combinaisons testées, et chaque tentative est répétée 5 fois pour contrer les aléas statistiques des algorithmes de *clustering* (*initialisation du clust.kmeans.cop*, ...) et d'échantillonnage (*choix des contraintes au hasard avec samp.random.full*, ...). Pour plus de détails sur ces algorithmes, référez-vous à l'ANNEXE C.2 pour avoir accès à leur description, à leurs paramètres et aux choix d'implémentation.

Pour évaluer l'équivalence entre la vérité terrain et notre segmentation des données obtenue au cours de la méthode, nous nous intéressons à l'évolution de la **v-measure** (ROSENBERG et HIRSCHBERG, 2007) entre ces deux jeu de données. Si le score du calcul de la **v-measure** est de 100%, cela signifierait que le *clustering* final et la vérité terrain propose une segmentation identique des données, donc que la vérité terrain a pu être retrouvée, et donc qu'il est possible d'obtenir une base d'apprentissage pour un assistant conversationnel à l'aide d'une méthodologie d'annotation basée sur le *clustering* interactif.

i Pour information : Les scripts de l'expérience (*notebooks Python* (VAN ROSSUM et DRAKE, 2009)) sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.1.1.b Résultats obtenus

La FIGURE 4.3 et la TABLE 4.1 représentent l'évolution moyenne de la **v-measure** du *clustering* en fonction du nombre d'itérations de la méthode. Les tentatives les plus rapides et les plus lentes sont représentées sur la figure.

Malgré une forte dispersion des résultats (écart-type de **v-measure** pouvant être supérieur à 20%, forte différence entre la tentative la plus rapide et la plus lente) et quelques sauts de performances (cf. à-coups de la tentative la plus lente sur la figure), une convergence générale vers la vérité terrain peut être constatée.

À l'itération 0, une tentative commence avec une moyenne de 19.05% de **v-measure** entre son *clustering* initial (sans contrainte) et la vérité terrain. Cette **v-measure** moyenne croît presque linéairement (pente de 0.97) jusqu'à l'itération 75 où elle atteint la performance de 92.08% (cf. TABLE 4.1).

Au delà de l'itération 75, la courbe de la **v-measure** moyenne tend vers une asymptote de 100% (cf. FIGURE 4.3). Cette asymptote est atteinte par toutes les 960 tentatives (192 combinaisons de paramètres, 5 tentatives pour chaque combinaison), la tentative l'ayant atteinte le plus tôt à l'itération 19 et celle le plus tard à l'itération 328.

La courbe se prolonge jusqu'à l'itération 393 pour que toutes les contraintes possibles sur le jeu de données puissent être annotées. On peut aussi noter que 756 tentatives (78.75%) convergent vers 100% de **v-measure** avant l'annotation exhaustive de toutes les contraintes : sur ces tentatives, la convergence peut ainsi s'observer en moyenne avec seulement 91.30% du nombre de contraintes possibles (min : 8.72%, max : 99.69%, écart-type : 18.60%).

4.1.1.c Discussion

Au regard des résultats décrits ci-dessus, les différentes simulations de la méthode ont bien convergé vers la vérité terrain (atteinte de l'asymptote à 100% de **v-measure**). Cette expérience

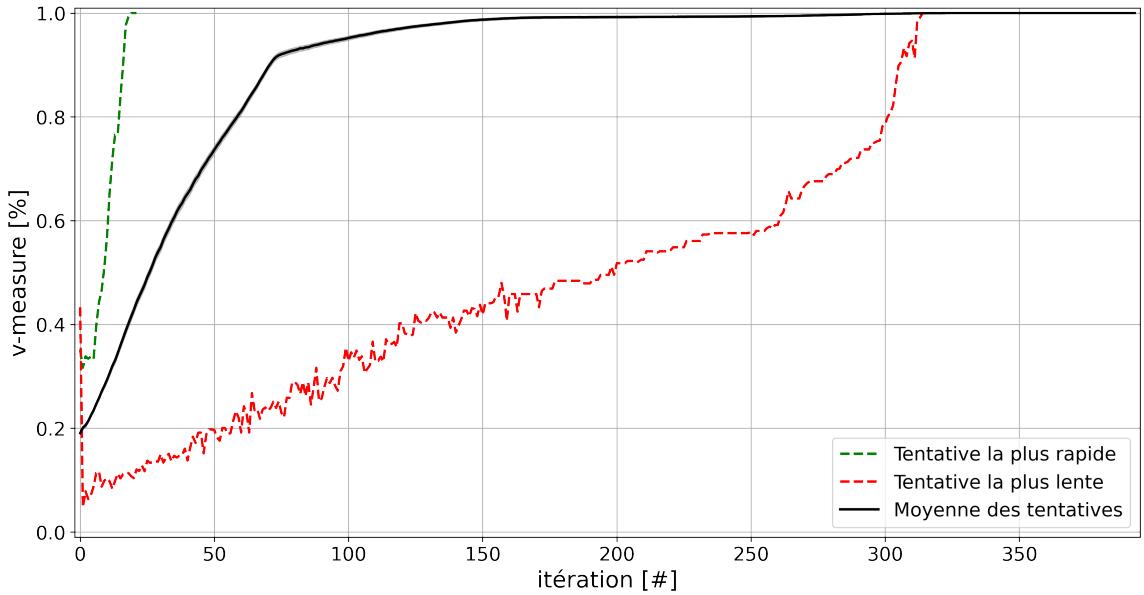


FIGURE 4.3 – Évolution de la moyenne de la *v*-measure entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de clustering interactif, moyenne réalisée itération par itération sur l'ensemble des tentatives. Représentation des tentatives ayant été les plus rapides (un prétraitement *prep.simple*, une vectorisation *vect.tfidf*, un clustering *clust.hier.comp* ou *clust.hier.ward*, et un échantillonnage *samp.closest.diff*) et les plus lentes (un prétraitement *prep.no*, une vectorisation *vect.tfidf*, un clustering *clust.spec*, et un échantillonnage de contraintes *samp.farthest.same*) pour atteindre 100 % de *v*-measure.

Annotations		Performances (<i>v</i> -measure)			
Itérations	Contraintes	Moyenne	Écart-type	Minimum	Maximum
0	0	19.05% (± 0.43)	13.38%	03.42%	47.75%
25	1 250	49.09% (± 0.82)	25.43%	09.09%	100.00%
50	2 500	73.66% (± 0.77)	23.98%	16.78%	100.00%
75	3 750	92.08% (± 0.54)	16.70%	21.74%	100.00%
100	5 000	95.19% (± 0.41)	12.67%	26.93%	100.00%
125	6 250	97.43% (± 0.29)	09.09%	34.99%	100.00%
150	7 500	98.73% (± 0.23)	07.22%	38.14%	100.00%
328	16 400	100.00% (± 0.00)	0.00%	100.00%	100.00%
394	19 700	100.00% (± 0.00)	0.00%	100.00%	100.00%

TABLE 4.1 – Détails de l'évolution de la moyenne de la *v*-measure entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de clustering interactif, moyenne réalisée itération par itération sur l'ensemble des tentatives.

permet donc de confirmer plusieurs espoirs portés sur la méthode.

Tout d'abord, la vérité terrain a été retrouvée sans formaliser concrètement la structure de données. Là où une annotation par label aurait requis au préalable une définition des catégories possibles pour les données à étiqueter (création d'un "type system"), la méthodologie employant le *clustering* interactif a permis de faire émerger naturellement cette structure de données. Cette

émergence provient directement des contraintes annotées par l'expert métier, traduisant ainsi ses connaissances à l'aide d'instructions simples : *les données sont-elles ou non similaires ?* Cela représente un net avantage pour l'opérateur qui n'a ainsi pas à maintenir en mémoire une modélisation complexe de la structure de données, rendant la tâche d'annotation plus accessible.

D'autre part, ces contraintes ont fait l'objet d'une annotation guidée par les besoins de la machine afin de s'améliorer d'itération en itération (voir la croissance globale de la **v-measure** sur la FIGURE 4.3). Ainsi, l'expert métier corrige la base d'apprentissage à chaque itération : soit en affinant les clusters en cours de construction, améliorant ainsi la cohérence des clusters (cf. pentes croissantes) ; soit en remaniant les clusters mal formés pour repartir sur de bonnes bases, détériorant la cohérence des clusters le temps de la réorganisation (cf. oscillations ou pentes décroissantes). Une telle assistance limite ainsi le nombre de contraintes non utiles au *clustering*, même si certains paramétrages semblent plus efficaces que d'autres (voir la forte dispersion des résultats).

De plus, nous remarquons que 76.75% des tentatives convergent vers la vérité terrain sans bénéficier d'une annotation exhaustive de toutes les contraintes possibles. Cela montre l'intérêt des interactions Homme/machine afin d'obtenir plus efficacement un résultat qu'un expert métier (aussi parfait soit-il) aurait obtenu seul. L'intérêt serait maintenant de déterminer la meilleure combinaison de paramétrages demandant d'annoter un nombre **suffisant** de contraintes afin d'obtenir ce même résultat de la manière la plus efficiente (cf. SECTION 4.2) et la plus robuste aux erreurs d'annotation (cf. SECTION 4.6).

Néanmoins, différentes pistes sont encore à explorer pour rendre le *clustering* interactif utilisable en situation réelle.

D'une part, nous échangeons le besoin de définir une structure de données contre la nécessité d'annoter un grand nombre de contraintes : pour 500 points de données, et en considérant que l'asymptote à 100% est atteinte en moyenne autour de l'itération 200, il faudrait 10 000 annotations de contraintes pour être exhaustif, ce qui correspond à près de 20 fois plus de contraintes que de données. Bien que l'annotation binaire demande a priori une charge mentale plus faible (HART et STAVELAND, 1988) et que l'opérateur n'a pas besoin de définir ou de maintenir en mémoire une structure de données complexe, un tel volume d'annotation représente tout de même une grande quantité de travail. Cela peut décourager les experts métiers en début de projet, surtout pour des projets ayant des jeux de données de plus grandes tailles. Toutefois, les résultats obtenus montrent une forte dispersion du nombre d'itérations nécessaire, et certaines tentatives ont été bien plus efficientes dans l'utilisation de leurs contraintes. La tentative la plus rapide a convergé à l'itération 19, soit 950 contraintes, ce qui est un volume d'annotation bien plus abordable ! On peut donc espérer trouver un paramétrage optimal de la méthode permettant de diminuer significativement le nombre moyen de contraintes nécessaires afin d'obtenir une base d'apprentissage exploitable avec un volume d'annotations acceptable. Cet aspect fait l'objet de l'étude décrite dans la SECTION 4.2 (hypothèse d'efficience).

D'autre part, le choix d'annoter toutes les contraintes possibles sur les données (**annotation exhaustive**) n'est pas forcément judicieux. En effet, si nous regardons la FIGURE 4.3, une moyenne de 90% de **v-measure** est déjà atteinte autour de l'itération 75, alors que l'asymptote à 100% n'est atteinte qu'au delà de l'itération 200. Afin d'être plus efficient, il faudrait envisager une **annotation partielle** permettant d'obtenir rapidement ces 90% de **v-measure** (cf. coude sur la FIGURE 4.3), quitte à affiner le résultat manuellement pour combler la "perte" moyenne de 10% de **v-measure**. Cet aspect sera ajouté à l'objectif de l'étude décrite dans la SECTION 4.2 (hypothèse d'efficience).

Pour finir, nous avons supposé dans cette étude que l'annotateur est un expert métier connaît-

sant parfaitement le domaine traité. Cette hypothèse forte n'est a priori pas valable en situation réelle : En effet, des différences d'annotations peuvent intervenir (*ambiguités sur les données, méconnaissance du domaine, erreurs d'inattention, différence d'opinions entre annotateurs, ...*), ce qui peut entraîner des divergences ou des incohérences dans la construction de la base d'apprentissage. Il semble donc nécessaire d'étudier les impacts de ces incohérences, ainsi que de proposer une méthode pour les prévenir ou les corriger. Cet aspect sera traité à la fin de ce chapitre dans la SECTION 4.6 (hypothèse de robustesse).

4.2 Évaluation de l'hypothèse d'efficience

Suite à la validation de l'hypothèse d'efficacité (convergence de la méthode, cf. SECTION 4.1), nous voulons déterminer les paramètres optimaux de la méthode afin de converger le plus rapidement vers la vérité terrain. Nous aimerions donc vérifier l'hypothèse suivante :

💡 Hypothèse d'efficience 💡

« La vitesse de convergence du *clustering* interactif peut être optimisée en ajustant différents paramètres afin de minimiser la charge de travail de l'opérateur. Nous étudions en particulier l'influence sur le nombre de contraintes requis du prétraitement des données, de la vectorisation des données, de l'échantillonnage des contraintes à annoter et du *clustering* sous contraintes. »

La FIGURE 4.4 illustre cette hypothèse et l'espérance d'une convergence "optimale" d'une base d'apprentissage en cours de construction vers sa vérité terrain.

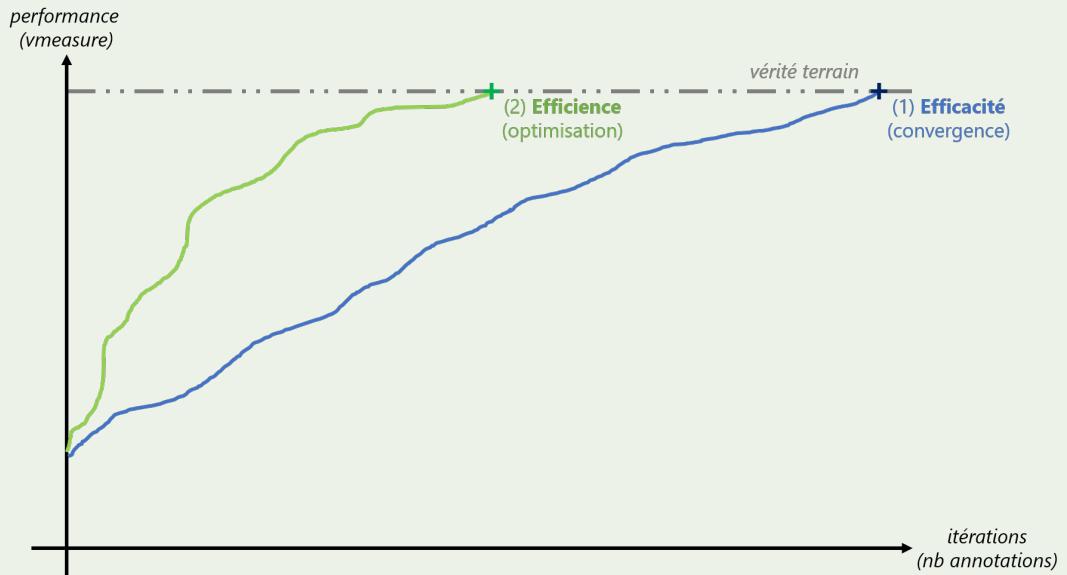


FIGURE 4.4 – Illustration des études réalisées sur le clustering interactif (étape 2/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en *v*-measure) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations par un expert métier).

Afin de vérifier cette hypothèse, nous mettons en place une expérience de ré-annotation d'une base d'apprentissage (qui servira ici de vérité terrain) à l'aide de notre méthode, en simulant l'annotation d'un expert, et nous réalisons l'analyse statistique de la taille d'effets de différents paramètres sur la **vitesse de convergence** du *clustering* itératif (cf. SECTION 4.2.1).

4.2.1 Étude d'optimisation des paramètres d'implémentation en analysant leurs tailles d'effets sur la vitesse de création d'une base d'apprentissage

Nous voulons étudier l'influence des paramètres de notre implémentation du *clustering* interactif sur la vitesse de création d'une base d'apprentissage pour un assistant conversationnel.

Nous allons donc compléter le protocole expérimental de l'étude de convergence en SECTION 4.1.1 visant à simuler la création d'une base d'apprentissage.

i Pour information : Cette étude a été l'objet d'une présentation à la conférence EGC (Extraction et Gestion des Connaissances) (SCHILD et al., 2021), et d'une extension dans le journal IJDWM (International Journal of Data Warehousing and Mining) (SCHILD, DURANTIN et al., 2022). Les résultats et la discussion de ces articles ont été mis à jour pour mieux s'intégrer au discours ce manuscrit.

4.2.1.a Protocole expérimental

⚠️ Attention : Comme dans l'étude précédente, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Cependant, cette hypothèse forte n'est pas toujours vérifiée en pratique, surtout lorsque l'on manipule des données non structurées. L'impact de ce point sur les résultats obtenus est discuté en fin de partie, et nous nous y intéressons plus en détails dans la SECTION 4.6 (hypothèse de robustesse).

Pour résumer le protocole expérimental adapté, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.2.

```

Données : jeu de données annotées (vérité terrain)
Entrées : combinaisons d'algorithmes et de paramètres à tester
1 pour chaque combinaison d'algorithmes et de paramètres à tester faire
2   initialisation (données) : récupérer les données et la vérité terrain ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   prétraitement : supprimer le bruit dans les données ;
5   vectorisation : transformer les données en vecteurs ;
6   clustering initial : regrouper les données par similarité des vecteurs ;
7   évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
8   répéter
9     échantillonnage : sélectionner de nouvelles contraintes à annoter ;
10    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
11    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
12    clustering : regrouper les données par similarité avec les contraintes ;
13    évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
14   jusqu'à annotation de toutes les contraintes possibles;
15 analyse : déterminer les tailles d'effets des algorithmes et paramètres ;
Résultat : meilleures combinaisons d'algorithmes et de paramètres

```

ALGORITHME 4.2 – Description en pseudo-code du protocole expérimental de l'étude d'optimisation de la convergence du clustering interactif vers une vérité terrain pré-établie.

En s'appuyant sur les résultats précédemment obtenus, nous allons analyser l'influence des différentes tâches employées (**prétraitement**, **vectorisation**, **clustering sous contraintes**,

échantillonnage) et de leurs paramètres sur la vitesse de convergence vers la vérité terrain. Nous utilisons à nouveau le jeu de données **Bank Cards** (v1.0.0) (cf. annexe A.1) comme vérité terrain, sur lequel nous testons 192 combinaisons de paramétrages, et chaque tentative est répétée 5 fois pour contrer les aléas statistiques de certains algorithmes. Pour plus de détails sur ces algorithmes, référez-vous à l'ANNEXE C.2.

Comme lors de l'étude sur la convergence de la méthode, nous nous intéressons à l'évolution de la **v-measure** (ROSENBERG et HIRSCHBERG, 2007) entre la vérité terrain et notre segmentation des données obtenue, et nous affinons notre évaluation en portant attention aux trois seuils d'annotations suivants :

1. le cas d'une **annotation partielle**, correspondant au nombre d'itérations nécessaires à la méthode pour avoir 90% de **v-measure**, c'est-à-dire un état de semi-parcours vers une convergence totale¹⁷ ;
2. le cas d'une **annotation suffisante**, correspondant au nombre d'itérations nécessaires à la méthode pour avoir 100% de **v-measure**, c'est-à-dire avoir suffisamment de contraintes annotées par l'expert métier pour retrouver la vérité terrain ;
3. le cas d'une **annotation exhaustive**, correspondant au nombre d'itérations nécessaires à la méthode pour parcourir toutes les contraintes possibles sur les données, et ainsi retranscrire exhaustivement la vision de l'expert métier¹⁸.

Enfin, nous utilisons une ANOVA à mesures répétées (GIRDEN, 1992) afin de déterminer l'effet des paramètres de notre implémentation sur le nombre d'annotations requis pour converger vers la vérité terrain. Le test de Tukey (HSD) (TUKEY, 1949) est utilisé pour les comparaisons post-hoc.

i Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009) et des scripts *R* (R CORE TEAM, 2017), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.2.1.b Résultats obtenus

Pour obtenir une **annotation partielle** (*atteindre une v-measure de 90%*), la moyenne des itérations est de 59.04 (min : 11, max : 315, écart-type : 42.14), soit une moyenne de 2 951.81 annotations (min : 550, max : 15 750, écart-type : 2 106.72). La FIGURE 4.5 représente la répartition de ces itérations au cours des différentes tentatives. On peut noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec un prétraitement des données **prep.no** ou **prep.simple** ou **prep.lemma**, une vectorisation des données **vect.tfidf**, un *clustering* sous contraintes **clust.hier.sing**, et un échantillonnage de contraintes **samp.closest.diff**. Ces tentatives ont requis 11 itérations, soit 550 annotations, dont 299 (respectivement 304 et 281) contraintes **MUST-LINK**.

17. Le seuil de 90% a été choisi au cours de l'étude de convergence (cf. hypothèse d'efficacité, SECTION 4.1, coude de la FIGURE 4.3).

18. Une annotation est a priori inutilisable en pratique (demande trop de contraintes, cf. hypothèse d'efficacité, SECTION 4.1), nous l'étudions toutefois pour avoir un point de comparaison.

- Les tentatives les plus lentes furent celles avec un prétraitement des données `prep.no`, une vectorisation des données `vect.tfidf`, un *clustering* sous contraintes `clust.spec`, et un échantillonnage de contraintes `samp.farthest.same`. Ces tentatives ont requis 315 itérations, soit 15 750 annotations, dont 1 032 contraintes MUST-LINK.

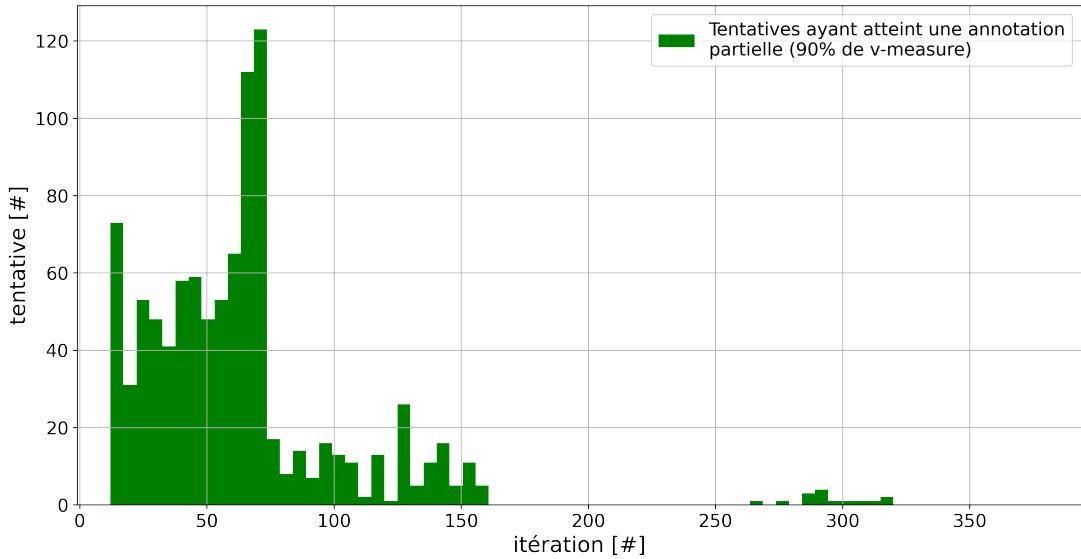


FIGURE 4.5 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation partielle, c'est-à-dire l'itération à laquelle elles parviennent à 90% de *v-measure* entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics pour simplifier l'affichage.

La TABLE 4.2 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaires pour atteindre une **annotation partielle** (*atteindre une v-measure de 90%*). Les analyses de variance mettent en relief l'effet significatif sur cette convergence du prétraitement (*eta-carré* : 0.320, *p-valeur* : $< 10^{-3}$), de la vectorisation (*eta-carré* : 0.388, *p-valeur* : $< 10^{-3}$), du *clustering* (*eta-carré* : 0.866, *p-valeur* : $< 10^{-3}$) et de l'échantillonnage (*eta-carré* : 0.968, *p-valeur* : $< 10^{-3}$). L'analyse post-hoc de ces effets indique que le meilleur paramétrage moyen pour atteindre une **annotation partielle** repose sur le prétraitement `prep.simple`, la vectorisation `vect.tfidf`, le *clustering* `clust.hier.avg`, et l'échantillonnage `samp.closest.diff`. La moyenne du nombre d'itération requis pour ce paramétrage est de 19.00 (écart-type : 0.79), soit 950 annotations (écart-type : 39.34).

Pour obtenir une **annotation suffisante** (*atteindre une v-measure de 100%*), la moyenne des itérations est de 76.29 (min : 19, max : 328, écart-type : 46.44), soit une moyenne de 3 801.19 annotations (min : 950, max : 16 400, écart-type : 2 314.91). La FIGURE 4.6 représente la répartition de ces itérations au cours des différentes tentatives. On peut noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec un prétraitement des données `prep.simple`, une vectorisation des données `vect.tfidf`, un *clustering* sous contraintes `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 19 itérations, soit 950 annotations, dont 638 (respectivement 641) contraintes MUST-LINK.

Description des facteurs analysés		Description statistique des itérations			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	p-valeur
prétraitement	prep.simple	61.90	(1)	0.32	0.320	$< 10^{-3}$ (***)
	prep.lemma	63.08	(2)			
	prep.no	63.70	(2)			
	prep.filter	71.90	(4)			
vectorisation	vect.tfidf	60.61	(1)	0.29	0.388	$< 10^{-3}$ (***)
	vect.frcorenewsmd	63.08	(2)			
clustering	clust.hier.avg	50.64	(1)	0.35	0.866	$< 10^{-3}$ (***)
	clust.kmeans.cop	52.43	(2)			
	clust.hier.sing	54.08	(3)			
	clust.hier.ward	72.41	(4)			
	clust.hier.comp	73.48	(5)			
	clust.spec	87.84	(6)			
échantillonnage	samp.closest.diff	33.66	(1)	0.32	0.968	$< 10^{-3}$ (***)
	samp.random.same	48.24	(2)			
	samp.random.full	65.83	(3)			
	samp.farthest.same	112.86	(4)			

TABLE 4.2 – ANOVA du nombre d’itérations nécessaires pour l’obtention de 90 % de v-mesure. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne Moyenne indiquent le classement des niveaux selon les analyses post-hoc.

- Les tentatives les plus lentes furent celles avec un prétraitement des données prep.no, une vectorisation des données vect.tfidf, un clustering sous contraintes clust.spec, et un échantillonnage de contraintes samp.farthest.same. Ces tentatives ont requis 394 itérations, soit 16 400 annotations, dont 1 309 contraintes MUST-LINK.

La TABLE 4.3 retranscrit l’influence de chacun des paramètres sur le nombre d’itérations nécessaires pour atteindre une **annotation suffisante**. Les analyses de variance mettent en relief l’effet significatif sur cette convergence du prétraitement (eta-carré : 0.987, p-valeur : $< 10^{-3}$), de la vectorisation (eta-carré : 0.991, p-valeur : $< 10^{-3}$), du clustering (eta-carré : 0.997, p-valeur : $< 10^{-3}$) et de l’échantillonnage (eta-carré : 0.998, p-valeur : $< 10^{-3}$). L’analyse post-hoc de ces effets indique que le meilleur paramétrage moyen pour atteindre une **annotation suffisante** repose sur le prétraitement prep.lemma, la vectorisation vect.tfidf, le clustering clust.kmeans.cop, et l’échantillonnage samp.closest.diff. La moyenne du nombre d’itération requis pour ce paramétrage est de 34.60 (écart-type : 7.44), soit 1 730 annotations (écart-type : 372.00).

Enfin, pour avoir une **annotation exhaustive** (*annoter toutes les contraintes possibles*), la moyenne des itérations est de 88.98 (min : 20, max : 394, écart-type : 68.21), soit une moyenne de 4 431.34 annotations (min : 1 000, max : 19 656, écart-type : 3 405.16). La FIGURE 4.7 représente la répartition de ces itérations au cours des différentes tentatives. On peut noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec un prétraitement des données prep.no ou

4.2. Évaluation de l'hypothèse d'efficience

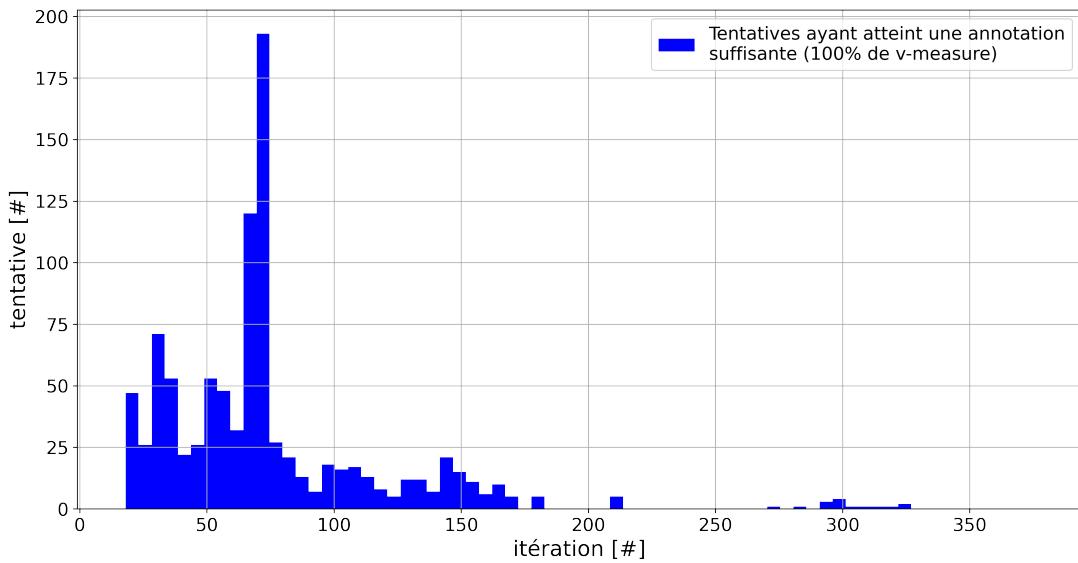


FIGURE 4.6 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation suffisante, c'est-à-dire l'itération à laquelle elles parviennent à 100 % de v -measure entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics pour simplifier l'affichage.

Description des facteurs analysés		Description statistique des itérations			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	p-valeur
prétraitement	prep.lemma	72.86	(1)	0.32	0.276	$< 10^{-3}$ (***)
	prep.simple	73.30	(2)			
	prep.no	75.24	(2)			
	prep.filter	83.77	(4)			
vectorisation	vect.tfidf	71.16	(1)	0.36	0.366	$< 10^{-3}$ (***)
	vect.frcorennewsmd	81.43	(2)			
clustering	clust.kmeans.cop	62.23	(1)	0.42	0.700	$< 10^{-3}$ (***)
	clust.hier.avg	65.13	(2)			
	clust.hier.sing	75.44	(3)			
	clust.hier.ward	80.44	(4)			
	clust.hier.comp	81.46	(5)			
	clust.spec	93.06	(6)			
échantillonnage	samp.closest.diff	50.29	(1)	0.39	0.950	$< 10^{-3}$ (***)
	samp.random.same	56.38	(2)			
	samp.random.full	71.95	(3)			
	samp.farhtest.same	126.55	(4)			

TABLE 4.3 – ANOVA du nombre d'itérations nécessaires pour l'obtention de 100 % de v -mesure. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne Moyenne indiquent le classement des niveaux selon les analyses post-hoc.

`prep.lemma`, une vectorisation des données `vect.tfidf`, un algorithme de *clustering* sous contraintes `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 20 itérations, soit 1 000 annotations, dont 653 (respectivement 668) contraintes MUST-LINK.

- Les tentatives les plus lentes furent celles avec un prétraitement des données `prep.simple`, une vectorisation des données `vect.frcorenewsmd`, un *clustering* `clust.hier.sing`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 394 itérations, soit 19 656 annotations, dont 682 contraintes MUST-LINK.

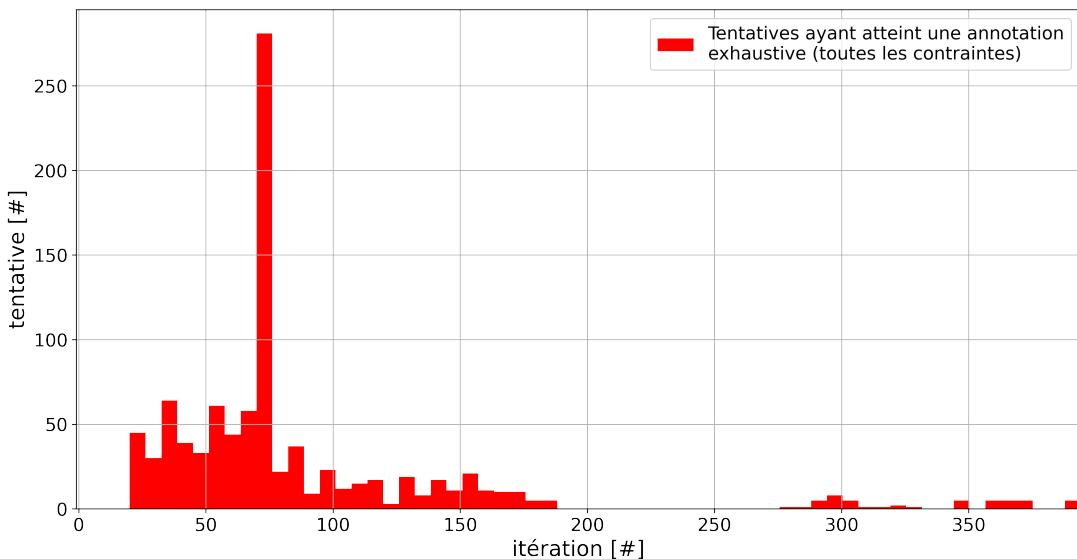


FIGURE 4.7 – Répartition des tentatives en fonction de l’itération de la méthode à laquelle elles atteignent le seuil d’une annotation exhaustive, c’est-à-dire l’itération à laquelle toutes les contraintes possibles entre les données ont été annotées. L’histogramme est réduit à 60 pics pour simplifier l’affichage.

La TABLE 4.4 retranscrit l’influence de chacun des paramètres sur le nombre d’itérations nécessaires pour atteindre une **annotation exhaustive**. Les analyses de variance mettent en relief l’effet significatif sur cette convergence du prétraitement (**eta-carré** : 0.909, **p-valeur** : $< 10^{-3}$), de la vectorisation (**eta-carré** : 0.985, **p-valeur** : $< 10^{-3}$), du *clustering* (**eta-carré** : 0.999, **p-valeur** : $< 10^{-3}$) et de l’échantillonnage (**eta-carré** : 0.997, **p-valeur** : $< 10^{-3}$). L’analyse post-hoc de ces effets indique que le meilleur paramétrage moyen pour atteindre une **annotation exhaustive** repose sur le prétraitement `prep.lemma`, la vectorisation `vect.tfidf`, le *clustering* `clust.kmeans.cop`, et l’échantillonnage `samp.random.same`. La moyenne du nombre d’itération requis pour ce paramétrage est de 32.60 (écart-type : 1.14), soit 1 630 annotations (écart-type : 57.00).

La FIGURE 4.8 représente les évolutions moyennes de la **v-measure** du *clustering* en fonction du nombre d’itération de la méthode pour les différentes valeurs des facteurs analysés (prétraitement en haut à gauche, vectorisation en haut à droite, *clustering* en bas à gauche, échantillonnage en bas à droite). La FIGURE 4.9 représente cette même évolution pour les meilleurs paramétrages moyens destinés à atteindre les trois seuils d’annotation définis (partiel, suffisant, exhaustif), où nous y constatons une baisse significative du nombre d’itérations nécessaire à la convergence par rapport à la moyenne des tentatives.

Description des facteurs analysés		Description statistique des itérations			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	p-valeur
prétraitement	prep.lemma	85.89	(1)	0.42	0.052	$< 10^{-3}$ (***)
	prep.filter	89.55	(2)			
	prep.simple	89.64	(2)			
	prep.no	90.81	(4)			
vectorisation	vect.tfidf	85.50	(1)	0.39	0.165	$< 10^{-3}$ (***)
	vect.frcorennewsmd	92.46	(2)			
clustering	clust.kmeans.cop	64.99	(1)	0.39	0.894	$< 10^{-3}$ (***)
	clust.hier.avg	78.54	(2)			
	clust.hier.ward	81.31	(3)			
	clust.hier.comp	82.49	(3)			
	clust.spec	93.78	(5)			
	clust.hier.comp	132.75	(6)			
échantillonnage	samp.random.same	57.23	(1)	0.42	0.930	$< 10^{-3}$ (***)
	samp.random.full	72.80	(2)			
	samp.closest.diff	98.38	(3)			
	samp.farhtest.same	132.75	(4)			

TABLE 4.4 – ANOVA du nombre d’itérations nécessaires pour annoter toutes les contraintes possibles. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne Moyenne indiquent le classement des niveaux selon les analyses post-hoc.

4.2.1.c Discussion

L’objectif de l’étude est de trouver une implémentation “efficiente” du *clustering* interactif permettant d’obtenir une base d’apprentissage correctement annotée en un minimum d’annotation. Pour trouver si une telle implémentation existe et quels en sont les paramètres optimaux, nous avons analysé l’impact de différentes paramétrages sur les tâches principales de la méthode (**prétraitement**, **vectorisation**, **clustering sous contraintes**, **échantillonnage**) en nous basant sur des simulations d’annotation d’un jeu de données.

Dans l’optique d’être efficient, nous excluons le désir d’annoter **exhaustivement** le jeu de données car la charge de travail estimée est trop importante. (cf. discussion de la SECTION 4.1 (hypothèse d’efficacité)) Nous préférons donc nous concentrer sur deux seuils d’annotation plus réalistes : celui d’une **annotation partielle** (atteindre 90% de v-measure avec la vérité terrain) et celui d’une **annotation suffisante** (atteindre 100% de v-measure avec la vérité terrain en un minimum de contraintes).

L’étude réalisée met en avant l’impact significatif des quatre tâches principales (**prétraitement**, **vectorisation**, **clustering sous contraintes**, **échantillonnage**) sur la vitesse de convergence de la méthode pour atteindre les seuils définis de 90% et 100% de v-measure. Il existe donc bien un paramétrage permettant d’optimiser l’implémentation proposée et de réduire le nombre de contraintes nécessaires à annoter :

- pour une **annotation partielle** (90% de v-measure), le meilleur paramétrage moyen

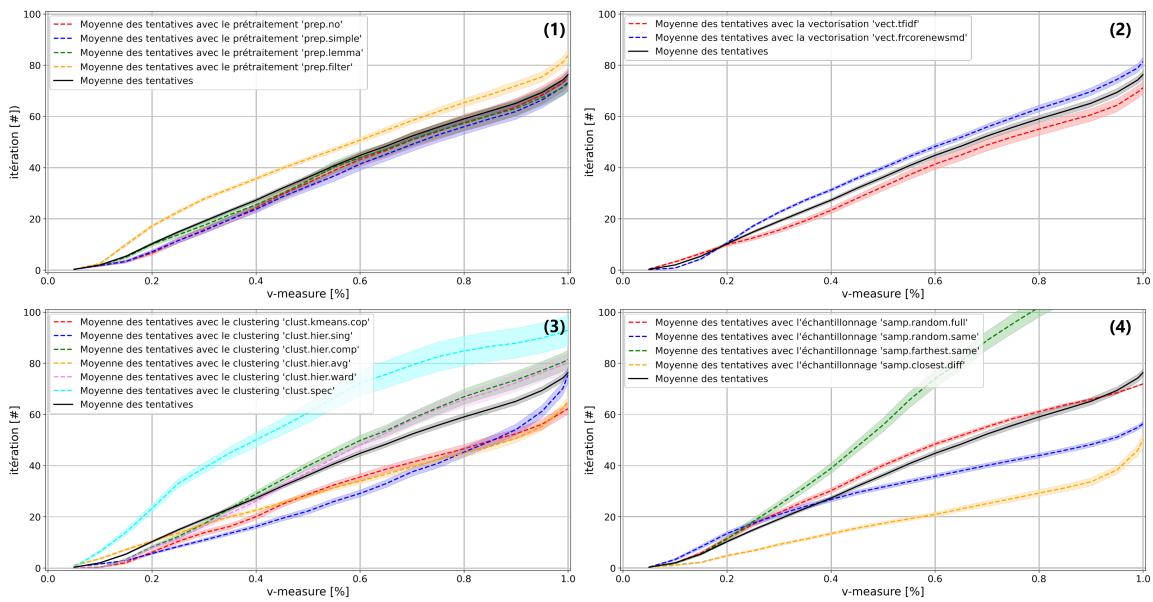


FIGURE 4.8 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de clustering interactif pour obtenir un seuil défini de *v-measure* entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes valeurs que peuvent prendre les facteurs analysés et affichées par facteur : (1) prétraitement, (2) vectorisation, (3) clustering et (4) échantillonnage.

Note : Le seuil d’annotation exhaustive (annoter toutes les contraintes possibles) n’étant pas exprimé en terme de *v-measure*, ce seuil n’est pas affiché ici.

est constitué du prétraitement simple (`prep.simple`), de la vectorisation TF-IDF (`vect.tfidf`), du *clustering* hiérarchique à lien moyen (`clust.hier.avg`) et de l’échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`). Avec ce paramétrage, il faut en moyenne 950 annotations de contraintes pour obtenir une *v-measure* de 90% ;

2. pour une **annotation suffisante** (100% de *v-measure*), le meilleur paramétrage moyen est constitué du prétraitement avec lemmatisation (`prep.lemma`), de la vectorisation TF-IDF (`vect.tfidf`), du *clustering* KMeans avec modèle COP (`clust.kmeans.cop`) et de l’échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`). Avec ce paramétrage, il faut en moyenne 1 750 annotations de contraintes pour obtenir une *v-measure* de 100% ;
3. le cas d’une **annotation exhaustive** (annoter toutes les contraintes possibles sur les données) n’est pas explicité ici mais peut se déduire des résultats décrits plus haut.

Notes de l'auteur : On note que les facteurs de prétraitement et de vectorisation sont significatifs possèdent toutefois de faibles valeurs de variance expliquée ($\eta^2 < 0.40$).

Les facteurs de *clustering* et d’échantillonnage ont quand à eux des valeurs plus fortes ($\eta^2 > 0.70$), dénotant ainsi un plus grand pouvoir explicatif de la variance des résultats. Notre attention s’attarde particulièrement sur l’échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`) qui s’avère très prometteur :

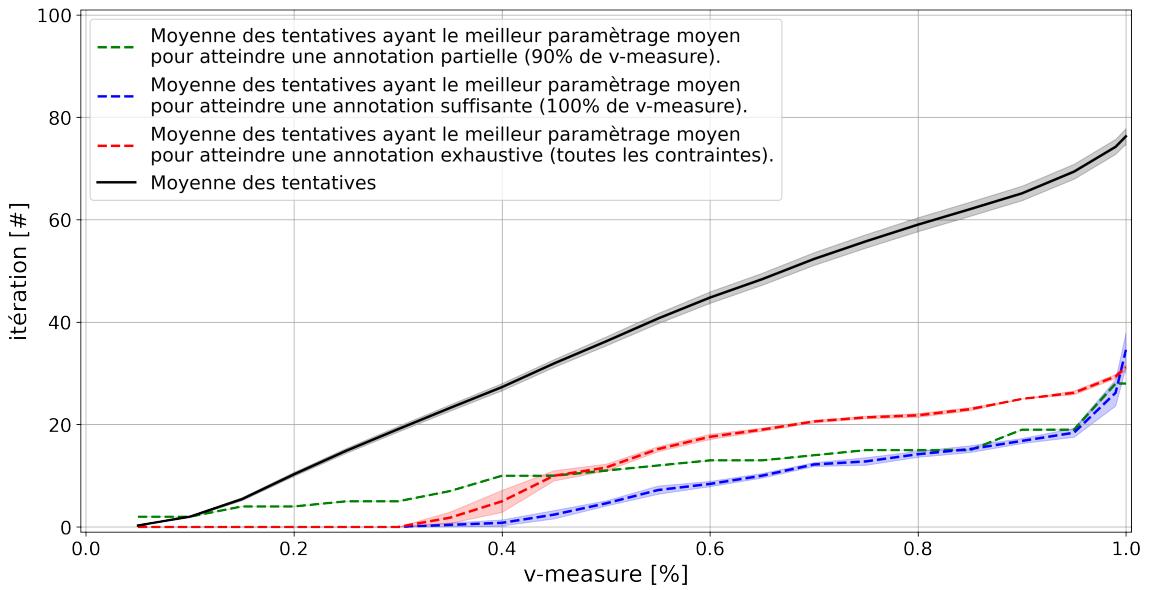


FIGURE 4.9 – Évolution des moyennes du nombre d'itérations nécessaire de la méthode de clustering interactif pour obtenir un seuil défini de v -measure entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes seuils d'annotations étudiés : l'annotation partielle (atteindre une v -measure de 90%), l'annotation suffisante (atteindre une v -measure de 100%) et l'annotation exhaustive (annoter toutes les contraintes possibles).

cette sélection semble permettre de corriger efficacement la frontière des *clusters* en favorisant l'ajout de contraintes MUST-LINK sur des données séparées à tord. Or les corrections introduisant des contraintes MUST-LINK sont plus explicites que les contraintes CANNOT-LINK dont l'utilisation est plus ambiguë (*dans le premier cas, on sait que les deux données sont désormais à mettre dans le même cluster ; dans le second, on sait qu'il faut séparer les deux données sans exprimer dans quels clusters ils doivent aller*). Ainsi, nous sommes d'avis que l'échantillonnage `sampl.closest.diff` est efficiente car elle permet d'introduire efficacement des contraintes dont l'utilité est immédiate pour corriger le *clustering*.

Ainsi, cette étude permet de répondre à la limite du nombre de contraintes requis (discutée dans l'hypothèse d'efficacité, SECTION 4.1). En effet, l'optimisation des paramètres de l'implémentation du *clustering* interactif permet de réduire considérablement le nombre de contraintes nécessaires pour obtenir une base d'apprentissage exploitable. En nous basant sur la TABLE 4.1 de l'étude de convergence, et dans le cadre de l'annotation d'un jeu de 500 données, nous sommes passé d'un paramétrage moyen nécessitant 3 750 (respectivement 10 000) contraintes à un paramétrage optimisé ne nécessitant que 950 (respectivement 1 750) contraintes pour atteindre un seuil de 90% (respectivement 100%) v -measure. L'ordre de grandeur de la charge de travail demandée aux annotateurs est donc située entre 2 et 4 fois la taille du jeu de données.

Cette estimation est plus raisonnable que celle réalisée en SECTION 4.1. De plus, en considérant que les annotations sont binaires et demandent a priori une charge mental plus faible que les annotations par attribution de label ("les données sont-elles similaires ?" vs "quel est l'étiquette de cette donnée ?", cf. HART et STAVELAND, 1988), nous pouvons espérer que la charge totale nécessaire à l'annotation avec une méthodologie basée sur le *clustering* interactif est comparable

à celles des méthodes traditionnelles. Nous confirmerons cette conclusion en étudiant le temps nécessaire à un opérateur pour annoter un lot de contraintes (cf. SECTION 4.3.1).

Afin de compléter cette analyse d'efficience, quelques pistes sont encore à explorer.

D'une part, une étude de coût est à réaliser pour trancher le choix de paramètre optimaux réalistes. En effet, il est intéressant d'étudier le coût machine (temps CPU utilisé) et le coût humain (temps d'annotation) afin d'affiner les choix techniques et de compléter les arguments sur l'utilisation en situation réelle d'une méthodologie d'annotation basée sur le *clustering* interactif. Cette étude sera l'occasion de rentrer en détail dans la comparaison de la charge demander à l'annotateur, tant sur la durée que sur la complexité de la tâche d'annotation. Cet aspect sera traité dans la SECTION 4.3 (hypothèse des coûts).

D'autre part, l'étude réalisée se base sur des seuils de performance par rapport à une vérité terrain. Or en situation réelle, cette comparaison avec la vérité terrain n'est pas possible car elle est précisément en cours de conception (la base d'apprentissage finale devant être la vérité terrain). De plus, un tel score n'est pas le plus explicite pour pour un expert métier pour qui un score de **v-measure** n'est pas révélateur de la pertinence métier de la segmentation proposée des données. Dans un registre similaire, il est possible que l'évolution du partitionnement des données passe par plusieurs états stables et pertinents, mais que l'annotateur soit obligé d'affiner sa vision en annotant certaines contraintes ambiguës. Cela peut être le cas avec des *clusters* traitant en fin de compte de sujets très similaires (*ajouter des MUST-LINK pour les fusionner*) ou avec un *cluster* qui regroupe finalement plusieurs thématiques (*ajouter des CANNOT-LINK pour le segmenter*). Il manque donc une stratégie d'évaluation de pertinence de la base d'apprentissage en cours de construction afin d'estimer la stabilité d'un partitionnement et de la suffisance des annotations réalisées pour faire refléter la vision de l'annotateur dans le résultat obtenu. Cet aspect sera traité dans la SECTION 4.4 (hypothèse de pertinence).

Pour finir, comme pour l'étude de convergence réalisée en SECTION 4.1, nous avons supposé dans cette étude que l'annotateur est un expert métier connaissant parfaitement le domaine traité. Cette hypothèse forte n'est a priori pas valable en situation réelle : En effet, des différences d'annotations peuvent intervenir (*ambiguïtés sur les données, méconnaissance du domaine, erreurs d'inattention, différence d'opinions entre annotateurs, ...*), ce qui peut entraîner des divergences ou des incohérences dans la construction de la base d'apprentissage. Il semble donc nécessaire d'étudier les impacts de ces incohérences, ainsi que de proposer une méthode pour les prévenir ou les corriger. Cet aspect sera traité à la fin de ce chapitre dans la SECTION 4.6 (hypothèse de robustesse).

4.3 Évaluation de l'hypothèse sur les coûts

Dans les deux sections précédentes, nous avons estimé le paramétrage du *clustering* interactif le plus efficient pour atteindre 90% de **v-measure** avec la vérité terrain, correspondant à ce que nous appelons une annotation partielle. Toutefois, pour compléter l'étude de faisabilité technique de notre méthode, nous devons nous intéresser aux coûts (matériel et humain) à investir pour atteindre notre objectif. Nous aimerions donc vérifier l'hypothèse suivante :

❖ Hypothèse sur les coûts ❖

« Il est possible d'estimer les coûts nécessaires d'une méthodologie d'annotation basée sur le *clustering* interactif pour obtenir une base d'apprentissage exploitable. Nous étudions en particulier les coûts relatifs au temps d'annotation, au temps de calcul des algorithmes, ainsi que la durée totale de la méthode en fonction de la taille du jeu de données. »

La FIGURE 4.10 illustre cette hypothèse et l'espoir de pouvoir caractériser la qualité de la base d'apprentissage en cours de construction en fonction d'un coût temporel au lieu d'un nombre abstrait d'itérations de la méthode.



FIGURE 4.10 – Illustration des études réalisées sur le *clustering* interactif (étape 3/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en *v-measure*) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine).

Afin de vérifier cette hypothèse, nous organisons plusieurs expériences pour simuler et déterminer ces durées :

- une étude du **temps d'annotation** par un expert métier, mesuré lors d'une expérience d'annotation de contraintes faisant intervenir plusieurs opérateurs (cf. SECTION 4.3.1) ;

- une étude du **temps de calcul** des algorithmes, modélisé en exécutant les différentes implémentations du *clustering* interactif avec diverses valeurs d'arguments (cf. SECTION 4.3.2) ;
- et une étude du **nombre de contraintes** nécessaires en fonction du nombre de données à traiter, estimé en simulant la création d'une base d'annotation avec notre méthodologie sur des jeux de données de différentes tailles (cf. SECTION 4.3.3).

Nous exposons nos conclusions sur l'estimation du **temps total** à investir et réalisons une comparaison avec une organisation plus traditionnelle d'un projet d'annotation en SECTION 4.3.4.

4.3.1 Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle

Nous voulons estimer le temps nécessaire à un opérateur pour annoter un lot de contraintes. Pour cela, nous allons chronométrier plusieurs experts métiers en train d'annoter un même échantillon et modéliser le nombre de contraintes par minute, ainsi que son évolution au cours de plusieurs sessions d'annotation. De plus, nous aimeraisons aussi confirmer que l'ajout de contraintes dans notre contexte s'apparente à une tâche "intuitive", c'est-à-dire que l'annotation se fait dans la réaction et non dans la réflexion (voir KAHNEMAN, 2011 qui distingue un *système 1* intuitif, rapide, de l'ordre de l'émotion, et un *système 2* plus lent, logique et réfléchi). Pour ce faire, nous estimons grossièrement le temps nécessaire à l'annotation réactive d'une contrainte, nous le comparons au temps moyen estimé lors de notre expérience, et nous nous demandons si la différence observée peut cacher un mécanisme cognitif plus complexe.

4.3.1.a Protocole expérimental

⚠️ Attention : Dans cette étude, nous supposons que les annotateurs de l'expérience connaissent parfaitement le domaine traité dans le jeu de données, et qu'ils sont capables de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Afin de pourvoir faire cette hypothèse forte, et ainsi limiter les bruits dans l'analyse des résultats, le jeu de données devra traiter d'un sujet de culture générale (ne nécessitant donc pas de connaissance particulière) et des réviseurs supprimeront en amont et d'un commun accord les données trop spécifiques ou trop ambiguës.

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.3.

Nous allons procéder en plusieurs étapes. D'abord, il faut choisir un jeu de données approprié : pour valider notre hypothèse forte sur les compétences de nos annotateurs, nous cherchons un jeu de données traitant d'un sujet de culture général. Pour cette expérience, nous avons donc choisi MLSUM : une collecte d'articles de journaux, classés par catégorie de publication et décrits par leur titre et leur résumé. Nous nous intéressons ici à la tâche de classification d'un titre d'article en fonction de sa catégorie de publication. Comme certains titres peuvent porter à confusion (un titre d'article n'étant pas toujours explicite sur son contenu), deux réviseurs (*une Data Scientist et moi-même*) sont chargés de choisir les données les plus explicites sur un échantillon d'un millier de données représentatives des catégories les plus communes. L'échantillon résultant, noté **MLSUM FR Train Subset (v1.0.0-schild)**, est composé de 744 titres d'articles rédigés en français et répartis en 14 classes (*économie, sport, ...*). Pour plus de détails, consultez l'annexe A.2.

```

Données : jeu de données annotées (vérité terrain)
Entrées : plusieurs réviseurs, plusieurs annotateurs
1 initialisation : définir et revoir le jeu de données entre réviseurs ;
2 échantillonnage : sélectionner une base de contraintes avec samp.rand.full ;
3 temps théorique : estimation du temps nécessaire à l'annotation d'une contrainte ;
4 pour chaque annotateur faire
5   tant que la base de contraintes n'a pas été entièrement annotée faire
6     chronomètre : START ;
7     annotation : annoter une partie des contraintes ;
8     revue : revue des contraintes en conflits d'annotation ;
9     chronomètre : STOP ;
10    mesure : estimer la différence de chronomètre pour cette session ;
11 analyse : modéliser le temps d'annotation d'un lot de contraintes ;
Résultat : modélisation du temps d'annotation d'un lot de contraintes

```

ALGORITHME 4.3 – Description en pseudo-code du protocole expérimental de l'étude du temps d'annotation d'un lot de contraintes par plusieurs experts métiers en situation réelle.

À partir de ces données, nous sélectionnons un lot de 1 000 contraintes à annoter. Comme nous nous intéressons exclusivement au temps d'annotation pour cette expérience (et que nous ne regardons pas le nombre d'itérations de la méthode), nous utilisons l'échantillonnage purement aléatoire (**samp.rand.full**). L'analyse de l'accord inter-annotateurs sera réalisé en SECTION 4.6.1.

Sur la base de cette échantillon, nous pouvons approximer le temps théorique nécessaire à l'annotation d'une contrainte à 6.8 secondes. En effet, il faut d'abord considérer la taille moyenne des titres d'article à lire et en déduire le temps dédié à la lecture des deux textes de la contraintes. En utilisant l'approximation d'une lecture silencieuse par un adulte à 238 mots par minute (BRYSBERT, 2019) et en mesurant la taille moyenne des titre d'article à 10.1 mots, on en déduit que le temps de lecture d'un texte est environ de 2.55 secondes. Ensuite, il convient d'intégrer la durée de traitement cognitif requis pour estimer si les deux phrases sont similaires ou discordantes. À cet effet, nous retenons 1 seconde¹⁹ (PURVES et BRANNON, 2013) en admettant que cette tâche est rapide. Enfin, nous ajoutons 1 seconde supplémentaire pour représenter la réaction motrice (clic de bouton) et le délais applicatif (rechargement de la page). Au total, nous obtenons ainsi un temps d'annotation moyen de 7 secondes. Bien entendu, cette durée reste approximative, mais elle nous permet de discuter de l'ordre de grandeur à manipuler durant l'annotation.

Ensuite, un groupe de 14 annotateurs vont annoter la sélection de 1 000 contraintes en plusieurs sessions. Les directives données aux opérateurs sont les suivantes :

- **Contexte de l'opérateur** : « *Vous êtes des experts de la presse et de l'actualité ; Vous voulez classer des articles dans des catégories en fonction de leur titre ; Vous ne savez pas précisément quelles catégories vous allez utiliser pour classer vos articles ; Mais vous savez caractériser la similitude de deux articles* » ;

19. Nous pourrions faire le parallèle avec la composante P600 communément admise en neuroscience pour caractériser la réaction provoquée par la dissonance grammaticale ou syntaxique d'une phrase. Nous arrondissons à 1 seconde pour garder une marge d'erreur.

- **Contexte sur le jeu de données :** « Le thème sont les catégories d'articles de presse ; La vérité terrain contient entre 10 et 20 catégories parmi les plus communes de la presse ; La vérité terrain contient entre 30 et 100 articles par catégorie ; Vous pouvez regarder le jeu de données non annoté autant que vous le voulez (disponible dans l'onglet TEXTS de l'application) » ;
- **Objectif de l'expérience :** « Je veux savoir le temps nécessaire pour annoter un certain nombre de contraintes ; Autrement dit : Pour annoter 1000 contraintes, combien de temps me faut-il ? » ;
- **Consignes d'annotations :** « Faites des séries de 15 minutes minimum pour avoir de la régularité ; Si possible, isolez-vous pour ne pas être dérangé et ne pas fausser les résultats ; Pour chaque série, notez le temps et le nombre de contraintes annotés ; Si vous ne savez pas quoi annoter (trop ambigu, vocabulaire inconnu, ...), passez au suivant sans annoter (vous êtes sensés être des experts de la presse !) ».

Pour réaliser l'annotation, les opérateurs auront accès à l'application web développée au cours de ce doctorat. Des captures d'écran sont disponibles en FIGURE 4.11 et FIGURE 4.12. Une description plus détaillée de l'application et de ses fonctionnalités est disponible en ANNEXE C.4.

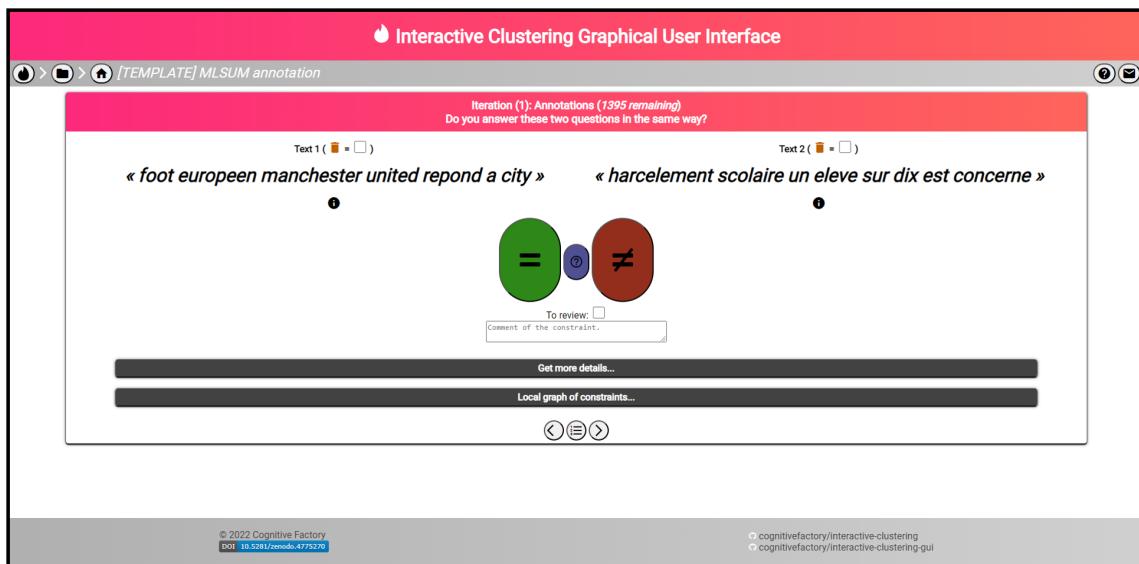


FIGURE 4.11 – Capture d'écran de l'application web permettant utilisant notre méthodologie de clustering interactif : page d'annotation de contraintes. Les deux textes à annoter sont disposés à gauche et droite de l'écran. Chacun dispose d'un cache à cocher si le texte n'est pas pertinent à analyser (ambigu, hors périmètre, incompréhensible, ...).

Les boutons à disposition permettent respectivement d'annoter un **MUST-LINK** si les données sont similaires (bouton en vert), un **CANNOT-LINK** si les données ne sont pas similaire (bouton en rouge), d'ignorer la contrainte pour laisser la main à l'algorithme de clustering (bouton en bleu), et d'ajouter un commentaire pour revoir la contrainte plus tard (case à chosir et champ de texte libre). Deux éléments déroulant permettent d'avoir des informations supplémentaires (metadata de sélection et de clustering, représentation graphique des liens entre contraintes annotées). Les boutons de navigation (boutons flèches et liste) sont disponibles en bas de page.

4.3. Évaluation de l'hypothèse sur les coûts

List of constraints								
Sort by: Iteration of san ▾ ascending ▾								
Text 1	Constraint	Text 2	Sampling Iteration	Last update	To annotate	To review	To fix	Go to
« comment lyon a banni les pesticides de ses parcs et jardins »	DAMNED	« pour marisol touraine la liberté d'installation des médecins est préservée »	1	05/06/2023, at 16:43:00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(>)
« le bouclier fiscal c est reporter la fiscalité des plus riches vers les moins riches »	DAMNED	« immobilier ces voisins qui coutent cher »	1	05/06/2023, at 16:43:11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(>)
« jeu de piste entre picasso et godard »	MUST	« le centre pompidou expose wifredo lam pour la première fois »	1	05/06/2023, at 16:43:17	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(>)
« harclement scolaire un eleve sur dix est concerne »	DAMNED	« a istanbul l'art a un air de defi »	1	05/06/2023, at 16:43:21	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(>)
« football luis fernandez est le nouveau selectionneur d israel »	DAMNED	« logements la correction du marche neuf devrait se poursuivre »	1	05/06/2023, at 16:43:25	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(>)
« deces d'egon bahr figure du spd et artisan de l'ostpolitik »	SKIP	« la droite s interroge sur la strategie de nicolas sarkozy »	1	Never	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(>)
« chomage les demandeurs d'emploi globalement satisfaisants du pole emploi »	SKIP	« d autres virus h7 potentiellement dangereux pour l'homme a l'instar de la grippe aviaire »	1	Never	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(>)
« bygmalion nicolas sarkozy directement vise »	SKIP	« logement la crise sans fin »	1	Never	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(>)

FIGURE 4.12 – Capture d'écran de l'application web permettant utilisant notre méthodologie de clustering interactif : *page d'inventaire des contraintes à annoter*.

La partie supérieure permet d'identifier le nombre de textes et de contraintes sur le projet, ainsi que les boutons destinés à calculer les transitivités entre les contraintes et à approuver le travail réalisé si aucune transitivité n'entre en conflit avec un contrainte annotée. La partie inférieure liste l'ensemble des contraintes du projet, avec les annotations réalisées, l'itération à laquelle la contraintes a été sélectionnée et annotée, si elle est à revoir ou si une incohérence la concernant est détectée.

Une fois les sessions d'annotations terminées, nous entraînons un modèle linéaire généralisé (*GLM*) pour estimer le temps d'annotation moyen pour un lot de contraintes (dont la taille est notée `batch_size`). Ce modèle sera caractérisé par le coefficient de détermination généralisé R^2 de Cox et Snel (DIAMOND et al., 1990), la log-vraisemblance `llf` (EDWARDS, 1992) et la log-vraisemblance `llf_null` du modèle *null*. Nous discutons aussi de l'évolution de la vitesse d'un opérateur au cours des différentes sessions d'annotation.

i Pour information : L'outil d'annotation utilisé est accessible dans SCHILD, TREMBLE et MISIAK, 2022. Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), ainsi que le projet à importer dans l'outil d'annotation, sont disponibles dans un dossier dédié de SCHILD, 2022b. Nous utilisons entre autres les librairies `datetime`²⁰ et `statsmodels`²¹ (SEABOLD et PERKTOLD, 2010).

4.3.1.b Résultats obtenus

Durant cette expérience, 14 annotateurs ont participé à l'annotation de 1 000 contraintes aléatoires sur un jeu de données. Ces opérateurs travaillent tous dans un service informatique

20. `datetime` : <https://pypi.org/project/datetime/>

21. `statsmodels` : <https://pypi.org/project/statsmodels/>

dédié à l'entraînement et l'amélioration de solutions de *Machine Learning* et sont répartis de la manière suivante :

- 4 femmes, 10 hommes ;
- 7 personnes entre 20 et 30 ans, 7 personnes entre 30 et 40 ans ;
- 9 *Data Scientist*, 4 développeurs et 1 ergonome ;
- 1 personne ayant révisé le jeu de données, 3 le découvrant pour la première fois dont 1 ayant travaillé auparavant dans le secteur de la presse.

Par manque de disponibilités, 4 annotateurs n'ont que partiellement réalisé leur tâche : nous avons toutefois intégré leurs participations car elles contenaient au minimum 150 annotations.

D'après les observations, un annotateur réalisait en moyenne 170.7 contraintes par session d'annotation (min : 43, max : 547, médiane : 138, écart-type : 106.4) ce qui lui demandait en moyenne 23.1 minutes (min : 3.0, max : 92.0, écart-type : 14.4). De plus, la vitesse d'annotation moyenne était de 7.7 contraintes par minute (min : 3.5, max : 14.3, écart-type : 2.9).

Le modèle linéaire généralisé entraîné sur les mesures de temps d'annotations ($R^2 : 0.910$, `llf` : -499.15, `llf_null` : -539.95) nous permet de déduire l'équation suivante :

$$\text{annotation_time [s]} \propto 7.8 \cdot \text{batch_size} \quad (4.1)$$

La FIGURE 4.13 représente cette modélisation du temps d'annotation en comparaison avec les mesures réalisées lors de l'expérience. Pour rappel, le temps théorique estimé précédemment est de 7 secondes par contrainte.

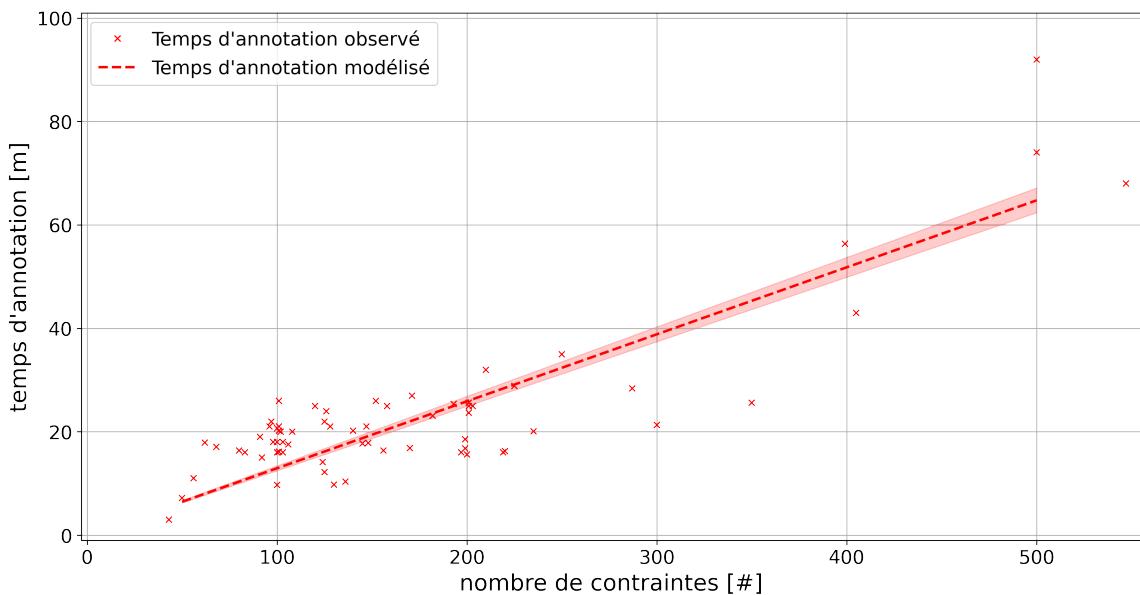


FIGURE 4.13 – Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.

En ce qui concerne l'évolution de la vitesse d'annotation au cours des sessions, aucune tendance significative n'a été identifiée. La FIGURE 4.14 représente l'évolution de vitesse d'annotation pour quatre opérateurs (les deux plus rapides et les deux plus lents). Ces données sont l'objet d'une étude de cas dans la discussion ci-dessous.

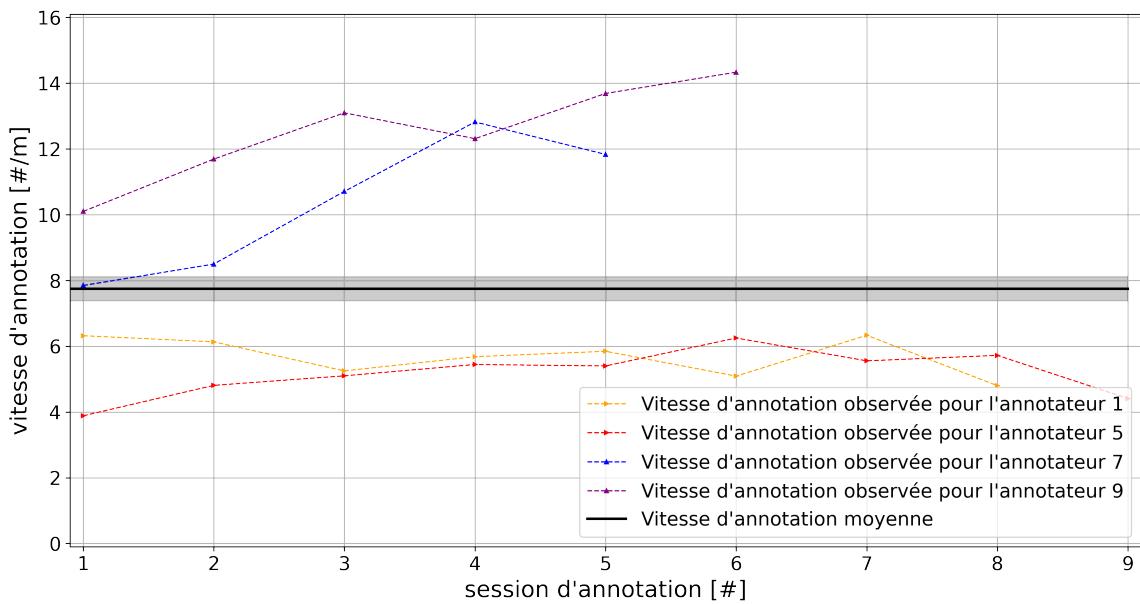


FIGURE 4.14 – Étude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations.

4.3.1.c Discussion

L'étude réalisée avec 14 annotateurs sur des lots de 1 000 contraintes a permis d'estimer à $7.8 \cdot \text{batch_size}$ le temps nécessaire (en secondes) pour annoter un lot de contraintes (cf. FIGURE 4.13).

Notes de l'auteur : Avant poursuivre la discussion, il est nécessaire de préciser qu'il est difficile de comparer ces résultats. D'une part, il y a une forte disparité des mesures, et il est idyllique de penser qu'une étude sur 14 annotateurs peut représenter la diversité du comportement humain sur une tâche aussi complexe que l'annotation de données textuelles. D'autre part, il y a un manque de repères concrets dans la littérature scientifique, entre autres à cause des nombreux facteurs intervenant dans une tâche d'annotation (*objectifs à réaliser, données à manipuler, nombre de choix proposés à l'opérateur, complexité sémantique des données, des compétences de l'opérateur, fréquence d'exécution de la tâche, ...*), mais aussi en raison du manque d'intérêt à l'analyse du temps nécessaire au profit de l'analyse de la cohérence et de la qualité intra- ou inter-annotateur (BALEDENT, 2023). De plus, les résultats peuvent différer en fonction des contraintes à caractériser : on peut supposer que des couples de données très similaires ou très différentes sont simples à annoter, mais que des données plus ambiguës peuvent nécessiter davantage de temps pour être intégrées et étiquetées.

Pour pallier ce problème, nous proposons confronter nos estimations à des mesures réalisées sur des tâches n'ayant pas le même objectif mais dont la complexité est comparable. Cette approche, bien qu'un peu rudimentaire, nous permettra ainsi de discuter des ordres de grandeurs à manipuler.

Analyse de l'annotation d'une contraintes. En premier lieu, nous voulions confirmer que l'annotation de contraintes est une tâche intuitive dont la durée est caractéristique d'un mécanisme de réaction et non d'une réflexion. Pour ce faire, nous avons estimé la durée théorique d'une annotation à 7 secondes par contrainte, comprenant les temps de lecture, d'analyse de la similitude et d'action de l'opérateur, et nous avons mesuré la durée d'annotation réelle à 7.8 secondes par contrainte. Bien que l'écart constaté (0.8 seconde) soit compris dans les bornes d'approximation, cette différence n'est pas suffisamment insignifiante pour nous permettre d'exclure avec certitude la présence d'un mécanisme cognitif supplémentaire.

Pour nous permettre de discuter du temps d'annotation mesuré et de son approximation théorique, nous utilisons utiliser plusieurs points de référence extraits de (SNOW et al., 2008). Dans cette étude, les auteurs délèguent quelques tâches d'annotation à Amazon Mechanical Turk²² :

1. Tâche de "*désambiguïsation du sens des mots*" (PRADHAN et al., 2007) : Elle consiste à catégoriser le contexte de phrases comportant le mot "*président*" suivant 3 options préformatées (*dirigeant d'une entreprise*, *dirigeant des États Unis*, *dirigeant d'un autre pays*). Il y a 177 phrases à labelliser par 10 annotateurs, et la tâche a été réalisée en un total de 8.59 heures, soit une moyenne de 17.5 secondes par annotation. Nous utilisons ce point de repère car la tâche s'apparente à une annotation d'une tâche de classification (3 catégories).
2. Tâche de "*caractérisation de similitude des mots*" (MILLER et CHARLES, 1991) : Elle consiste à ordonner des couples de mots du plus similaire au moins similaire en fonction de leur proximité sémantique afin de mettre en avant les meilleurs couples de synonymes. Il y a 30 paires de synonymes à ordonner par 10 annotateur, et la tâche a été réalisée en un total de 0.17 heures, soit une moyenne de 2.0 secondes par annotation. Nous utilisons ce point de repère car la tâche s'apparente à l'annotation de contraintes (*laquelle de ces deux paires est la plus adéquate ?*) qui ne fait pas appel à un mécanisme de réflexion (*peu de vocabulaire, similarité intrinsèque triviale entre deux mots*).
3. Tâche de "*reconnaissance de l'implication textuelle*" (DAGAN et al., 2005) : Elle consiste à confirmer ou infirmer si une phrase est la conséquence d'une autre (*l'annotation est donc binaire*). Il y a 800 paires de phrases à valider par 10 annotateur, et la tâche a été réalisée en un total de 89.3 heures, soit une moyenne de 40.2 secondes par annotation. Nous utilisons ce point de repère car la tâche s'apparente à l'annotation de contraintes (*est-ce que l'implication est vraie ?*) faisant intervenir un mécanisme de réflexion (*comprendre une implication logique*).

En utilisant la tâche de "*désambiguïsation du sens des mots*" (1.), nous avons déjà un point de comparaison entre notre annotation de contraintes et une annotation classique de classes. Nous constatons une nette différence entre les deux approches (7.8 secondes par contrainte vs 17.5 secondes par donnée), cela met en avant qu'une annotation de contraintes est plus rapide qu'une annotation par label.

Ensuite, en utilisant la tâche de "*caractérisation de similitude des mots*" (2.), nous pouvons confirmer que notre approximation théorique (faisant intervenir un traitement cognitif de 1 seconde) semble adaptée pour représenter un mécanisme de l'ordre de la réaction. En effet, le coût très faible de la caractérisation d'une similarité entre deux mots (2.0 secondes par paire de mots) s'avère être en adéquation avec cette approximation (2.5 secondes par paire de phrases de taille

22. Amazon Mechanical Turk est une plateforme de travail collaboratif en ligne (*crowdsourcing*) : nous avions évoqué leur fonctionnement, leur avantages et leurs inconvénients en SECTION 2.3.2.C, notamment le risque de l'abandon de la qualité des annotations au profit de la quantité.

1).

Enfin, en utilisant la tâche de "reconnaissance de l'implication textuelle" (3.), nous comparons deux annotations binaires dont une fait nettement appel à un mécanisme de réflexion (déduction logique dans une implication). Nous constatons une très nette différence entre les deux approches (7.8 secondes par contrainte vs 40.2 secondes par implication), ce qui nous permet d'exclure la présence de mécanisme cognitif trop complexe.

Points à retenir : Mettant bout à bout ces comparaisons, et en gardant à l'esprit que ces références restent approximatives, nous pouvons conclure que (1) **l'annotation de contraintes est une tâche plus rapide qu'une annotation par label** et que (2) **cette annotation binaire ne semble pas faire pas intervenir de traitement cognitif complexe** (*c'est une piste à explorer plus en détails pour expliquer le gain de temps observé*).

Analyse d'une session d'annotation de contraintes. Nous avons analysé l'évolution de la vitesse d'annotation au cours des sessions d'annotation, en espérant observer une accélération des annotations au fur et à mesure que l'annotateur s'habitue avec la tâche, ainsi qu'un effet de fatigue pour des sessions d'annotations trop longues. Cependant, aucune de nos analyses n'a montré de résultats significatifs (on peut constater la forte dispersion des résultats grâce à la FIGURE 4.14). Nous ne pouvons donc pas conclure sur de telles tendances.

Notes de l'auteur : Nos intuitions initiales concernaient deux points :

- la diminution du **temps d'adaptation** au cours de sessions d'annotations : au fur et à mesure qu'il annote, l'opérateur pourrait entrer plus facilement dans sa tâche, lui permettant d'atteindre plus rapidement sa vitesse de croisière et ainsi gagner en efficacité sur plusieurs sessions. D'après ANDERSON, 2013, ce temps d'adaptation pourrait se définir en trois étapes : une phase déclarative (*besoin d'instructions détaillées, exécution lente et avec erreurs*), une phase associative (*quelques rappels clés suffisent pour retrouver les instructions, donc gain de vitesse*) et une phase autonome (*les consignes sont acquises, donc exécution rapide et sans erreur*) ;
- l'intervention d'un **effet de fatigue** : si une session d'annotation dure trop longtemps, l'opérateur pourrait perdre en efficacité par manque de concentration et augmenter ses chances de faire des erreurs. D'après JONES et al., 2015, la fatigue est considérée comme un inconfort qui s'installe après une tâche excessive, et ELKOSANTINI et GIEN, 2009 décrit cet état de fatigue par des capacités de travail réduites.

Ces différentes intuitions ont aussi été remontées par les annotateurs de notre expériences, mais aucun effet significatif n'a pu être observé.

Par extension, nous ne pouvons pas non plus conclure sur la taille optimale d'échantillon de contraintes à sélectionner pour une session d'annotation. Dans nos précédentes études, nous avions arbitrairement fixé la taille de lot à 50 pour bénéficier d'itérations brèves, permettant à l'algorithme de *clustering* de l'améliorer régulièrement avec les dernières contraintes. Mais des petits lots d'annotation démultiplient le nombre d'itérations à réaliser, et donc le nombre d'algorithmes à exécuter. Il serait donc judicieux d'adapter le nombre d'annotations à réaliser

pour d'améliorer l'expérience utilisateur en situation réelle de l'opérateur. Malheureusement, aucun repère significatif ne peut être déduit de nos résultats pour prédire la fin d'un temps d'adaptation ou le début d'un effet de fatigue.

Afin de proposer tout de même un ordre de grandeur de taille de lot, nous pouvons nous intéresser au nombre moyen de contraintes annotées lors des sessions réalisées par les opérateurs de notre expérience. Bien que ces informations n'ont pas été collectées initialement à cette fin, on peut supposer que les opérateurs ont interrompu leur session pour se reposer (*par fatigue, ennui, agacement, ...*) ou répondre à une autre sollicitation (*intervention d'un collègue, mail important, pause café, ...*) Après un entretien avec les opérateurs de notre expérience, il semble y avoir deux possibilités : soit l'opérateur ne se fixait pas d'objectif et s'arrêtait par fatigue ; soit il se fixait un objectif (*de nombre ou de durée*), mais adaptait son prochain objectif en fonction de la fatigue ressentie en fin de session. Dans les deux cas, nous pouvons faire l'hypothèse que le nombre moyen d'annotation par session tend à représenter une borne supérieure de la taille maximale d'un lot à considérer pour ne pas entamer l'effet de fatigue. Sur l'expérience réalisée, cette moyenne est de 170.70 contraintes annotées par session (écart-type : 106.37, erreur standard : 13.19). En prenant en compte une marge d'erreur pour minimiser ce résultat, nous retenons 150 contraintes comme seuil à ne pas dépasser.

Points à retenir : Pour une session d'annotation, **nous conseillons une taille d'échantillon entre 50 et 150 contraintes**. Attention aux échantillons trop petits qui multiplient le nombre d'itérations à réaliser ; Attention aussi aux échantillons trop gros qui peuvent introduire un effet de fatigue chez l'opérateur et casser la dynamique d'interactions avec la machine. La discussion finale de ce chapitre affinera cette fourchette grâce à une vue d'ensemble sur les coûts de la méthode.

Analyse des fonctionnalités de l'application d'annotation. Pour finir cette discussion, nous nous intéressons à l'utilisation du logiciel par nos opérateurs au cours de cette étude. En effet, il est logique de penser que la conception de l'application et les fonctionnalités qu'elle dispose peut grandement impacter l'expérience utilisateur de notre méthodologie de *clustering* itératif. Un entretien avec les opérateurs a permis de remonter plusieurs pistes d'amélioration sur son ergonomie.

Un premier point concerne l'affichage des textes d'une contraintes. Comme montré en FIGURE 4.11, nous avions choisi d'afficher des données normalisées à l'écran pour masquer le bruit provoqué par les accents, les majuscules, la ponctuation. Bien que cette fonctionnalité peut servir pour des textes bruts (issues de conversation clients ou de forum par exemple), cela a plutôt nuit à la compréhension des données par les opérateurs. Nous avons donc envisager de laisser la données brutes à disposition, dans une infobulle ou grâce à une option permettant d'interchanger le format des données.

Une seconde proposition concerne l'ordre des contraintes à annoter. Nous pouvons facilement admettre que la compréhension rapide d'un grand nombre de texte est un tâche pénible. Pour soulager les opérateurs et limiter le nombre de transitions, nous avons trier les contraintes par ordre alphabétique. Ainsi, toutes les contraintes associées à une même donnée peuvent être traitées à la suite. Cette solution permet de limiter le nombre de changement de contexte et peut faciliter la caractérisation d'une similitude en analysant à la chaîne plusieurs données du même type. À cet effet, une option e tri à été ajouté sur la liste de contraintes à traiter (cf. FIGURE 4.12).

Enfin, une dernière idée concerne l'affichage des données à annoter. Nous avions jusqu'à présent considérer l'annotation entre deux données (cf. FIGURE 4.11), mais il peut être judicieux d'afficher plusieurs données à caractériser simultanément. Une telle fonctionnalité permettrait ainsi de regrouper rapidement un grand nombre de données similaires ou de distinguer avec moins d'ambiguïté certaines données en s'appuyant sur leur voisinage.

i Pour information : Ces différentes évolutions sont en cours d'analyse ou ont déjà été intégrées dans l'application que nous proposons (SCHILD, TREMBLE et MISIAK, 2022).

4.3.2 Étude du temps de calcul nécessaire aux algorithmes implémentés en chronométrant des exécutions dans différentes situations

Maintenant que nous avons pu modéliser le temps nécessaire à un expert pour annoter un lot de contraintes, nous nous intéressons au temps nécessaire à la machine pour interpréter ces annotations et proposer une nouvelle segmentation des données.

Comme les différents algorithmes manipulent des contraintes sur les données, l'estimation théorique du temps d'exécution par l'analyse de la complexité ne semble pas fiable : en effet, quelques contraintes bien placées peuvent suffire à simplifier le fonctionnement d'un algorithme de *clustering* alors qu'une grande quantité de contraintes mal placées vont au contraire le pénaliser. Nous préférons donc une approche empirique en chronométrant plusieurs exécutions isolées des algorithmes intervenant dans notre implémentation du *clustering* interactif et en évaluant l'importance de leurs différents arguments d'entrée (la taille du jeu de données, le nombre de clusters et le nombre de contraintes annotées, ...). Nous profitons aussi de ces modélisations du temps de calcul pour confirmer le choix de paramétrage réalisé lors de l'étude d'efficience en SECTION 4.2, et ainsi faire un compromis entre l'algorithme le plus efficient et l'algorithme le plus rapide.

4.3.2.a Protocole expérimental

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer aux pseudo-code décrit dans ALGORITHME 4.4.

Nous utilisons le jeu de données **Bank Cards** (v2.0.0) comme référence pour cette expérience : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 1 000 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte**, **carte avalée**, **commande de carte**, ...). Pour plus de détails, consultez l'annexe A.1. Cependant, un seul jeu de données ne nous permet pas d'analyser l'impact du nombre de données sur le temps d'exécution des algorithmes. Pour utiliser facilement plusieurs jeux de données de tailles différentes tout en maîtrisant leur contenu, nous avons donc dupliqué aléatoirement des données issues du jeu de référence en y insérant des fautes de frappes.

⚠️ Attention : Dans le cadre de cette étude, nous faisons l'hypothèse que cette création artificielle de données n'a pas d'impact majeur sur le temps d'exécution des différents algorithmes.

À l'aide de ces données, nous lançons plusieurs exécutions de chaque algorithme de notre implémentation du *clustering* interactif (cf. ANNEXE C.2) avec différentes variations de contexte d'utilisation. Cela comprend les tâches, algorithmes et contextes d'utilisation suivants :

```

Données : jeux de données annotées (vérités terrains) de tailles différentes
Entrées : combinaisons d'algorithmes et de paramètres à tester

1 pour chaque combinaison d'algorithmes et de paramètres à tester faire
2   initialisation (données) : récupérer ou générer le jeu de données ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   si estimation de la tâche de prétraitement alors
5     chronomètre : START ;
6     prétraitement (à étudier) : supprimer le bruit dans les données ;
7     chronomètre : STOP ;
8   sinon si estimation de la tâche de vectorisation alors
9     prétraitement : supprimer le bruit dans les données avec prep.simple ;
10    chronomètre : START ;
11    vectorisation (à étudier) : transformer les données en vecteurs ;
12    chronomètre : STOP ;
13   sinon si estimation de la tâche de clustering alors
14     prétraitement : supprimer le bruit dans les données avec prep.simple ;
15     vectorisation : transformer les données en vecteurs avec vect.tfidf ;
16     échantillonnage initial : sélectionner des contraintes avec samp.rand.full ;
17     simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
18     intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
19     chronomètre : START ;
20     clustering (à étudier) : regrouper les données par similarité ;
21     chronomètre : STOP ;
22   sinon si estimation de la tâche d'échantillonnage alors
23     prétraitement : supprimer le bruit dans les données avec prep.simple ;
24     vectorisation : transformer les données en vecteurs avec vect.tfidf ;
25     échantillonnage initial : sélectionner des contraintes avec samp.rand.full ;
26     simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
27     intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
28     clustering initial : regrouper les données avec clust.kmeans.cop ;
29     chronomètre : START ;
30     échantillonnage (à étudier) : sélectionner de nouvelles contraintes à annoter ;
31     chronomètre : STOP ;
32 pour chaque algorithme à modéliser faire
33   cadrage : définir les facteurs et les interactions intervenant dans la modélisation ;
34   simplification : restreindre la modélisation aux facteurs les plus corrélés ;
35   analyse : modéliser le temps d'exécution avec les facteurs retenus ;
Résultat : modélisation du temps d'exécution des différents algorithmes

```

ALGORITHME 4.4 – Description en pseudo-code du protocole expérimental de l'étude du temps d'exécution des algorithmes du clustering interactif.

1. le **prétraitement** des données...

- avec les algorithmes suivants : **simple** (noté `prep.simple`), **avec lemmatisation** (noté `prep.lemma`) et **avec filtres** (noté `prep.filter`) ;
- avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`) ;

2. la **vectorisation** des données...

- avec les algorithmes suivants : **TF-IDF** (noté `vect.tfidf`) et **SpaCy** (noté `vect.frcorenewsmd`) ;
- avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`) ;
- précédé par un prétraitement **simple** ;

3. le **clustering sous contraintes** des données...

- avec les algorithmes suivants : **KMeans** (modèle *COP* noté `clust.kmeans.cop`), **Hiérarchique** (lien *single* noté `clust.hier.sing` ; lien *complete* noté `clust.hier.comp` ; lien *average* noté `clust.hier.avg` ; lien *ward* noté `clust.hier.ward`) et **Spectral** (modèle *SPEC* noté `clust.spec`) ;
- avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`), le **nombre de contraintes annotés** (variant de 0 à 5 000 par pas de 500, noté `previous_nb_constraints`) et le **nombre de clusters à trouver** (variant de 5 à 50 par pas de 5, noté `algorithm_nb_clusters`) ;
- précédé par un prétraitement **simple** et une vectorisation **TF-IDF** et un échantillonnage initial **purement aléatoire** ;

4. l'**échantillonnage** des contraintes à annoter...

- avec les algorithmes suivants : **purement aléatoire** (noté `samp.random.full`), **pseudo-aléatoire** (noté `samp.random.same`), **même cluster et étant les plus éloignées** (noté `samp.farhest.same`) et **clusters différents et étant les plus proches** (noté `samp.closest.diff`) ;
- avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`), le **nombre de contraintes annotés** (variant de 0 à 5 000 par pas de 500, noté `previous_nb_constraints`), le **nombre de clusters existant** (variant de 10 à 50 par pas de 10, noté `previous_nb_clusters`) et le **nombre de contraintes à sélectionner** (variant de 50 à 250 par pas de 50, noté `algorithm_nb_constraints`) ;
- précédé par un prétraitement **simple**, une vectorisation **TF-IDF**, un *clustering* initial **KMeans** (modèle *COP*) et un échantillonnage initial **purement aléatoire** ;

Il y a donc 8 825 combinaisons d'algorithmes (15 pour le prétraitement, 10 pour la vectorisation, 3 330 pour le *clustering*, 5 550 pour l'échantillonnage), et chaque combinaison est répétée 5 fois pour contrer les aléas statistiques des exécutions. De plus, chaque jeu de données est généré 5 fois pour contrer les aléas statistiques de création, donc il y a 220 625 exécutions d'algorithmes

(375 pour le prétraitement, 250 pour la vectorisation, 82 500 pour le *clustering*, 137 500 pour l'échantillonnage).

Sur la base de ces mesures, nous cherchons à modéliser le temps d'exécution de chaque algorithme en fonction de son contexte d'utilisation (dépendant de ses arguments d'entrée), et les interactions doubles entre paramètres sont envisagées. Afin de réduire la complexité des modélisations, nous ordonnons les interactions de facteurs possibles en fonction de leur corrélation avec le temps mesuré (la corrélation **r** de Pearson (KIRCH, 2008) est utilisée) et nous nous limitons aux variables responsables d'un maximum de la variance des mesures (la méthode d'*Elbow* (THORNDIKE, 1953) est utilisée pour choisir les facteurs pertinents). Sur cette base, nous entraînons un modèle linéaire généralisé (*GLM*, cf. NELDER et WEDDERBURN, 1972) pour représenter le temps d'exécution moyen de l'algorithme : ce modèle sera caractérisé par le coefficient de détermination généralisé R^2 de Cox et Snel (DIAMOND et al., 1990), la log-vraisemblance **llf** (EDWARDS, 1992) et la log-vraisemblance **llf_null** du modèle *null*. Pour finir, nous discutons des valeurs des coefficients obtenus sur l'impact du temps d'exécution.

i Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b. Nous utilisons entre autres les librairies **datetime**²³ et **statsmodels**²⁴ (SEABOLD et PERKTOLD, 2010).

4.3.2.b Résultats obtenus

En ce qui concerne la tâche de **prétraitement**, une première analyse montre que les modélisations des trois implémentations sont similaires (**p-valeur** : > 0.980). Nous faisons donc une seule modélisation.

Pour les algorithmes de prétraitements (**prep.simple**, **prep.lemma** et **prep.filter**), l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur **dataset_size** (**r** : 0.997). Le modèle linéaire généralisé retenu (R^2 : > 0.999, **llf** : -432.43, **llf_null** : -1 353.98) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{prep}) [s] \propto 6.55 \cdot 10^{-3} \cdot \text{dataset_size} \quad (4.2)$$

La FIGURE 4.15 représente cette modélisation du temps de calcul des algorithmes de prétraitements en comparaison avec les mesures réalisées lors de l'expérience.

En ce qui concerne la tâche de **vectorisation**, une première analyse montre que les modélisations des deux implémentations sont différentiables (**p-valeur** : < 10^{-3}). Nous faisons donc une modélisation par algorithme.

Pour les algorithmes de vectorisation **vect.tfidf**, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur **dataset_size** (**r** : 0.977). Le modèle linéaire généralisé retenu (R^2 : > 0.999, **llf** : 259.89, **llf_null** : 70.04) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{vect.tfidf}) [s] \propto 9.16 \cdot 10^{-5} \cdot \text{dataset_size} \quad (4.3)$$

23. **datetime** : <https://pypi.org/project/datetime/>

24. **statsmodels** : <https://pypi.org/project/statsmodels/>

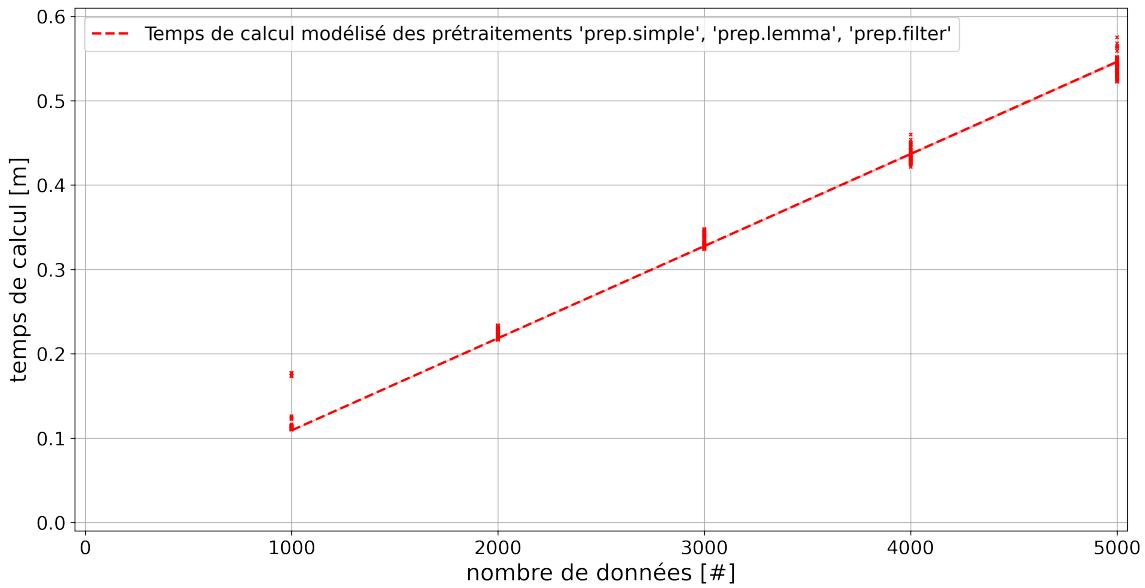


FIGURE 4.15 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de prétraitement en fonction du nombre de données à traiter. Les paramétrages `prep.simple`, `prep.lemma` et `prep.filter` ayant des temps de calcul similaires, leurs modélisations n'ont pas été séparées.

Pour les algorithmes de vectorisation `vect.frcorenewsmd`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size` ($r : 0.983$). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, `llf` : -214.44 , `llf_null` : -399.39) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{vect.frcorenewsmd}) [s] \propto 4.62 \cdot 10^{-3} \cdot \text{dataset_size} \quad (4.4)$$

La FIGURE 4.16 représente ces modélisations de temps de calcul des algorithmes de vectorisation en comparaison avec les mesures réalisées lors de l'expérience.

En ce qui concerne la tâche de **clustering sous contraintes**, une première analyse montre que les modélisations des six implémentations sont différentiables (`p-valeur` : $< 10^{-3}$). Nous faisons donc une modélisation par algorithme.

⚠️ Attention : Plusieurs exécutions des algorithmes de type *hiérarchique* ont été annulées pour les jeux données de tailles supérieures à 4 000 car la durée excédé plusieurs heures. Nous limitons donc l'analyse de `clust.hier.sing`, `clust.hier.comp`, `clust.hier.avg` et `clust.hier.ward` aux tailles de 1 000 à 3 000.

Pour les algorithmes du *clustering* sous contraintes `clust.kmeans.cop`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size` ($r : 0.837$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_clusters` ($r :$

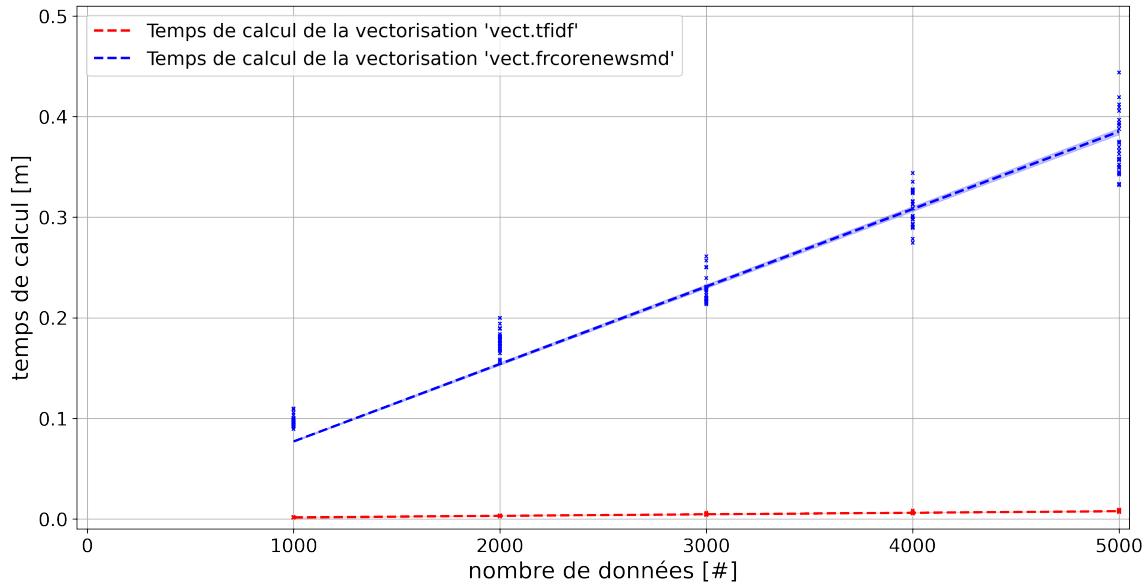


FIGURE 4.16 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de vectorisation en fonction du nombre de données à traiter.

0.545). Le modèle linéaire généralisé retenu ($R^2 : 0.802$, $l1f : -9.37 \cdot 10^4$, $l1f_null : -1.00 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{clust.kmeans.cop}) [s] \propto 1.45 \cdot 10^{-1} \cdot \text{dataset_size} \quad (4.5)$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.sing`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.940$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_clusters` ($r : 0.729$). Le modèle linéaire généralisé retenu ($R^2 : 0.987$, $l1f : -5.54 \cdot 10^4$, $l1f_null : -6.10 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{clust.hier.sing}) [s] \propto 5.00 \cdot 10^{-4} \cdot \text{dataset_size}^2 \quad (4.6)$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.comp`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.938$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_clusters` ($r : 0.736$). Le modèle linéaire généralisé retenu ($R^2 : 0.984$, $l1f : -5.56 \cdot 10^4$, $l1f_null : -6.11 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{clust.hier.comp}) [s] \propto 4.99 \cdot 10^{-4} \cdot \text{dataset_size}^2 \quad (4.7)$$

Pour les algorithmes du *clustering* sous contraintes `clust.hier.avg`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.915$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_clusters` ($r : 0.713$). Le modèle linéaire généralisé retenu ($R^2 : 0.981$, $l1f : -5.90 \cdot 10^4$, $l1f_null : -6.45 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{clust.hier.avg}) [s] \propto 8.51 \cdot 10^{-4} \cdot \text{dataset_size}^2 \quad (4.8)$$

4.3. Évaluation de l'hypothèse sur les coûts

Pour les algorithmes du *clustering* sous contraintes `clust.hier.ward`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.945$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_clusters` ($r : 0.734$). Le modèle linéaire généralisé retenu ($R^2 : 0.989$, `llf` : $-5.57 \cdot 10^4$, `llf_null` : $-6.14 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{clust.hier.ward}) [s] \propto 5.30 \cdot 10^{-4} \cdot \text{dataset_size}^2 \quad (4.9)$$

Pour les algorithmes du *clustering* sous contraintes `clust.spec`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.658$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_clusters` ($r : 0.595$). Le modèle linéaire généralisé retenu ($R^2 : 0.527$, `llf` : $-7.89 \cdot 10^5$, `llf_null` : $-8.27 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{clust.spec}) [s] \propto 8.18 \cdot 10^{-6} \cdot \text{dataset_size}^2 \quad (4.10)$$

La FIGURE 4.17 représente ces modélisations de temps de calcul des algorithmes de *clustering* en comparaison avec les mesures réalisées lors de l'expérience.

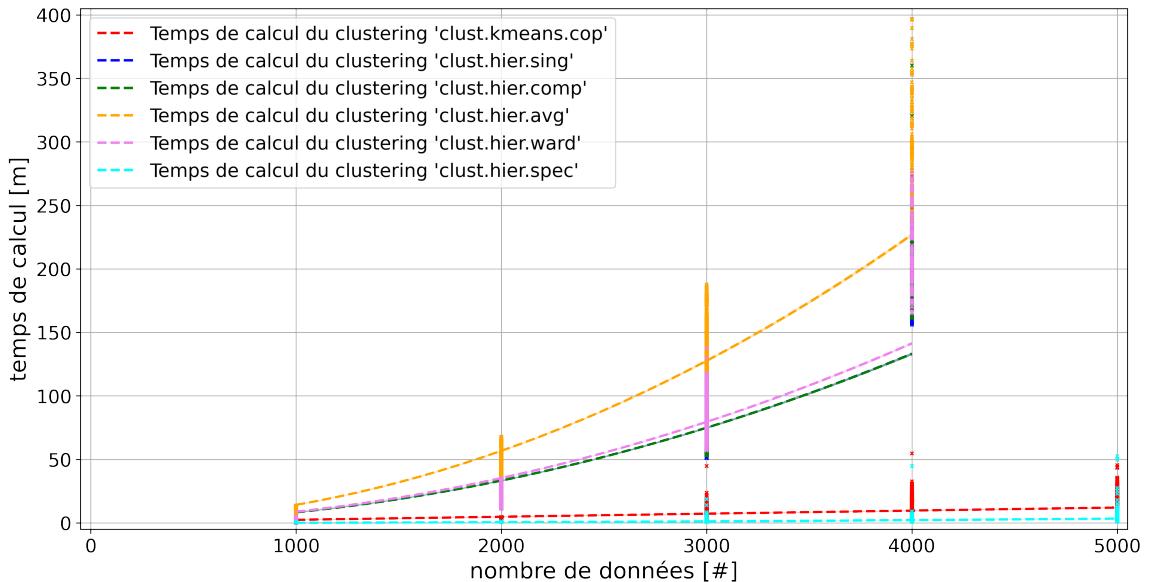


FIGURE 4.17 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de *clustering* en fonction du nombre de données à traiter.

En ce qui concerne la tâche d'**échantillonnage de contraintes**, une première analyse montre que les modélisations des quatre implémentations sont différentiables (p -valeur : $< 10^{-3}$). Nous faisons donc une modélisation par algorithme.

Pour les algorithmes de l'échantillonnage de contraintes `samp.rand.full`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.993$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · previous_nb_clusters` ($r :$

0.791). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, $llf : -4.52 \cdot 10^4$, $llf_null : -1.17 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{samp.rand.full}) [s] \propto 8.20 \cdot 10^{-7} \cdot \text{dataset_size}^2 \quad (4.11)$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.rand.same`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.939$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · algorithm_nb_constraints` ($r : 0.611$). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, $llf : -3.20 \cdot 10^4$, $llf_null : -6.84 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{samp.rand.same}) [s] \propto 1.85 \cdot 10^{-7} \cdot \text{dataset_size}^2 \quad (4.12)$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.farhtest.same`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.981$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · previous_nb_clusters` ($r : 0.700$). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, $llf : -4.56 \cdot 10^4$, $llf_null : -1.02 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{samp.farhtest.same}) [s] \propto 5.19 \cdot 10^{-7} \cdot \text{dataset_size}^2 \quad (4.13)$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.closest.diff`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.995$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · previous_nb_clusters` ($r : 0.815$). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, $llf : -5.96 \cdot 10^4$, $llf_null : -1.36 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$\text{computation_time}(\text{samp.closest.diff}) [s] \propto 1.43 \cdot 10^{-6} \cdot \text{dataset_size}^2 \quad (4.14)$$

La FIGURE 4.18 représente ces modélisations de temps de calcul des algorithmes d'échantillonnage en comparaison avec les mesures réalisées lors de l'expérience.

4.3.2.c Discussion

Dans cette étude, nous avons estimé le temps de calcul des différents algorithmes implémentés afin de confirmer le choix de paramétrage pour une convergence optimal (cf. hypothèse d'efficience en SECTION 4.2). Ces estimations ont été réalisées sur la base de plusieurs exécutions et fonction de divers contextes d'utilisation : nombre de données, nombre de contraintes annotées, nombre de contraintes à sélectionner, nombre de *clusters* existant, nombre de *clusters* à trouver.

En premier lieu, on peut constater que les différentes modélisations dépendent majoritairement de la taille du jeu de données manipulé (`dataset_size` ou `dataset_size2`) avec un score de corrélation r avec le temps mesuré généralement supérieur à 0.9 et des modèles *GLM* avec des coefficients de détermination généralisé R^2 généralement proches de 0.999. Bien que d'autres facteurs peuvent intervenir dans ces estimations (notamment les interactions doubles entre la taille du jeu de données et le nombre de *clusters* ou le nombre de contraintes), ces derniers semblent avoir un impact négligeable sur le temps d'exécution.

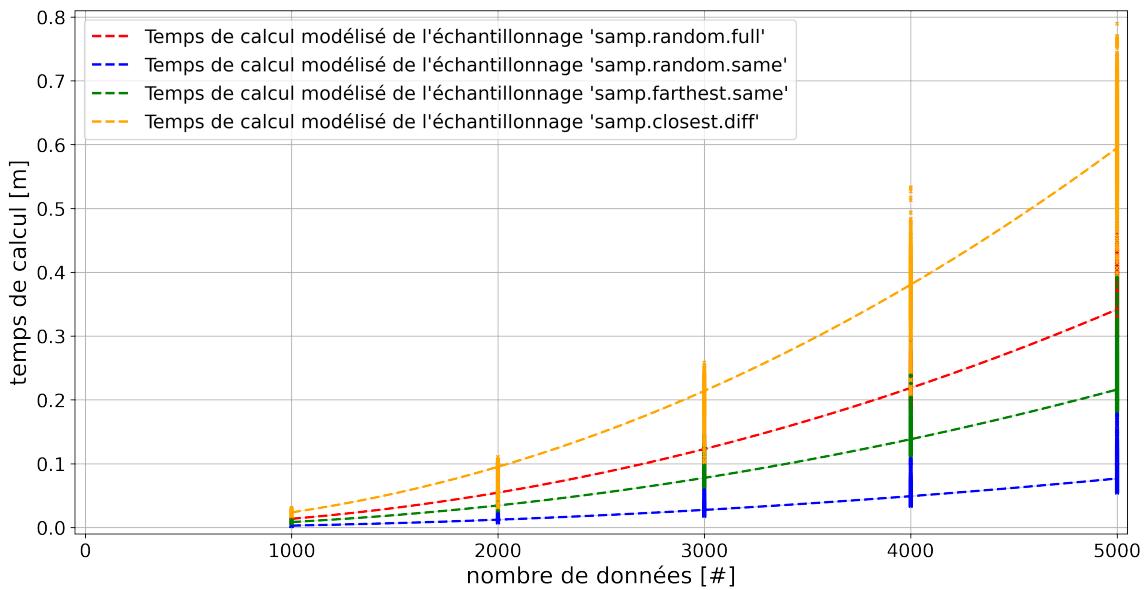


FIGURE 4.18 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche d'échantillonnage de contraintes en fonction du nombre de données à traiter.

Notes de l'auteur : Certains paramétrages de la méthode du *clustering* interactif semblent cependant avoir un temps de calcul décroissant au cours des itérations, mais nous n'avons cependant pas pu montrer de tendances globales significatives. Il est probable que l'ajout de contraintes judicieusement placées permettent à certains algorithmes de *clustering* de s'exécuter plus rapidement, notamment lorsque ceux-ci exploitent les composants connexes du graphe de contraintes (cf. ANNEXE C.2.2). En effet, :

- les *clustering* hiérarchiques s'initialisent autant de *clusters* que de groupes de données liées entre elles par des contraintes **MUST-LINK** : or s'il y a plus de contraintes, alors les composants connexes sont davantage développés, donc il y a moins de *clusters* à initialiser et donc moins d'époques de l'algorithme ;
- le *clustering KMeans* (modèle **COP**) attire auprès d'un barycentre l'ensemble des données liées par un **MUST-LINK** : or s'il y a plus de contraintes, alors il y a des données attirées, donc les noyaux de *clusters* peuvent se stabiliser plus rapidement.

Toutefois, ces suppositions n'ont pas pu être démontrées, et certains contre-exemples tendent à conclure que ces comportements sont très dépendants du jeu de données manipulé et de l'ordre d'ajout des contraintes. Par exemple :

- l'ajout d'un trop grand nombre de contraintes **CANNOT-LINK** peut engendrer un surplus de vérification pour estimer quelles formations de *clusters* sont autorisées sans violer de contraintes ;
- l'algorithme **KMeans** (modèle **COP**) peut osciller autour de plusieurs noyaux de *clusters* instables si les contraintes violent trop la similarité intrinsèque des données.

En ce qui concerne la tâche de *clustering*, on note des différences significatives dans les

temps d'exécution des divers algorithmes implémentés. En effet, l'algorithme **KMeans** (modèle COP) est nettement plus rapide (complexité estimée en $\mathcal{O}(\text{dataset_size})$, nécessitant quelques dizaines de minutes pour 5 000 données) que les implémentations du *clustering* hiérarchique (complexité estimée en $\mathcal{O}(\text{dataset_size}^2)$, nécessitant plusieurs heures dès 3 000 données). Cette différence, visible en FIGURE 4.17, a un réel impact sur l'expérience utilisateur de l'opérateur. En effet, bien qu'il soit théoriquement plus efficient pour atteindre une annotation suffisante (cf. hypothèse d'efficience en SECTION 4.2), l'usage d'un *clustering* hiérarchique imposerait de longs temps d'attente à l'opérateur, interdisant des interactions rapides avec la machines. Or l'intérêt principal de notre méthodologie d'annotation à l'aide du *clustering* interactif repose sur ces interactions homme-machine via l'ajout régulier de contraintes pertinentes (cf. hypothèse d'efficacité en SECTION 4.1). Nous décidons donc d'exclure l'usage des algorithmes de *clustering* hiérarchique au profit du *clustering* **KMeans** (modèle COP).

i Pour information : Dans le cadre du projet étudiant avec l'école Télécom Physique Strasbourg visant à implémenter d'autres algorithmes de *clustering* sous contraintes, un raisonnement similaire a été utilisé pour filtrer les algorithmes. Ainsi, l'implémentation de **KMeans** (modèle MPC) a été exclu (complexité estimée en $\mathcal{O}(\text{dataset_size}^3)$) et l'implémentation de la propagation par affinité écarte la gestion des contraintes CANNOT-LINK pour avoir un temps d'exécution comparable au *clustering* KMeans (modèle COP). L'algorithme DBScan (modèle C-DBScan) est quand à lui un rival possible avec une complexité estimée en $\mathcal{O}(\text{dataset_size})$.

En ce qui concerne les tâches de prétraitements (cf. FIGURE 4.15), de vectorisation (cf. FIGURE 4.16), et d'échantillonnage de contraintes (cf. FIGURE 4.18) ont des complexités presque négligeables en représentant moins de 10% des temps d'exécution du *clustering* (pour 5 000 données : moins de 1 minute pour les trois algorithmes, contre 12.1 minutes pour `clust.kmeans.cop` et 3.5 heures pour `clust.hier.sing`). Nous maintenons donc les paramétrages obtenus pour ces tâches en SECTION 4.2 sans analyses complémentaires, et nous utilisons l'estimation temporelle du *clustering* `clust.kmeans.cop` majorée de 10%.

Points à retenir : Dans l'optique d'atteindre de manière efficiente 90% de `v-measure`²⁵ avec un coût global minimal, nous retenons l'usage du **paramétrage favori** constitué du prétraitement simple (`prep.simple`), de la vectorisation TF-IDF (`vect.tfidf`), du *clustering* KMeans avec modèle COP (`clust.kmeans.cop`) et de l'échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`). On estime le temps d'exécution de ce paramétrage avec l'équation suivante²⁶ :

$$\text{computation_time}(\text{settings.favorite}) [s] \propto 0.17 \cdot \text{dataset_size} \quad (4.15)$$

25. 90% de `v-measure` : cas d'une annotation dite partielle, dont le paramétrage le plus efficient est constitué du prétraitement simple (`prep.simple`), de la vectorisation TF-IDF (`vect.tfidf`), du *clustering* hiérarchique à lien moyen (`clust.hier.avg`) et de l'échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`).

26. Temps du paramétrage favori : environ 2.8 minutes pour 1 000 données; environ 14.2 minutes pour 5 000 données.

4.3.3 Étude du nombre de contraintes nécessaires à la convergence vers une vérité terrain pré-établie en fonction de la taille du jeu de données

Avec les deux précédentes études, nous sommes capable d'estimer le temps nécessaire à un expert pour annoter des contraintes et le temps nécessaire à la machine pour proposer un nouveau *clustering* adapté aux suggestions de l'expert. Pour poursuivre nos études et pouvoir estimer le coût total d'un projet d'annotation, il nous reste à estimer le nombre total de contraintes à devoir renseigner en fonction de la taille du jeu de données.

Pour cela, nous allons simuler la création de cette base d'apprentissage en adaptant le protocole utilisé lors de notre étude d'efficacité (cf. SECTION 4.1.1) : nous employons notre méthode de *clustering* interactif avec notre **paramétrage favori**²⁷ sur des jeux de données de différentes tailles et mesurons le nombre de contraintes nécessaires pour converger vers la vérité terrain.

4.3.3.a Protocole expérimental

Attention : Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble.

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.5.

Données : jeux de données annotées (vérités terrains) de tailles différentes 1 pour chaque jeux de données à tester faire 2 initialisation (données) : récupérer ou générer les données et la vérité terrain ; 3 initialisation (contraintes) : créer une liste vide de contraintes ; 4 prétraitement : supprimer le bruit dans les données avec <code>prep.simple</code> ; 5 vectorisation : transformer les données en vecteurs avec <code>vect.tfidf</code> ; 6 clustering initial : regrouper les données par similarité avec <code>clust.kmeans.cop</code> ; 7 évaluation : estimer l'équivalence entre le <i>clustering</i> et la vérité terrain ; 8 répéter 9 échantillonnage : sélectionner des contraintes avec <code>samp.closest.diff</code> ; 10 simulation d'annotation : déterminer les contraintes avec la vérité terrain ; 11 intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ; 12 clustering : regrouper les données par similarité avec <code>clust.kmeans.cop</code> ; 13 évaluation : estimer l'équivalence entre le <i>clustering</i> et la vérité terrain ; 14 jusqu'à annotation de toutes les contraintes possibles ; 15 analyse : entraîner un modèle linéaire généralisé du nombre de contraintes nécessaires ; Résultat : modélisation du nombre de contraintes nécessaires pour un jeu de données

ALGORITHME 4.5 – Description en pseudo-code du protocole expérimental de l'étude du nombre de contraintes nécessaires pour converger vers une vérité terrain pré-établie avec notre paramétrage favori du clustering interactif.

Nous utilisons deux vérités terrains comme références pour cette expérience :

27. Paramétrage favori (atteindre 90% de v-measure avec un coût minimal) : prétraitement simple (`prep.simple`), vectorisation TF-IDF (`vect.tfidf`), *clustering KMeans* avec modèle COP (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des clusters différents (`samp1.closest.diff`)

- le jeu de données **Bank Cards** (v2.0.0) : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 1 000 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte, carte avalée, commande de carte**, ...). Pour plus de détails, consultez l'annexe A.1 ;
- le jeu de données **MLSUM FR Train Subset** (v1.0.0-schild) : ce dernier concerne les titres d'articles de journaux issus des catégories de publication les plus communes. Il est composé de 744 titres d'articles rédigés et répartis en 14 classes (**économie, sport, ...**). Pour plus de détails, consultez l'annexe A.2 ;

Cependant, deux jeux de données ne nous permettent pas d'analyser l'impact du nombre de données sur le nombre de contraintes nécessaires pour converger vers une vérité terrain. Pour utiliser facilement plusieurs jeux de données de tailles différentes tout en maîtrisant leur contenu, nous avons donc dupliqué aléatoirement des données issues de ces jeux de référence en y insérant des fautes de frappes. La taille des jeux de données générés, notée **dataset_size**, varie entre 1 000 à 5 000 par pas de 250, et chaque taille de jeu est générée 3 fois pour contrer les aléas statistiques de création. Il y a donc 51 variations de chaque jeu de références, soit 102 jeux utilisés de tailles différentes.

⚠️ Attention : Dans le cadre de cette étude, nous faisons l'hypothèse que cette création artificielle de données n'a pas d'impact majeur sur le nombre de contraintes nécessaires pour converger vers une vérité terrain.

Sur chacun de ces jeux générés, une tentative complète²⁸ de la méthode du *clustering* interactif en utilisant notre paramétrage favori est exécuté, et chaque tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions. Il y a donc 510 tentatives de *clustering* interactif réalisées.

Pour chacune de ces tentatives, nous nous intéressons au nombre de contraintes nécessaires pour atteindre le seuil d'annotation partielle (caractérisé par 90% de **v-measure** entre la vérité terrain et la segmentation des données obtenue), et nous entraînons un modèle linéaire généralisé (*GLM*) pour modéliser le nombre de contraintes requis en fonction de la taille du jeu de données (noté **dataset_size**). Ce modèle sera caractérisé par le coefficient de détermination généralisé **R²** de *Cox et Snel* (DIAMOND et al., 1990), la log-vraisemblance **llf** (EDWARDS, 1992) et la log-vraisemblance **llf_null** du modèle *null*. Pour finir, nous discutons des valeurs des coefficients obtenus sur l'impact du nombre d'itérations de la méthode à prévoir.

ℹ️ Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b. Nous utilisons entre autres la librairie **statsmodels**²⁹ (SEABOLD et PERKTOLD, 2010).

4.3.3.b Résultats obtenus

Le modèle linéaire généralisé entraîné sur les mesures du nombre de contraintes requis pour atteindre 90% de **v-measure** ($R^2 > 0.999$, **llf** : -4 327.6, **llf_null** : -4 942.9) nous permet

28. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

29. **statsmodels** : <https://pypi.org/project/statsmodels/>

de déduire l'équation suivante :

$$\text{constraints_needed}(\text{settings.favorite}) [\#] \propto 3.15 \cdot \text{dataset_size} \quad (4.16)$$

La FIGURE 4.19 représente cette modélisation.

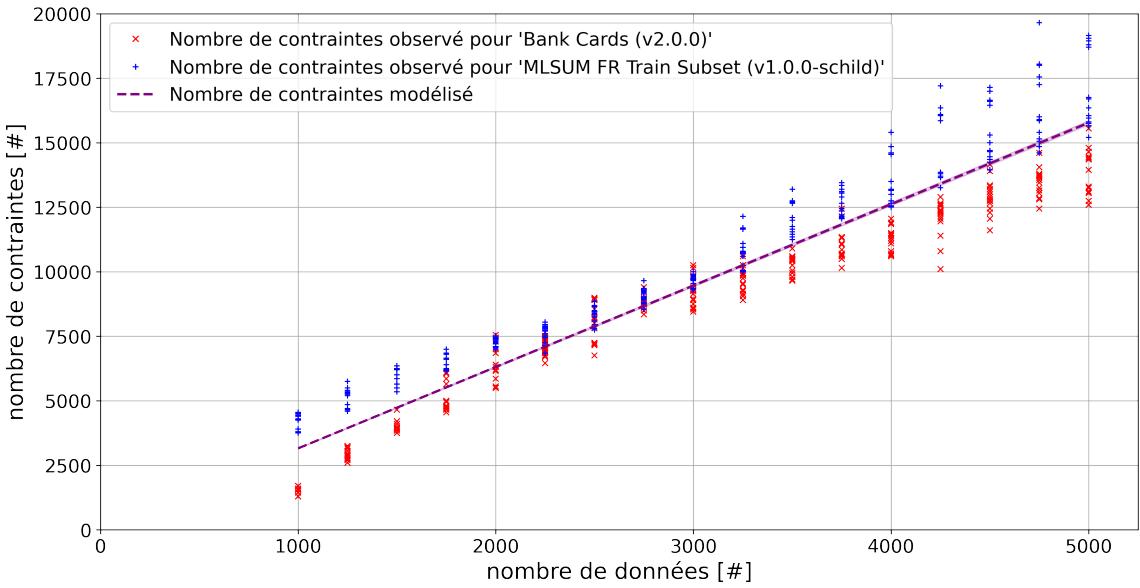


FIGURE 4.19 – Estimation du nombre moyen de contraintes nécessaire à notre paramétrage *favori* du clustering interactif afin d'obtenir une annotation partielle (atteindre une *v-measure* de 90%) en fonction de la taille du jeu de données à modéliser.

Notes de l'auteur : On peut considérer les points de références suivants :

- le nombre de contraintes possibles (avec doublons) est de dataset_size^2 (*caractériser chaque couple de données présent dans la matrice d'adjacence*) ;
- le nombre de contraintes possibles (sans doublons) est de $\frac{1}{2} \cdot (\text{dataset_size}^2 - \text{dataset_size})$ (*considérer la symétrie des contraintes, donc seul le triangle supérieur de la matrice d'adjacence a besoin d'être renseigné*) ;
- le nombre minimal de contraintes à annoter pour être exhaustif sur une partition en k clusters $\{K_1, K_2, \dots, K_k\}$ est estimé à $\sum_{1 \leq i \leq k} (\|K_i\| - 1) + \sum_{1 \leq i \leq k} (k - i)$ (*il faut d'abord considérer les chemins minimaux pour parcourir les composants connexes avec des contraintes MUST-LINK, correspondant à $\|K_i\| - 1$ contraintes MUST-LINK pour chaque partition $\|K_i\|$, puis ajouter le nombre minimal des contraintes CANNOT-LINK pour distinguer chacun de ses composants connexes en cluster, correspondant au nombre de arrangements sans répétition de deux partitions*).

La FIGURE 4.20 illustre ces propos sur un jeu d'exemple comportant 10 points de données réparties en 3 classes, et met en avant l'explosion du nombre de contraintes possibles même sur un petit jeu de données (cf. 4.20 (2)).

Avec ces références, le nombre de contraintes est borné approximativement entre 1 035 et 499 500 pour un jeu de 1 000 données équilibré en 10 classes, et entre 6 175 et 12 497 500 pour un jeu de 5 000 données équilibré en 50 classes.

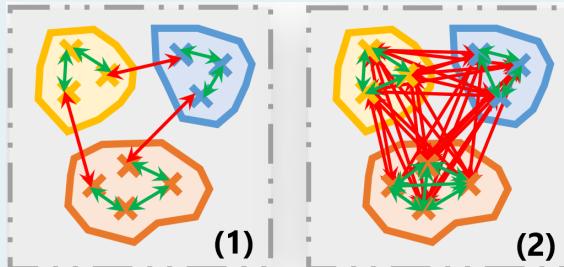


FIGURE 4.20 – Exemple de caractérisation exhaustive d'un jeu de données (10 données, 3 classes) en ajoutant un nombre minimal de contraintes (cf. (1)) ou en ajoutant toutes les contraintes possibles (cf. (2)).

4.3.3.c Discussion

L'objectif de cette étude était de déterminer le nombre moyen de contraintes à devoir annoter pour modéliser un jeu de données avec un accord 90% de **v-measure** avec la vérité terrain utilisée. Cette estimation, dépendant de la taille du jeu de données manipulé, est représentée par l'**EQUATION 4.16**.

On peut constater que la relation entre la taille du jeu de données et le nombre de contraintes à annoter est linéaire (pente de 3.15) : doubler la taille d'un jeu de données doublera donc la charge de travail incomptant à l'expert métier. À première vue, une telle estimation représente une lourde charge d'annotation : **pour un jeu de 5 000 données, il faut caractériser 15 750 contraintes, ce qui correspond environ à 34 heures d'annotation** d'après l'**EQUATION 4.1** ! Néanmoins, comme le nombre de contraintes possibles évolue en $\mathcal{O}(\text{dataset_size})$, cette estimation aurait pu être bien pire et représenter 12 497 500 contraintes (cf. FIGURE 4.20 dans notre précédente précédente). Mieux encore, le nombre théorique minimal moyen de contraintes à annoter pour 5 000 données n'est que 2.55 fois plus faible que notre estimation (6 175 vs 15 750, cf. notre précédente précédente), alors que cette borne minimale nécessite un échantillonnage "parfait" permettant d'identifier le chemin minimal parcourant les clusters. Nous pouvons donc relativiser l'estimation faite avec l'**EQUATION 4.16** et en conclure que notre méthode un nombre de contraintes raisonnable à annoter.

Bien évidemment, une telle estimation est sensible au jeu de données utilisé comme référence (cf. FIGURE 4.19). Ici, la différence de pente mesurée est de 0.25 (**p-valeur** : > 0.999), soit un écart moyen d'environ 8% par rapport à la modélisation moyenne. Toutefois, comme l'impact semble limité, nous maintenons la modélisation moyenne représentée par l'**EQUATION 4.16** pour la suite de nos estimations de coûts.

Notes de l'auteur : Il n'y a pas davantage de matière à discussion pour cette étude, car le principal résultat (l'**EQUATION 4.16**) est un résultat temporaire nécessaire à

l'estimation du coût global d'un projet utilisant une méthodologie de *clustering* interactif.

4.3.4 Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts

Résumons l'ensemble des modélisations réalisées lors des précédentes études (cf. sections 4.3.1, 4.3.2 et 4.3.3) afin d'estimer le coût total d'un projet d'annotation employant une méthodologie basée sur le *clustering* interactif et utilisant notre **paramétrage favori**³⁰. Dans les notations, `dataset_size` représente la taille du jeu de données à modéliser, et `batch_size` représente le nombre de contraintes que l'expert annote à chaque itération.

4.3.4.a Synthèse des résultats

Tout d'abord, nous pouvons estimer le **temps moyen d'une itération de la méthode**, comprenant d'une part les temps d'exécution des algorithmes (*prétraitement, vectorisation, clustering, échantillonnage*) et d'autre part le temps d'annotation d'un lot de contraintes, grâce aux équations suivantes :

$$\begin{cases} \text{computation_time [s]} & \propto 0.17 \cdot \text{dataset_size} \\ \text{annotation_time [s]} & \propto 7.8 \cdot \text{batch_size} \\ \text{iteration_time(sequential) [s]} & \propto \text{computation_time} + \text{annotation_time} \end{cases} \quad (4.17)$$

Ensuite, nous sommes en mesure d'anticiper le **nombre moyen de contraintes à annoter** pour modéliser le jeu de données avec un seuil de 90% de **v-measure**, et donc de déduire le nombre d'itérations nécessaire de la méthode, grâce aux équations suivantes :

$$\begin{cases} \text{constraints_needed [\#]} & \propto 3.15 \cdot \text{dataset_size} \\ \text{iterations_needed [\#]} & \propto \frac{\text{nb_constraints}}{\text{batch_size}} \end{cases} \quad (4.18)$$

Enfin, il suffit de combiner EQUATION 4.17 et EQUATION 4.18 pour estimer le temps total nécessaire à un projet d'annotation utilisant le *clustering* interactif (c'est-à-dire en enchaînant successivement des étapes d'échantillonnage, d'annotation et de *clustering*, cf. FIGURE 4.21 (1)) pour converger vers 90% de **v-measure** :

$$\text{total_time(sequential) [s]} \propto \text{iteration_time} \cdot \text{iterations_needed} \quad (4.19)$$

Ces estimations globales sont représentées sur la FIGURE 4.22 en fonction de plusieurs taille de jeu de données et plusieurs tailles de lots d'annotation.

4.3.4.b Discussion du coût total

L'objectif de cette section consistait à déterminer les coûts relatifs à la tâche d'annotation de contraintes par un expert métier et au temps d'exécution des algorithmes intervenant dans notre implémentation du *clustering* interactif. Pour cela, nous avons chronométré des annotateurs en

30. Paramétrage favori (atteindre 90% de **v-measure** avec un coût minimal) : prétraitement simple (`prep.simple`), vectorisation TF-IDF (`vect.tfidf`), *clustering KMeans* avec modèle COP (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`).

situation réelle (cf. SECTION 4.3.1), estimé le temps de calcul de chaque algorithme implémenté (cf. SECTION 4.3.2) et trouvé le moyen de prédire le nombre de contraintes à annoter sur un jeu de données (cf. SECTION 4.3.3). Nous avons pu montré qu'une annotation de contraintes est plus rapide qu'une annotation par label et nous conseillons, d'après notre analyse empirique, des sessions d'annotation de moins de 150 contraintes pour ne pas épuiser l'annotateur. Sur le paramétrage de la méthode, nous avons rejeté l'usage d'algorithmes de *clustering* de type hiérarchiques à cause de leur lenteur, au profit du KMeans avec modèle COP (`clust.kmeans.cop`). Notre paramétrage favori, permettant d'atteindre 90% de **v-measure** avec un coût minimal, est ainsi constitué d'un prétraitement simple (`prep.simple`), d'une vectorisation TF-IDF (`vect.tfidf`), d'un *clustering* KMeans avec modèle COP (`clust.kmeans.cop`) et d'un échantillonage des données les plus proches dans des clusters différents (`sampl.closest.diff`). Enfin, pour atteindre cet objectif de **v-measure**, le nombre moyen de contraintes à annoter avec notre méthodologie semble rester linéairement proportionnel à la taille du jeu de données à modéliser, ce qui est encourageant au regard de la combinatoire de contraintes possibles.

La mise en commun de ces résultats se retrouvent dans EQUATION 4.17, EQUATION 4.18 et EQUATION 4.19. Comme le nombre de données à annoter est inversement proportionnel au nombre d'itération à réaliser, nous avons le dilemme suivant : soit nous annotons de petits lots (50 contraintes) pour rapidement intégrer les contraintes et permettre à l'annotateur de se reposer régulièrement, mais ce dernier va en contrepartie devoir attendre plus souvent la fin des exécutions du *clustering*; soit nous annotons des lots plus conséquents (150 contraintes) pour diminuer le nombre d'itérations et exécuter moins de *clustering*, mais cela risque d'épuiser l'opérateur avec des grosses charges d'annotation. Dans les deux cas, le **coût total semble élevé** : pour un jeu de 5 000 points de données, et avec des tailles d'échantillons compris entre 50 et 150 contraintes, il faut entre 59 et 110 heures de travail (*34 heures d'annotations et entre 25 et 76 heures d'attente de la fin d'exécution d'algorithmes suivant la taille des lots*). En considérant une journée de travail de 7 heures, cela représente une charge contenue entre 8.4 et 15.7 jours pour avoir un jeu de donnée fiable à 90% de **v-measure**. Pour finir, nous ajoutons 1 jour pour combler l'écart théorique de 10% de **v-measure** en corrigeant manuellement le résultat obtenu, et 1 jour supplémentaire pour interpréter et nommer chaque *cluster* (voir. ALGORITHME 3.1, ligne 13). Au final, pour un ensemble de 5 000 données, il faut donc entre 8.4 (+2) et 15.7 (+2) jours de travail à un expert métier pour obtenir une base d'apprentissage avec une méthodologie basée sur le *clustering* interactif.

Pour critiquer l'approximation que nous avons faite lors de cette section, nous essayons de la comparer le temps nécessaire qu'il aurait fallu à un projet d'annotation traditionnelle, comme décrit en SECTION 2.2.1 (cycle MATTER), et notre proposition de cycle d'annotation avec le *clustering* interactif, visible en FIGURE 4.21 (1). En nous basant sur un ensemble de 5 000 données à annoter, nous estimons qu'il faut : 1 jour de travail pour définir une représentation des données en modèle de classification ; 4 jours de travail pour annoter 5 000 données (à raison de 20 secondes par données, estimation haute inspirée de PRADHAN et al., 2007 où la "*désambiguïsation du sens des mots*", présentée comme une tâche de classification, demande 17.5 secondes par données) ; 2 jours de marge d'erreur pour changer de modèle de représentation s'il ne semble pas adapté aux données en cours d'annotation. Au total, nous estimons donc qu'il faut 5 (+2) jours de travail³¹ à un expert métier pour obtenir une base d'apprentissage avec une méthodologie

31. 5 (+2) jours de travail : Notons que cette estimation n'est bien entendu pas généralisable. En effet, le temps nécessaire aux phases de modélisation et de revues peuvent fortement augmenter si le cas d'usage est plus complexe. Par exemple, la classification de questions en une centaine d'intentions peut prendre plusieurs jours voir quelques semaines d'étude alors que la classification de sentiments sur trois niveaux (positif, négatif, neutre)

traditionnelle. En l'état, nous pouvons donc conclure que le *clustering* interactif que nous proposons est en moyenne deux fois plus coûteux qu'une annotation manuelle classique (8.4 (+2) à 15.7 (+2) jours vs 5 (+2) jours), ce qui peut être un frein à son utilisation en situation réelle.

Afin d'accélérer la phase d'annotation et de diminuer le nombre d'itérations, il est bien entendu possible d'augmenter la taille des échantillons de contraintes à annoter ou d'ajouter plusieurs opérateurs. Cependant, une telle solution ne permet toujours pas d'être compétitif avec l'annotation traditionnelle si cette dernière dispose aussi de plusieurs opérateurs (*avec 2 annotateurs : de 6.5 (+2) à 13.3 (+2) jours vs 3 (+2) jours ; avec 4 annotateurs : de 4.8 (+2) à 12.1 (+2) jours vs 2 (+2) jours*). Une autre piste, plus prometteuse, consiste plutôt à adapter notre méthode pour exploiter les temps d'attente lors de l'exécution d'un *clustering*.

4.3.5 Ouverture vers une annotation en parallèle du *clustering*

Notre méthodologie d'annotation basée sur le *clustering* interactif est pénalisée par la séquentialité des actions à réaliser. En effet, l'annotateur doit attendre la fin d'un de l'exécution des algorithmes de *clustering* et d'échantillonnage avant de pouvoir travailler. D'après nos précédentes estimations sur un jeu de 5 000 données, l'opérateur doit attendre entre 25 et 76 heures en fonction de la taille de lots d'annotation choisie, et une idée à explorer consiste à optimiser ce temps d'attente.

L'amélioration envisagée consiste à **réaliser l'annotation de contraintes en parallèle de l'exécution du *clustering***, comme représenté dans la FIGURE 4.21 (2). Dans cette version, l'échantillonnage se base toujours sur le résultat du *clustering* de l'itération précédente, mais le *clustering* intègre les contraintes annotées avec un décalage d'une itération. Un tel changement permet de limiter le temps d'attente de l'opérateur et d'optimiser l'enchaînement des algorithmes.

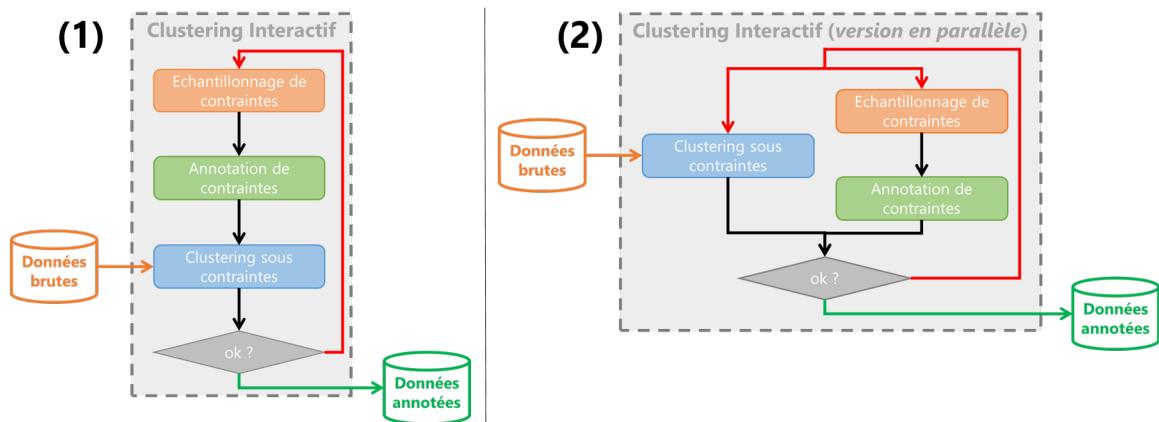


FIGURE 4.21 – Schéma comparatif des architectures du clustering interactif : (1) représente la version séquentielle initialement présentée en CHAPITRE 3 où le clustering s'adapte avec les annotation de l'itération en cours ; (2) représente l'évolution en mode parallèle où le clustering s'adapte avec les annotations de l'itération précédente (décalage d'une itération).

Avec cette version, le temps nécessaire à une itération correspond à la durée la plus longue entre le temps d'annotation et le temps de calcul des algorithmes. Nous pouvons donc adapter est presque triviale.

l'EQUATION 4.17 par l'équation suivante :

$$\begin{cases} \text{computation_time [s]} & \propto 0.17 \cdot \text{dataset_size} \\ \text{annotation_time [s]} & \propto 7.8 \cdot \text{batch_size} \\ \text{iteration_time(parallel) [s]} & \propto \max(\text{computation_time}, \text{annotation_time}) \end{cases} \quad (4.20)$$

Ensuite, afin de limiter les pertes de temps (humain et machine), nous pouvons choisir une taille de lot d'annotation rendant ces deux durées équivalentes. Nous déduisons donc les changements suivants dans l'EQUATION 4.18 :

$$\begin{cases} \text{optimal_batch_size [\#]} & \propto \frac{\text{computation_time}}{7.8} \propto 0.0218 \cdot \text{dataset_size} \\ \text{iterations_needed [\#]} & \propto \frac{\text{nb_constraints}}{\text{optimal_batch_size}} \propto 144.5 \end{cases} \quad (4.21)$$

Enfin, il suffit de combiner EQUATION 4.20 et EQUATION 4.21 pour estimer le temps total nécessaire à un projet d'annotation pour converger vers 90% de **v-measure** utilisant la version parallèle du *clustering* interactif :

$$\begin{cases} \text{total_time(parallel) [s]} & \propto \text{iteration_time} \cdot \text{iterations_needed} \\ \text{total_time(parallel) [s]} & \propto 24.6 \cdot \text{dataset_size} \end{cases} \quad (4.22)$$

Ces estimations mises à jour sont représentées sur la FIGURE 4.22 en fonction de plusieurs taille de jeu de données, et permettent de faire la comparaison avec la version séquentielle initialement présentée.

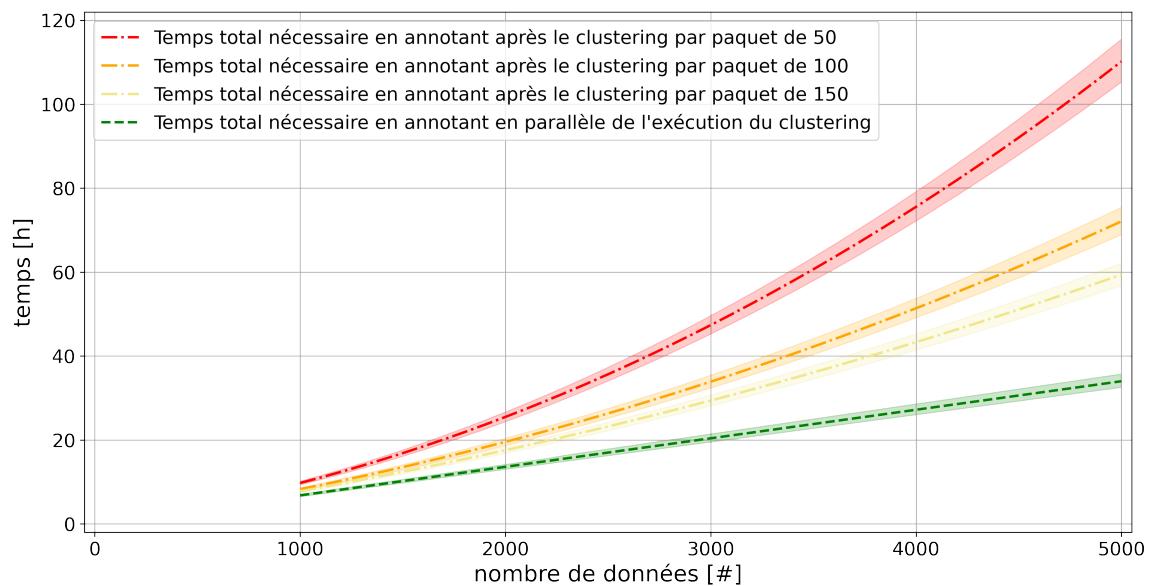


FIGURE 4.22 – Estimation du temps total nécessaire (en heures) pour modéliser un jeu de données avec notre paramétrage favori du clustering interactif afin d'obtenir une annotation partielle (atteindre une *v-measure* de 90%), en fonction de plusieurs taille de jeu de données, plusieurs tailles de lots d'annotation, et mettant en opposition l'approche séquentielle (annotation puis le clustering) et l'approche parallèle (annotation pendant le clustering).

Nous pouvons déjà remarquer que le coût d'annotation du projet devient linéaire en nombre de données (pente de 24.6 secondes) et nécessite un nombre fixe de 145 itérations. En reprenant une base de 5 000 données et une marge de 2 jours pour corriger et nommer les *clusters*³², cela représente 4.8 (+2) jours de travail, un estimation équivalente à une annotation traditionnelle qui nécessite 5 (+2) jours de travail d'après nos approximations. De plus, si nous ajoutons des plusieurs opérateurs, cette version parallèle reste compétitive (*avec 2 annotateurs* : 2.5 (+2) *jours vs 3 (+2) jours*; *avec 4 annotateurs* : 1.3 *jours vs 2 (+2) jours*). Cette découverte est très encourageante, car cela confirme qu'une méthodologie basée sur notre implémentation du *clustering* interactif permet d'obtenir une base d'apprentissage avec un coût temporel équivalent à un projet traditionnel utilisant une annotation par label. Cette méthode est d'autant plus intéressante qu'elle fait intervenir un mécanisme d'annotation rapide et intuitif pour un expert métier.

 **Points à retenir :** Au cours de cette étude de coûts, nous avons pu déduire que :

- ✓ L'annotation d'une contrainte nécessite en moyenne 8 secondes : cette tâche est rapide et intuitive (cf. SECTION 4.3.1) ;
- ✓ Notre paramétrage favori, permettant d'atteindre 90% de **v-measure** avec un coût minimal, est constitué du prétraitement simple (`prep.simple`), de la vectorisation TF-IDF (`vect.tfidf`), du *clustering KMeans* avec modèle COP (`clust.kmeans.cop`) et de l'échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`). Ce paramétrage a un coût moyen de $0.17 \cdot \text{dataset_size}$ secondes (cf. SECTION 4.3.2) ;
- ✓ Une adaptation optimale de notre méthodologie consiste à paralléliser l'exécution du *clustering* et l'annotation de contraintes afin de limiter les temps d'attente inutiles. Une telle méthode a un coût moyen de $24.6 \cdot \text{dataset_size}$ pour atteindre 90% de **v-measure**, auquel on ajoute 2 jours de travail pour raffiner les clusters et les nommer. Ce temps est compétitif à une annotation traditionnelle (cf. SECTION 4.3.4) ;
- ✓ Cette étude met en avant l'intérêt des interactions homme-machine : (1) l'expert métier se recentre sur son domaine de compétence avec une caractérisation proche de ses connaissances ("les données sont-elles similaires ?") et (2) la machine optimise l'intervention de l'expert pour que ce dernier soit toujours pertinent dans ses contributions.

Dans les sections suivantes, nous allons nous intéresser à l'analyse des résultats de cette méthode. En effet, en situation réelle, nous n'avons pas accès à la vérité terrain car elle est justement en cours de construction. Il nous est donc impossible d'estimer notre seuil de **v-measure**, et donc incapable de s'arrêter à 90% de **v-measure**. Nous nous intéressons donc à l'estimation de la valeur métier d'un résultat de *clustering* (cf. hypothèse de pertinence en SECTION 4.4) et à la définition de cas d'arrêt agnostique d'une vérité terrain (cf. hypothèse de rentabilité en SECTION 4.5).

32. Nommer les *clusters* : voir. ALGORITHME 3.1, ligne 13.

4.4 Évaluation de l'hypothèse de pertinence

Jusqu'à présent, nous avons analysé la performance et l'évolution des résultats de notre implémentation du *clustering* interactif en calculant la similarité (en **v-measure**) avec une vérité terrain. Cependant, une telle référence n'est pas accessible en situation réelle car l'objectif de notre méthode est précisément de la construire cette vérité terrain. Nous devons donc nous intéresser à d'autres moyens pour estimer la pertinence et l'exploitabilité des bases d'apprentissages obtenues. Ainsi, nous aimerions vérifier l'hypothèse suivante :

💡 Hypothèse de pertinence 💡

« Au cours d'une méthodologie d'annotation basée sur le *clustering* interactif, il est possible à un expert métier d'évaluer rapidement la pertinence de la base d'apprentissage en construction sans utiliser de vérité terrain. »

La FIGURE 4.23 illustre cette hypothèse et l'espoir de pouvoir caractériser la qualité de la base d'apprentissage en cours de construction en fonction d'une valeur métier exprimée par un expert.



FIGURE 4.23 – Illustration des études réalisées sur le *clustering* interactif (étape 4/6) en schématisant l'évolution de la pertinence (valeur métier évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine).

Afin de vérifier cette hypothèse, nous explorons trois approches :

- une **validation par un expert** du partitionnement des données obtenus, en parcourant manuellement le contenu des *clusters* et en donnant un avis sur l'exploitabilité de ces derniers (cf. SECTION 4.4.1) ;
- une analyse des **patterns linguistiques saillants** dans la base d'apprentissage à l'aide

- d'une stratégie de sélection des composantes principales d'un modèle (cf. SECTION 4.4.2),
- et une approche utilisant un **résumé automatique de thématique** par un large modèle de langage (LLM), permettant de décrire succinctement le contenu des *clusters* en une phrase (cf. SECTION 4.4.3).

4.4.1 Étude d'une validation manuelle et non assistée de la valeur métier d'une base d'apprentissage par un expert

Afin d'estimer la pertinence d'un résultat de *clustering*, notre première intuition consiste à demander simplement l'avis d'un expert sur la base d'apprentissage en cours de construction. En lui posant certaines questions, nous espérons obtenir une description qualitative de chaque *cluster* et ainsi déduire quand le résultat du *clustering* interactif devient exploitable pour définir et entraîner un modèle de classification.

4.4.1.a Protocole expérimental

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.6.

Données : jeu de données annotées (vérité terrain)

1 **pour chaque** *jeux de données à tester faire*

2 **initialisation (données)** : récupérer les données et la vérité terrain ;

3 **initialisation (contraintes)** : créer une liste vide de contraintes ;

4 **prétraitement** : supprimer le bruit dans les données avec **prep.simple** ;

5 **vectorisation** : transformer les données en vecteurs avec **vect.tfidf** ;

6 **clustering initial** : regrouper les données par similarité avec **clust.kmeans.cop** ;

7 **évaluation manuelle** : juger de l'exploitabilité de chaque *cluster* ;

8 **labellisation manuelle** : nommer chaque *cluster* exploitable ;

9 **répéter**

10 **échantillonnage** : sélectionner des contraintes avec **samp.closest.diff** ;

11 **simulation d'annotation** : déterminer les contraintes avec la vérité terrain ;

12 **intégration** : ajouter les nouvelles contraintes au gestionnaire de contraintes ;

13 **clustering** : regrouper les données par similarité avec **clust.kmeans.cop** ;

14 **évaluation manuelle** : juger de l'exploitabilité de chaque *cluster* ;

15 **labellisation manuelle** : nommer chaque *cluster* exploitable ;

16 **jusqu'à annotation de toutes les contraintes possibles;**

17 **analyse** : afficher l'évolution de l'exploitabilité de chaque itération de *clustering* ;

Résultat : discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

ALGORITHME 4.6 – Description en pseudo-code du protocole expérimental de l'étude de validation manuelle non assistée de la valeur métier d'une base d'apprentissage.

Nous utilisons comme vérité terrain le jeu de données **Bank Cards** (v1.0.0) : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte**, **carte avalée**, **commande de carte**, ...). Pour plus de détails, consultez l'annexe A.1.

Sur ce jeu de données, nous exécutons une tentative complète³³ de la méthode du *clustering* interactif en utilisant notre paramétrage favori³⁴ (voir SECTION 4.3), et cette tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions.

Au cours des itérations, un expert qualifie chaque *cluster* en donnant son avis sur sa valeur métier. Afin d'encadrer ses réponses, nous lui demandons d'analyser trois aspects :

- est-ce que le *cluster* a une thématique principale bien définie ? (*en effet, comment interpréter un cluster sans définition claire ?*)
- est-ce que le *cluster* est constitué par un nombre suffisant de données ? (*en effet, comment entraîner un modèle de classification sans données ?*)
- est-ce que le *cluster* n'est pas trop bruité ? (*en effet, comment avoir de bonnes performances si la base d'apprentissage n'est pas fiable ?*)

L'avis exprimé par l'expert métier est alors classé en trois niveaux :

- **exploitable** : le *cluster* possède (1) une thématique bien définie, (2) un nombre de données suffisant pour entraîner un modèle de classification et (3) peu de bruit ; ce *cluster* peut donc être exploité en l'état ou avec peu de modifications manuelles ;
- **partiellement exploitable** : soit le *cluster* est composé de plusieurs de thématiques (*deux ou trois*), soit il ne comporte pas assez de données (*moins d'une vingtaine*), soit il est bruité (*au moins un quart de bruit*) ; ce *cluster* donne une première base pour créer une classe, mais un travail manuel est nécessaire (*ajout de données, tri du bruit, ...*) ;
- **non exploitable** : soit le *cluster* ne contient pas ou contient trop de thématique, soit c'est un *cluster* singleton ou un *cluster* de données trop hétérogènes, soit ce *cluster* est complètement bruité ; dans tous les cas, il n'est absolument pas exploitable sans un gros travail manuel.

Pour limiter la charge de travail de l'opérateur, nous ne demandons l'expertise que toutes les 5 itérations d'une tentative.



Attention : Par manque de personnes aptes à qualifier le jeu de données utilisé, les annotations de cette étude ont été réalisées par un seul opérateur (*moi-même, ayant participé à la création de ce jeu de données*). Nous supposons que cette contrainte n'est pas pénalisante pour l'analyse : en effet, dans une situation réelle, cet opérateur serait responsable des choix à entreprendre pour concevoir le jeu de données, il est donc le mieux placer pour juger la pertinence d'un *clustering* par rapport à sa propre modélisation du problème. Les problèmes d'accords inter-annotateurs seront plutôt discuté en SECTION 4.6. Nous réalisons toutefois 5 tentatives différentes de la méthode pour limiter les biais intra-individuels.

33. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

34. Paramétrage favori (atteindre 90% de `v-measure` avec un coût minimal) : prétraitement simple (`prep.simple`), vectorisation TF-IDF (`vect.tfidf`), *clustering KMeans* avec modèle COP (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`).

i Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.4.1.b Résultats obtenus

La FIGURE 4.24 met en avant l'évolution de la pertinence moyenne estimée par l'opérateur sur la base des contenu des *clusters*. Nous allons nous intéresser à trois phases s'y distinguant.

À l'initialisation (itération 0), la majeure partie des *clusters* sont inexploitables (environ 60%) et seul 35% d'entre eux semblent exploitables. Dans le top 3 des classes facilement identifiables à ce stade, nous retrouvons *gestion_sans_contact* (5/5), *consultation_solde* (3/5) et *gestion_carte_virtuelle* (3/5).

Nous constatons ensuite une première phase de remaniement des *clusters*, située entre les itérations 0 et 10, où le taux d'inexploitables chute au profit des *clusters* partiellement exploitables, dont la proportion augmente de 10 à près de 40%. À l'itération 10, le top 3 des classes identifiables mais bruitées ou en cohabitation dans un *cluster* sont *gestion_carte_virtuelle* (4/5), *alerte_perte_vol_carte* (4/5) et *commande_carte* (4/5).

Une seconde phase de consolidation se présente entre les itérations 10 et 25. Durant cette phase, les taux de *clusters* non exploitables et de partiellement exploitables diminuent alors que le taux d'exploitables monte en flèche (de 35% à 90% en 15 itérations). La majeure partie des *clusters* sont ainsi exploitables en l'état ou après la correction de quelques points aberrants. Après l'itération 25, le *cluster* le plus récalcitrant concerne un mélange des classes *alerte_perte_vol_carte* et *gestion_decouvert* (5/5).

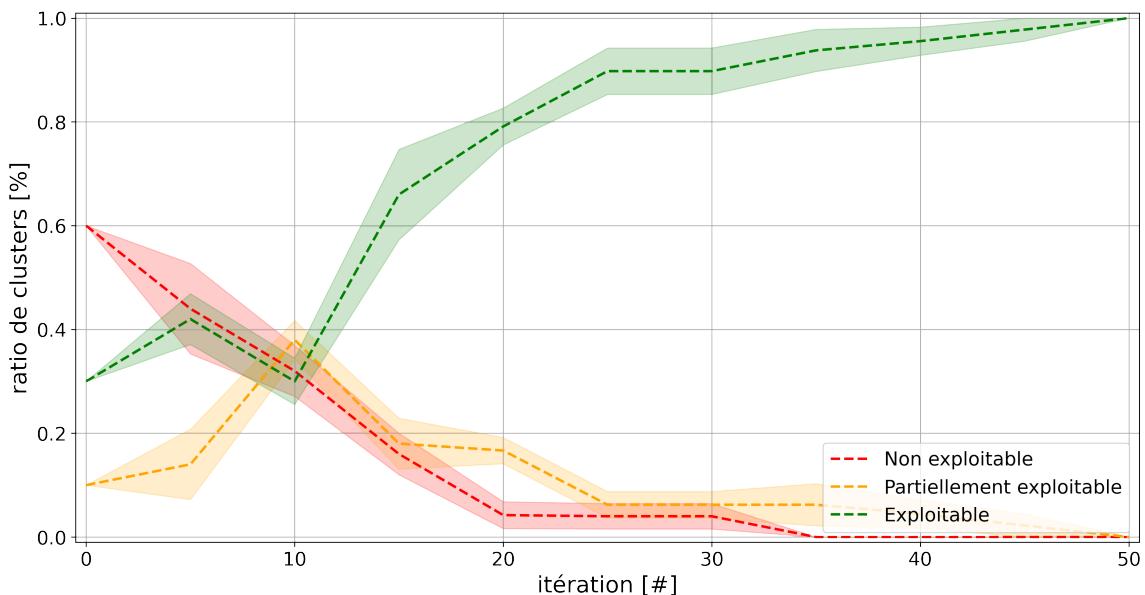


FIGURE 4.24 – Évolution de la pertinence métier moyenne estimée manuellement au cours des itérations du résultat du clustering interactif avec notre paramétrage favori. Cette pertinence, exprimée en proportion du nombre de clusters, est retranscrite en trois niveaux : *exploitable* en vert, *partiellement exploitable* en orange, et *non exploitable* en rouge.

4.4.1.c Discussion

Cette première étude visait à observer comment un expert métier peut interpréter un résultat de *clustering* proposé par notre méthode. Nous avons donc valider manuellement chaque *cluster* et essayé de les labelliser.

Comme l'analyse n'a pu être faite que par un seul opérateur, nous ne nous attardons pas sur la précision des taux d'exploitabilité des clusters. Nous pouvons déjà reconnaître que trois phases sont présentes :

- une première **phase exploratoire** (cf. itérations 0 à 10), où de premiers *clusters* partiellement exploitables apparaissent. Ces derniers contiennent souvent une thématique bruitée ou quelques thématiques mal séparées, ce qui permet toutefois à l'expert de se faire une idée des thématiques contenu dans la collecte de données. Cependant, s'arrêter à ce stade demanderait beaucoup de travail manuel pour obtenir une base d'apprentissage opérationnelle ;
- une seconde **phase de consolidation** (cf. itérations 10 à 25), où les *clusters* partiellement exploitable se raffinent. De plus en plus de *clusters* bien définis naissent, avec une seul thématique et peu de bruits. Ces derniers pourraient être extraits et exploités en l'état ;
- une **phase de parachèvement** (cf. itérations après 25), où la plupart des *clusters* sont exploitables en l'état mais quelques-uns nécessitent encore du travail. Cette phase est la moins rentable car les derniers *clusters* et points aberrants sont corrigés petit à petit, mais sans impact notable sur leur valeur métier.

Au cours de ces validation, il apparaît que la complexité réside dans l'analyse des *clusters* partiellement exploitables. En effet, les clusters totalement exploitable ou totalement inexploitables sont souvent simples à identifier. les premiers se repèrent facilement, surtout quand ils sont déjà présents lors d'itérations précédentes (exemple de la classe `gestion_sans_contact`, présente dès l'itération 0) ; les seconds, plutôt présents au début de la méthode, peuvent mélangler un grand nombre de thématique et s'apparenter rapidement à des *clusters* poubelle. En revanche, les *clusters* partiellement exploitables peuvent avoir des limites subjectives, altérant parfois l'avis de l'expert.

Pour aller plus loin, la difficulté de cette tâche est aussi dû aux contraintes suivantes :

- il faut être capable de maintenir en mémoire de grands ensembles de données ;
- il faut estimer la thématique principale à l'aide de données désordonnées ;
- il faut examiner la cohérence de cette thématique alors que le vocabulaire employé diffère ;
- il faut juger de l'importance du bruit contenu dans les *clusters* ;
- il faut prendre du recul pour repérer la dispersion d'une thématique dans plusieurs *clusters* ;
- ...

Tous ces facteurs peuvent donc nuire à la qualité, tant sur l'estimation de l'exploitabilité que sur le nommage des *clusters*.

Ainsi, nous déduisons aisément que l'expérience utilisateur proposée à l'expert métier induit une charge mentale élevée. Comme l'analyse de l'exploitabilité d'un *cluster* semble être une tâche complexe, il semble inconcevable de demander sa réalisation sur tous les *clusters* de toutes les

itérations ! De plus, sans aide supplémentaire et sans vis-à-vis pour se confronter à ses conclusions d'analyse, l'opérateur peut difficilement vérifier la cohérence et la reproductibilité de son travail. Il est donc nécessaire de considérer des pistes d'amélioration pour ne pas abandonner l'expert métier à lui-même dans cette tâche cruciale.

Quelques pistes peuvent être étudiées pour accompagner l'opérateur :

- employer plusieurs experts pour valider par consensus leurs appréciation sur un résultat de *clustering* : cette méthode est efficace pour rattraper les thématiques non identifiées et pour s'accorder sur la valeur d'un *cluster* ;
- préparer le travail d'analyse en réalisant une étude linguistique des *clusters*, et permettre ainsi d'identifier grossièrement les thématiques présentes en fonction du vocabulaire employé (cf. SECTION 4.4.2) ;
- automatiser une partie du travail d'analyse en utilisant les capacités des larges modèles de langage (*large language models*, LLM), afin d'alléger la charge de travail demandée aux experts métiers (cf. SECTION 4.4.3).

4.4.2 Étude des patterns linguistiques pertinents à l'aide de la Maximisation des Traits pour assister la validation d'une base d'apprentissage

Nous venons de conclure que la validation manuelle d'un résultat de *clustering* interactif est fonctionnelle, mais qu'elle souffre d'une très mauvaise expérience utilisateur. Afin d'améliorer cette aspect, une première idée consiste à mettre en valeur le vocabulaire caractéristique de chaque *cluster* et d'examiner si ces informations sont utiles à l'appréciation de leur valeur métier.

4.4.2.a Protocole expérimental

Pour résumer le protocole expérimental adapté, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.7.

Nous nous appuyons sur le même protocole que l'expérience précédente (cf. SECTION 4.4.1) : nous utilisons donc comme vérité terrain le jeu de données Bank Cards (v1.0.0), nous réalisons 5 tentatives complètes de la méthode du *clustering* interactif en utilisant notre paramétrage favori (voir SECTION 4.3), et nous demandons toutes les 5 itérations à un expert de qualifier les *clusters* obtenus entre trois catégories (**exploitable**, **partiellement exploitable** et **non exploitable**).

Cependant, avant de demander l'avis de l'expert, nous réalisons une analyse du vocabulaire employé. Pour cela, nous utilisons une sélection des patterns linguistiques pertinents basée sur la maximisation des traits (notée FMC, cf. LAMIREL et al., 2017) : cette méthode permet de trouver les composantes vectorielles caractéristiques et discriminantes de chaque *cluster* en attribuant un score à chaque couple (*cluster*, *composante*). Dans notre cas, en utilisant une vectorisation basée sur la fréquence du vocabulaire dans un document comme **TF-IDF** (SPARCK JONES, 1972), nous pouvons déterminer les mots les plus représentatifs de chaque *cluster*. Ainsi, nous pouvons à la fois décrire chaque groupe de questions par une liste de mots clés caractéristiques, mais aussi surligner ces mots dans les questions à parcourir pour attirer l'attention de l'expert sur ce qui semble être statistiquement discriminant.

Nous adaptons aussi la tâche de l'expert afin qu'il donne son avis sur chaque *cluster* en répondant aux questions suivantes :

- est-ce que la liste des patterns linguistiques caractéristiques du *cluster* est suffisamment complète pour permettre d'identifier une thématique principale **bien définie** ? (*en effet*,

```
1 Données : jeu de données annotées (vérité terrain)
2 pour chaque jeux de données à tester faire
3   initialisation (données) : récupérer les données et la vérité terrain ;
4   initialisation (contraintes) : créer une liste vide de contraintes ;
5   prétraitement : supprimer le bruit dans les données avec prep.simple ;
6   vectorisation : transformer les données en vecteurs avec vect.tfidf ;
7   clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
8   analyse linguistique : caractériser les clusters grâce à la FMC ;
9   évaluation assistée : juger de l'exploitabilité de chaque cluster ;
10  labellisation assistée : nommer chaque cluster exploitable ;
11  répéter
12    échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
13    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
14    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
15    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
16    analyse linguistique : caractériser les clusters grâce à la FMC ;
17    évaluation assistée : juger de l'exploitabilité de chaque cluster ;
18    labellisation assistée : nommer chaque cluster exploitable ;
19  jusqu'à annotation de toutes les contraintes possibles;
20 analyse : afficher l'évolution de l'exploitabilité de chaque itération de clustering ;
Résultat : discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité
```

ALGORITHME 4.7 – Description en pseudo-code du protocole expérimental de l'étude des patterns linguistiques pertinents pour vérifier la valeur métier d'une base d'apprentissage.

comment interpréter un cluster sans définition claire ?)

- est-ce que la liste des patterns linguistiques caractéristiques du *cluster* identifie plusieurs thématiques ou bruits dans le *cluster* ? (*en effet, comment avoir de bonnes performances si la base d'apprentissage n'est pas fiable ?*)

💡 Idées : Nous pourrions aussi analyser l'impact ergonomique qu'apporte la mise en exergue des patterns linguistiques pertinents dans le texte de chaque *cluster*. Une telle étude pourrait ainsi mesurer le gain de temps et de qualité par rapport à une validation manuelle non assistée comme présentée en SECTION 4.4.1.

❶ Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b. L'implémentation de la maximisation des traits est accessible ici dans SCHILD, 2023.

4.4.2.b Résultats obtenus

⚠️ Attention : Par manque de moyen, la vérification manuelle des *clusters* n'a pas été réalisée. Nous présentons donc simplement quelques exemples d'analyses linguistiques réalisées grâce à la FMC.

Prenons quelques *clusters* et suivons l'évolution de leur analyse linguistiques au cours des itérations. Nous nous référons aux tableaux ci-contre pour connaître le top 10 des termes caractéristiques des différents *clusters* ainsi qu'un extrait de questions issu de ces *clusters* avec une mise en évidence des termes caractéristiques dans le texte.

D'abord, prenons l'exemple suivi dans la TABLE 4.5. Nous suivons ici l'évolution d'un *cluster* bien formé dès l'itération 0. En effet, il aisément de juger ce *cluster* exploitable et d'en déduire sa thématique : le *cluster* est de taille suffisante (plus de 45 questions), son vocabulaire caractéristique est fourni (36 à 38 patterns) et la liste des patterns mis en avant sont cohérents (*sans contact*, *paiement sans contact*, *activer le*, *nfc*, ...). En parcourant son contenu, on arrive rapidement à associer sa thématique à la classe *gestion_sans_contact* de la vérité terrain.

Ensuite, prenons l'exemple décrit dans la TABLE 4.6. À l'itération 0, il est impossible d'exploiter ce *cluster* : il n'y a que deux patterns mis en avant ne traitant pas d'une thématique, et les questions semblent toutes traiter de sujets différents. À l'itération 10, le résultat semble un peu plus exploitable. Nous pouvons d'ailleurs déduire deux thématiques principales : la première, mise en avant par des patterns linguistiques de type "*numero*", "*online*", et "*de carte virtuelle*", permet d'imaginer un sujet sur la gestion des numéros de cartes virtuelles ; la seconde, identifiée par les termes "*débloquer*" et "*réactiver*", oriente plutôt vers la création d'un thème pour débloquer une carte bancaire. Quelques bruits sont présent, mais ce *cluster* à l'itération 10 peut donc être associé aux classes *gestion_carte_virtuelle* et *déblocage_carte*. Dès l'itération 15, ces thématiques se séparent en deux clusters (1 et 4) : nous pouvons les identifier à l'aide de leurs listes de termes bien fournies.

Identification du cluster	Analyse linguistique (avec la FMC)	Aperçu du cluster (avec emphase)
Tentative : 1 Itération : 0 Cluster : 0 Avis initial : Exploitable	- sans contact - contact - sans - mode - le mode - sur ma carte - le sans contact - le sans - paiement sans contact (Total : 36)	- activer le moyen de paiement nfc sur ma carte gold - enlever le mode sans contact de ma carte - gerer le mode de paiement nfc sur ma carte - je souhaite gerer le mode nfc sur mes cartes bancaires - l option sans contact ne fonctionne pas sur ma carte - modifier le mode sans contact - modifier le mode nfc sur ma carte de paiement - peut on annuler le paiement sans contact - puis je activer le sans contact depuis l application (Total : 46)
Tentative : 1 Itération : 15 Cluster : 0 Avis initial : Exploitable	- sans contact - contact - sans - sur ma - mode - le mode - sur ma carte - nfc - le sans contact (Total : 38)	- activer le moyen de paiement nfc sur ma carte gold - enlever le mode sans contact de ma carte - gerer le mode de paiement nfc sur ma carte - je souhaite gerer le mode nfc sur mes cartes bancaires - l option sans contact ne fonctionne pas sur ma carte - modifier le mode sans contact - modifier le mode nfc sur ma carte de paiement - peut on annuler le paiement sans contact - puis je activer le sans contact depuis l application (Total : 50)

TABLE 4.5 – Extrait de l’analyse linguistique de clusters exploitables dès la première itération. Ces clusters représentent la thématique *gestion_sans_contact* entre l’itération 0 (initialisation) et l’itération 15 (atteinte de la vérité terrain). La troisième colonne expose un aperçu du contenu des clusters en mettant l’emphase sur les termes caractéristiques identifiés grâce à la FMC, et la deuxième colonne représente le top 10 de ces termes les plus caractéristiques pour chaque cluster.

4.4.2.c Discussion

Cette seconde étude avait pour objectif de proposer une assistance à la validation manuelle des *clusters* par un expert métier afin d’en qualifier la pertinence métier. Pour cela, nous avons choisi une analyse linguistique à l’aide de la maximisation de traits pour mettre en avant les mots représentatif et discriminants de chaque groupe de questions. Nous discutons ici de l’intérêt d’une telle analyse.

Tout d’abord, concernant les *clusters* jugés exploitables, nous constatons que la FMC permet d’identifier aisément la thématique principale. C’est le cas dans les exemples présentés dans les TABLE 4.5 et TABLE 4.6), où les thématiques sont bien représentées par leurs patterns (*gestion_sans_contact* avec "mode", "sans", "contact", "nfc"; *gestion_carte_virtuelle* avec "numero", "virtuel", "online"; *deblocage_carte* avec "debloquer", "deverrouiller", "carte") De plus, la présence des différentes variantes d’un même pattern (avec ses pluriels, au sein d’un groupe de termes, ...) permet de rapidement d’intégrer le champ lexical présent, aidant ainsi à interpréter le *cluster* comme probablement exploitable.

Les thématiques présentent dans les *clusters* partiellement exploitables sont aussi identifiables grâce à la FMC, mais à moindre mesure. En effet, comme plusieurs thématiques se mélangent, l’ensemble de termes caractéristiques est aussi plus hétérogène, des variantes ne sont pas identifiées, et des patterns dénués de sens peuvent être mis en avant. Par exemple, dans le *cluster* 2 de la tentative 1 à l’itération 0, deux classes sont présentes avec leurs termes caractéristiques (*consultation_solde* avec "solde" et "compte"; *gestion_carte_virtuelle* avec

4.4. Évaluation de l'hypothèse de pertinence

Identification du cluster	Analyse linguistique (avec la FMC)	Aperçu du cluster (avec emphase)
Tentative : 1 Itération : 0 Cluster : 1 Avis initial : Non exploitable	- carte avalee - nouvelle carte bancaire (Total : 2)	- ai je le droit d avoir un decouvert bancaire - bonjour pouvez vous debloquer ma carte merci - carte bancaire avalee - choisir une nouvelle carte bancaire - comment signaler un vol de carte bleue - diminuer le plafond d une carte gold - le rapatriement est il couvert par ma carte bancaire - que faire pour activer une carte bancaire virtuelle - quelle est ma situation financiere (Total : 157)
Tentative : 1 Itération : 10 Cluster : 2 Avis initial : Partiellement exploitable	- un numero - numero - de carte virtuelle - un numero de - numero de carte - numero de - numeros - debloquer ma - débloquer ma carte (Total : 25)	- activer les achats avec un numero virtuel - comment debloquer ma mastercard - comment reactiver sa carte - j aimerais debloquer ma carte svp - pouvez vous debloquer ma carte - obtenir une carte online - ou en est ma situation financiere - ou puis je gerer mes numeros virtuels - supprimer une carte virtuelle (Total : 80)
Tentative : 1 Itération : 15 Cluster : 2 Avis initial : Exploitable Tentative : 1 Itération : 15 Cluster : 4 Avis initial : Exploitable	- virtuelle - carte virtuelle - un numero - numero - de carte virtuelle - un numero de - numero de carte - numero de - numeros (Total : 34) - reactiver - debloquer - debloquer ma - debloquer ma carte - bloquee - reactiver sa - reactiver ma - deverrouiller - reactiver sa carte (Total : 24)	- activer les achats avec un numero virtuel - comment consulter ses numeros de carte virtuelle - comment obtenir un numero de carte virtuelle - comment supprimer un numero de carte online - creer une carte bancaire virtuelle - faire un achat avec un numero de carte online - j aimerais utiliser une carte virtuelle - ou peut on gerer ses numeros virtuels - supprimer un numero de carte virtuel (Total : 49) - bonjour pouvez vous debloquer ma carte merci - comment deverrouiller sa carte - comment reutiliser une carte bancaire bloquee - debloquer sa carte apres trois mauvais codes - j ai besoin de deverrouiller ma carte de paiement - j ai retrouve ma carte puis je la reactiver - je souhaite debloquer ma carte bleue - pouvez vous debloquer ma carte - reactiver une carte suspendue (Total : 48)

TABLE 4.6 – Extrait de l'analyse linguistique de clusters évoluant de non exploitables à exploitables. Ces clusters représentent la conception des thématiques *gestion_card_virtuelle* et *debloque_card*, entre l'itération 0 (initialisation) et l'itération 15 (atteinte de la vérité terrain). La troisième colonne expose un aperçu du contenu des clusters en mettant l'emphase sur les termes caractéristiques identifiés grâce à la FMC, et la deuxième colonne représente le top 10 de ces termes les plus caractéristiques pour chaque cluster.

"*carte virtuelle*" et "*numero*"), mais certains termes quelconques sont pourtant sont présentés comme représentatifs ("*de mes*", "*avec un*") et d'autres termes intéressants ne sont pas identifiés ("*compte*" et "*numeros*" ne sont présents qu'au singulier). Ces petits indices peuvent donc aiguiller l'expert sur des ajustements nécessaires ou un *cluster* réalisé sur des bases fragiles.

Enfin, en ce qui concerne les *clusters* jugés non exploitables, la FMC permet de confirmer leur manque de cohérence (pour le *cluster* 1 de la tentative 2 à l'itération 0 : "carte gold", "numeros virtuels", "découvert bancaire", "carte paiement differe", ...) ou leur absence de valeur métier (seulement 2 termes caractéristiques pour le *cluster* 1 de la tentative 1 à l'itération 0). Nous pouvons aussi constater que les termes estimés comme caractéristiques pour ces *clusters* non exploitables sont souvent des groupes de mots trop spécifiques (dans notre dernier exemple : "numeros virtuels" uniquement au pluriel), ce qui est plutôt caractéristique de termes qui distinguent le *cluster* des autres mais qui ne l'identifient pas (voir LAMIREL et al., 2017 pour comprendre la balance entre *Features Recall* (identification) et *Features Predominance* (discrimination)).

Cependant, une telle approche pour qualifier les *clusters* risque de ne pas répondre à nos attentes en l'état, car certains problèmes identifiés dans la section précédente subsistent (4.4.1). Entre autres, malgré une abstraction des *clusters* par leurs termes les plus caractéristiques (cf. deuxième colonne dans les TABLE 4.5 et TABLE 4.6) et une mise en exergue dans le texte (cf. troisième colonne de ces tableaux), il faut toujours parcourir un grand nombre de données pour juger de la pertinence métier, ce qui reste une tâche chronophage s'il faut la réaliser à chaque itération. Ainsi, même si le travail est simplifié par l'identification rapide des termes importants du corpus, la charge de travail reste élevée.

D'autre part, les résultats remontés par l'analyse linguistiques sont à affiner davantage pour faciliter leur interprétation. Par exemple, les mots pourraient être réduits à leur forme racine (lemmatisation) afin d'éviter de considérer les nombreuses variations possibles (formes conjuguées, pluriels, ...) et les mots vides (*stopwords*) pourraient être supprimés pour accentuer l'absence de termes caractéristiques à valeur métier dans les *clusters* inexploitables. En ce qui concerne l'affichage des patterns identifiés dans les textes, un code couleur pourrait être introduit pour faciliter la lecture, avec l'usage de nuance de couleurs pour marquer la prépondérance d'un terme en fonction du *cluster*. L'exemple ci-dessous (issu du *cluster* 0 de la tentative 1 à l'itération 0) illustre l'intérêt de cette coloration : nous y distinguons facilement que le *cluster* traite des paiements sans contact (NFC en vert) mais que les cartes Visa (en rouge) ont dû être regroupées au sein d'un autre *cluster*.

Q Exemples : Possibilité d'affichage en couleur d'un texte et de son analyse linguistique : les patterns caractéristiques du *cluster* auquel appartient le texte en vert, les patterns caractéristiques des autres *clusters* en rouge, les patterns non caractéristiques en noir.

« *est ce que le mode de paiement nfc est disponible sur ma carte visa* »

Mais malgré ces améliorations de l'approche linguistique, il est probable qu'elle soit trop éloignée des compétences réelles des experts : en effet, ces derniers disposent de connaissances sur leur domaine métier (dans notre cas, des sujets concernant la banque, l'assurance et la finance), mais ils ne disposent pas forcément d'aptitudes permettant d'examiner les nuances linguistiques présentes dans un *cluster*, ni l'impact de la prépondérance des pluriels, des bigrams, des mots vides de sens, ...

Notes de l'auteur : Nous utilisons notre expérience personnelle pour avancer cet avis. En effet, lors de précédents travaux industriels (non publiés), nous avons réalisé une

modélisation thématique (*topic modeling*) sur des corps de mail en utilisant la LDA (BLEI et al., 2003). Comme il est coutume, nous avons réalisé une abstraction en énumérant le vocabulaire employé par chaque *topic*. Puis, nous avons fait intervenir des experts métiers afin d'estimer la valeur de chaque *topic*, de les raffiner, et de les nommer grâce aux abstractions réalisées. Toutefois, nous avons constaté que les experts intervenants dans ce projet avaient du mal à manipuler de telles abstractions, notamment car ils n'arrivaient pas à se projeter sur les *topic* peu ou pas exploitable. Au final, les experts ont préféré modéliser manuellement le corpus de mail, sans utiliser les résultats de la LDA.

Une partie de cet échec est probablement lié à l'utilisation en tant que telle de la LDA (peu d'interactivité entre l'expert et l'algorithme de modélisation), mais nous avons aussi eu des retours directs des experts sur leur difficulté à utiliser des champs lexicaux pour juger de la qualité et de la cohérence d'une modélisation. Ainsi, même si une abstraction linguistique semble sémantiquement pertinente, il se peut que les intervenants du projet ne soient pas à l'aise avec son utilisation. Une étude spécifique à ce sujet pourrait vérifier ce ressenti.

Pour conclure, nous retenons que l'analyse linguistique est pleine de potentiel et qu'elle permet de mettre en exergue des mots importants lors de l'affichage du contenu des clusters. Toutefois, nous conservons quelques doutes sur l'expérience utilisateur d'une telle approche pour des experts métiers qui, n'étant pas des experts en linguistique, pourraient se perdre sur des considérations trop techniques durant leur analyse. De fait, si une approche linguistique est trop abstraite pour qualifier un *cluster*, nous nous intéressons pour la suite à une approche plus pragmatique pour identifier sa thématique principale (cf. SECTION 4.4.3).

4.4.3 Étude d'un résumé automatique des *clusters* à l'aide d'un large modèle de langage

Comme nous l'avons vu dans la section précédente, une analyse linguistique peut paraître trop abstraite pour un expert métier. Nous nous intéressons donc à un moyen de simplifier l'identification de thématiques dans un *cluster*, et envisageons l'automatisation de cette tâche en utilisant les capacités d'un large modèle de langage (LLM). En effet, plusieurs de ces modèles ont montré leur efficacité sur les tâches de résumé de documents (ZHANG et al., 2019, LEWIS et al., 2019, RADFORD et al., 2019, BROWN et al., 2020), une fonctionnalité que nous allons adapter³⁵ et étudier ici.

4.4.3.a Protocole expérimental

Pour résumer le protocole expérimental adapté, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.8.

Nous nous appuyons sur le même protocole que l'expérience précédente (cf. SECTION 4.4.1) : nous utilisons donc comme véritable terrain le jeu de données Bank Cards (v1.0.0), nous réalisons 5 tentatives complètes de la méthode du *clustering* interactif en utilisant notre paramétrage favori (voir SECTION 4.3), et nous demandons toutes les 5 itérations à un expert de qualifier les *clusters* obtenus entre trois catégories (**exploitable**, **partiellement exploitable** et **non exploitable**).

³⁵ Nous nous inspirons notamment de ALAMMAR et GREFENSTETTE, 2022 qui propose un boîte à outils en Python permettant de nommer des *topics* en utilisant BERT.

```
1   Données : jeu de données annotées (vérité terrain)
2   pour chaque jeux de données à tester faire
3       initialisation (données) : récupérer les données et la vérité terrain ;
4       initialisation (contraintes) : créer une liste vide de contraintes ;
5       prétraitement : supprimer le bruit dans les données avec prep.simple ;
6       vectorisation : transformer les données en vecteurs avec vect.tfidf ;
7       clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
8       synthèse automatique : résumer les thématiques des clusters par un LLM ;
9       évaluation assistée : juger de l'exploitabilité de chaque cluster ;
10      labellisation assistée : nommer chaque cluster exploitable ;
11      répéter
12          échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
13          simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
14          intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
15          clustering : regrouper les données par similarité avec clust.kmeans.cop ;
16          synthèse automatique : résumer les thématiques des clusters par un LLM ;
17          évaluation assistée : juger de l'exploitabilité de chaque cluster ;
18          labellisation assistée : nommer chaque cluster exploitable ;
19      jusqu'à annotation de toutes les contraintes possibles;
20
21      analyse : afficher l'évolution de l'exploitabilité de chaque itération de clustering ;
22      Résultat : discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité
```

ALGORITHME 4.8 – Description en pseudo-code du protocole expérimental de l'étude d'un résumé automatique des clusters à l'aide d'un large modèle de langage pour vérifier la valeur métier d'une base d'apprentissage.

Cependant, avant de demander l'avis de l'expert, nous utilisons un large modèle de langage pour résumer automatiquement le contenu du *cluster* à analyser. Pour cela, nous utilisons le modèle gpt-3.5-turbo mis à disposition par OpenAI³⁶. Le *prompt* du modèle, adapté à l'usage de notre jeu de données, est composé de trois parties :

- un **contexte d'utilisation**, destiné à centrer les réponses sur le domaine général traité : « *Tu es un expert des secteurs banque, assurance et finance.* » ;
- une **description de la tâche** avec les consignes de restitution : « *Résume-moi en une phrase la thématique traitée dans les textes suivants* : » ;
- les **textes** du *cluster*, énumérés sous la forme d'une liste à puces. Si la taille maximale du *prompt* ne peut pas prendre en compte l'ensemble des données du *cluster*, il est possible de ne prendre qu'un échantillon.

Exemples : Exemple de *prompt* pour le *cluster* 0 de la tentative 1 à l'itération 0.

Tu es un expert des secteurs banque, assurance et finance.

Résume-moi en une phrase la thématique traitée dans les textes suivants :

- *activer le moyen de paiement nfc sur ma carte gold*
- *enlever le mode sans contact de ma carte*
- *gerer le mode de paiement nfc sur ma carte*
- ... (43 autres) ...

À l'aide de ces résumés, nous pouvons ainsi préparer le travail de l'expert en mettant en avant une synthèse des informations contenus dans chaque *cluster*.

Nous adaptons aussi la tâche de l'expert afin qu'il donne son avis sur chaque *cluster* en répondant aux questions suivantes :

- est-ce que le résumé est suffisamment précis pour identifier une thématique principale bien définie dans le *cluster* ? (*en effet, comment interpréter un cluster sans définition claire ?*)
- est-ce que le résumé identifie plusieurs thématiques ou bruits dans le *cluster* ? (*en effet, comment avoir de bonnes performances si la base d'apprentissage n'est pas fiable ?*)

Idées : Nous pourrions aussi analyser l'impact ergonomique qu'apporte l'automatisation de la synthèse thématique de chaque *cluster*. Une telle étude pourrait ainsi mesurer le gain de temps et de qualité par rapport à une validation manuelle non assistée comme présentée en SECTION 4.4.1. Ceci pourrait faire l'objet d'études ultérieures.

³⁶ OpenAI est une entreprise fournissant des services d'intelligence artificielle sur le Cloud. Elle est entre autres connue pour les modèles d'IA DALL-E (RAMESH et al., 2021) et GPT (BROWN et al., 2020, OPENAI, 2023)

⚠️ Attention : Par manque de personnes aptes à qualifier le jeu de données utilisé, les annotations de cette étude ont été réalisées par un seul opérateur (*moi-même, ayant participé à la création de ce jeu de données*). Nous supposons que cette contrainte n'est pas pénalisante pour l'analyse : en effet, dans une situation réelle, cet opérateur serait responsable des choix à entreprendre pour concevoir le jeu de données, il est donc le mieux placer pour juger la pertinence d'un *clustering* par rapport à sa propre modélisation du problème. Les problèmes d'accords inter-annotateurs seront plutôt discuté en SECTION 4.6. Nous réalisons toutefois 5 tentatives différentes de la méthode pour limiter les biais intra-individuels.

ℹ️ Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b. L'appel à un large modèle de langage, sous la forme d'un appel API, se fait grâce à la librairie `openai`³⁷.

4.4.3.b Résultats obtenus

La FIGURE 4.25 met en avant l'évolution de la pertinence moyenne estimée par l'opérateur sur la base des résumés automatiques des *clusters*. Comme lors de l'analyse manuelle (cf. SECTION 4.4.1), il est normal de retrouver une tendance générale à la diminution du nombre de *clusters* inexploitables au profit de *clusters* exploitables. La différence principale réside dans l'absence du pic de croissance du nombre de *clusters* partiellement exploitables, dont l'apogée était précédemment situé à l'itération 10.

Pour aller plus loin, reprenons les *clusters* que nous avons utilisé comme cas d'étude lors de l'analyse linguistique à l'aide de la maximisation des traits (cf. SECTION 4.4.2).

D'abord, reprenons l'exemple de l'évolution d'un *cluster* bien formé dès l'itération 0, précédemment détaillé dans la TABLE 4.5 et dont les résumés automatiques sont présentés dans la TABLE 4.7. Nous constatons effectivement que les synthèses générées par le modèle identifie sans ambiguïté le thème du paiement sans contact, même si le résumé de l'itération 0 énumère longuement les actions réalisables sur ce sujet.

Ensuite, intéressons nous l'exemple de l'évolution des *clusters* en cours de formations détaillés dans la TABLE 4.5 et dont les résumés automatiques sont présentés dans la TABLE 4.7. À l'itération 0, le résumé proposé est une longue énumération de 9 thématiques différentes sur la gestion de carte bancaires : puisque le jeu de données entier traite des cartes bancaires, nous identifions clairement ce *cluster* comme non exploitable. À l'itération 10, nous ne distinguons plus que deux sujets principaux : le déblocage de carte, et l'utilisation de numéros de cartes virtuelles, ce qui est en accord avec notre précédente analyse. À partir de l'itération 15, ces deux thématiques se retrouvent bien séparées dans deux *clusters* différents, et chacune est identifiable via le résumé proposé : on note toutefois que si la thématique principale est identifiée, alors l'énumération de détails est plutôt portée sur les actions réalisables avec cette thématique ("*création*", "*activation*", "*suppression*" pour la classe `gestion_carte_virtuelle`).

De manière général, la majorité des résumés automatiques permettent d'identifier sans ambiguïté les mêmes thématiques qu'une validation manuelle.

37. `openai` : <https://pypi.org/project/openai/>

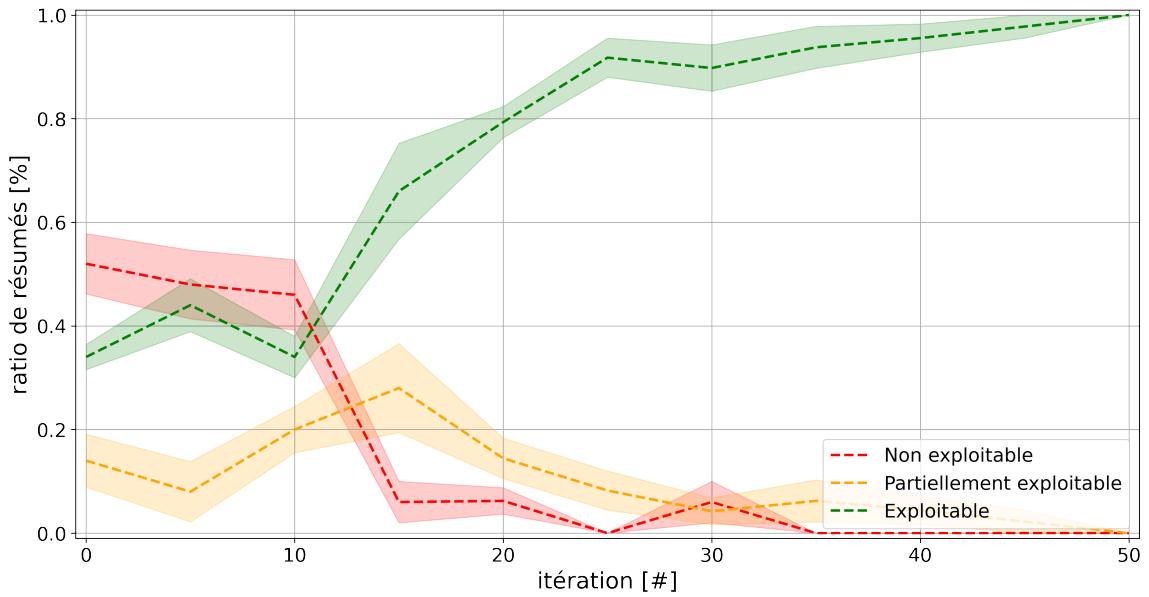


FIGURE 4.25 – Évolution de la pertinence métier moyenne en fonction du nombre d’itérations de la méthode. Cette pertinence, exprimée en proportion du nombre de clusters, est estimée sur la base du résumé automatique des clusters par un large modèle de langage et est retranscrite en trois niveaux : *exploitable* en vert, *partiellement exploitable* en orange, et *non exploitable* en rouge.

Identification du cluster	Résumé automatique du cluster (LLM)
Tentative : 1 Itération : 0 Cluster : 0 Avis initial : Exploitable	La thématique traitée dans ces textes est la gestion de l’activation, la désactivation, la modification ou la nécessité d’utiliser le paiement sans contact ou le NFC (Near Field Communication) sur les cartes de paiement bancaires.
Tentative : 1 Itération : 15 Cluster : 0 Avis initial : Exploitable	Les textes traitent de la gestion et de l’utilisation du paiement sans contact (NFC) sur les cartes bancaires.

TABLE 4.7 – Extrait de l’analyse de résumés automatiques de clusters exploitables dès la première itération. Ces clusters représentent la thématique *gestion_sans_contact* entre l’itération 0 (initialisation) et l’itération 15 (atteinte de la vérité terrain). La seconde colonne expose le résumé obtenu en appelant un large modèle de langage (*gpt-3.5-turbo*) sur une tâche de résumé.

- En ce qui concerne les itérations où le *clustering* atteint la vérité terrain, tous les résumés permettent d’identifier les thématiques présentes (190 *clusters* concernés, toutes itérations confondues) ;
- En ce qui concerne les autres itérations (dont le *clustering* ne correspond pas encore à la vérité terrain), les résumés automatiques permettent d’identifier les mêmes thématiques qu’une validation manuelle dans 86% des cas (331 cas sur 386 *clusters*, toutes itérations confondues).

Identification du cluster	Résumé automatique du cluster (avec un LLM)
Tentative : 1 Itération : 0 Cluster : 1 Avis initial : Non exploitable	La thématique des textes est la gestion des cartes bancaires, incluant la sécurité, les pertes, les vols, les blocages, les récupérations, le changement ou la commande de cartes, les garanties et assurances, les découverts, les paiements virtuels, les numéros virtuels et les plafonds de paiement autorisés.
Tentative : 1 Itération : 10 Cluster : 2 Avis initial : Partiellement exploitable	Les textes traitent principalement de la gestion des cartes de paiement, de leur déblocage ou leur réactivation, et de l'utilisation de numéros de carte virtuelle pour les achats en ligne.
Tentative : 1 Itération : 15 Cluster : 2 Avis initial : Exploitable Tentative : 1 Itération : 15 Cluster : 4 Avis initial : Exploitable	Les textes concernent la gestion et l'utilisation des numéros de carte virtuelle pour les achats en ligne, notamment la création, l'activation, la suppression et la gestion de ces numéros virtuels. La thématique traitée dans ces textes est le déblocage, le déverrouillage ou la réactivation de cartes bancaires bloquées.

TABLE 4.8 – Extrait de l’analyse de résumés automatiques de clusters évoluant de non exploitables à exploitables. Ces clusters représentent la conception des thématiques *gestion_carte_virtuelle* et *debloage_carte*, entre l’itération 0 (initialisation) et l’itération 15 (atteinte de la vérité terrain). La seconde colonne expose le résumé obtenu en appelant un large modèle de langage (*gpt-3.5-turbo*) sur une tâche de résumé.

confondues).

4.4.3.c Discussion

Cette troisième et dernière étude sur l'estimation de la valeur métier d'un résultat de *clustering* interactif avait pour but de simplifier la tâche d'analyse de l'expert en évaluant les capacités un large modèle de langage à synthétiser les thématiques présentes. Nous comparons les résultats obtenus avec ceux obtenus dans les deux sections précédentes.

En premier lieu, nous constatons que l'approche est efficace, car elle permet dans près de 86% des cas à l'expert de parvenir aux mêmes conclusions sur l'analyse de la valeur métier d'un *cluster*. Concernant les *clusters* exploitables, la synthèse générée est généralement très explicite, à l'image des résultats présentés dans la TABLE 4.7, ce qui rend la tâche de labellisation des *clusters* presque triviale. On retrouve notamment ces résultats lorsque le *clustering* atteint la vérité terrain : toutes les descriptions permettent de décrire sans ambiguïté les thématiques traitées, confirmant à la fois que le jeu de données est bien annoté et que le *clustering* est exploitable. De manière similaire, les *clusters* non exploitables peuvent aussi être rapidement identifiés, notamment par la présence de longues énumérations incohérentes et des tournures de phrases ambiguës, à l'image du premier exemple de la TABLE 4.8. Les résultats obtenus ci-dessus permettent donc de conclure sans hésitation à l'approche est adéquate pour assister un expert

dans sa tâche d'évaluation.

En plus de la justesse des résumés obtenus, il y a aussi un gain réel de confort pour l'expert métier. En effet, la tâche de celui-ci consiste désormais à simplement à lire une description textuelle et de confirmer si elle correspond à un cas d'usage métier. Ainsi, plus besoin de parcourir de grands ensembles de données ou de réaliser des analyses linguistiques complexes : l'exercice est simple, peu chronophage, et est centrer sur les compétences réelles de l'expert. Nous pourrions aller plus loin en déclinant cette approche sur d'autres analyses, par exemple en réduisant cette synthèse à quelques mots pour nommer le *clusters*, ou en demandant d'identifier les potentielles données aberrantes à supprimer pour augmenter la cohérence du regroupement de questions.

Bien entendu, l'automatisation de cette tâche peut aussi orienter l'expert vers d'autres conclusions, voire le mener vers une fausse piste. Nous estimons à 14% le taux de différences d'identification de thématiques avec une approche purement manuelle, et nous constatons que la majorité des écarts entre ces approches résident dans les cas de figure suivants :

- le modèle peut générer des hallucinations n'ayant rien à voir avec les données en entrée : c'est le cas avec le cluster 9 de la tentative 2 à l'itération 10, où le cluster est composé de 2 questions (« *Comment obtenir une Mastercard ?* » et « *Désactiver les numéros virtuels.* ») et où le résumé parle de "sécurisation des transactions bancaires" ;
- le résumé peut être trop concis et certaines thématiques peuvent être ignorées dans la synthèse : l'expert sera tenté de conclure à un *cluster* exploitable alors qu'il est potentiellement bruité ;
- à l'inverse, les données aberrantes ou isolées d'un *cluster* peuvent influencer le résumé : cela peut donner l'illusion que plusieurs thématiques sont présentes alors qu'une seule ne l'est réellement (voir aussi FALKE et al., 2019 qui met en avant l'importance de l'ordre des informations transmises au modèle) ;
- le résumé peut aussi mettre en avant des thématiques auxquelles l'expert n'aurait pas pensé : c'est le cas dans les *clusters* 7 et 8 de la tentative 1 à l'itération 0, où les thématiques `gestion_carte_mastercard` et `gestion_carte_visa` sont proposées dans la synthèse, mais auraient probablement été considérées par un expert (il n'y a pas de différences significatives entre les deux réseaux de cartes bancaires pour justifier une gestion séparée, même si les clusters sont pourtant bien formés).

Ces différents exemples confirment qu'une étape de validation reste nécessaire. Celle-ci n'a pas besoin d'être systématique, elle peut être réalisée pour confirmer la fin des itérations de *clustering* interactif, limitant ainsi la charge de travail de l'expert.

Toutefois, le principal obstacle à l'utilisation des larges modèles de langage concerne des problématiques techniques pour disposer, installer et utiliser ces modèles. En effet, à l'heure de rédaction de ce manuscrit (juillet 2023), l'entraînement requiert des serveurs aux configurations pharaoniques, composés de plusieurs milliers de GPU, du stockage et de la bande passante pour manipuler des Téra octets de données, et de toute l'infrastructure auxiliaire nécessaire pour contrôler et refroidir les machines, soit un investissement de plusieurs millions voire milliards de dollars. L'inférence n'est pas simple non plus, car les machines doivent entre autres disposer de suffisamment de RAM pour charger les modèles et de GPU pour les exécuter. Ainsi, seules quelques entreprises peuvent investir et mettre à disposition ce genre d'infrastructures, souvent sous forme de services hébergés sur le *Cloud*, comme la plateforme *Meta Research Super Cluster* (LEE et

SENGUPTA, 2022) ou le service *Microsoft Azure AI* (ROACH, 2023))³⁸.

Une autre limite en découlant concerne la confidentialité des données. En effet, comme l'utilisation des LLM se fait via des services externes, il est possible que certaines plateforme ré-entraînent leurs modèles à l'aides des données communiquées. Ainsi, HUANG et al., 2022 et O'NEILL et CONNOR, 2023 mettent en avant les risques qu'un modèle soit capable d'intégrer puis de restituer des données privées ou confidentielles. Dans cette expérience, nous n'avions pas de craintes car les données sont publiques, mais cela peut pénaliser un projet manipulant des données plus sensibles.

En conclusion, l'utilisation d'un LLM semble très prometteuse pour aider un expert métier à estimer la valeur métier d'un partitionnement de données. Nous constatons notamment un réel impact sur l'expérience utilisateur car la synthèse automatique d'un *cluster* réduit drastiquement la charge et la complexité de travail de l'expert. Néanmoins, nous notons plusieurs zones d'ombres sur la confiance que nous pouvons apporter à l'automatisation de cette tâche, tant sur l'utilisation des modèles (infrastructure technique lourde, confidentialité des données, ...) que sur l'exploitation de leurs résultats (hallucinations, ambiguïtés, ...).

4.4.4 Mise en commun des stratégies d'évaluation de la pertinence métier d'un résultat de *clustering* interactif

 **Points à retenir :** Au cours de cette étude de pertinence, nous avons pu voir que :

- La tâche de validation et de labellisation de résultats de *clustering* est plutôt complexe et fastidieuse si elle n'est pas assistée : il est donc conseillé de faire intervenir plusieurs experts afin de consolider leurs analyses et confronter les points de vues ;
- Nous pouvons utiliser des larges modèles de langage (LLM) pour réaliser une synthèse automatique d'un *cluster* : cela permet d'estimer rapidement la pertinence d'un regroupement de questions et d'identifier les thématiques qui y sont présentes, offrant ainsi un gain d'efficacité et de confort aux experts métier (cf. SECTION 4.4.3) ;
- Cette approche automatisée n'étant pas infaillible, il est recommandé de croiser différentes approches d'analyses et de toujours vérifier manuellement le contenu des *clusters* avant d'arrêter le projet d'annotation : pour faciliter cette revue, il est possible de réaliser une analyse linguistique grâce à la FMC pour identifier les termes caractéristique de chaque *cluster* et les mettre en avant dans le texte (cf. SECTION 4.4.2).

Pour compléter l'étude que nous venons de réaliser, il peut être intéressant de définir un cas d'arrêt des itérations du *clustering* interactif afin de pouvoir prédire quand stopper l'annotation et demander aux experts d'évaluer la pertinence de la base d'apprentissage obtenue. Nous étudions cet aspect dans la prochaine section (cf. hypothèse de rentabilité en SECTION 4.5).

38. À titre d'exemple, prenons *Llama 2* de *Meta* (TOUVRON et al., 2023) : pour l'entraînement, il a fallu 2 000 GPU sur près de 3 millions d'heures (temps GPU cumulé) pour un modèle de 70 millions de paramètres.

4.5 Évaluation de l'hypothèse de rentabilité

Dans les études précédentes, le cas d'arrêt de notre méthodologie d'annotation basée sur le *clustering* interactif était conditionné à la vérité terrain. En effet, nous utilisions un seuil de 90% de **v-measure**, caractérisant une annotation dite "partielle" de la base d'apprentissage. Cependant, une telle référence n'est pas accessible en situation réelle car l'objectif de notre méthode est précisément de la construire cette vérité terrain. Nous devons donc nous intéresser à d'autres moyens pour estimer la rentabilité d'une itération supplémentaire et pouvoir ainsi définir de nouveaux cas d'arrêt pour le *clustering* interactif. Pour cela, nous aimerions vérifier l'hypothèse suivante :

❖ Hypothèse de rentabilité ❖

« Au cours d'une méthodologie d'annotation basée sur le *clustering* interactif, il est possible d'estimer la rentabilité d'une itération supplémentaire de la méthode, et ainsi d'établir des cas d'arrêt indépendant d'une vérité terrain pour obtenir une base d'apprentissage satisfaisante. »

La FIGURE 4.26 illustre cette hypothèse et l'espérance de pouvoir estimer le rapport entre le gain de pertinence obtenu et le coût nécessaire pour l'obtenir.

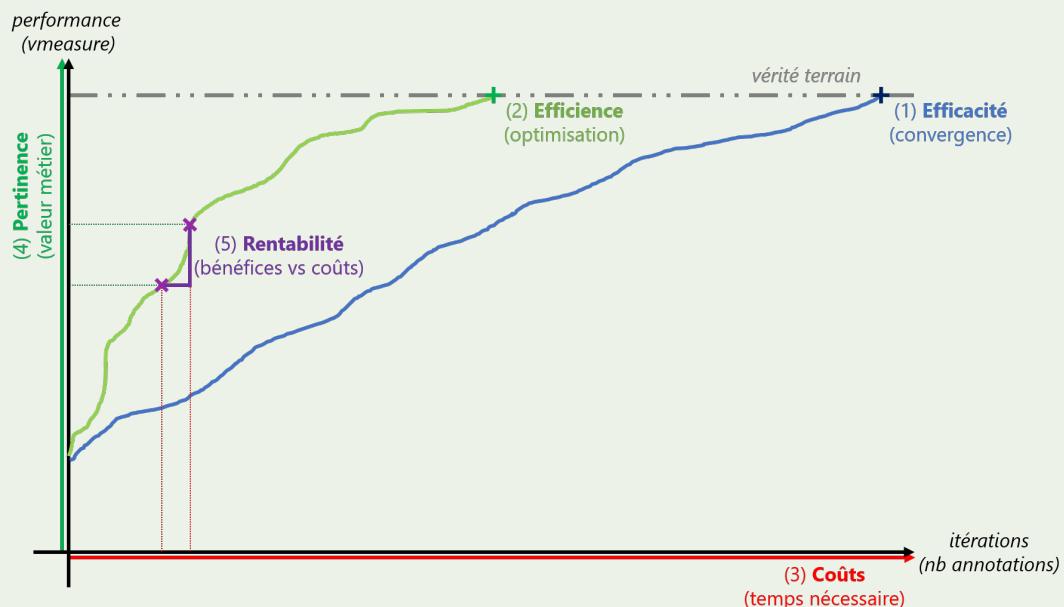


FIGURE 4.26 – Illustration des études réalisées sur le *clustering* interactif (étape 5/6) en schématisant l'évolution de la pertinence (valeur métier évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine), ainsi que la rentabilité de chaque itération de la méthode (rapport entre le gain potentiel de pertinence et le coût à investir).

Afin de vérifier cette hypothèse, nous explorons deux approches :

- l'évolution de l'accord entre l'annotation de l'expert et le *clustering* sur lequel est

- basé l'échantillon d'annotation, permettant d'estimer si la machine doit encore être corrigée par l'annotateur (cf. SECTION 4.5.1) ;
- et l'évolution de la **différence entre deux *clusterings* successifs**, permettant de mesurer s'il y a eu des changements visibles dans le partitionnement des données après l'ajout des dernières contraintes (cf. SECTION 4.5.2).

4.5.1 Étude de l'évolution d'accord entre l'annotation et le *clustering*

Nous cherchons à trouver un cas d'arrêt du *clustering* interactif ne nécessitant pas de comparaison avec une vérité terrain, et notre première intuition concerne l'étude des annotations réalisées. En effet, à chaque itération, l'expert annote un échantillon de contraintes dans le but de confirmer ou de corriger le *clustering* de l'itération précédente. Or, après un nombre suffisant d'itérations, le *clustering* commence à se stabiliser : il devrait donc y avoir davantage d'annotations qui confirment le *clustering* que d'annotations qui le corrigent, puis n'avoir que des accords entre les annotations et le *clustering*. Ainsi, nous allons étudier l'évolution du nombre de contraintes annotées qui approuvent le partitionnement des données obtenu et essayer d'adapter cette analyse en cas d'arrêt pour notre méthode d'annotation.

4.5.1.a Protocole expérimental

⚠️ Attention : Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble.

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.9.

Nous utilisons comme vérité terrain le jeu de données **Bank Cards** (v1.0.0) : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 500 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte**, **carte avalée**, **commande de carte**, ...). Pour plus de détails, consultez l'annexe A.1.

Sur ce jeu de données, nous exécutons une tentative complète³⁹ de la méthode du *clustering* interactif en utilisant notre paramétrage favori⁴⁰ (voir SECTION 4.3), et cette tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions. À chaque itération, un lot de 50 contraintes est sélectionné puis annotés en simulant l'action d'un expert métier, et nous évaluons l'accord entre ces nouvelles annotations et la proposition de partitionnement des données réalisé par le *clustering* à l'itération précédente :

- il y a **accord** lorsqu'une contrainte de deux données issues d'un même *cluster* est annotée **MUST-LINK**, ou lorsqu'une contrainte de deux données issues de deux *clusters* différents est annotée **CANNOT-LINK** (cf. FIGURE 4.27 (1)) ;

39. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

40. Paramétrage favori (atteindre 90% de **v-measure** avec un coût minimal) : prétraitement simple (**prep.simple**), vectorisation **TF-IDF** (**vect.tfidf**), *clustering KMeans* avec modèle **COP** (**clust.kmeans.cop**) et échantillonnage des données les plus proches dans des clusters différents (**sampl.closest.diff**).

Données : jeu de données annotées (vérité terrain)

```

1 pour chaque jeux de données à tester faire
2   initialisation (données) : récupérer les données et la vérité terrain ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   prétraitement : supprimer le bruit dans les données avec prep.simple ;
5   vectorisation : transformer les données en vecteurs avec vect.tfidf ;
6   clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
7   répéter
8     échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
9     simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
10    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
11    rentabilité : calculer l'accord entre l'annotation et le clustering précédent ;
12    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
13    jusqu'à annotation de toutes les contraintes possibles;
14 analyse 1 : afficher l'évolution de l'accord entre annotation et clustering ;
15 analyse 2 : calculer la corrélation entre le score d'accord et le score de performance ;
  Résultat : discussion sur la rentabilité d'après l'accord entre annotation et clustering

```

ALGORITHME 4.9 — *Description en pseudo-code du protocole expérimental de l'étude de l'évolution d'accord entre l'annotation et le clustering.*

- il y a **désaccord** lorsqu'une contrainte de deux données issues d'un même *cluster* est annotée **CANNOT-LINK**, ou lorsqu'une contrainte de deux données issues de deux *clusters* différents est annotée **MUST-LINK** (cf. FIGURE 4.27 (2)).

Nous pouvons ainsi calculer un score d'accord défini par la ratio entre le nombre d'accords et le nombre de contraintes annotées. Pour nous permettre de discuter de l'utilité de ce score pour prédire la stabilisation du *clustering* et ainsi définir un cas d'arrêt de notre méthodologie d'annotation, nous calculons aussi le score de corrélation entre cet accord et la performance obtenu à l'aide d'une vérité terrain (la corrélation *r* de Pearson (KIRCH, 2008) est utilisée).

💡 Idées : Nous concentrons l'étude sur notre paramétrage favori (voir SECTION 4.4.3). Cependant, afin de compléter notre discussion avec d'autres points de comparaison, nous analysons aussi les autres paramétrages implémentés, notamment les meilleurs paramétrages moyens identifiés lors de l'hypothèse d'efficience (voir SECTION 4.2).

ℹ️ Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.5.1.b Résultats obtenus

La FIGURE 4.28 représente l'évolution moyenne du score d'accord entre annotation et *clustering* pour les quatre paramétrages mis en avant lors de nos études. Nous pouvons constater une tendance générale à la croissance de ce score d'accord : pour le paramétrage favori (4), l'accord est plutôt faible au début de la méthode (inférieur à 45% avant l'itération 15), puis devient de plus en plus fort (dépassant les 60%) pour finalement atteindre les 100% vers l'itération 45.

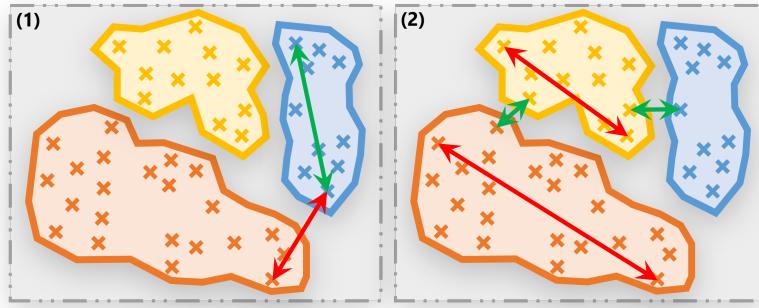


FIGURE 4.27 – Exemples d'accords et de désaccord entre les annotations d'une itération et le résultat du clustering de l'itération précédente. Des contraintes *MUST-LINK* (flèches vertes) et *CANNOT-LINK* (flèches rouges) sont représentées dans deux situations : (1) montre des cas d'accords (*MUST-LINK* dans un même cluster, *CANNOT-LINK* entre deux clusters différents), et (2) montre des cas de désaccords (*MUST-LINK* entre deux clusters différents, *CANNOT-LINK* dans un même cluster).

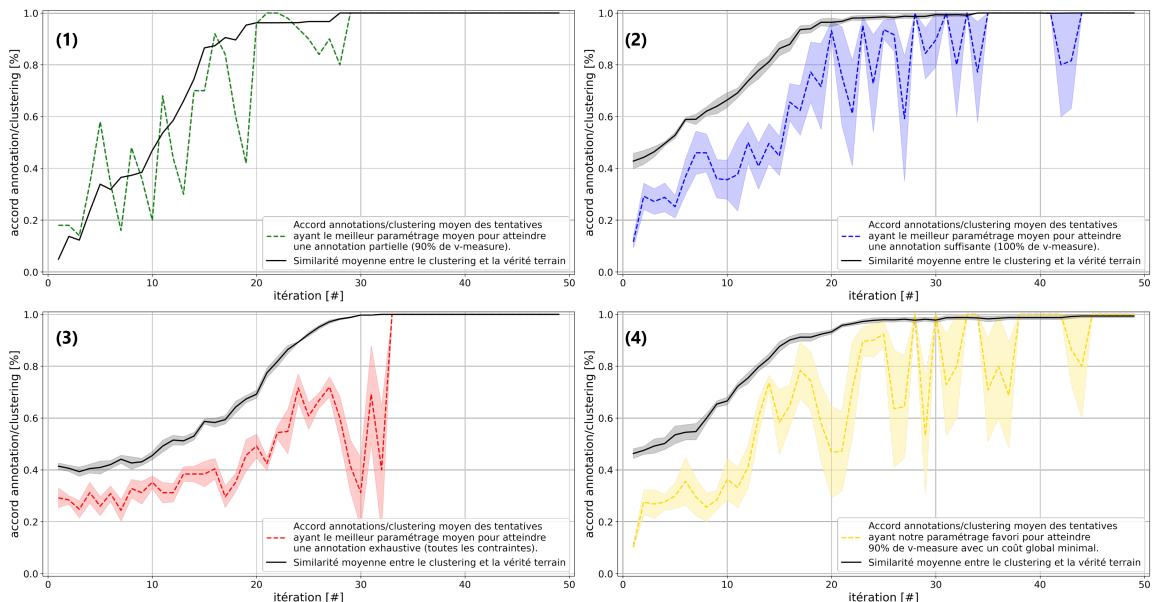


FIGURE 4.28 – Évolution au cours des itérations de l'accord entre l'annotation de contraintes d'un expert et le résultat de clustering sur lequel est basé l'échantillonnage de contraintes. Ces accords sont exprimés grâce à des lots de 50 contraintes annotées. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori. À titre d'information, les courbes en noir représentent l'évolution de la *v-measure* entre le clustering et la vérité terrain.

La TABLE 4.9 contient le score de corrélation entre cet accord et la performance théoriques obtenue grâce à la vérité terrain. Cette corrélation est modérée : 0.49 sur l'ensemble des tentatives,

0.69 sur les tentatives utilisant notre paramétrage favori.

Paramétrage	Corrélation r
Meilleur paramétrage moyen pour une annotation partielle (1)	0.92
Meilleur paramétrage moyen pour une annotation suffisante (2)	0.74
Meilleur paramétrage moyen pour une annotation exhaustive (3)	0.57
Paramétrage favori (4)	0.69
Moyenne des 960 tentatives	0.49

TABLE 4.9 – Score de corrélation r de Pearson entre la performance du clustering obtenu à l'aide d'une vérité terrain (v -measure) et le score d'accord entre annotation et clustering.

Cependant, la tendance constatée est aussi saccadée par de nombreux pics pouvant faire perdre ou gagner jusqu'à 40% d'accord entre deux itérations. Des chutes d'accord peuvent intervenir à des itérations où la similarité du *clustering* avec la vérité terrain est pourtant forte, comme c'est le cas autour des itérations 29 et 36 où l'accord chute de plus de 25% alors que la *v-measure* avec la vérité terrain est constamment au dessus de 95%.

Les autres paramétrages représentés dans (1), (2) et (3) comportent des tendances similaires (corrélation forte mais variations soudaines d'accord, chute d'accords malgré des *clustering* aux performances élevées, ...).

4.5.1.c Discussion

Dans cette étude, nous avons analysé l'évolution de l'accord entre les annotations et le partitionnement de données proposé par un *clustering* dans l'espoir de définir un cas d'arrêt de notre méthodologie d'annotation qui soit indépendant d'une vérité terrain pré-établie. Cependant, en considérant les résultats obtenus, ce score d'accord ne semble pas répondre à cette objectif.

Tout d'abord, malgré une corrélation acceptable avec la performance théorique du *clustering* (moyenne à 0.49, voir TABLE 4.9), l'évolution du score d'accord reste instable. En effet, les nombreuses variations et saccades rendent toute analyse de rentabilité difficile voire impossible, ce qui ne permet pas de définir un cas d'arrêt pour notre méthode d'annotation.

Exemples : Concernant l'évolution du paramétrage favori (FIGURE 4.28 (4)), nous ne pouvons pas précisément définir à partir de quelle itération les résultats semblent intéressant car le score d'accord oscille longuement entre 50% et 100% avec des pics de plus de 25% entre deux itérations.

Notes de l'auteur : Après réflexion, ce score d'accord est probablement infructueux à cause du fonctionnement même de notre méthode, dont l'objectif est de corriger le partitionnement des données en utilisant un minimum de contraintes. En effet, dans le cadre de l'optimisation des paramètres réalisée en SECTION 4.2, nous avons retenu dans notre paramétrage favori la sélection des contraintes les plus proches entre deux *clusters* différents (`samp.closest.diff`) : cette sélection permet ainsi de décrire efficacement l'emplacement des frontières de *clusters*.

Or, cet échantillonnage reste une méthode non-supervisée : aux premières itérations, les contraintes sélectionnées ont de bonnes chances de mettre en avant une frontière mal positionnée, mais au fur et à mesure que des contraintes s'ajoutent, les nouvelles contraintes ont moins de chances de trouver des bordures de *clusters* qui ne soient pas encore caractérisées. De ce fait, il se peut que les dernières sélections n'identifient aucune nouvelle frontière, qu'elles se concentrent sur des frontières déjà bien positionnées ou déjà décrites par d'autres contraintes, ou qu'elles nécessitent plusieurs itérations pour caractériser des frontières complexes (le comportement des autres méthodes de sélections représentées en FIGURE 4.28 peut être illustré par des raisonnements similaires). L'ensemble de ces cas de figures peut ainsi expliquer les nombreuses saccades dans l'évolution du score d'accord : tantôt la sélection semble pertinente, tantôt la sélection semble inutile.

Pour aller plus loin, nous pouvons aussi critiquer le score de corrélation qui ne semble pas montrer de lien fort entre les performances théoriques et les accords calculés, tant sur l'ensemble des tentatives que pour le paramétrage favori. Il est même rare d'observer des chutes importantes d'accords qui soient accompagnées d'une variation significative de **v-measure** avec la vérité terrain. Au final, ce score d'accord n'est donc pas vraiment représentatif de la rentabilité d'une itération ou de l'évolution de la pertinence du *clustering*.

Notes de l'auteur : Pour expliquer cette absence de corrélation, il est possible que l'analyse des annotations réalisées ait été une idée infructueuse : les 50 contraintes annotées peuvent peut-être exprimer un désaccord avec le précédent *clustering*, mais ce n'est pas pour autant que l'ajout de ces nouvelles contraintes impacte significativement la pertinence globale du partitionnement des données.

En conclusion, le **score d'accord entre l'annotation courante et le clustering précédent n'est pas adéquat pour estimer un cas d'arrêt de notre méthode d'annotation**, principalement car il est trop instable et qu'il ne représente pas bien les bénéfices obtenus à chaque itération. Ainsi, comme l'analyse de l'annotation n'est pas fructueuse, nous nous tournons vers l'analyse basée sur les différences entre deux résultats de *clustering*.

4.5.2 Étude de l'évolution de la différence entre deux *clusterings* consécutifs

Nous venons de conclure que l'analyse de l'accord entre l'annotation et le partitionnement des données ne permet pas d'estimer la rentabilité d'une itération de notre méthode d'annotation. Parmi les explications possibles, nous avons mis en cause l'analyse du lot de contraintes annotées : en effet, ce n'est pas parce que l'annotation de contraintes est en désaccord avec le précédent partitionnement des données que les correctifs associés auront un impact significatif sur le prochain partitionnement. Ainsi, nous voulons analyser l'évolution de la différence entre deux *clusterings* successifs : en effet, si une itération apporte des correctifs ayant un impact, alors il devrait y avoir des différences visibles entre les deux itérations de *clustering*.

4.5.2.a Protocole expérimental

⚠️ Attention : Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble.

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.10.

Données : jeu de données annotées (vérité terrain)

```

1 pour chaque jeux de données à tester faire
2   initialisation (données) : récupérer les données et la vérité terrain ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   prétraitement : supprimer le bruit dans les données avec prep.simple ;
5   vectorisation : transformer les données en vecteurs avec vect.tfidf ;
6   clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
7   répéter
8     échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
9     simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
10    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
11    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
12    rentabilité : calculer la différence entre les deux précédents clusterings ;
13   jusqu'à annotation de toutes les contraintes possibles ;
14 analyse 1 : afficher l'évolution de la différence entre deux clustering consécutifs ;
15 analyse 2 : calculer la corrélation entre le score de différence et le score de performance ;
Résultat : discussion sur la rentabilité d'après la différence entre clusterings
```

ALGORITHME 4.10 – Description en pseudo-code du protocole expérimental de l'étude de l'évolution de la différence entre deux clustering consécutifs.

Nous nous appuyons sur le même protocole que l'expérience précédente (cf. SECTION 4.5.1) : nous utilisons comme vérité terrain le jeu de données **Bank Cards** (v1.0.0), nous réalisons 5 tentatives complètes de la méthode du *clustering* interactif en utilisant notre paramétrage favori (voir SECTION 4.3), et nous simulons l'*annotation* par un expert d'un lot de 50 contraintes à chaque itération.

Cependant, au lieu de calculer un score d'accord entre *annotation* et *clustering*, nous estimons la différence entre le *clustering* précédent et le *clustering* obtenu grâce aux dernières annotations. Cette différence entre deux *clustering* X et Y est obtenue par la formule $1 - v\text{-measure}(X, Y)$ où la **v-measure** caractérise la ressemblance entre deux partitionnements des données (ROSENBERG et HIRSCHBERG, 2007). Pour nous permettre de discuter de l'utilité de ce score pour prédire la stabilisation du *clustering* et ainsi définir un cas d'arrêt de notre méthodologie d'*annotation*, nous calculons aussi le score de corrélation entre cette différence et la performance obtenue à l'aide d'une vérité terrain (la corrélation **r** de Pearson (KIRCH, 2008) est utilisée).

💡 Idées : Comme précédemment, nous concentrons l'étude sur notre paramétrage favori (voir SECTION 4.4.3). Cependant, afin de compléter notre discussion avec d'autres points de comparaison, nous analysons aussi les autres paramétrages implémentés, no-

tamment les meilleurs paramétrages moyens identifiés lors de l'hypothèse d'efficience (voir SECTION 4.2).

i Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.5.2.b Résultats obtenus

La FIGURE 4.29 représente l'évolution moyenne du score de différence entre deux *clusterings* pour les quatre paramétrages mis en avant lors de nos études. Nous pouvons constater une tendance générale à la décroissance vers 0% de ce score de différence : pour le paramétrage favori, la différence moyenne entre deux *clustering* est initialement comprise entre 25% et 35% jusqu'à l'itération 10, elle chute ensuite pour être inférieure à 5% après l'itération 20, et elle termine en oscillant très légèrement ($\pm 1\%$) autour de 0% jusqu'à la fin des annotations.

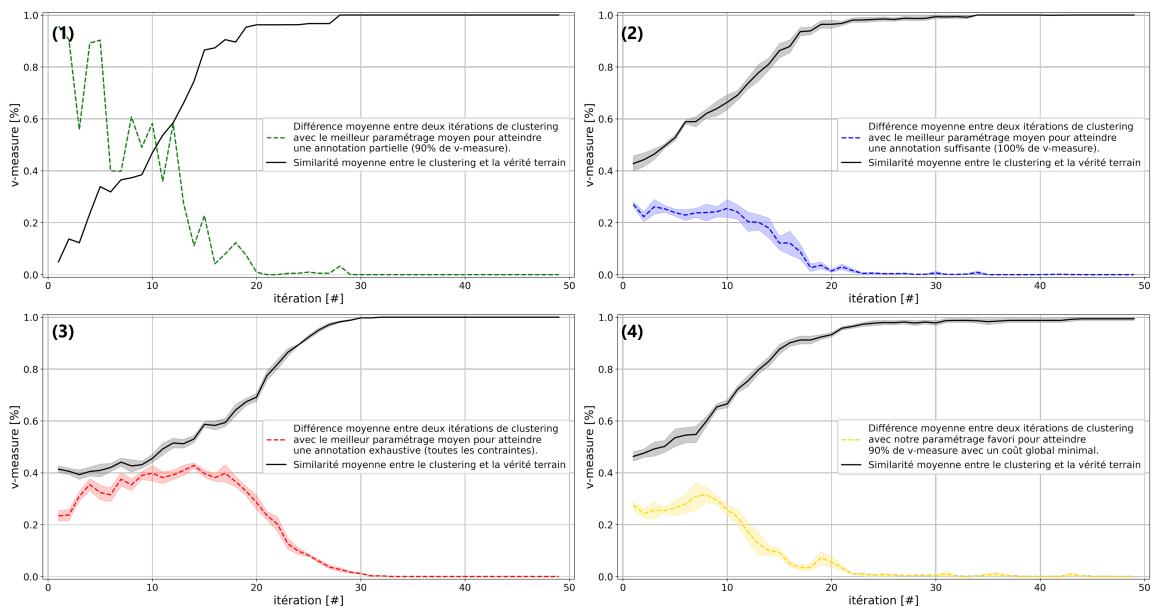


FIGURE 4.29 – Évolution de la différence de résultats entre deux itérations de clustering. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori. À titre d'information, les courbes en noir représentent l'évolution de la *v-measure* entre le clustering et la vérité terrain.

La TABLE 4.10 contient le score de corrélation entre cette différence et la performance théoriques obtenue grâce à la vérité terrain. Cette corrélation est forte : 0.75 sur l'ensemble des tentatives, 0.93 sur les tentatives utilisant notre paramétrage favori. La FIGURE 4.29 confirme cette corrélation :

- un score de *v-measure* avec la vérité terrain proche de 100% est accompagné d'un score de

différence proche de 0% (après l'itération 20 pour (1), après l'itération 20 pour (2), après l'itération 30 pour (3) et après l'itération 22 pour (4)) ;

- une croissance de performance est généralement accompagnée d'un score non nul de différence (voir (2) et (4) entre les itérations 0 et 20), et plusieurs pics de performance sont accompagnés de scores forts de différence (particulièrement visible sur (1) vers l'itération 5 et entre les itérations 10 et 15) ;
- il est toutefois à noter que l'inverse n'est pas vrai : un score non nul de différence n'accompagne pas forcément une croissance de performance, mais peut simplement caractériser un changement de partitionnement, comme c'est le cas dans (3) entre les itérations 0 et 10 où des modifications ont lieu (score de différence non nul) mais où la performance par rapport à la vérité terrain stagne.

Paramétrage	Corrélation
Meilleur paramétrage moyen pour une annotation partielle (1)	0.96
Meilleur paramétrage moyen pour une annotation suffisante (2)	0.92
Meilleur paramétrage moyen pour une annotation exhaustive (3)	0.85
Paramétrage favori (4)	0.93
Moyenne des 960 tentatives	0.75

TABLE 4.10 – Score de corrélation r de Pearson entre la performance du clustering obtenu à l'aide d'une vérité terrain ($v\text{-measure}$) et le score de différence entre deux clusterings consécutifs.

Les autres paramétrages représentés dans (1), (2) et (3) comportent des tendances similaires (décroissance générale, forte corrélation avec la performance théorique) à quelques détails ((1) commence avec des scores de différence très forts avant décroître avec de nombreux pics; (3) croît légèrement avant d'entamer sa décroissance, ...).

4.5.2.c Discussion

Dans cette étude, nous avons analysé l'évolution du score de différence entre deux itérations de *clustering* dans l'espoir de définir un cas d'arrêt de notre méthodologie d'annotation qui soit indépendant d'une vérité terrain pré-établie.

Tout d'abord, nous pouvons affirmer qu'il y une forte corrélation entre l'évolution de ce score de différence et l'évolution du score de performance (voir TABLE 4.10 : r moyen de 0.75 ; r supérieur à 0.85 pour les paramétrages mis en avant). Cette corrélation est confirmée visuellement grâce à la FIGURE 4.29 : plus les différences entre *clusterings* sont faibles, plus les performances des *clusterings* sont fortes.

Un point d'attention est toutefois à retenir : une modification du partitionnement des données n'entraîne pas forcément un gain de performance (voir (3) entre les itérations 0 et 10 et (4) entre les itérations 0 et 8). Nous ne pouvons donc pas conclure que l'analyse de la différence entre eux itération de *clustering* permet de caractériser totalement la rentabilité d'une itération.

Cependant, nous pouvons tout de même nous servir de ce score pour définir un cas d'arrêt pour notre méthodologie d'annotation lorsque la différence entre deux *clusterings* est faible. Pour cela, il nous suffit de fixer un seuil bas du score de différence en dessous duquel il n'est plus rentable de faire de nouvelles itérations de la méthode car les performances devraient être suffisantes. Une analyse manuelle ou semi-manuelle (voir hypothèse de pertinence en SECTION 4.4)

reste nécessaire pour confirmer la valeur métier du résultat obtenu.

💡 Idées : Si nous restons sur notre seuil théorique de 90% de **v-measure** (voir SECTION 4.2) et que nous nous basons sur la FIGURE 4.29 (4), nous pouvons visuellement fixer ce seuil autour de 5% de différences. Le réglage fin de ce seuil pourra être le sujet de futures analyses complémentaires.

En conclusion, le score de différences entre deux résultats de *clustering* semble être un bon indicateur pour estimer un cas d'arrêt de notre méthodologie d'annotation, et nous proposer d'utiliser un seuil de 5% pour implémenter ce cas d'arrêt.

4.5.3 Mise en commun des stratégies d'évaluation de la rentabilité d'une itération de la méthode et définition d'un cas d'arrêt indépendant d'une vérité terrain.

📘 Points à retenir : Au cours de cette étude de rentabilité, nous avons pu voir que :

- ☒ l'analyse du score d'accord entre l'annotation courante et le *clustering* précédent ne permet pas d'estimer la rentabilité d'une itération, ni de définir un cas d'arrêt de notre méthodologie d'annotation (cf. SECTION 4.5.1) ;
- ☑ l'analyse des différences entre deux itérations de *clusterings* est une approche prometteuse pour estimer la rentabilité d'une itération, bien qu'une modification significative entre deux résultats de *clustering* n'implique pas forcément un gain de performance (les deux *clustering* peuvent avoir une **v-measure** équivalente avec la vérité terrain) ;
- ☑ l'usage de différences entre deux itérations de *clusterings* permet de définir un cas d'arrêt de notre méthodologie d'annotation : si les différences sont faibles (par exemple : inférieures à 5%), alors les performances stagnent ou plafonnent, donc il peut être intéressant d'interrompre le *clustering* interactif après avoir vérifier la manuellement pertinence des résultats obtenus (cf. SECTION 4.5.2) et SECTION 4.4.4).

Pour terminer nos différentes analyses, il convient maintenant d'anticiper la présence de différences d'annotation. En effet, nous avons fait jusqu'à présent l'hypothèse que l'annotateur ne se trompe jamais et que deux annotateurs n'ont jamais de désaccords, mais cette hypothèse forte n'est pas toujours vérifiée en pratique. Pour estimer l'impact de ces incohérences d'annotation, nous devons donc réaliser une analyse de robustesse de notre méthode d'annotation : celle-ci sera réalisée en SECTION 4.6.

4.6 Évaluation de l'hypothèse de robustesse

Dans les précédentes études, nous avons presque toujours analysé le *clustering* interactif en supposant que l'annotateur connaît parfaitement le domaine traité par le jeu de données et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Bien entendu, cette hypothèse forte n'est pas toujours vérifiée en situation réelle : l'interprétation du langage peut contenir certaines ambiguïtés, l'opérateur peut faire des erreurs d'inattention, et deux annotateurs peuvent avoir des avis contraire sur un même sujet. Or, comme notre méthode d'annotation est itérative, elle est a priori sensible aux dérives fonctionnement liées à ce type de contradictions. Dans cette section, nous nous intéressons donc à la robustesse du *clustering* interactif en présence d'incohérences dans les contraintes et aux moyens de les contrer. Pour cela, nous aimerions donc vérifier l'hypothèse suivante :

💡 Hypothèse de robustesse 💡

« Au cours d'une méthodologie d'annotation basée sur le *clustering* interactif, il est possible d'estimer le taux d'incohérences dans les contraintes ainsi que leur impact sur les résultats de la méthode. »

La FIGURE 4.30 illustre cette hypothèse et l'espoir de estimer l'impact de différences d'annotations sur le nombre d'itérations de la méthode.

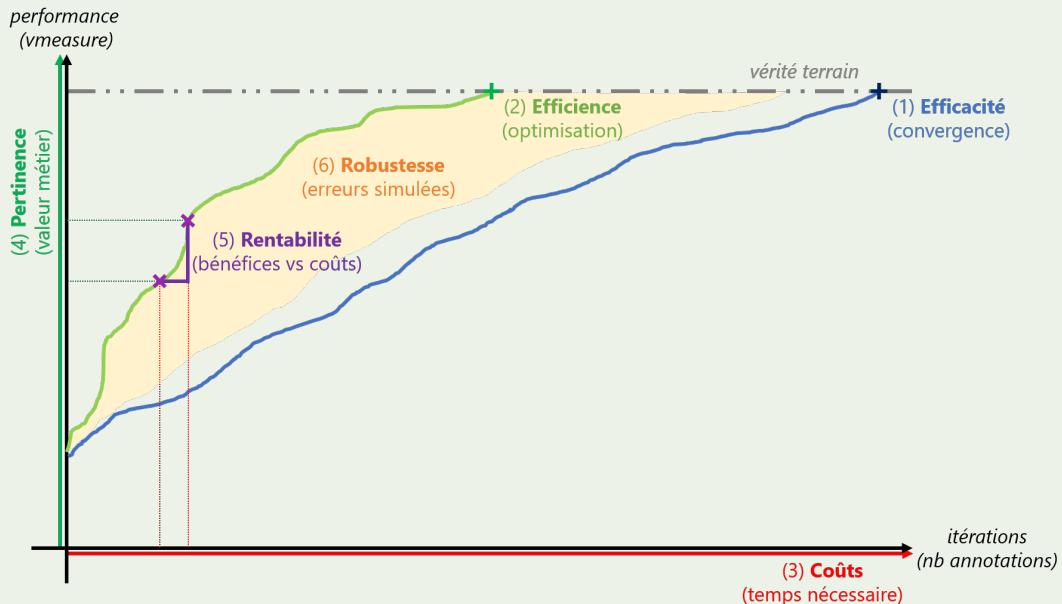


FIGURE 4.30 – Illustration des études réalisées sur le *clustering* interactif (étape 6/6) en schématisant l'évolution de la pertinence (valeur métier évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps nécessaire à l'expert métier et à la machine), ainsi que les marges d'erreurs représentant l'impact de différences d'annotation sur le nombre d'itérations nécessaire à la méthode.

Afin de vérifier cette hypothèse, nous organisons trois expériences :

- une étude de cas d'un **score inter-annotateurs** obtenu lors d'une annotation de contraintes en situation réelle avec plusieurs opérateurs, permettant d'estimer une borne maximale du taux de désaccords d'annotation pour les autres études (cf. SECTION 4.6.1) ;
- une étude de l'**impact d'une erreur d'annotation** et de l'**intérêt de la corriger**, en simulant l'insertion d'erreurs d'annotation et en mesurant la similarité du *clustering* obtenu avec la vérité terrain (cf. SECTION 4.6.2) ;
- une étude de l'**impact de la subjectivité de l'annotation sur la divergence des résultats de clustering**, en simulant l'insertion de différences d'annotation et en mesurant la similarité entre les clustering obtenus (cf. SECTION 4.6.3).

4.6.1 Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle

Afin d'affiner le champ de recherche de nos futures études, nous voulons analyser le score d'accord inter-annotateurs calculé lors d'une expérience d'annotation de contraintes par plusieurs experts métiers en situation réelle. Pour cela, nous reprenons l'expérience de la SECTION 4.3.1 visant à estimer le temps moyen d'annotation d'un lot de contraintes, et nous réutilisons ces résultats pour en estimer l'accord inter-annotateurs. Comme l'objectif de cette précédente étude n'était pas d'étudier la qualité des annotation, aucun guide ni aucune règles d'annotation précises n'avaient été fournies : nous espérons donc pourvoir **estimer une borne maximale grossière du désaccord entre annotateurs** lors de l'utilisation de notre méthode, permettant ainsi d'affiner notre discussion.

4.6.1.a Protocole expérimental

⚠️ Attention : Pour l'étude d'annotation en SECTION 4.3.1, nous avions supposé que les annotateurs de l'expérience connaissaient parfaitement le domaine traité dans le jeu de données, et qu'ils sont capables de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Afin de pourvoir faire cette hypothèse forte, et ainsi limiter les bruits dans l'analyse des résultats, le jeu de données choisi devait traiter d'un sujet de culture générale (ne nécessitant donc pas de connaissance particulière) et des réviseurs avaient supprimer en amont et d'un commun accord les données trop spécifiques ou trop ambiguës.

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.11.

Concernant la précédente expérience d'annotation en situation réelle, nous avions procédé en plusieurs étapes. D'abord, il fallait choisir un jeu de données approprié : pour valider notre hypothèse forte sur les compétences de nos annotateurs, nous cherchions un jeu de données traitant d'un sujet de culture général. Pour cette expérience, nous avions donc choisi MLSUM : une collecte d'articles de journaux, classés par catégorie de publication et décrits par leur titre et leur résumé. Nous nous intéressions ici à la tâche de classification d'un titre d'article en fonction de sa catégorie de publication. Comme certains titres pouvaient porter à confusion (un titre d'article n'étant pas toujours explicite sur son contenu), deux réviseurs (*une Data Scientist et moi-même*) furent chargés de choisir les données les plus explicites sur un échantillon d'un millier de données représentatives des catégories les plus communes. L'échantillon résultant, noté **MLSUM FR Train**

Données : jeu de données annotées (vérité terrain)

Entrées : plusieurs réviseurs, plusieurs annotateurs

1 initialisation : définir et revoir le jeu de données entre réviseurs ;

2 échantillonnage : sélectionner une base de contraintes équilibrée ;

3 pour chaque annotateur faire

4 tant que la base de contraintes n'a pas été entièrement annotée faire

5 annotation : annoter une partie des contraintes ;

6 revue : revue des contraintes en conflits d'annotation ;

Résultat : modélisation du score inter-annotateurs sur le lot de contraintes

ALGORITHME 4.11 – Description en pseudo-code du protocole expérimental de l'étude du score inter-annotateurs d'annotation d'un lot de contraintes par plusieurs experts métiers en situation réelle.

Subset (v1.0.0-schild), est composé de 744 titres d'articles rédigés en français et répartis en 14 classes (*économie, sport, ...*). Pour plus de détails, consultez l'annexe A.2.

À partir de ces données, nous avions sélectionné un lot de 400 contraintes à annoter. Pour faciliter l'analyse, l'échantillonnage fut un tirage aléatoire équilibré d'après la vérité terrain en 200 MUST-LINK et en 200 CANNOT-LINK.

Ensuite, un groupe de 3 annotateurs ont annoté la sélection des 400 contraintes en plusieurs sessions. Les directives données aux opérateurs étaient les suivantes :

- **Contexte de l'opérateur** : « *Vous êtes des experts de la presse et de l'actualité ; Vous voulez classer des articles dans des catégories en fonction de leur titre ; Vous ne savez pas précisément quelles catégories vous allez utiliser pour classer vos articles ; Mais vous savez caractériser la similitude de deux articles* » ;
- **Contexte sur le jeu de données** : « *Le thème sont les catégories d'articles de presse ; La vérité terrain contient entre 10 et 20 catégories parmi les plus communes de la presse ; La vérité terrain contient entre 30 et 100 articles par catégorie ; Vous pouvez regarder le jeu de données non annoté autant que vous le voulez (disponible dans l'onglet TEXTS de l'application)* » ;
- **Consignes d'annotations** : « *Faites des séries de 15 minutes minimum pour avoir de la régularité ; Si possible, isolez-vous pour ne pas être dérangé et ne pas fausser les résultats ; Pour chaque série, notez le temps et le nombre de contraintes annotés ; Si vous ne savez pas quoi annoter (trop ambigu, vocabulaire inconnu, ...), passez au suivant sans annoter (vous êtes sensés être des experts de la presse !)* ».

Pour réaliser l'annotation, les opérateurs eurent accès à l'application web développée au cours de ce doctorat. Des captures d'écran sont disponibles en FIGURE 4.11 et FIGURE 4.12. Une description plus détaillée de l'application et de ses fonctionnalités est disponible en ANNEXE C.4. Il est à noter que l'autre réviseur a aussi participé à l'annotation de ces contraintes : nous avons retenu ses résultats, mais nous les analyserons séparément du groupe d'annotateurs.

Pour cette étude, nous allons calculer le score d'accord inter-annotateurs global et deux à deux. Pour ce faire, nous utilisons l' α de *Krippendorff*⁴¹ (KRIPPENDORFF, 2004) implémenté dans

41. Choix de l' α de *Krippendorff* : L'utilisation de κ de *Cohen* (LANDIS et KOCH, 1977) est aussi fréquemment utilisée, mais elle n'est pas adaptée pour plus de deux opérateurs ni pour manipuler des absences d'annotations.

la librairie `simplifiedorff`⁴² (PERRY, 2021). Nous rappelons qu'un accord est considéré comme *faible* si $\alpha < 0.667$, *acceptable* si $0.667 \leq \alpha < 0.8$, *fort* si $0.8 \leq \alpha < 1.0$ et *parfait* si $\alpha = 1.0$. De plus, un score α négatif représente une opposition d'accord.

i Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.6.1.b Résultats obtenus

Durant cette expérience, 4 opérateurs ont participé à l'annotation de 400 contraintes issues d'un tirage aléatoire équilibré d'après la vérité terrain en 200 MUST-LINK et en 200 CANNOT-LINK. Ces opérateurs travaillent tous dans un service informatique dédié à l'entraînement et l'amélioration de solutions de *Machine Learning* et sont répartis de la manière suivante :

- 2 femmes, 2 hommes ;
- 4 personnes entre 20 et 30 ans ;
- 4 *Data Scientist* ;
- 1 personne ayant révisé le jeu de données, 3 le découvrant pour la première fois.

Par manque de disponibilités, 4 annotateurs n'ont que partiellement réalisé leur tâche : nous avons toutefois intégré leurs participations car elles contenaient au minimum 150 annotations.

La TABLE 4.11 expose les accords des opérateurs (1 réviseur, 3 annotateurs) par rapport à la vérité terrain. Nous pouvons constater les points suivants :

- le réviseur possède un accord *fort* avec la vérité terrain ($\alpha = 0.892$), confirmant sa connaissance de celle-ci ;
- les autres annotateurs, ne connaissant pas la vérité terrain, ont tous un accord *acceptable* avec celle-ci $\alpha \geq 0.685$; d'après le taux d'accord brut, le pourcentage de désaccords d'annotations concernent entre 13% et 16% de contraintes pour chaque annotateur ;
- malgré ces désaccords, l'accord inter-opérateurs global par rapport à vérité terrain reste *acceptable* ($\alpha = 0.697$ et $\alpha = 0.735$).

La TABLE 4.12 expose les accords inter-opérateurs (1 réviseur, 3 annotateurs). Nous pouvons constater les points suivants :

- les accords deux à deux sont concernent à chaque fois au moins 80.25% des contraintes ;
- un seul couple d'opérateurs est considéré comme ayant un accord *faible* ($\alpha = 0.597 < 0.667$) ;
- les autres annotateurs, ne connaissant pas la vérité terrain, ont tous un accord *acceptable* avec celle-ci $\alpha \geq 0.685$; d'après le taux d'accord brut, le pourcentage de désaccords d'annotations concernent entre 13% et 16% de contraintes pour chaque annotateur ;
- malgré ces désaccords, l'accord inter-opérateurs global reste *acceptable* ($\alpha = 0.669$ et $\alpha = 0.715$).

42. `simplifiedorff` : <https://pypi.org/project/simplifiedorff/>

Opérateurs	Accord avec la vérité terrain		
	Accord brut	α Krippendorff	Interprétation α
1 (réviseur)	94.75%	0.892	<i>fort</i>
7 (annotateur)	87.50%	0.750	<i>acceptable</i>
9 (annotateur)	84.25%	0.685	<i>acceptable</i>
12 (annotateur)	87.00%	0.737	<i>acceptable</i>
Annotateurs 7, 9 et 12	72.00%	0.697	<i>acceptable</i>
Opérateurs 1, 7, 9 et 12	71.75%	0.735	<i>acceptable</i>

TABLE 4.11 – Score d'accord avec la vérité terrain des 4 opérateurs (1 réviseur, 3 annotateurs) sur un lot commun de 400 contraintes (200 *MUST-LINK*, 200 *CANNOT-LINK*). L'accord brut représente le pourcentage de contraintes ayant la même annotation et l'accord α représente la mesure de Krippendorff. Les numéros d'opérateurs correspondent à leur identifiants leur de l'expérience.

Opérateurs	Accord inter-annotateurs		
	Accord brut	α Krippendorff	Interprétation α
1 (réviseur) et 7 (annotateur)	91.50%	0.825	<i>fort</i>
1 (réviseur) et 9 (annotateur)	86.00%	0.711	<i>acceptable</i>
1 (réviseur) et 12 (annotateur)	89.50%	0.780	<i>acceptable</i>
7 (annotateur) et 9 (annotateur)	85.75%	0.714	<i>acceptable</i>
7 (annotateur) et 12 (annotateur)	85.25%	0.699	<i>acceptable</i>
9 (annotateur) et 12 (annotateur)	80.25%	0.597	<i>faible</i>
Annotateurs 7, 9 et 12	75.50%	0.669	<i>acceptable</i>
Opérateurs 1, 7, 9 et 12	74.00%	0.715	<i>acceptable</i>

TABLE 4.12 – Score d'accord inter-opérateurs (1 réviseur, 3 annotateurs) sur un lot commun de 400 contraintes (200 *MUST-LINK*, 200 *CANNOT-LINK*). L'accord brut représente le pourcentage de contraintes ayant la même annotation et l'accord α représente la mesure de Krippendorff. Les numéros d'opérateurs correspondent à leur identifiants leur de l'expérience.

4.6.1.c Discussion

L'objectif de cette étude est l'estimation du taux de désaccords d'annotation qui peuvent apparaître en utilisant notre méthode. Pour cela, nous avons réutilisé les annotations réalisées dans une précédente expérience (voir l'étude du temps d'annotation en SECTION 4.3.1), en espérant ainsi déterminer une borne maximale de ce désaccord entre annotateurs.

Pour rappel, nous avions fait l'hypothèse que les opérateurs étaient des experts du domaine qu'ils annotent. Afin de valider cette hypothèse, nous avions choisi une vérité terrain traitant d'un sujet de culture générale (ici : la presse) et deux réviseurs avaient revu cette vérité terrain en amont dans le but de supprimer au mieux les données ambiguës. Cette hypothèse est a priori valable en considérant que les opérateurs ont eu de bons scores d'accord avec la vérité terrain : un accord *fort* pour le réviseur ($\alpha = 0.892$), et des accords *acceptables* pour les annotateurs ($\alpha \geq 0.685$).

Cependant, nous constatons aussi que nos consignes d'annotations n'étaient probablement pas assez explicites. En effet, aucun annotateur n'a d'accord *fort* avec la vérité terrain ($\alpha < 0.750$), et

le score d'accord inter-opérateurs des trois annotateurs est simplement *acceptable*. Nous pouvons donc assimiler cette expérience à un projet d'annotation dont les règles sont légèrement ambiguës, empêchant ainsi d'avoir un accord *fort* entre les annotateurs.

Ainsi, nous utilisons les taux de désaccords bruts pour affiner notre champ de recherche pour notre prochaines simulations d'erreurs (voir SECTION 4.6.2) et de désaccords (voir SECTION 4.6.3) :

- Sur la base des résultats rapportés dans la TABLE 4.11, nous observons un accord brut avec la vérité terrain sur au moins 84.25% des contraintes annotées, c'est-à-dire que le désaccord maximal concerne au plus 15.75% des annotations ;
- Sur la base des résultats rapportés dans la TABLE 4.12, nous observons un accord brut inter-annotateurs sur au moins 80.25% des contraintes annotées, c'est-à-dire que le désaccord maximal concerne au plus 19.75% des annotations .

Par conséquent, et afin de prendre en compte ces résultats, **nous décidons de considérer 25% comme une borne maximale des taux d'erreurs de désaccords dans nos futures simulations.**

 **Notes de l'auteur :** Il est à noter qu'un taux d'erreurs ou de désaccords de 25% semble toutefois considérable pour une simple annotation binaire. En effet, nous avons pu voir dans la TABLE 4.12 qu'un taux de désaccords de 19.75% étaient déjà considéré comme représentatif d'un accord *faible*. De ce fait, un taux d'accord encore plus faible pourrait plutôt être imputé à une mauvaise organisation du projet d'annotation (*par exemple avec des opérateurs non formés à la labellisation ou des règles d'annotation minimalistes*).

Dans les deux prochaines études, nous allons tout de même simuler des erreurs et des désaccords jusqu'à 25%, mais nous garderons à l'esprit que des désaccords de cette ordre de grandeur sont plutôt caractéristiques d'un problème de gestion de projet.

4.6.2 Étude de l'impact d'une erreur d'annotation et l'intérêt de la corriger

Dans cette seconde étude, nous cherchons à estimer la robustesse du *clustering* interactif en nous intéressant plus particulièrement aux erreurs d'annotation d'un seul opérateur. En effet, comme nous l'avons vu en SECTION 2.3.3.B, des **différences de comportements intra-annotateur** peuvent être observées au cours de la labellisation, ces dernières s'apparentant à des erreurs d'inattention ou des contradictions. Nous allons ici simuler de telles erreurs dans l'annotation de contraintes dans le but :

- d'observer leur détection par le gestionnaire de contraintes ;
- d'évaluer la perte de performances par rapport à la vérité terrain si les contradictions détectées ne sont pas corrigés.

4.6.2.a Protocole expérimental

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.12.

Nous utilisons comme vérité terrain le jeu de données **Bank Cards** (v1.0.0) : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire.

Données : jeu de données annotées (vérité terrain)

Entrées : liste de stratégies de correction de conflits et de taux d'erreurs à insérer

```

1 pour chaque stratégie de correction et taux d'erreurs à insérer faire
2   initialisation (données) : récupérer les données et la vérité terrain ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   prétraitement : supprimer le bruit dans les données avec prep.simple ;
5   vectorisation : transformer les données en vecteurs avec vect.tfidf ;
6   clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
7   évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
8   répéter
9     échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
10    choix des erreurs : définir les contraintes erronées ;
11    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
12    si stratégie de correction naïve alors
13      intégration naïve : ajouter les nouvelles contraintes au gestionnaire de
        contraintes, et ignorer les conflits avec le gestionnaire de contraintes ;
14    sinon si stratégie avec correction alors
15      intégration corrective : ajouter les nouvelles contraintes au gestionnaire de
        contraintes, et ré-annoter les annotations en conflit ;
16    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
17    évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
18  jusqu'à annotation de toutes les contraintes possibles;
19  analyse locale : afficher l'évolution de la similarité entre les clustering de la
    tentative courante et la vérité terrain ;
20 analyse générale : déterminer l'impact des stratégies de correction en fonction des
    taux d'erreurs insérées ;
Résultat : discussion sur l'impact des erreurs et l'intérêt de les corriger

```

ALGORITHME 4.12 – Description en pseudo-code du protocole expérimental de l'étude d'impact d'une erreur d'annotation et l'intérêt de la corriger.

Il est composé de 500 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte, carte avalée, commande de carte, ...**). Pour plus de détails, consultez l'annexe A.1.

Sur ce jeu de données, nous exécutons une tentative complète⁴³ de la méthode du *clustering* interactif en utilisant notre paramétrage favori⁴⁴ (voir SECTION 4.3). Toutefois, contrairement aux précédents expériences, nous allons ajouter un pourcentage de contraintes erronées à chaque itération pour simuler les variations de comportement de l'annotateur :

- Le taux d'erreurs insérées, variant de 0% à 25%⁴⁵ par pas de 5%, reste fixe tout au long d'une même tentative de notre méthode : nous pouvons ainsi analyser l'impact d'un taux d'erreur fixe sur les résultats au courant des itérations ;
- Les contraintes erronées à insérer sont tirées aléatoirement parmi le lot de contraintes qui aurait été échantillonnées au cours d'une tentative sans erreurs : ainsi, nous pouvons comparer itération par itération toutes ces simulations car elles partagent la même base de contraintes (aux valeurs de **MUST-LINK** et **CANNOT-LINK** près) ;

Puisque nous introduisons des erreurs d'annotations, des conflits peuvent apparaître dans le gestionnaire de contraintes. Pour rappel, un conflit est détecté dans le cas où l'ajout d'une nouvelle contrainte annotée contredit ce qui a été précédemment déduit grâce aux propriétés de transitivité des contraintes de types **MUST-LINK** et **CANNOT-LINK** (voir FIGURE C.1 en ANNEXE C.2.2). Pour les traiter, nous allons tester deux approches :

- une approche *naïve* ignorant les conflits : si la prochaine contrainte à ajouter est incompatible avec la base de contraintes déjà intégrées au gestionnaire, alors nous ignorons simplement son existence sans remettre en question les précédentes annotations ;
- une approche *avec correction* des conflits : pour simuler la correction d'un expert, nous re-créons à chaque itération le gestionnaire de contraintes en intégrant d'abord les contraintes correctes puis les contraintes erronées ; ainsi, les conflits ne peuvent arriver qu'à l'ajout d'une contrainte erronée, et il suffit d'ajouter sa version exacte pour simuler la correction de l'expert.

Ainsi, il y a donc 6 taux d'erreurs d'annotation à simuler, chacun suivant 2 approches de gestion de conflits, et chacune de ces simulations d'erreurs seront répétées 10 fois sur chaque tentative complète de la méthode pour contrer les aléas statistiques des tirages de contraintes erronées, ce qui représente 120 simulations par tentatives. Enfin, chaque tentative complète de *clustering* interactif est répétée 5 fois pour contrer les aléas statistiques des exécutions, ce qui représente un total de 600 tentatives complètes à simuler.

Enfin, nous affichons l'évolution de la performance moyenne du *clustering* obtenu en fonction des divers taux d'erreurs simulées, et nous discutons de la significativité de la perte de performances due à l'absence de corrections des conflits.

43. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

44. Paramétrage favori (atteindre 90% de **v-measure** avec un coût minimal) : prétraitement simple (**prep.simple**), vectorisation TF-IDF (**vect.tfidf**), *clustering KMeans* avec modèle **COP** (**clust.kmeans.cop**) et échantillonnage des données les plus proches dans des clusters différents (**sampl.closest.diff**).

45. Choix de 0% à 25% : nous utilisons ici les estimations grossières de bornes maximale d'erreurs réalisées en SECTION 4.6.1.

i Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.6.2.b Résultats obtenus

La FIGURE 4.31 représente l'évolution moyenne de la **v-measure** du *clustering* en fonction du nombre de contraintes annotées au cours des itérations de la méthode, et cette évolution est déclinée pour les 6 taux d'erreurs simulées et les 2 approches de gestion des conflits. Les contraintes utilisées sont basées sur les échantillonnages réalisées au cours des tentatives n'introduisant pas d'erreurs d'annotation par rapport à la vérité terrain : comme les mêmes contraintes sont donc utilisées (aux valeurs d'annotations près), toutes les courbes sont comparables point par point.

⚠️ Attention : Toutefois, il est important de noter que les tentatives sans contraintes ont besoin de maximum 3 000 contraintes pour annoter toutes les contraintes possibles et leurs transitivités (moyenne : 2 488, écart-type : 327). Au delà de cette limite, il faudrait échantillonner de nouvelles contraintes pour les tentatives introduisant des erreurs, mais les bases de contraintes utilisées ne seraient alors plus comparables. Nous décidons donc de tronquer les différentes courbes à 3 000 contraintes, que la convergence eut lieu ou non, et nous analysons ces résultats partiels obtenus pour ce nombre de contraintes.

Tout d'abord, observons l'approche *naïve*, ignorant simplement les conflits (voir FIGURE 4.31 (1)). Nous pouvons constater que les performances (basées sur la vérité terrain) plafonnent très rapidement si des incohérences sont introduites : dès 5% d'erreurs, le score de **v-measure** stagne autour de 65% à partir de 1 000 contraintes, alors qu'une performance théorique d'environ 90% serait attendue pour une tentative idéale n'introduisant pas d'erreurs. D'autre part, si le taux de contraintes erronées est supérieur à 15%, nous pouvons remarquer que ces performances diminuent avant de stagner à des valeurs inférieures à 40% de **v-measure**. Pour finir sur cette première approche, tout porte à croire que ces faibles seuils performances perdurent au delà des 3 000 contraintes (figure tronquée) : les tentatives s'éloignent donc significativement de la vérité terrain si des incohérences d'annotation sont insérées.

Observons désormais l'approche *avec correction*, ré-évaluant les contraintes lorsqu'un conflit est détecté par le gestionnaire de contraintes (voir FIGURE 4.31 (2)). Nous pouvons constater que les tentatives introduisant des contraintes erronées subissent aussi un retard de performances, mais celui-ci est bien plus faible que celui encaissé par les approches naïves. Nous observons que toutes les courbes restent globalement croissantes, bien que le score moyen de 90% de **v-measure** par rapport à la vérité terrain n'est pas atteint en moins de 3 000 contraintes par les tentatives ayant un taux d'erreurs supérieur à 15%. Nous pouvons toutefois espérer que cette convergence se poursuit au delà des 3 000 contraintes (figure tronquée).

4.6.2.c Discussion

L'objectif de cette étude est l'analyse de l'impact des différences de comportements intra-annotateur sur les résultats de notre méthode, plus particulièrement l'intérêt de corriger les incohérences d'annotations lorsqu'elles sont détectées par le gestionnaire de contraintes. Pour cela, nous avons comparé deux approches : une approche naïve ignorant simplement les conflits, et une deuxième approche les corrigeant lorsque ces derniers sont détectées.

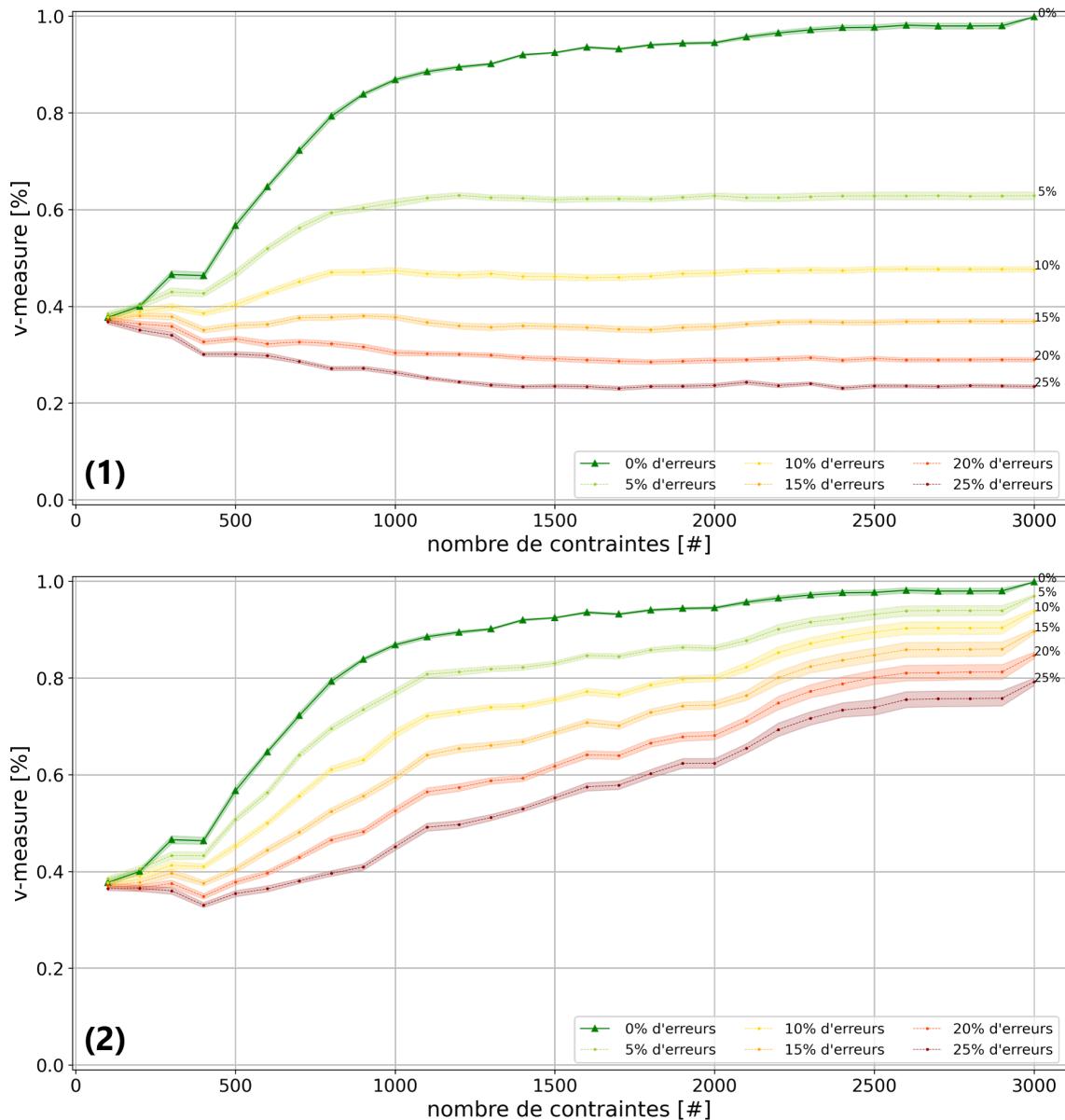


FIGURE 4.31 – Évolution des similitudes moyennes (calculées en terme de *v-measure*) des résultats de clustering des tentatives introduisant des erreurs d'annotation par rapport à la vérité terrain au cours des itérations. Les dégradés de couleurs des courbes représentent les déclinaisons de ces évolutions en fonction des différents taux d'annotations erronées (allant de 0 % et 25 %). (1) représente l'approche naïve ignorant les conflits d'annotation et (2) représente l'approche corrigeant les conflits détectés par le gestionnaire de contraintes. Toutes les courbes sont tronquées à 3 000 contraintes (nombre maximum de contraintes nécessaire à une tentatives n'introduisant pas erreurs pour converger vers la vérité terrain).

D'après l'analyse de la FIGURE 4.31, nous pouvons clairement déduire que l'absence de correction des incohérences pénalise significativement la méthode. En effet, sur le jeu de données utilisé comme vérité terrain, nous constatons une perte irréversible d'au moins 35% de *v-measure* dès l'introduction de 5% d'erreurs d'annotation, caractérisant ainsi une dérive importante des

résultats si les conflits d'annotations ne sont pas corrigés. Toutefois, la mise en oeuvre d'un mécanisme simple de correction semble estomper en partie ces régressions de performances et permet à certaines tentatives introduisant des erreurs de rester compétitives (*toutes les tentatives peuvent espérer atteindre 80% de v-measure en corrigeant leurs incohérences, alors que l'approche naïve les condamnait à une v-measure plafonnée*). Nous pouvons donc assurément conclure en faveur de la **nécessité de vérifier la base de contraintes et de corriger toute incohérence s'y trouvant.**

Une telle conclusion confirme la sensibilité aux erreurs de cette approche incrémentale : si une erreur apparaît, elle peut rapidement se propager et impacter les résultats de la méthode. Dans notre cas, nous avons optimisé l'implémentation de notre *clustering* interactif pour obtenir un corpus d'apprentissage pertinent en un minimum d'annotation de contraintes. Or obtenir un nombre minimal implique de limiter la redondance parmi les contraintes annotées. De ce fait, certaines incohérences mal placées peuvent fortement influencer la base de contraintes et ainsi faire diverger les résultats.

Pour contrer ce problème, nous avons deux pistes pouvant être explorées :

- **introduire intentionnellement de la redondance** dans la base de contraintes annotées à l'aide d'une nouvelle méthode d'échantillonnage : ce moyen tire parti des propriétés de transitivité du graphe de contraintes pour identifier d'éventuelles incohérences d'annotation isolée ou masquée. Cette piste a cependant le désavantage d'ajouter de nouvelles contraintes peu informatives dans le simple but de mieux détecter certaines incohérences, introduisant donc un léger coût supplémentaire pour l'annotateur ;
- **confronter plusieurs des annotateurs** sur les mêmes contraintes : en faisant annoter les mêmes contraintes par deux opérateurs différents, les erreurs d'annotation peuvent être révélées. Cette piste engendre aussi un surcoût car elle nécessite d'embaucher plusieurs annotateurs.

Notes de l'auteur : Dans les deux cas, il y a un choix à faire : **veut-on privilégier la qualité** (*ajouter de la redondance et une double vérification, au prix d'un surcoût d'annotation*) **ou la rapidité de conception** (*optimiser l'annotation pour une convergence en un minimum de contraintes, au risque d'introduire des incohérences dans la modélisation*) ?

4.6.3 Étude de l'impact de la subjectivité de l'annotation sur la divergence des résultats obtenus

Dans la section précédente, nous nous sommes intéressés aux différences de comportement intra-annotateur et aux pistes permettant de limiter les erreurs d'annotation d'un seul opérateur. Cependant, nous devons aussi nous intéresser aux **différences inter-annotateurs** : en effet, nous avons pu voir en SECTION 2.3.3.A que la labellisation est une tâche subjective, et que la complexité du phénomène à modéliser ainsi que la diversité de profils d'annotateurs peuvent introduire des différences d'annotation. Mais ces différences ne sont pas synonymes d'erreurs, elles peuvent aussi être les témoins de **différences d'opinion entre les experts**.

Notes de l'auteur : Pour aller plus loin, il est même mal avisé de parler d'*erreurs* d'annotation car cela suppose qu'une comparaison avec une vérité terrain soit possible. Or, en situation réelle, cette vérité terrain est précisément en cours de construction à l'aide de notre méthodologie de *clustering* interactif : l'expert est responsable de la vision qu'il veut appliquer aux données durant son annotation, il est donc difficile de parler d'*erreurs* sans porter de jugement sur sa vision. Il convient donc mieux de parler de *différences* d'annotation si deux experts ont une divergence de point de vue sur une donnée à annoter.

Dans cette dernière étude, nous allons donc estimer la robustesse du *clustering* interactif en nous intéressant plus particulièrement aux différences d'annotation entre deux opérateurs. Pour cela, nous allons simuler de telles différences entre deux annotateurs fictifs : le premier sera représenté par la vérité terrain, et le second sera simulé en introduisant un certain taux de désaccords d'annotation à chaque itération. Nous discutons ensuite de l'écart de similarité entre les résultats de *clustering* obtenus lorsque l'annotateur de référence atteint le seuil d'annotation partielle de son jeu de données (*accord théorique de 90% de v-measure sur la vérité terrain qu'il recherche*) Nous réalisons cette analyse sur des jeux de données de différentes tailles et pour plusieurs taux de désaccords insérées.

4.6.3.a Protocole expérimental

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.13.

Pour information : Nous reprenons dans les grandes lignes le protocole expérimental de la précédente étude sur l'impact des erreurs d'annotations et l'intérêt de les corriger (voir SECTION 4.6.2). Cependant, nous utilisons ici la vérité terrain pour représenter un opérateur de référence, et nous représentons le second opérateur par les différences d'annotation introduites.

Nous utilisons cette fois deux vérités terrains comme références, représentant un **annotateur de référence** (*et sa propre vision métier*) :

- le jeu de données **Bank Cards** (v2.0.0) : ce dernier traite des demandes les plus fréquentes des clients en ce qui concerne la gestion de leur carte bancaire. Il est composé de 1 000 questions rédigées en français et réparties en 10 classes (**perte ou vol de carte, carte avalée, commande de carte, ...**). Pour plus de détails, consultez l'annexe A.1 ;
- le jeu de données **MLSUM FR Train Subset** (v1.0.0-schild) : ce dernier concerne les titres d'articles de journaux issus des catégories de publication les plus communes. Il est composé de 744 titres d'articles rédigés et répartis en 14 classes (**économie, sport, ...**). Pour plus de détails, consultez l'annexe A.2 ;

Pour utiliser facilement plusieurs jeux de données de tailles différentes tout en maîtrisant leur contenu, nous avons donc dupliqué aléatoirement des données issues de ces jeux de référence en y insérant des fautes de frappes. La taille des jeux de données générés varie entre 1 000 à 5 000 par pas de 500. Il y a donc 9 variations de chaque jeu de références, soit 18 jeux utilisés de tailles différentes.

Données : jeux de données annotées (vérités terrains) de tailles différentes

Entrées : liste de taux de désaccords à insérer

```

1 pour chaque jeux de données à tester et taux de désaccords à insérer faire
2   initialisation (données) : récupérer les données et la vérité terrain ;
3   initialisation (contraintes) : créer une liste vide de contraintes ;
4   prétraitement : supprimer le bruit dans les données avec prep.simple ;
5   vectorisation : transformer les données en vecteurs avec vect.tfidf ;
6   clustering initial : regrouper les données par similarité avec clust.kmeans.cop ;
7   évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
8   répéter
9     échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
10    choix des désaccords : définir les contraintes erronées ;
11    simulation d'annotation : déterminer les contraintes avec la vérité terrain ;
12    intégration corrective : ajouter les nouvelles contraintes au gestionnaire de
        contraintes, et ré-annoter les annotations en conflit ;
13    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
14    évaluation : estimer l'équivalence entre le clustering et la vérité terrain ;
15    jusqu'à annotation de toutes les contraintes possibles ;
16    analyse locale : afficher l'évolution de la similarité entre les clustering de la
        tentative courante et les clustering de la tentative de référence (n'ayant pas de
        différence d'annotation par rapport à la vérité terrain) ;
17 analyse générale : analyse des différences de résultats de clustering obtenus par taille
        de jeux de données et par taux de désaccords insérés ;
Résultat : discussion sur l'impact de la subjectivité de l'annotation sur la divergence
        des résultats de clustering obtenu

```

ALGORITHME 4.13 – Description en pseudo-code du protocole expérimental de l'impact de la subjectivité de l'annotation sur la divergence des résultats.

⚠️ Attention : Dans le cadre de cette étude, nous faisons l'hypothèse que cette création artificielle de données n'a pas d'impact majeur sur le nombre de contraintes nécessaires pour converger vers une vérité terrain.

Sur ces jeux de données, nous exécutons une tentative complète⁴⁶ de la méthode du *clustering* interactif en utilisant notre paramétrage favori⁴⁷ (voir SECTION 4.3). À nouveau, nous allons ajouter un pourcentage de contraintes erronées à chaque itération, représentant un **autre annotateur** (*et sa propre vision métier du problème à modéliser*) :

- Le taux de désaccords insérés, variant de 0% à 25%⁴⁸ par pas de 5%, reste fixe tout au long d'une même tentative de notre méthode : nous pouvons ainsi analyser l'impact d'un taux de désaccords fixe sur les résultats au courant des itérations ;
- Les contraintes divergentes à insérer sont tirées aléatoirement parmi le lot de contraintes qui aurait été échantillonné au cours d'une tentative sans introduction de différences : ainsi, nous pouvons comparer itération par itération toutes ces simulations car elles partagent la même base de contraintes (aux valeurs de MUST-LINK et CANNOT-LINK près) ;

Puisque nous introduisons des différences d'annotations, des conflits peuvent apparaître dans le gestionnaire de contraintes. Pour rappel, un conflit est détecté dans le cas où l'ajout d'une nouvelle contrainte annotée contredit ce qui a été précédemment déduit grâce aux propriétés de transitivité des contraintes de types MUST-LINK et CANNOT-LINK (voir FIGURE C.1 en ANNEXE C.2.2). En tirant parti des conclusions de la précédente étude, nous choisissons de corriger ces désaccords dès leur détection, et supposant que le second annotateur se range à la vision de l'annotateur de référence. Pour simuler cette correction réalisée par les experts, nous recréons à chaque itération le gestionnaire de contraintes en intégrant d'abord les contraintes correctes puis les contraintes divergentes ; ainsi, les conflits ne peuvent arriver qu'à l'ajout d'une contrainte divergente, et il suffit d'ajouter sa version exacte pour simuler la correction des experts.

Ainsi, il y a donc 6 taux de désaccords, et chacune de ces simulations de désaccords seront répétées 2 fois sur chaque tentative complète de la méthode pour limiter au mieux les aléas statistiques des tirages de contraintes divergentes, ce qui représente 12 simulations par tentatives. Enfin, chaque tentative complète de *clustering* interactif est répétée 2 fois pour limiter au mieux les aléas statistiques des exécutions, soit un nombre de 24 tentatives complètes pour chacun des 18 jeux de données, ce qui représente un total de 432 tentatives à simuler au cours de cette expérience.

Nous réalisons ensuite l'analyse des différences de résultats obtenu en estimant la similarité des *clustering* entre une tentative introduisant des différences et sa tentative de référence n'en introduisant pas. Pour cela, nous procédons en trois temps :

- nous estimons d'abord, pour chaque taille de jeu de données, le nombre moyen de contraintes nécessaires aux tentatives de référence pour atteindre un score de 90% de **v-measure** par rapport à leur vérité terrain ;

46. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

47. Paramétrage favori (atteindre 90% de **v-measure** avec un coût minimal) : prétraitement simple (`prep.simple`), vectorisation TF-IDF (`vect.tfidf`), *clustering KMeans* avec modèle COP (`clust.kmeans.cop`) et échantillonnage des données les plus proches dans des clusters différents (`sampl.closest.diff`).

48. Choix de 0% à 25% : nous utilisons ici les estimations grossières de bornes maximale d'erreurs réalisées en SECTION 4.6.1.

- nous estimons ensuite, pour chaque tentative introduisant des différences d'annotation, le score de **v-measure** entre leur résultat de *clustering* et le résultat de *clustering* de leur tentative de référence pour le nombre de contraintes déterminé précédemment (celui nécessaire à la tentative de référence pour atteindre un score de 90% par rapport à sa vérité terrain) ;
- enfin, nous discutons de ces scores moyens pour les différentes tailles de jeu de données et les différents taux de désaccords introduits.

Un exemple est illustré avec la FIGURE 4.32.

Notes de l'auteur : Il est à noter que nous aurions pu estimer ce nombre de contraintes grâce à l'ÉQUATION 4.16⁴⁹, mais comme cette estimation théorique est une moyenne qui aurait pu décaler légèrement nos résultats. Nous avons donc préféré mesurer directement le nombre exact de contraintes nécessaires pour chaque tentative afin de ne pas avoir à analyser une double moyenne.

Attention : Ces simulations étant plus lourdes que les précédentes, et en considérant le manque de temps pour réaliser cette étude, nous avons du diminuer le nombre de répétitions de nos tentatives (1 pour le génération des jeux de données, 2 pour l'insertion des différences, 2 pour l'exécution des tentatives complètes de la méthode). Les résultats obtenus nous permettent tout de même de discuter des tendances générales, mais il serait intéressant de compléter a posteriori les résultats de cette étude pour en améliorer la fiabilité.

Pour information : Les scripts de l'expérience, réalisés avec des *notebooks Python* (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022b.

4.6.3.b Résultats obtenus

Commençons par un exemple de résultats pour un jeu de 5 000 données. La FIGURE 4.32 représente l'évolution moyenne de la **v-measure** du *clustering* en fonction du nombre de contraintes annotées au cours des itérations de la méthode. Sur cette figure, la courbe ayant 0% de différence d'annotation représente l'évolution moyenne des tentatives de l'opérateur de référence : celles-ci convergent pas-à-pas vers la vérité terrain (100 de **v-measure**). Les autres courbes représentent les tentatives du second opérateur et la divergence de ses *clustering* due à l'introduction de différence d'annotation. Dans un soucis de lisibilité, nous avons toutefois tronqué la figure à 50 000 contraintes.

Grâce à cette figure, nous pouvons estimer à 16 250 le nombre moyen de contraintes nécessaires aux tentatives de référence pour atteindre 90% de **v-measure** par rapport à la vérité terrain (*ici : celle ayant 5 000 données*). Sur cette base, nous pouvons identifier la similitude moyenne des résultats *clustering* entre chaque tentative introduisant des désaccords et leur tentative de

49. ÉQUATION 4.16 : $\text{constraints_needed} \propto 3.15 \cdot \text{dataset_size}$

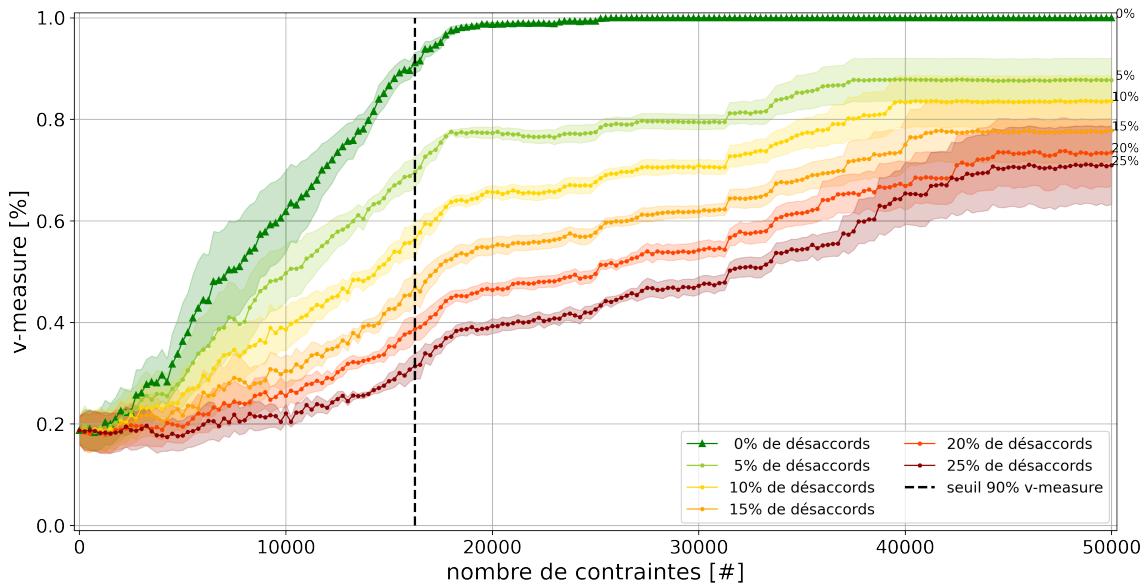


FIGURE 4.32 – Exemple d’une évolution de similitudes moyennes (calculées en terme de *v-measure*) de résultats de clustering de tentatives introduisant des différences d’annotation par rapport à la vérité terrain au cours des itérations, exemple pour une vérité terrain ayant une taille de 5 000 données. Les dégradés de couleurs des courbes représentent les déclinaisons de ces évolutions en fonction des différents taux d’annotations divergentes (allant de 0% et 25%). La barre verticale indique le nombre moyen de contraintes nécessaires aux tentatives n’introduisant pas de désaccords pour obtenir un score de 90% de *v-measure* (ici : 16 250 contraintes).

référence, calculée pour un nombre fixe de 16 250 contraintes⁵⁰. Ces scores de similitudes moyen sont rapportés dans la TABLE 4.13, chaque analyse pour une taille de jeu de données étant retranscrite dans une colonne dédiée.

Nous pouvons constater les points suivants :

- la similarité est bien entendu de 100% pour un taux d’incohérence de 0% (*une tentative de référence est comparée à elle-même*) ;
- à taille de jeu de données fixe, le taux de similarité diminue lorsque le taux de désaccords introduits augmentent ; l’amplitude de diminution varie environ entre 35 points (*pour une taille de 1 000 données*) et 73 points (*pour une taille de 4 500 données*) .
- à un taux fixe de désaccords introduits, le taux de similitude diminue lorsque la taille du jeu de données augmentent ; l’amplitude de diminution varie environ entre 0 points (*pour l’introduction de 0% de désaccords*) et 33 points (*pour l’introduction de 25% de désaccords*) .

4.6.3.c Discussion

L’objectif de cette étude est l’analyse de l’impact des différences de comportements inter-annotateurs sur les résultats de notre méthode, plus particulièrement l’impact de la subjectivité

50. Observation de 16 250 contraintes nécessaires : lors d’une analyse théorique utilisant l’ÉQUATION 4.16, nous aurions estimé une moyenne de $3.15 \cdot 5\ 000 \propto 15\ 750$ contraintes, soit un écart de 500 contraintes qui aurait introduit une légère variation dans nos résultats.

	Contraintes annotées	Taille des jeux de données								
		1 000	1 500	2 000	2 500	3 000	3 500	4 000	4 500	5 000
Taux de désaccords simulés	4 250	4 250	6 000	7 250	8 750	10 250	11 250	12 250	13 500	16 250
	0%	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)	100.00% (±0.00)
	5%	86.16% (±3.52)	78.56% (±1.50)	80.09% (±1.27)	74.81% (±1.62)	75.06% (±1.02)	74.72% (±1.84)	71.86% (±0.91)	70.07% (±1.29)	71.03% (±3.09)
	10%	79.84% (±6.22)	66.02% (±2.00)	66.95% (±1.46)	61.13% (±1.55)	61.64% (±1.05)	60.93% (±1.92)	57.70% (±1.52)	54.00% (±1.84)	57.01% (±3.65)
	15%	73.94% (±8.48)	58.64% (±2.21)	57.13% (±1.97)	51.01% (±1.43)	48.25% (±2.14)	52.33% (±2.04)	48.52% (±1.61)	42.13% (±1.43)	47.41% (±3.27)
	20%	68.96% (±9.61)	51.92% (±2.30)	48.85% (±2.31)	42.01% (±0.96)	41.57% (±2.00)	43.37% (±1.71)	40.56% (±1.94)	34.87% (±1.15)	39.08% (±2.51)
	25%	65.05% (±10.78)	44.12% (±2.30)	40.19% (±2.01)	35.88% (±0.98)	35.31% (±1.25)	37.31% (±2.04)	32.47% (±1.46)	26.73% (±1.56)	31.90% (±2.56)

TABLE 4.13 – Estimation de la similitude moyenne (calculée en terme de v-measure) des résultats de clustering des tentatives introduisant des désaccords d'annotation par rapport aux résultats de clustering de leurs tentatives de référence.

Cette similitude est rapportée en fonction de la taille du jeu de données utilisé et du taux de désaccords introduits lors des tentatives. Pour chaque taille de jeu de données, les calculs sont réalisés avec un nombre de contraintes fixe, choisi comme étant le nombre de contraintes nécessaires à une tentative de référence pour atteindre une v-measure moyenne de 90% avec sa vérité terrain (ce nombre est rapporté en deuxième ligne).

des opérateurs manifestée par des désaccords d'annotation sur le résultat du clustering obtenu. Pour cela, nous avons analysé l'évolution de la similarité en simulant des désaccords d'annotation pour plusieurs tailles de jeux de données.

Sur la base des résultats rapportés dans la TABLE 4.13, nous pouvons constater que les résultats de clustering divergent rapidement si des désaccords d'annotation sont présents. En effet, l'introduction de 5% de différences fait diverger les résultats d'au moins 14 points de v-measure, et cette divergence est plus forte lorsque la taille du jeu de données augmente. Si nous prenons le cas extrême (25% de désaccord, taille de 4 500 données), la similitude des clustering obtenu n'est en moyenne plus que de 26.73% au bout de 13 500 contraintes. Nous pouvons donc conclure qu'en l'état, le clustering interactif est sensible à la subjectivité des annotateurs.

Une telle conclusion est en accord avec le caractère incrémental de la méthode : une différence d'annotation peut rapidement se propager et faire diverger les résultats de la méthode. Dans notre cas, nous avons optimisé l'implémentation de notre clustering interactif pour obtenir un corpus d'apprentissage pertinent un minimum d'annotation de contraintes. Or cette notion de pertinence est subjective à la vision de l'expert annotant les données. De ce fait, si deux annotateurs ne sont pas d'accord sur la vision à appliquer aux données lors de l'annotation, il est normal de voir leurs résultats de clustering diverger. Nous pourrions résumer cette situation par le fait que **ces deux experts ne recherchent pas la même vérité terrain, d'où la divergence significative de leurs résultats**. Ainsi, la sensibilité de notre méthode n'est pas la racine du problème, elle met plutôt en lumière le besoin de confronter les opinions des experts afin qu'ils puissent s'accorder.

En nous basant sur notre revue de littérature (notamment la SECTION 2.3.3.A), nous conseillons à un projet d'annotation voulant utiliser notre implémentation du clustering interactif pour concevoir une base d'apprentissage d'**employer au moins 3 opérateurs et de**

les confronter aux mêmes contraintes à chaque itération de la méthode. Ainsi, si ces derniers ont des différences d'opinion sur la modélisation du problème, celles-ci se manifesteront en observant le score d'accord inter-annotateurs. Il conviendra alors d'organiser une session de débat pour ré-annoter les désaccords et décider des adaptations à prévoir dans le guide d'annotation du projet.

Idées : Pour aider à trouver un accord lors de la revue, une prise de recul peut être nécessaire. Pour ce faire, les experts pourraient par exemple **observer le graphe de contraintes** déjà annotées. En effet, le gestionnaire de contraintes gère les propriétés de transitivité entre celle-ci, créant ainsi des composants connexes de données liées par les contraintes MUST-LINK et distinguées par les contraintes CANNOT-LINK (voir ANNEXE C.2.2). Or le règlement d'un désaccord va nécessairement impacter ces composants connexes : en les rapprochant si le désaccord est tranché en faveur d'un MUST-LINK, en les éloignant sinon. Par conséquent, il serait intéressant d'aiguiller le débat pour anticiper les conséquences du consensus à trouver : ces composants sont-ils à rapprocher ? à distinguer ? ou sont-ils éventuellement à remettre en question ?

Si malgré cette prise de recul, aucun accord n'a été trouvé, il est possible de ne pas caractériser cette contrainte et de **laisser l'algorithme de clustering trancher le débat**. En effet, la philosophie de la méthode d'annotation basée sur le *clustering* interactif repose sur la coopération entre l'Homme et la machine : dans cette situation, il peut être intéressant de simplement observer quelle solution est proposée par la machine pour segmenter les données tout en respectant les autres contraintes déjà annotées.

Bien entendu, une telle approche engendre un coût supplémentaire aux estimations réalisées dans la SECTION 4.3 :

- d'une part, nous triplons la charge salariale à investir dans le but de redonner les contraintes annotées ;
- d'autre part, nous introduisons un délai supplémentaire en organisant des revues de désaccords de contraintes, du moins tant que des désaccords de visions subsistent.

Néanmoins, ces surcoûts participent à l'amélioration de la stabilité de la base d'apprentissage en cours de construction. Par ailleurs, nous pouvons noter que :

- un projet d'annotation classique emploie aussi plusieurs opérateurs : ce surcoût correspond donc plutôt à un investissement supplémentaire au profit d'une fiabilisation de la qualité de ses résultats ;
- un projet d'annotation classique est déjà confronté au besoin d'organiser des sessions de revue de sa modélisation (voir l'étape *Revise* du cycle MATTER) : l'avantage de notre méthode réside néanmoins sur la possibilité de discuter directement des divergences d'opinions en analysant des cas d'usage métier (*évaluation d'une contrainte à l'aide de compétences métiers*) plutôt que de discuter de divergences d'opinions sur l'interprétation d'une abstraction du problème (*remise en cause d'une modélisation à l'aide de compétences analytiques*).

Notes de l'auteur : Nous concluons cette discussion par la même remarque faite à la fin de la précédente étude de robustesse : **il y a un choix à faire entre privilégier**

la qualité (*ajouter de la redondance pour clarifier les désaccords d'opinion, au prix d'un surcoût d'annotation*) ou **la rapidité de conception** (*optimiser l'annotation pour une convergence en un minimum de contraintes, au risque d'introduire des incohérences dans la modélisation*).

4.6.4 Bilan concernant la robustesse du *clustering* interactif

- Points à retenir : Au cours de cette étude de robustesse, nous avons pu voir que :
- ✓ Le *clustering* interactif, comme tout autre approche incrémentale, est **sensible aux erreurs et aux désaccords d'annotation** : en effet, la méthode est optimisée pour converger vers une base d'apprentissage en un minimum de contraintes, donc toute différence d'annotation peut rapidement se propager ;
 - ✓ Pour **combattre les erreurs d'annotations** (*divergences intra-annotateur*), il peut être intéressant d'**ajouter de la redondance dans les annotations** : cela permet au gestionnaire de contraintes de vérifier les propriétés de transitivité des contraintes MUST-LINK et CANNOT-LINK, contribuant ainsi à la détection d'éventuelles incohérences ;
 - ✓ Pour **combattre les différences d'opinions** (*désaccords inter-annotateurs*), il est nécessaire de **confronter plusieurs opérateurs sur les mêmes annotations contraintes** : cela permet de débattre des désaccords d'interprétation de cas d'usages métier dissimulés derrière les différences de labellisation, et de compléter le guide d'annotation si besoin ;
 - ✓ Il y a un choix à faire entre **privilégier la qualité** (*vérification des annotations et harmonisation des opinions, au prix d'un surcoût d'annotation*) ou **la rapidité de conception** (*optimisation de la convergence en un minimum d'annotation, au risque d'introduire des incohérences dans la modélisation*) .

4.7 Autres hypothèses non vérifiées

Lors des études précédentes, nous avons vérifié un certain nombre d'hypothèses et avons exploré plusieurs détails pratiques pour mettre en oeuvre une méthodologie d'annotation basée sur le *clustering* interactif. Toutefois, certains points n'ont pas pu être étudiés en profondeurs lors de ce doctorat, par manque de temps ou de moyens. Nous exposons ici un ensemble de pistes intéressantes pouvant nourrir de futurs travaux afin d'améliorer la notre méthode.

4.7.1 Étude du nombre de clusters optimal

Un problème ouvert de la recherche lors de l'utilisation d'algorithmes de *clustering* concerne le choix du nombre de *clusters* à trouver. En effet, à part une connaissance à priori du nombre de thématiques présentes dans le jeu de données, il est difficile d'estimer le nombre optimal de *clusters*, d'autant plus que celui-ci peut changer en fonction de la granularité de modélisation requise pour répondre au cas d'usage.

Nous avons déjà exploré partiellement deux pistes :

- **l'exploration du graphe de contraintes** : en effet, il est possible d'estimer le nombre maximal de *clusters* grâce aux composants connexes de contraintes **MUST-LINK**, et d'estimer le nombre minimal de *clusters* grâce à la coloration du graphe de contraintes **CANNOT-LINK** ;
- **les études de pertinence** avec l'analyse des patterns linguistiques et le résumé thématique des *clusters* (cf. SECTION 4.4) : ces deux approches permettent de rapidement constater si les thématiques obtenues sont trop générales (*i.e.* *s'il n'y a pas assez de clusters*) ou si elles semblent trop spécifiques (*i.e.* *s'il y en a trop*).

Toutefois, pour aller plus loin, deux pistes potentielles pourraient être explorées :

- l'exploration brute du nombre de *clusters* par la **méthode du coude** : bien que ces approches sont plus coûteuses en temps de calcul, elles permettent d'estimer le nombre de *clusters* pour lequel la stabilité du *clustering* est la plus élevée ;
- l'utilisation d'algorithme n'ayant pas de nombre de clusters en paramètres comme des versions contraintes de **DBScan** (par exemple dans sa version **C-DBScan**, RUIZ et al., 2010) ou de la **propagation par affinité** (GIVONI et FREY, 2009) : ces alternatives semblent prometteuses car elles retirent la complexité due à ce paramétrage abstrait.

i Pour information : L'étude de **C-DBScan** a été en partie réalisée dans le cadre d'un projet étudiant avec l'école d'ingénieur Télécom Physique Strasbourg. Les résultats montraient que le temps de calcul était similaire à celui du **KMeans** (dans sa version **COP**). La difficulté d'utilisation résidait plutôt sur la définition du rayon de voisinage **eps** à parcourir pour établir des liens entre données. Celui-ci peut être estimé en analysant la densité vectorielle du jeu de données. Le code informatique est disponible dans **SCHILD**, 2022a⁵¹.

51. Implémentation de **C-DBScan** : *Pull Request* en attente pour une version 0.6.0 après ajout de documentation et de tests unitaires.

4.7.2 Étude d'autres méthodes de vectorisation

Au début de ce doctorat, nous avons conclu que les algorithmes de vectorisation n'avaient pas d'impact réel sur l'efficience de notre méthodologie d'annotation. Toutefois, les modèles de langues se sont largement développés, et il est fort probable que l'utilisation d'un **modèle pré-entraîné** permettent désormais d'avoir un gain de performance.

Nous pourrions par exemple tester les **architectures à base de Transformers** (USZKOREIT, 2017) comme **BERT** (DEVLIN et al., 2019) et essayer différents modèles pré-entraînés sur des données françaises pour compléter nos études réalisées dans SCHILD, 2022b

4.7.3 Étude d'autres méthodes d'échantillonnage

Comme nous avons pu le voir dans SECTION 4.6, il peut-être intéressant d'introduire un mécanisme de création de redondance dans le graphe de contraintes annotées pour identifier les erreurs d'annotation. Un tel mécanisme n'a pas encore été implémenté mais pourrait facilement être intégré aux implémentations Python déjà existantes (SCHILD, 2022a).

Pour ce faire, le parcours de graphe et la création de cycle permettraient de vérifier la présence de conflits et ainsi de **provoquer des phases de revues de contraintes** si cela est nécessaire. Une telle page de revue pourrait aussi être complétée par des analyses complémentaires, comme l'estimation du taux de contraintes n'ayant pas de redondance et représentant ainsi des erreurs cachées potentielles.

4.7.4 Étude de techniques de transfert d'apprentissage

Dans la SECTION 2.3, nous avions déjà évoqué le fait que la modélisation d'un phénomène peut être assister par des techniques telles que la pré-annotation (DANDAPAT et al., 2009) ou le transfert d'apprentissage (ZHUANG et al., 2021). Nous pourrions nous inspirer davantage de ces approches pour démarrer plus efficacement les premières itérations d'un *clustering* interactif.

Voici quelques idées inspirées de ces méthodes :

- **pré-annoter** certaines contraintes simples à l'aide de règles (*basées par exemple sur la présence de mots de vocabulaire en commun*) ou grâce à l'utilisation d'un modèle déjà disponible ;
- **introduire des données synthétiques ou empruntées** à d'autres bases d'apprentissage pour initialiser le *clustering*, et permettre ainsi ajouter d'emblée des connaissances générales dans la modélisation.

4.7.5 Étude ergonomique de l'interface d'annotation

L'application web développée au cours de ce doctorat (SCHILD, TREMBLE et MISIAK, 2022) permet d'essayer rapidement notre méthodologie d'annotation. Cependant, cette dernière n'a pu faire l'objet d'études poussées pour estimer la meilleure disposition des composants ou l'intérêt de certaines fonctionnalités d'annotation.

Parmi les pistes potentielles à explorer, nous avons évoqué la possibilité d'**annoter plusieurs contraintes** dans une même interface (*par exemple : annoter visuellement un mini-graphes de 4 données plutôt que d'annoter simplement un couple de données*) et le besoin de **réaliser des analyses rapides** sur les *clusters* ou sur le graphe de contraintes (voir SECTION 4.4 et SECTION 4.5). Pour aller plus loin, BAE et al., 2021 propose d'autres listes d'interactions qui sont

possibles d'avoir avec un algorithmes de *clustering*, notamment sur la manipulation de son résultats (*fusion*, *suppression*, *verrouillage*, ...) et de ses hyperparamètres (*nombre de clusters*, *adaptation du vocabulaire autorisé*, ...)

Toutes ces idées pourraient être l'objet de développements et d'études dédiées avec des groupes d'annotateurs différentes pour voir l'impact sur les performances et les biais de conception de modèles.

Chapitre 5

Bilan et Guide d'utilisation du *Clustering Interactif*

5.1 Présentation rapide du clustering interactif et de ses avantages

SECTION À RÉDIGER :

- trouver une base d'apprentissage acceptable, puis corriger manuellement
- intervention d'experts métiers sur la base de leur connaissance métiers
- revues d'annotations basée sur leur connaissance métiers
- aide à la modélisation par le clustering

5.2 Conseils pour organiser l'annotation

SECTION À RÉDIGER :

- bien définir l'objectif à modéliser (par action ? par objet ?)
- laisser à la mach
- faire référence aux maximes de LEECH, 1993 ? 1. VOIR LA DONNEE NON PRETRAI-TEE : It should always be possible to come back to initial data (example BC). Note : can be hard after normalization ("l'arbre !" vs "le arbre", etc.)
- 2. ANNOTER DES DIFFERENCES VISIBLES : Annotations should be extractable from the text
- 3. DOCUMENTER L'AJOUT DE CONTRAINTES : The annotation procedure should be documented (ex : Brown Corpus annotation guide, Penn Tree Bank annotation guide)
- 4. DOCUMENTER LES COMPETENCES DES ANNOTATEURS : Mention should be made of the annotator(s) and the way annotation was made (manual/automatic annotation, number of annotators, manually corrected/uncorrected...)
- 5. SUBJECTIVITE => 3 ANNOTATEURS : Annotation is an act of interpretation (cannot be infallible)
- 6. MIEUX VAUT NE PAS LIER QUE LIER DE MANIERE AMBIGUE : Annotation schemas should be as independent as possible on formalisms
- 7. PLUSIEURS VISIONS POSSIBLES, IL FAUT EN CHOISIR UNE ET S'Y TENIR : No annotation schema should consider itself a standard (it possibly becomes one)

5.3 Conseils pour analyser les résultats

SECTION À RÉDIGER :

- Cas d'arrêt quand le clustering stagne à 5% : si change pas, alors l'annotation n'a plus d'effet...
- utiliser le graphe de contraintes pour voir les données liées entre elles et les données isolées
- Analyse avec résumer par LLM pour identifier facilement les thématiques qui se dégagent
- Analyse avec FMC pour identifier le vocabulaire qui caractérise chaque thématique
- utiliser ces analyses pour régler le nombres d clusters ?

5.4 Conseils pour paramétrier la méthode

SECTION À RÉDIGER :

- Architecture parallele : pour gagner du temps
- paramétrage optimal : simple + tfidf + kmeans + closest
- taille de batch d'annotation dépendant de la taille du jeu de données (dépend du temps de clustering)

5.5 Estimation des coûts du projet d'annotation

SECTION À RÉDIGER :

- rappeler les équations de temps : environ 24 x taille de dataset, sans les revues d'annotations
- besoin de 3 annotateurs pour confronter les vision et modéliser dans de bonnes conditions. organiser les revues sur la base des différences entre cas d'usage métier
- ajouter de la redondance si le cas d'usage est complexe, mais ça ralenti
- avantage par rapport aux méthodes usuelles : moins abstrait/complexe, moins d'essai-erreur

Chapitre 6

Conclusion

6.1 (*Occupying the Niche*)

SECTION : TITRE À TROUVER : "*Occupying the Niche*"

SECTION : À RÉDIGER

6.2 (*Gap*)

SECTION : TITRE À TROUVER : "*Gap*"

SECTION : À RÉDIGER

6.3 (*Establishing a Niche*)

SECTION : TITRE À TROUVER : "*Establishing a Niche*"

SECTION : À RÉDIGER

6.4 (*Asset centrality*)

SECTION : TITRE À TROUVER : "*Asset centrality*"

SECTION : À RÉDIGER

Annexe A

Annexe des jeux de données utilisés dans nos études

Pour les différentes études réalisées au cours de ce doctorant (cf. CHAPITRE 4), nous avons utilisé les deux jeux de données suivants.

Sommaire

A.1	Jeu de données Bank Cards	155
A.2	Jeu de données MLSUM	156

A.1 Bank Cards : Jeu d'entraînement en français d'assistants conversationnels traitant des demandes courantes sur les cartes bancaires

Description : Cet ensemble de données représente des exemples de demandes usuelles des clients concernant la gestion des cartes bancaires. Il peut être utilisé comme jeu d'entraînement pour un petit assistant conversationnel destiné à traiter ces demandes courantes.

Contenu : Les questions sont formulées en français. L'ensemble de données est divisé en 10 intentions (classes) dont un aperçu est disponible dans la TABLE A.1. Ces intentions sont construites de telle manière que toutes les questions issues d'une même intention ont la même réponse ou action. La version 1.0.0 du jeu de données contient de 50 questions par intention, soit un total de 500 questions ; La version 2.0.0 du jeu de données contient de 100 questions par intention, soit un total de 1 000 questions.

Origine : Le périmètre des intentions est inspiré d'un chatbot actuellement en production. Les données ont été sélectionnées aléatoirement et reformulées manuellement pour garantir la confidentialité des utilisateurs : aucune données personnelles ne subsistent dans ce jeu de données. Enfin, deux réviseurs extérieurs à l'équipe de recherche (*des Data Analyst*), ayant un profil d'analystes métiers du domaine bancaire, ont validé le périmètre et le contenu de ces intentions.

Disponibilité : Le jeu de données est archivé sur la plateforme Zenodo et est accessible ici : SCHILD, 2022c

Annexe A. Annexe des jeux de données utilisés dans nos études

Intention	Définition	Exemple
alerte_perte_volee_carte	Affichage de la procédure de blocage d'une carte perdue ou volée	Comment signaler une perte de carte de paiement ?
carte_avalee	Affichage de la procédure de récupération d'une carte avalée	Comment récupérer une carte avalée ?
commande_carte	Affichage des cartes disponibles, de la procédure de commande,	Je souhaite changer de carte bancaire.
consultation_soldes	Affichage d'une synthèse des soldes bancaires du client.	Où retrouver le solde de mon compte ?
couverture_assurance	Affichage d'une synthèse des garanties d'assurances de la carte bancaire du client	Que couvre ma carte bancaire en cas d'hospitalisation ?
deblocage_carte	Affichage de gestion du statut des cartes du client.	ma carte de paiement est bloquée, que faire ?
gestion_carte_virtuelle	Affichage de gestion des cartes virtuelles du client.	Comment faire pour créer une carte de paiements virtuelle ?
gestion_decouvert	Affichage d'une synthèse des autorisations de découverts de leur procédure de gestion	Est-ce que j'ai un découvert autorisé ?
gestion_plafond	Affichage de gestion des plafonds des cartes du client.	Le plafond de ma carte est trop bas, que faire ?
gestion_sans_contact	Affichage de gestion des fonctionnalités des cartes du client.	Je veux désactiver le sans contact sur ma carte.

TABLE A.1 – Présentation du jeu de données *Bank Cards* avec quelques exemples. La version 2.0.0 contient 100 questions par intention.

A.2 MLSUM (The Multilingual Summarization Corpus) : Échantillon de titre d'article de journaux en français associés à leur classification thématique

Description : Cet ensemble de données est constitué d'articles de journaux avec leur titre, leur résumé et leur classification thématique. Nous l'utilisons (1) pour estimer le temps nécessaire pour annoter la similarité des titres avec des contraintes (**MUST-LINK**, **CANNOT-LINK**) et (2) pour tester la méthodologie de *clustering* interactif (annotation de contraintes et *clustering* sous contraintes).

Contenu : Les titres de journaux sont formulés en français. L'ensemble de données est divisé en 14 thèmes (classes) dont un aperçu est disponible dans la TABLE A.2. La version 1.0.0 [subset: fr+train+filtered] contient 744 articles.

Origine : L'ensemble de données MLSUM a été proposé par SCIALOM et al., 2020. Notre ensemble de données en est un échantillon (*sélectionner au hasard de 75 articles dans les 14 sujets les plus utilisés*) filtré (*conserver les articles qui ont un sujet évident par rapport à leur titre, sans leur corps*). Deux réviseurs (*une Data Scientist et moi-même*) ont travaillé sur cette tâche afin de limiter la subjectivité du filtrage. L'échantillon final contient 744 articles après révision.

Disponibilité : Le jeu de donnée original est archivé sur arXiv et est accessible ici : SCIALOM et al., 2020. L'échantillon réalisé par nos soins est archivé sur la plateforme Zenodo et est accessible ici : SCHILD et ADLER, 2023.

Thème	Définition	Exemple	Taille
arts	Actualités artistiques (spectacles, oeuvres, événements, expositions)	<i>La rencontre de l'art et de la gastronomie au château du Feij</i>	50
disparitions	Actualités nécrologiques (décès ou disparition)	<i>Le traducteur Jean-Pierre Carasso est mort à 73 ans</i>	48
écologie	Actualités sur la pollution et la transition écologique	<i>Comment Lyon a banni les pesticides de ses parcs et jardins</i>	34
economie	Actualités économiques, financières et boursières	<i>La guerre des prix s'intensifie sur le marché du mobile en Israël</i>	41
education	Actualités liées à l'éducation et à la filière enseignante	<i>Plainte de parents d'élève sur des notes jugées trop basses au bac</i>	62
emploi	Actualités liées au marché du travail et aux actions syndicales	<i>Plus d'un tiers des CDI prennent fin avant la première année</i>	54
immobilier	Actualités liées au marché de l'immobilier et logements locatifs	<i>Depuis la fin des années 2000, l'accession à la propriété se complique en France</i>	65
meteo	Actualités météorologiques (bulletins, catastrophes, canicule)	<i>L'Eure et l'est de la France balayés par les intempéries</i>	35
musiques	Actualités liées aux chanteurs, concerts et sorties d'albums	<i>Opéra : Elsa Dreisig, une soprano à voix nue</i>	55
police-justice	Actualités liées aux affaires policières et aux tribunaux	<i>Bygmalion : Nicolas Sarkozy directement visé</i>	67
politique	Actualités de la scène politique et législative	<i>Le Sénat donne son aval à la prolongation de l'état d'urgence</i>	52
sante	Actualités sanitaires	<i>Chine : un nouveau cas de grippe aviaire H7N9</i>	70
sciences	Actualités scientifiques et vulgarisation	<i>L'ordinateur quantique au banc d'essai</i>	47
sport	Actualités sportives	<i>F1 : Webber partira en tête à Monaco</i>	64

TABLE A.2 – Présentation du jeu de données échantillonné à partir de MLSUM avec quelques exemples.

Annexe A. Annexe des jeux de données utilisés dans nos études

Annexe B

Annexe des architectures d'assistants conversationnels (*chatbot*)

Au début de ce doctorat (octobre 2019), on pouvait noter que :

- selon COSTELLO et LODOLCE, 2019, « seuls 4% des clients de Gartner [déclaraient] utiliser des *chatbot* sur leur lieu de travail, mais 40% [avaient] l'intention de les mettre en oeuvre à court terme » ;
- et selon GOASDUFF, 2019, « d'ici 2022, 70% des employés [interagiraient] quotidiennement avec les plateformes conversationnelles ».

Aujourd'hui (octobre 2023), le mot *chatbot* est sur toutes les lèvres, surtout depuis la révolution des IA génératives lancées par ChatGPT (OPENAI, 2023) :

- selon COSTELLO et LODOLCE, 2022, une entreprise sur deux aurait actuellement recours à une forme de *chatbot* pour gérer sa relation client, et « d'ici 2027, les *chatbot* deviendront le principal canal de service client pour environ un quart des organisations ».

Dans cette annexe, nous allons détailler brièvement les philosophies de conceptions d'assistants conversationnels.

i Pour information : Cette présentation s'inspire des articles de CHEN et al., 2017 et de ADAMOPOULOU et MOUSSIADES, 2020.

A REDIGER

D'après CHEN et al., 2017 et ADAMOPOULOU et MOUSSIADES, 2020, nous pouvons distinguer grossièrement les *chatbots* en deux approches en fonction de leur objectif principal :

- d'une part, il y a les *chatbots* « *task-oriented* », axés sur l'accomplissement d'une tâche précise ;
- d'autre part, il y a les *chatbots* « *chat-oriented* », dont l'utilité première est basée sur leur capacité à entretenir une conversation avec l'utilisateur.

La FIGURE B.1 représente ces deux approches par leurs architectures les plus communes.

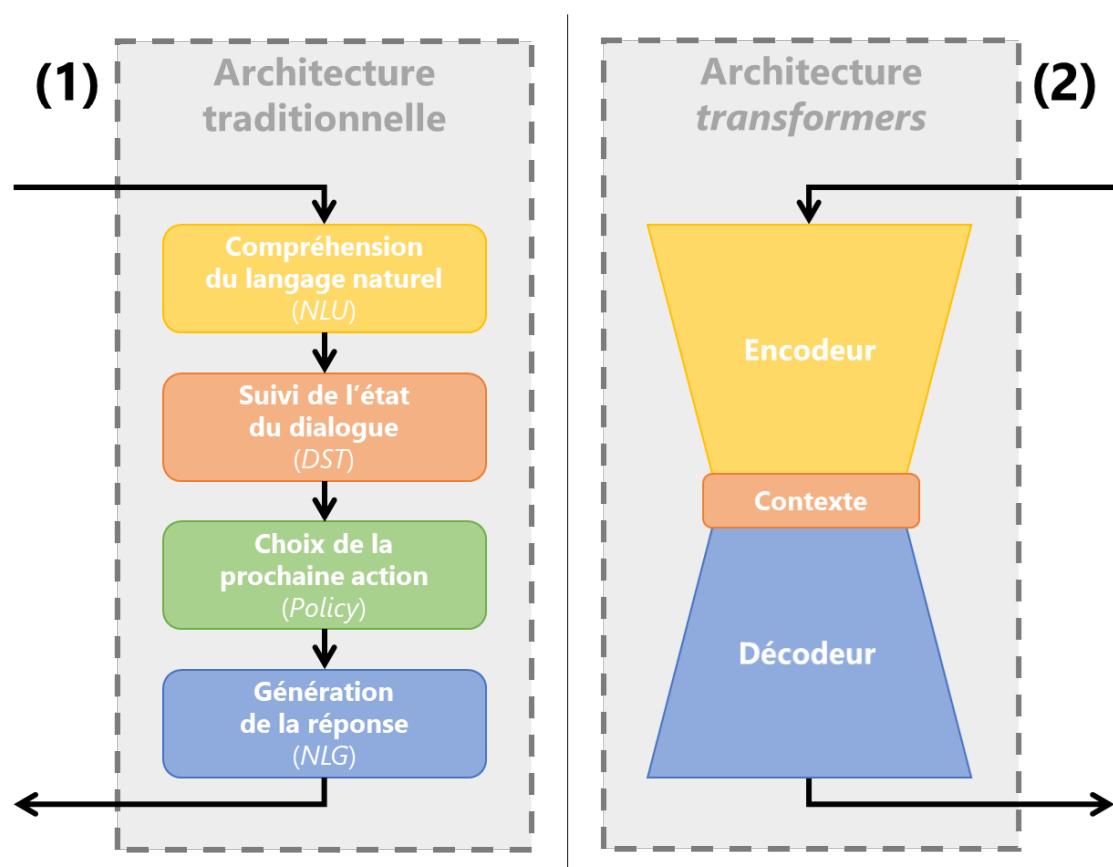


FIGURE B.1 – Schéma illustrant les deux architectures usuelles pour implémenter un assistant conversationnel : (1) représente les *approches symboliques* avec un schéma d'architecture de gestion d'états de dialogue, et (2) représente les *approches génératives* avec un schéma d'architecture à base de transformers, composée d'un encodeur et d'un décodeur.

Annexe C

Annexe des implémentations de notre *clustering interactif*

Au cours de ce doctorat, nous avons réalisé un ensemble d'implémentations en Python afin de mettre en oeuvre notre méthodologie de *clustering interactif*. Celle-ci est répartie en trois librairies :

1. `cognitivefactory-interactive-clustering`⁵² (SCHILD, 2022a), regroupant les gestions de données et des contraintes, les algorithmes de *clustering* et d'échantillonnage ;
2. `cognitivefactory-features-maximization-metric`⁵³ (SCHILD, 2023), disposant d'une méthode de sélection des patterns linguistiques pertinents d'un jeu de données labellisées, permettant ainsi d'analyser la pertinence d'un résultat de *clustering* ;
3. `cognitivefactory-interactive-clustering-gui`⁵⁴ (SCHILD, TREMBLE et MISIAK, 2022), intégrant la logique de la méthodologie dans une application web.

i Pour information : Ces implémentations sont disponibles sur le GitHub <https://github.com/cognitivefactory>. Les *pipeline* d'intégration continue contiennent les étapes de formatage du code (*grâce aux librairies isort*⁵⁵ et *black*⁵⁶), de vérification de la qualité et du typage du code (*grâce aux librairies flake8*⁵⁷ et *mypy*⁵⁸), de la vérification des vulnérabilités et des failles de sécurité (*grâce à la librairie safety*⁵⁹), l'exécution de tests unitaires et la vérification de la couverture du code testé (*grâce aux librairies pytest*⁶⁰ et *coverage*⁶¹), et la génération de la documentation technique (*grâce à la librairie mkdocs*⁶²).

52. <https://pypi.org/project/cognitivefactory-interactive-clustering/>
53. <https://pypi.org/project/cognitivefactory-features-maximization-metric/>
54. <https://pypi.org/project/cognitivefactory-interactive-clustering-gui/>
55. `isort` : <https://pypi.org/project/isort/>
56. `black` : <https://pypi.org/project/black/>
57. `flake8` : <https://pypi.org/project/flake8/>
58. `mypy` : <https://pypi.org/project/mypy/>
59. `safety` : <https://pypi.org/project/safety/>
60. `pytest` : <https://pypi.org/project/pytest/>
61. `coverage` : <https://pypi.org/project/coverage/>
62. `mkdocs` : <https://pypi.org/project/mkdocs/>

Dans cette annexe, nous allons détailler ces implémentations, leurs fonctionnalités et certains des choix de mises en oeuvres.

Sommaire

C.1	Table des nomenclatures utilisées	162
	C.2	Implémentation de la librairie
	cognitivefactory-interactive-clustering	163
C.2.1	Gestion des données	163
C.2.2	Gestion des contraintes	165
C.2.3	Algorithme de <i>clustering</i> sous contraintes	167
C.2.4	Algorithme d'échantillonnage de contraintes	169
	C.3	Implémentation de la librairie
	cognitivefactory-features-maximization-metric	171
	C.4	Implémentation de l'application web
	cognitivefactory-interactive-clustering-gui	172

C.1 Table des nomenclatures utilisées

Symbol	Définition	Référence
prep.no		
prep.simple		
prep.lemma		
prep.filter		
vect.tfidf		
vect.frcorenewsmd		
clust.kmeans.cop		
clust.kmeans.mpc		
clust.hier.sing		
clust.hier.comp		
clust.hier.avg		
clust.hier.ward		
clust.spec		
clust.cdbscan		
clust.affprop		
samp.rand.full		
samp.rand.same		
samp.farhtest.same		
samp.closest.diff		

TABLE C.1 – *Table de nomenclatures des noms d'algorithmes utilisés.*

Symbol	Définition	Référence
<code>dataset_size</code>		
<code>algorithm_nb_constraints</code>		
<code>previous_nb_constraints</code>		
<code>algorithm_nb_clusters</code>		
<code>previous_nb_clusters</code>		

TABLE C.2 – Table de nomenclatures des noms de variables et de paramètres utilisés.

C.2 Implémentation de la librairie cognitivefactory-interactive-clustering

La librairie cognitivefactory-interactive-clustering (SCHILD, 2022a) ...

introduction à rédiger

i Pour information : La documentation technique est accessible au lien suivant : <https://cognitivefactory.github.io/interactive-clustering/>.

Pour les sections suivantes, nous suivrons l'exemple suivant (cf. CODE C.1) pour présenter nos implémentations.

```
1 # Définir les données.
2 dict_of_texts = {
3     "0": "Comment signaler un vol de carte bancaire ?",
4     "1": "J'ai égaré ma carte bancaire, que faire ?",
5     "2": "J'ai perdu ma carte de paiement",
6     "3": "Le distributeur a avalé ma carte !",
7     "4": "En retirant de l'argent, le GAB a gardé ma carte...",
8     "5": "Le distributeur ne m'a pas rendu ma carte bleue.",
9     "# ..."
10    "N": "Pourquoi le sans contact ne fonctionne pas ?",
11 }
```

CODE C.1 – Jeu exemple pour présenter notre implémentation du clustering interactif.

C.2.1 Gestion des données

Tout d'abord, en ce qui concerne la **manipulation de données**, nous utilisons le module `utils` de la librairie cognitivefactory-interactive-clustering. Les données sont stockées dans un dictionnaire Python afin de tracer les manipulations à l'aide d'une clé servant d'identifiant de la donnée.

Nous avons d'une part la partie `utils.preprocessing`⁶³ qui permet de normaliser les données. Par défaut :

63. `utils.preprocessing` : https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/utils/preprocessing/

- le texte est passé en *minuscule* (de "Bonjour" à "bonjour"),
- la *ponctuation* est supprimée,
- les *accents* sont enlevés (de "crédit" à "credit"),
- et les multiples *espaces blancs* sont convertis en un unique espace simple (de "au revoir" à "au revoir").

Si besoin, trois options "avancées" sont disponibles pour réaliser un prétraitement plus destructif :

- la suppression des mots vides (NOTHMAN et al., 2018),
- la conversion des mots vers leur forme racine (MANNING et SCHÜTZE, 2000),
- et la suppression des mots en fonction de leur profondeur dans l'arbre de dépendances syntaxiques (NIVRE, 2006).

Ces traitements sont réalisés en bénéficiant des fonctionnalités mises à disposition d'un modèle de langue de type SpaCy (HONNIBAL et MONTANI, 2017), avec par défaut l'utilisation du modèle **fr-core-news-md**.

Pour nos études, nous définissons quatre niveaux de prétraitements facilement identifiables :

1. **L'absence de prétraitement**, soit la conservation de la donnée brute, noté **prep.no** ;
2. **Le prétraitement simple**, correspondant au traitement de base (minuscules, ponctuations, accents, espaces blancs), noté **prep.simple** ;
3. **Le prétraitement lemmatisé**, correspondant au traitement de base auquel s'ajoute la lemmatisation des mots, noté **prep.lemma** ;
4. **le prétraitement avec filtres**, correspondant au traitement de base avec l'élagage de l'arbre de dépendance syntaxique de la phrase, noté **prep.filter**.

D'autre part, la partie `utils.vectorization`⁶⁴ permet de transformer les données en une représentation exploitable pour la machine. Deux modes de vectorisation sont mis à disposition :

1. **TF-IDF** (RAMOS, 2003), utilisant la fréquence d'occurrence des mots pour représenter une phrase, et noté `vect.tfidf` pour nos études ;
2. **SpaCy** (HONNIBAL et MONTANI, 2017), utilisant le modèle de langue **fr-core-news-md**, et noté `vect.frcorenewsmd`.

Vous avez un exemple d'utilisation des modules de prétraitements et de vectorisation dans CODE C.2.

```
1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.utils.preprocessing
3     import preprocess
3 from cognitivefactory.interactive_clustering.utils.vectorization
4     import vectorize
4
5 # Prétraitement des données.
```

64. `utils.vectorization` : https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/utils/vectorization/

```

6 dict_of_preprocess_texts = preprocess(
7 dict_of_texts=dict_of_texts,
8 apply_stopwords_deletion=False,
9 apply_parsing_filter=False,
10 apply_lemmatization=False,
11 spacy_language_model="fr_core_news_md",
12 )
13 """
14 {"0": "comment signaler un vol de carte bancaire",
15 "1": "j ai egare ma carte bancaire , que faire",
16 "2": "j ai perdu ma carte de paiement",
17 "3": "le distributeur a avale ma carte",
18 "4": "en retirant de l argent le gab a garde ma carte",
19 "5": "le distributeur ne m a pas rendu ma carte bleue",
20 "# ...
21 "N": "pourquoi le sans contact ne fonctionne pas"}
22 """
23
24 # Vectorisation des données.
25 dict_of_vectors = vectorize(
26 dict_of_texts=dict_of_preprocess_texts,
27 vectorizer_type="tfidf",
28 )

```

CODE C.2 – Démonstration de notre implémentation du prétraitement et de la vectorisation sur le jeu d'exemple.

C.2.2 Gestion des contraintes

En ce qui concerne la **manipulation de contraintes**, nous utilisons le module `constraints`⁶⁵ de la librairie `cognitivefactory-interactive-clustering`.

Deux types de contraintes sont prises en charge (cf. WAGSTAFF et CARDIE, 2000) :

- les contraintes **MUST-LINK** permettant de réunir deux données,
- et les contraintes **CANNOT-LINK** permettant à l'inverse de les séparer.

Ces types de contraintes respectent les propriétés de transitivités décrites dans l'EQUATION C.1) et sont illustrées dans la FIGURE C.1 ((1) et (2)). On note ainsi qu'il est possible de déduire la troisième contrainte d'un triangle de trois points si nous connaissons déjà les deux premières.

$$(\forall A, B, C) \left\{ \begin{array}{l} \text{MUST_LINK}(A, B) \wedge \text{MUST_LINK}(B, C) \Rightarrow \text{MUST_LINK}(A, C) \\ \text{MUST_LINK}(A, B) \wedge \text{CANNOT_LINK}(B, C) \Rightarrow \text{CANNOT_LINK}(A, C) \end{array} \right. \quad (\text{C.1})$$

Pour respecter ces propriétés, le gestionnaire de contraintes doit ainsi calculer les transitivités à chaque ajout ou suppression de contraintes. On distinguera donc une contrainte ajoutée (`added`) d'une contrainte déduite par transitivité (`inferred`).

⁶⁵. `constraints` : https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/constraints/

Il se peut que la contrainte en cours d'ajout contredise les contraintes précédemment déduites : nous parlons alors d'incohérence ou de conflit (cf. FIGURE C.1 et EQUATION C.2). Dans ce cas, l'ajout de la dernière contrainte n'est pas prise en compte et le gestionnaire renvoie une erreur permettant d'identifier ce conflit. Ce conflit peut venir simplement venir d'une erreur d'inattention, mais peut aussi venir d'une déduction basée sur des ajouts antérieurs erronés. Sémantiquement, un conflit indique une contradiction dans la gestion des données, car les données concernées doivent à la fois être réunies et séparées...

$$(\exists A, B, C) \text{ MUST_LINK}(A, B) \wedge \text{MUST_LINK}(B, C) \wedge \text{CANNOT_LINK}(A, C) \quad (\text{C.2})$$

À partir d'une donnée D , et par application de la propriété de transitivité des MUST-LINK, nous appelons **composant connexe** de D l'ensemble des données D_i liées par une succession de contraintes MUST-LINK à D (cf. FIGURE C.1). Ce composant peut être vu comme un noyau de *cluster*. Il pourra être associé à d'autres noyaux par similarité pour former un *cluster* plus conséquent, ou être distingué d'autres noyaux pour former plusieurs *clusters*.

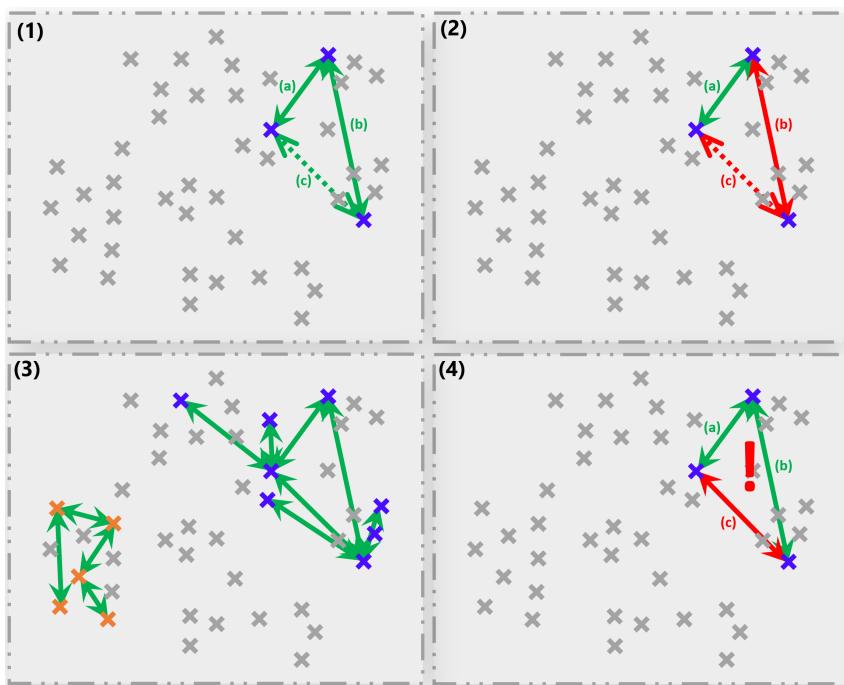


FIGURE C.1 – Exemples des propriétés de transitivité des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges). (1) et (2) représente les possibilités de déduction d'une contrainte ((c)) en fonction des deux autres ((a) et (b)). (3) représente deux composants connexes définis par la transitivité des contraintes MUST-LINK. Enfin, (4) représente un cas de conflit où une contrainte ((c)) ne correspond pas à sa déduction faite à partir des autres contraintes ((a) et (b)).

Un exemple d'utilisation du module de gestion de contraintes est consultable dans CODE C.3.

```

1 # Import des dépendances .
2 from cognitivefactory.interactive_clustering.constraints.factory
  import managing_factory
3
4 # Création du gestionnaire de contraintes .

```

```

5 constraints_manager = managing_factory(
6 manager="binary",
7 list_of_data_IDs = list(dict_of_texts.keys()), # ["0", "1", "2",
8     "3", "4", "5", ..., "N"]
9 )
10 # Ajout de contraintes.
11 constraints_manager.add_constraint(
12 data_ID1="0", # "Comment signaler un vol de carte bancaire ?"
13 data_ID2="1", # "J'ai égaré ma carte bancaire, que faire ?"
14 constraint_type="MUST_LINK",
15 )
16 constraints_manager.add_constraint(
17 data_ID1="3", # "Le distributeur a avalé ma carte !"
18 data_ID2="4", # "En retirant de l'argent, le GAB a gardé ma carte
19     ...
20 constraint_type="MUST_LINK",
21 )
22 constraints_manager.add_constraint(
23 data_ID1="0", # "Comment signaler un vol de carte bancaire ?"
24 data_ID2="N", # "Pourquoi le sans contact ne fonctionne pas ?"
25 constraint_type="CANNOT_LINK",
26 )
27 # NB: ajouter une contrainte "MUST_LINK" entre "1" et "N" lèverait
28     une erreur.
29
30 # Récupération des composants connexes.
31 connected_components = constraints_manager.get_connected_components()
32 """
33 [[ '0 ', '1 '],
34 [ '2 '],
35 [ '3 ', '4 '],
36 [ '5 '],
37 [ 'N ']]
38 """

```

CODE C.3 – Démonstration de notre implémentation de gestion des contraintes sur le jeu d'exemple.

C.2.3 Algorithme de *clustering* sous contraintes

En ce qui concerne le **regroupement automatique** des données par similarité, nous utilisons le module **clustering**⁶⁶ de la librairie **cognitivefactory-interactive-clustering**.

Ce module met à disposition six algorithmes de *clustering* sous contraintes :

⁶⁶. clustering : https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/clustering/

1. **KMeans**, dans sa version COP-KMeans (WAGSTAFF et al., 2001), noté `clust.kmeans.cop`, et sa version MPC-KMeans (KHAN et al., 2012), noté `clust.kmeans.mpc`;
2. **DBscan**, dans sa version C-DBScan (RUIZ et al., 2010), noté `clust.cdbscan`;
3. **Hiérarchique** (DAVIDSON et RAVI, 2005), avec quatre métriques de distances : `single` (noté `clust.hier.sing`), `complete` (noté `clust.hier.comp`), `average` (noté `clust.hier.avg`) et `ward` (noté `clust.hier.ward`);
4. **Spectral**, dans sa version SPEC (KAMVAR et al., 2003), noté `clust.spec`;
5. **Propagation par affinité** (GIVONI et FREY, 2009), noté `clust.affprop`.

Une classe abstraite définit les prérequis des algorithmes implémentés (avoir une méthode `cluster`) et une *factory* est disponible pour instancier rapidement un objet de *clustering*. Enfin, un exemple d'utilisation ce module est consultable dans CODE C.4.

```

1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.clustering.factory
3     import clustering_factory
4
5 # Initialiser un objet de clustering.
6 clustering_model = clustering_factory(
7     algorithm="kmeans",
8     model="COP",
9     random_seed=42,
10 )
11
12 # Lancer le clustering.
13 clustering_result = clustering_model.cluster(
14     constraints_manager=constraints_manager, # contient les contraintes
15     nb_clusters=2,
16     vectors=dict_of_vectors,
17 )
18 """
19 {"0": 0, # "Comment signaler un vol de carte bancaire ?"
20  "1": 0, # "J'ai égaré ma carte bancaire, que faire ?"
21  "2": 0, # "J'ai perdu ma carte de paiement"
22  "3": 1, # "Le distributeur a avalé ma carte !"
23  "4": 1, # "En retirant de l'argent, le GAB a gardé ma carte...""
24  "5": 1, # "Le distributeur ne m'a pas rendu ma carte bleue."
25  "# ..."
26  "N": 1} # "Pourquoi le sans contact ne fonctionne pas ?"
27 """

```

CODE C.4 – Démonstration de notre implémentation du clustering sous contraintes sur le jeu d'exemple.

i Pour information : Dans le cadre d'un projet étudiant avec l'école d'ingénieur Télécom Physique Strasbourg, les implémentations des algorithmes MPC-KMeans, C-DBScan

et propagation par affinité ont été ajoutées. Les élèves ont conclu ce projet d'extension en suggérant de se concentrer sur l'étude du C-DBScan car les deux autres algorithmes étaient soit trop instables, soit trop gourmand en temps de calcul. Les autres algorithmes (**COP-KMeans**, hiérarchique et spectral) ont été implémentés au début de ce doctorat.

C.2.4 Algorithme d'échantillonnage de contraintes

En ce qui concerne l'**échantillonnage** de contraintes à annoter, nous utilisons le module `sampling`⁶⁷ de la librairie `cognitivefactory-interactive-clustering`.

Cet échantillonnage correspond à la sélection de couple de données. Par défaut, l'échantillonnage est purement aléatoire. Cependant, plusieurs options sont disponibles :

- une restriction sur la *distance* pouvant imposer aux données d'être les plus proches ou les plus éloignées du corpus ;
- une restriction sur le *résultat du clustering* pouvant imposer aux données d'être issues d'un même cluster ou de clusters différents,
- une restriction pour exclure les contraintes *déjà annotées*,
- et enfin une restriction pour exclure les contraintes *déjà déduites* par transitivité.

Sur cette base, nous définissons quatre niveaux d'échantillonnage facilement identifiables pour nos études :

1. Un échantillonnage **purement aléatoire** en excluant toutes les contraintes déjà annotées ou déduites, noté `samp.random.full` ;
2. Un échantillonnage **pseudo-aléatoire** de données issues d'un **même cluster**, en excluant toutes les contraintes déjà annotées ou déduites, noté `samp.random.same` ;
3. Un échantillonnage des données issues d'un **même cluster** et étant **les plus éloignées** les unes des autres, noté `samp.farhtest.same` (cf. FIGURE C.2) ;
4. Un échantillonnage des données issues de **clusters différents** et étant **les plus proches** les unes des autres, noté `samp.closest.diff` (cf. FIGURE C.2).

Une classe abstraite définit les prérequis des algorithmes implémentés (avoir une méthode `sample`) et une *factory* est disponible pour instancier rapidement un objet d'échantillonnage. Un exemple d'utilisation ce module est consultable dans CODE C.5.

```

1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.sampling.factory import
   sampling_factory
3
4 # Initialiser un objet d'échantillonnage.
5 sampler = sampling_factory(
6   algorithm="random",
7   random_seed=42,

```

⁶⁷. `sampling` : https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/sampling/



FIGURE C.2 – Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (*samp.farhest.same*), et de données issues de clusters différents et étant les plus proches les unes des autres (*samp.closest.diff*).

```

8 )
9
10 # Run sampling.
11 selection = sampler.sample(
12 constraints_manager=constraints_manager,
13 nb_to_select=2,
14 clustering_result=clustering_result, # optionnel pour "random"
15 vectors=dict_of_vectors, # optionnel pour "random"
16 )
17 """
18 [ ("0" , '5") , # "Comment signaler un vol de carte bancaire ?" vs "Le
19     distributeur ne m'a pas rendu ma carte bleue."
20 ("0" , '2") , # "Comment signaler un vol de carte bancaire ?" vs "J'
21     ai perdu ma carte de paiement"
22 ("2" , 'N")] # "J'ai perdu ma carte de paiement" vs "Pourquoi le
23     sans contact ne fonctionne pas ?"
24 """

```

CODE C.5 – Démonstration de notre implémentation de l'échantillonnage sur le jeu d'exemple.

C.3 Implémentation de la librairie `cognitivefactory-features-maximization-metric`

La librairie `cognitivefactory-features-maximization-metric` (SCHILD, 2023) ...

introduction à rédiger

i Pour information : La documentation technique est accessible au lien suivant :
<https://cognitivefactory.github.io/features-maximization-metric/>.

C.4 Implémentation de l'application web cognitivefactory-interactive-clustering-gui

la librairie texttcognitivefactory-interactive-clustering-gui (SCHILD, TREMBLE et MISIAK, 2022) a été implémenté au cours de ce doctorat dans le but d'intégrer notre méthodologie de *clustering* interactif au sein d'une application web. Celle-ci dispose de plusieurs fonctionnalités telles que :

- la gestion du projet, de ses données et de ses paramétrages ;
- la gestion et l'annotation de contraintes, ainsi que la vérification des propriétés de transitivités ;
- la gestion des étapes d'une itération de la méthode ;
- l'exécution asynchrone des divers algorithmes de la méthode (prétraitement, vectorisation, clustering, échantillonnage) ;
- quelques scripts d'analyses.

Nous présenterons succinctement cette application ci-dessous à l'aide de captures d'écrans.

i Pour information : La documentation technique est accessible au lien suivant : <https://cognitivefactory.github.io/interactive-clustering-gui/>.

💬 Notes de l'auteur : Suite aux diverses études menées au cours de ce doctorat, certaines pages sont en cours de refonte, notamment les pages d'analyses (suite aux conclusions des SECTION 4.4 et SECTION 4.5) et les pages de documentations (suite à discussion en CHAPITRE 5).

C.4. Implémentation de l'application web cognitivefactory-interactive-clustering-gui

Page d'accueil de l'application (FIGURE C.3) : C'est la page de bienvenu de l'application. Nous y trouvons une description rapide de la méthode ainsi qu'une liste des questions fréquentes à son sujet. Le bouton en haut à gauche permettra toujours de revenir sur cette page.

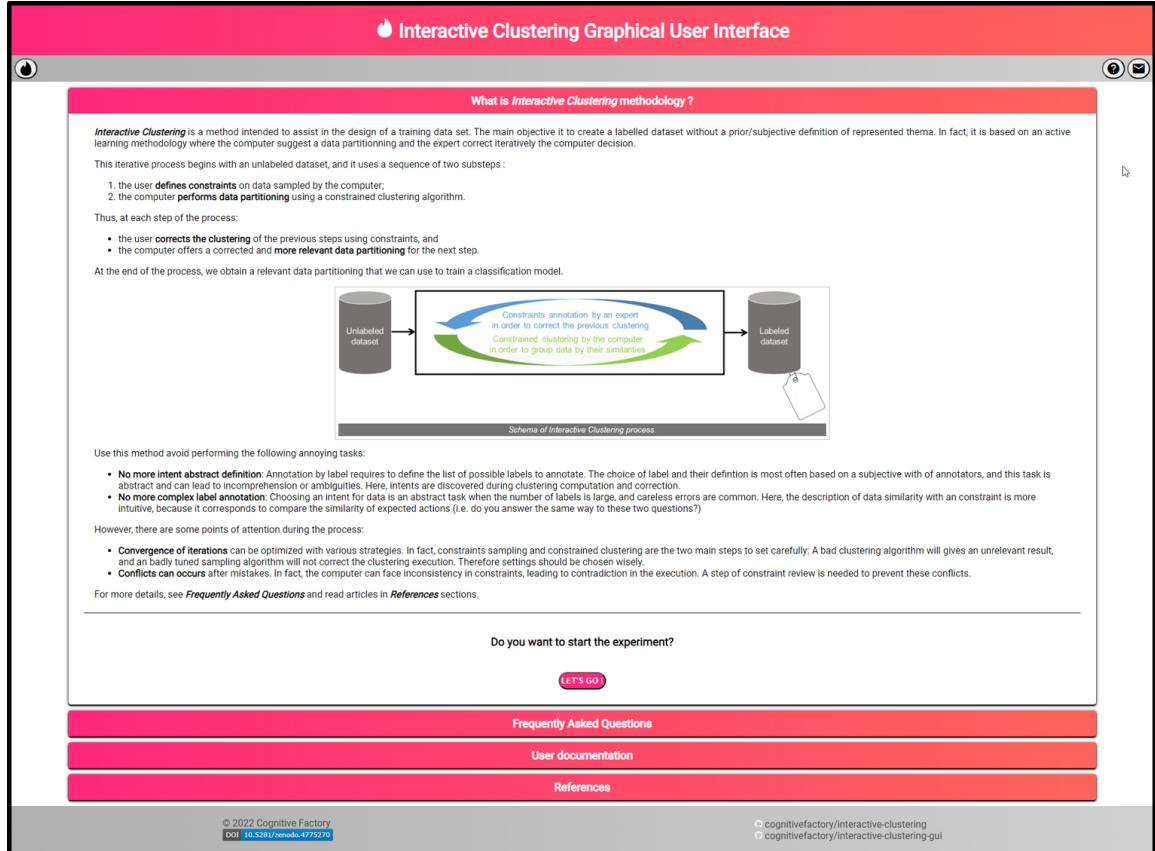


FIGURE C.3 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page d'accueil de l'application.

Annexe C. Annexe des implémentations de notre clustering interactif

Page de gestion des projets (FIGURE C.4) : Cette page liste les projets existants sous la forme de tuiles contenant les informations importantes : nom, date de création, nombre d’itérations de la méthode, et statut du projet (nous y reviendrons plus tard). Il est possible de télécharger un projet au format .zip ou de le supprimer. Pour créer un projet, le bouton ADD NEW ouvre un petit formulaire demandant le nom du projet et la liste des textes à annoter (au format .csv séparateur ‘;’). Il est aussi possible d’importer un projet à l’aide d’une archive .zip téléchargée au préalable. Le bouton LOAD mène à la page d’accueil du projet sélectionné.

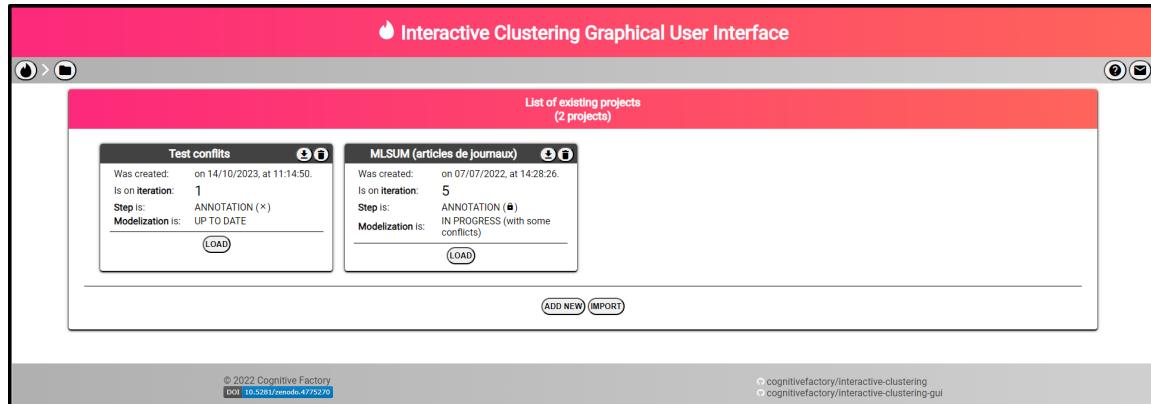


FIGURE C.4 – Capture d’écran de l’application web implémentant notre méthodologie de clustering interactif : page de gestion des projets.

C.4. Implémentation de l'application web cognitivefactory-interactive-clustering-gui

Page d'accueil du projet en cours (FIGURE C.5) :
à rédiger

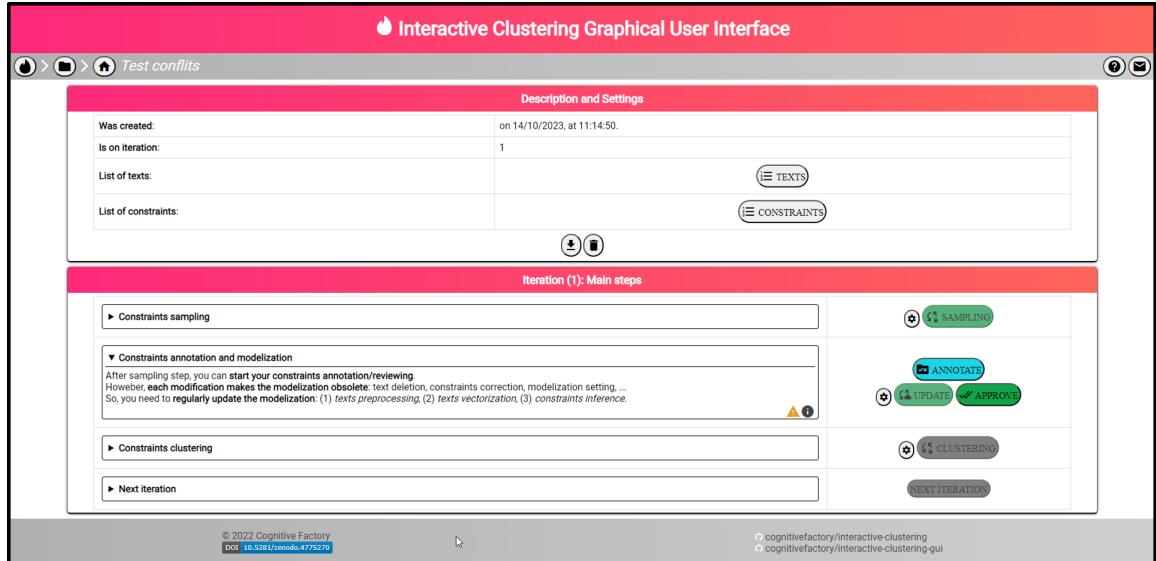


FIGURE C.5 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page d'accueil du projet en cours.

Annexe C. Annexe des implémentations de notre clustering interactif

Diagramme d'états de l'application et gestion des exécutions asynchrones (FIGURE C.6) :

à rédiger

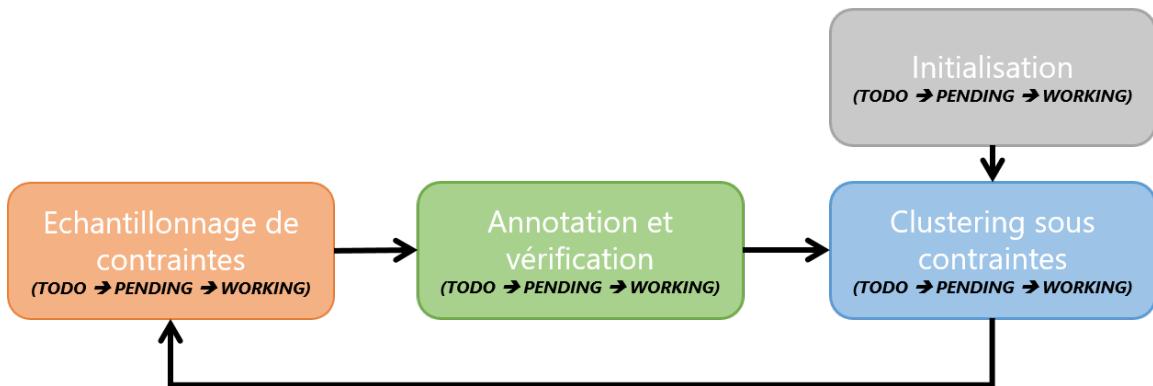


FIGURE C.6 – *Diagramme d'états simplifié de l'application web implémentant notre méthodologie de clustering interactif.*

Page de gestion des paramètres (FIGURE C.7) :
à rédiger

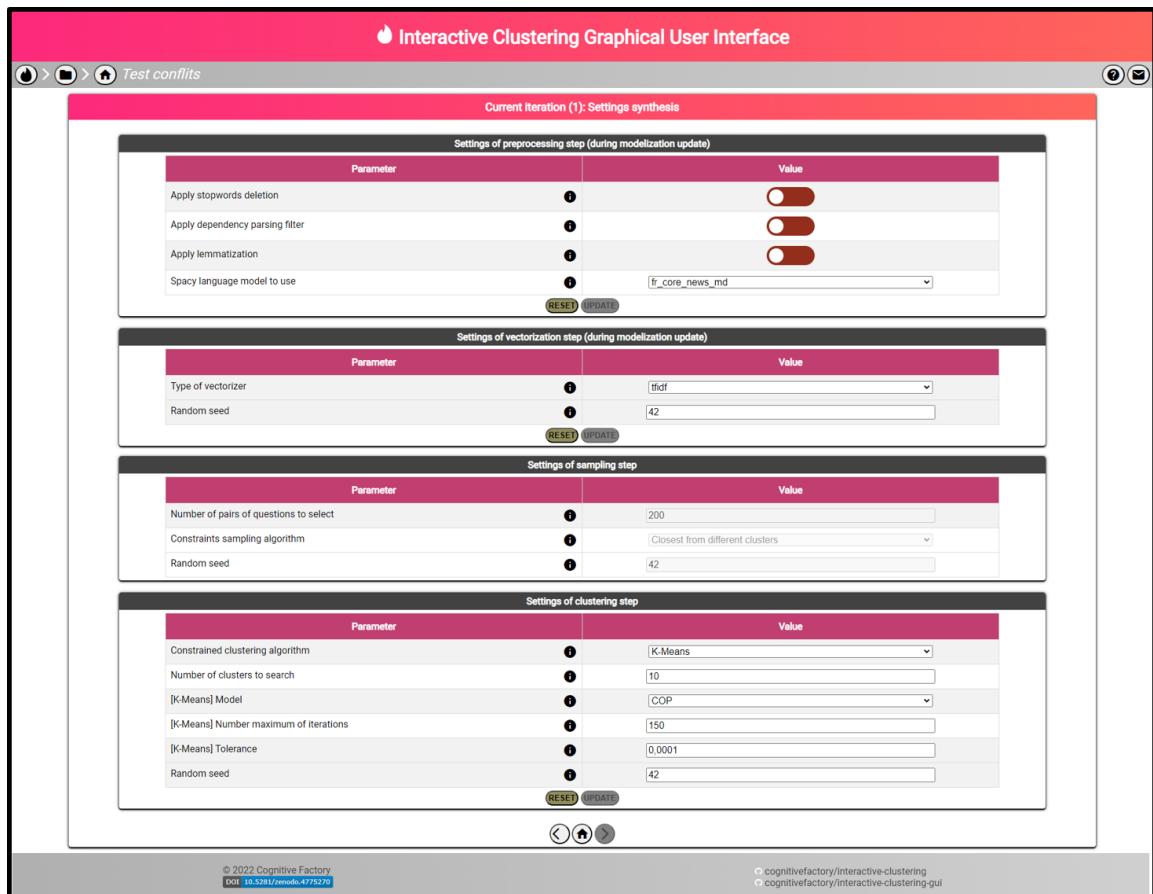


FIGURE C.7 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page de gestion des paramètres.

Annexe C. Annexe des implémentations de notre clustering interactif

Page d'inventaire des textes (FIGURE C.8) :

à rédiger

The screenshot shows a web application window titled "Interactive Clustering Graphical User Interface". The main title bar has a red background with white text. Below it, a header bar is also red with white text. The main content area has a pink header "Iteration (1): Texts synthesis". Underneath, there's a table with three rows: "Number of texts: 498 texts", "Number of constraints: 184 constraints and 12 need your annotation.", and "Modelization state: Modelization is up to date. Any modification will lead to an outdated modelization." To the right of the table are two buttons: "CONSTRAINTS" and "UPDATE". Below the table is a section titled "List of texts" with a table header "Sort by: default | descending". The table has three columns: "Preprocess text", "Current text", and "Deleted". There are seven rows of text pairs, each with edit and delete icons. The "Deleted" column contains checkboxes, some of which are checked.

Preprocess text	Current text	Deleted
« a combien s'eleve le solde de mon compte »	« À combien s'élève le solde de mon compte ? »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« hello there »	« Hello there ! »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« general kenobi you are a bold one »	« General Kenobi... you are a bold one ! »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« comment obtenir une autorisation de découvert »	« Comment obtenir une autorisation de découvert ? »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« comment obtenir une carte virtuelle pour mes achats sur internet »	« Comment obtenir une carte virtuelle pour mes achats sur internet ? »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« comment obtenir une mastercard »	« Comment obtenir une Mastercard ? »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« comment peut on commander une cartes de paiement »	« Comment peut-on commander une cartes de paiement ? »	<input type="checkbox"/> <input checked="" type="checkbox"/>
« comment pouvoir payer d'avantage avec sa carte visa »	« Comment pouvoir payer d'avantage avec sa carte Visa ? »	<input type="checkbox"/> <input checked="" type="checkbox"/>

FIGURE C.8 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page d'inventaire des textes.

Page d'inventaire des contraintes (FIGURE C.9) :
à rédiger

The screenshot shows a web application window titled "Interactive Clustering Graphical User Interface". The main title bar has a red background with white text. Below it, the window is divided into sections:

- Iteration (1): Constraints synthesis**: A header section with statistics: "Number of texts: 500 texts", "Number of constraints: 186 constraints and 13 need your annotation.", and a note: "Modelization is up to date. Any modification will lead to an outdated modelization. You can approve your work if all annotations are done." It includes buttons for "TEXTS", "ANNOTATE", "APPROVE", and "UPDATE".
- List of constraints**: A table with columns: Text 1, Constraint, Text 2, Sampling iteration, Last update, To annotate, To review, To fix, and Go to. The table lists several constraint pairs with their status (e.g., PAIN, MUST, SKIP) and annotations.

Text 1	Constraint	Text 2	Sampling iteration	Last update	To annotate	To review	To fix	Go to
« puis je avoir une autorisation de decouvert »	PAIN	« puis je avoir une nouvelle mastercard »	1	14/10/2023, at 11:20:54.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(i)
« comment changer de carte bleue »	MUST	« comment changer de carte de credit s il vous plait »	1	14/10/2023, at 11:22:10.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(i)
« quel est le montant de decouvert possible »	SKIP	« quel est le montant maximal que je peux retirer avec ma carte bancaire »	1	14/10/2023, at 11:37:29.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	?	(i)
« gerer la limite de depense d une carte de credit »	SKIP	« changer la limite de depense de ma carte bleue »	1	Never	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	?	(i)
« je suis victime d un vol de carte bancaire »	MUST	« je suis victime d un vol de carte »	1	14/10/2023, at 11:18:19.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(i)
« je me suis fait voler ma carte bancaire »	MUST	« je me suis fait voler ma carte bleue »	1	14/10/2023, at 11:18:21.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(i)
« comment deverrouiller sa carte »	MUST	« comment deverrouiller sa carte bancaire »	1	14/10/2023, at 11:19:22.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	(i)

FIGURE C.9 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page d'inventaire des contraintes.

Annexe C. Annexe des implémentations de notre clustering interactif

Page d'annotation d'une contrainte (FIGURE C.10 et FIGURE C.11) :
à rédiger

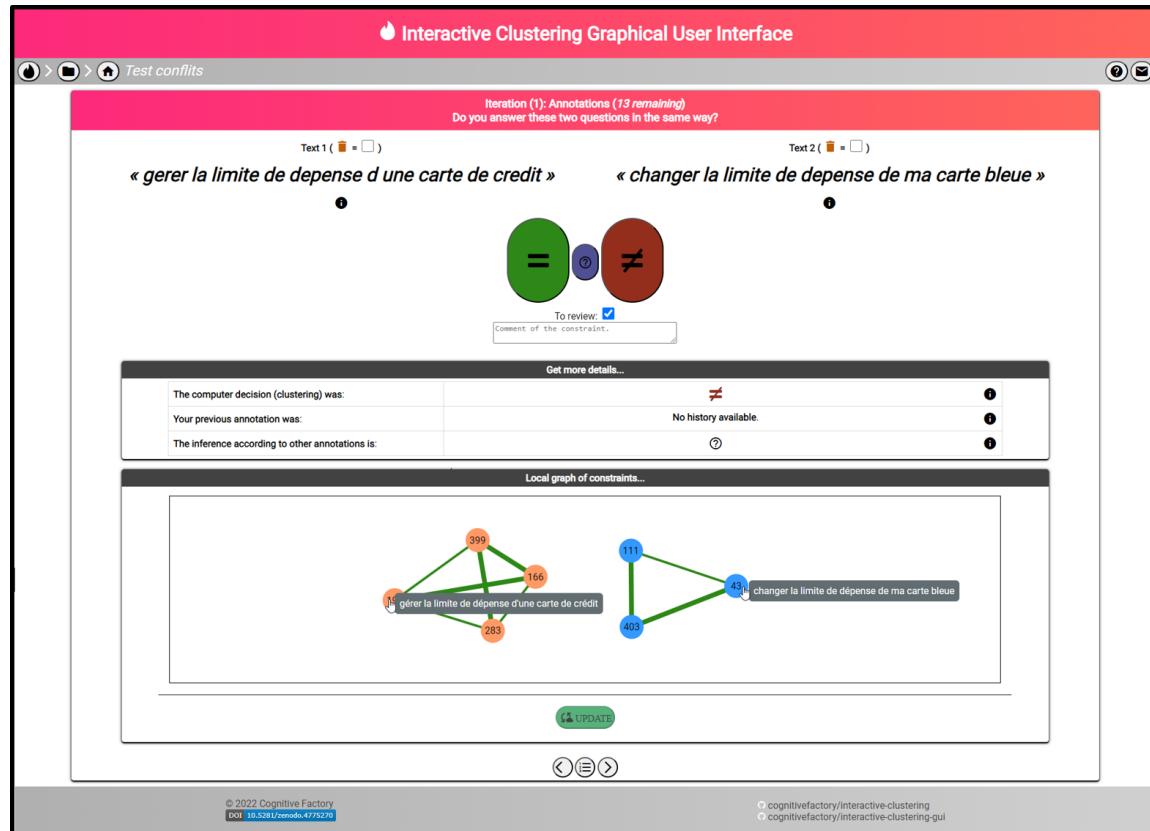


FIGURE C.10 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : page d'annotation d'une contrainte.

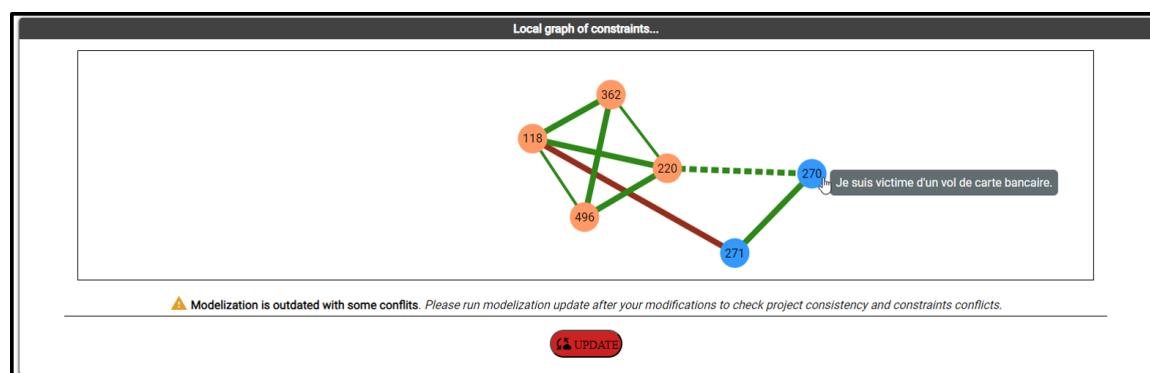


FIGURE C.11 – Capture d'écran de l'application web implémentant notre méthodologie de clustering interactif : graphe de contraintes présentant un conflit d'annotation.

Annexe D

Annexe technique

à rédiger

Sommaire

D.1	v-measure : métrique d'évaluation de <i>clustering</i>	181
-----	--	-----

D.1 v-measure : métrique d'évaluation de *clustering*

ANNEXE : v-measure

Bibliographie

- ADAMOPOULOU, E., & MOUSSIADES, L. (2020). An Overview of Chatbot Technology. In I. MAGLOGIANNIS, L. ILIADIS & E. PIMENIDIS (Éd.), *Artificial Intelligence Applications and Innovations* (p. 373-383). Springer International Publishing. <https://doi.org/10.ghj8>
- AGARWAL, P., ALAM, M. A., & BISWAS, R. (2011). Issues, Challenges and Tools of Clustering Algorithms. *IJCSI International Journal of Computer Science Issues*, 8(3). <https://doi.org/10.48550/ARXIV.1110.2610>
- AIZED AMIN SOOFI & ARSHAD AWAN. (2017). Classification Techniques in Machine Learning : Applications and Issues. *J. Basic Appl. Sci.*, 13, 459-465. <https://doi.org/10.6000/1927-5129.2017.13.76>
- ALAMMAR, J., & GREFENSTETTE, E. (2022). *Cohere Sandbox*. <https://github.com/cohere-ai/sandbox-topically>
- ALASADI, S. A., & BHAYA, W. S. (2017). Review of Data Preprocessing Techniques in Data Mining. *Journal of Engineering and Applied Sciences*, 12(16), 4102-4107. <https://www.academia.edu/download/54509277/4102-4107.pdf>
- ALEXA INTERNET. (2018). Keyword Research, Competitor Analysis, & Website Ranking | Alexa. <https://www.alexa.com>
- ANDERSON, J. R. (2013, novembre 19). *The Architecture of Cognition* (0^e éd.). Psychology Press. Récupérée juin 7, 2023, à partir de <https://www.taylorfrancis.com/books/9781317759539>
- ARTSTEIN, R., & POESIO, M. (2008). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4), 555-596. <https://doi.org/10.1162/coli.07-034-R2>
- ASHER, N., NASR, A., & PERROTIN, R. (2017). Manuel d'annotation en actes de dialogue pour le corpus Datcha.
- AUDACITY TEAM. (2000). *Audacity : Free Audio Editor and Recorder*. <https://www.audacityteam.org/>
- AWEL, M. A., & ABIDI, A. I. (2019). Review on Optical Character Recognition. 06(06).
- BAE, J., HELLDIN, T., RIVEIRO, M., NOWACZYK, S., BOUGUELIA, M.-R., & FALKMAN, G. (2021). Interactive Clustering : A Comprehensive Review. *ACM Comput. Surv.*, 53(1), 1-39. <https://doi.org/10.1145/3340960>
- BALEDENT, A. (2023, décembre 1). *De la complexité de l'annotation manuelle : méthodologie, biais et recommandations*. <https://theses.hal.science/tel-04011353>
- BAYERL, P. S., & PAUL, K. I. (2011). What Determines Inter-Coder Agreement in Manual Annotations ? A Meta-Analytic Investigation. *Computational Linguistics*, 37(4), 699-725. https://doi.org/10.1162/COLI_a_00074
- BERCHMANS, D., & KUMAR, S. S. (2014). Optical Character Recognition : An Overview and an Insight. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 1361-1365. <https://doi.org/10.1109/ICCICCT.2014.6993174>

BIBLIOGRAPHIE

- BIBER, D. (1993). Representativeness in Corpus Design. *Literary and Linguistic Computing*, 8(4). <https://doi.org/10.1093/llc/8.4.243>
- BLEI, D. M., NG, A. Y., & JORDAN, M. I. (2003). Latent Dirichlet Allocation. *Journal of machine Learning research*, (3), 993-1022.
- BÖHMOVÁ, A., HAJIČ, J., HAJIČOVÁ, E., & HLADKÁ, B. (2003). The Prague Dependency Treebank. In A. ABEILLÉ (Éd.). N. IDE & J. VÉRONIS (typeredactors), *Treebanks* (p. 103-127, T. 20). Springer Netherlands. Récupérée septembre 21, 2023, à partir de http://link.springer.com/10.1007/978-94-010-0201-1_7
- BRABRA, H., BAEZ, M., BENATALLAH, B., GAALOUL, W., BOUGUELIA, S., & ZAMANIRAD, S. (2022). Dialogue Management in Conversational Systems : A Review of Approaches, Challenges, and Opportunities. *IEEE Trans. Cogn. Dev. Syst.*, 14(3), 783-798. <https://doi.org/10.1109/TCDS.2021.3086565>
- BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D. M., WU, J., WINTER, C., ... AMODEI, D. (2020). Language Models Are Few-Shot Learners. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2005.14165>
- BRYSBERT, M. (2019). How many words do we read per minute ? A review and meta-analysis of reading rate. *Journal of Memory and Language*, 109, 104047. <https://doi.org/10.1016/j.jml.2019.104047>
- CALLISON-BURCH, C., & DREDZE, M. (2010). Creating Speech and Language Data with Amazon's Mechanical Turk. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 1-12. <https://aclanthology.org/W10-0701>
- CHEN, H., LIU, X., YIN, D., & TANG, J. (2017). A Survey on Dialogue Systems : Recent Advances and New Frontiers. *SIGKDD Explor. Newsl.*, 19(2), 25-35. <https://doi.org/10.1145/3166054.3166058>
- CLEMMENSEN, L. H., & KJAERSGAARD, R. D. (2022). Data Representativity for Machine Learning and AI Systems. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2203.04706>
- COLLINS, W. (2017, avril). Chapter 7 : Overfitting. In *Algorithms To Live By : The Computer Science of Human Decisions* (p. 149-168).
- CORTES, C., & VAPNIK, V. (1995). Support-vector networks. *Mach Learn*, 20(3), 273-297. <https://doi.org/10.1007/BF00994018>
- COSTELLO, K., & LoDOLCE, M. (2019). Gartner Top Technologies and Trends Driving the Digital Workplace [newspaper]. *Gartner, Inc.* Récupérée octobre 23, 2020, à partir de <https://www.gartner.com/smarterwithgartner/top-10-technologies-driving-the-digital-workplace/>
- COSTELLO, K., & LoDOLCE, M. (2022). Gartner Predicts Chatbots Will Become a Primary Customer Service Channel Within Five Years [newspaper]. *Gartner, Inc.* Récupérée octobre 9, 2023, à partir de <https://www.gartner.com/en/newsroom/press-releases/2022-07-27-gartner-predicts-chatbots-will-become-a-primary-customer-service-channel-within-five-years>
- CREATIVE COMMONS. (2013). *CC BY-NC 4.0 LEGAL CODE - Attribution-NonCommercial 4.0 International*. Récupérée septembre 29, 2023, à partir de <https://creativecommons.org/licenses/by-nc/4.0/legalcode.en>
- CVAT.AI CORPORATION. (2019, octobre 17). *Computer Vision Annotation Tool (CVAT)*. Récupérée septembre 27, 2023, à partir de <https://www.cvat.ai/>

- DAGAN, I., GLICKMAN, O., & MAGNINI, B. (2005). The PASCAL Recognising Textual Entailment Challenge. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, 3944, 177-190. https://doi.org/10.1007/11736790_9
- DANDAPAT, S., BISWAS, P., CHOUDHURY, M., & BALI, K. (2009). Complex Linguistic Annotation – No Easy Way out! A Case from Bangla and Hindi POS Labeling Tasks. *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, 10-18. <https://aclanthology.org/W09-3002>
- DATA BIRD. (2023, juillet). *Les 10 métiers data les plus recherchés en 2023*. DataBird. Récupérée septembre 26, 2023, à partir de <https://www.data-bird.co/blog/metiers-data>
- DAVIDSON, I., & RAVI, S. S. (2005). Agglomerative Hierarchical Clustering with Constraints : Theoretical and Empirical Results (A. M. JORGE, L. TORGÓ, P. BRAZDIL, R. CAMACHO & J. GAMA, Éd.). *Knowledge Discovery in Databases : PKDD 2005*, 3721, 59-70. Récupérée octobre 22, 2020, à partir de http://link.springer.com/10.1007/11564126_11
- DEVLIN, J., CHANG, M.-W., LEE, K., & TOUTANOVA, K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv preprint*. Récupérée juin 10, 2020, à partir de <http://arxiv.org/abs/1810.04805>
- DIAMOND, I., COX, D. R., & SNELL, E. J. (1990). Analysis of Binary Data. 2nd Edn. *Applied Statistics*, 39(2), 260. <https://doi.org/10.2307/2347766>
- DIPPER, S., GOTZE, M., & SKOPETEAS, S. (2004). Towards User-Adaptive Annotation Guidelines. *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, 23-30. <https://aclanthology.org/W04-1904>
- DZIEZA, J. (2023). AI is a lot of work [newspaper]. *New York Magazine : Artificial Intelligence*. Récupérée septembre 29, 2023, à partir de <https://nymag.com/intelligencer/article/ai-artificial-intelligence-humans-technology-business-factory.html>
- EDWARDS, A. W. F. (1992). *Likelihood* (Expanded ed). Johns Hopkins Univ. Press.
- ELKOSANTINI, S., & GIEN, D. (2009). Integration of human behavioural aspects in a dynamic model for a manufacturing system. *International Journal of Production Research*, 47(10), 2601-2623. <https://doi.org/10.1080/00207540701663490>
- ESTER, M., KRIEGEL, H.-P., & XU, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.
- FALKE, T., RIBEIRO, L. F. R., UTAMA, P. A., DAGAN, I., & GUREVYCH, I. (2019). Ranking Generated Summaries by Correctness : An Interesting but Challenging Application for Natural Language Inference. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2214-2220. <https://doi.org/10.18653/v1/P19-1213>
- FINLAYSON, M. A., & ERJAVEC, T. (2016). Overview of Annotation Creation : Processes & Tools. *ArXiv preprint*. Récupérée juin 14, 2021, à partir de <http://arxiv.org/abs/1602.05753>
- FIUMARA, J., CIERI, C., WRIGHT, J., & LIBERMAN, M. (2020). LanguageARC : Developing language resources through citizen linguistics. *Proceedings of the LREC 2020 Workshop on "Citizen Linguistics in Language Resource Development"*, 1-6. <https://aclanthology.org/2020.clrrd-1.1>
- FORT, K. (2017). Experts Ou (Foule de) Non-Experts ? La Question de l'expertise Des Animateurs Vue de La Myriadisation (Crowdsourcing). *corela*. <https://doi.org/10.4000/corela.4835>
- FORT, K., EHREMAN, M., & NAZARENKO, A. (2009). Vers Une Méthodologie d'annotation Des Entités Nommées En Corpus ? *Traitemet Automatique Des Langues Naturelles 2009*. <https://hal.science/hal-00402321>

BIBLIOGRAPHIE

- FORT, K., NAZARENKO, A., & ROSSET, S. (2012). Modeling the complexity of manual annotation tasks : a grid of analysis. *Proceedings of COLING 2012*, 895-910. <https://hal.science/hal-00769631>
- FORT, K., & SAGOT, B. (2010). Influence of Pre-Annotation on POS-Tagged Corpus Development. *Proceedings of the Fourth Linguistic Annotation Workshop*, 56-63. <https://aclanthology.org/W10-1807>
- GARSIDE, R., LEECH, G. N., & MCENERY, T. (Éd.). (1997). *Corpus annotation : linguistic information from computer text corpora*. Longman.
- GIRDEN, E. (1992). ANOVA. SAGE Publications, Inc. Récupérée juillet 6, 2023, à partir de <https://methods.sagepub.com/book/anova>
- GIVONI, I. E., & FREY, B. J. (2009). Semi-Supervised Affinity Propagation with Instance-Level Constraints.
- GOASDUFF, L. (2019). Chatbots Will Appeal to Modern Workers [newspaper]. *Gartner, Inc.* Récupérée octobre 23, 2020, à partir de <https://www.gartner.com/smarterwithgartner/chatbots-will-appeal-to-modern-workers/>
- GOYAL, A., GUPTA, V., & KUMAR, M. (2018). Recent Named Entity Recognition and Classification techniques : A systematic review. *Computer Science Review*, 29, 21-43. <https://doi.org/10.1016/j.cosrev.2018.06.001>
- GUILLAUME, B., FORT, K., & LEFÈBVRE, N. (2016). Crowdsourcing Complex Language Resources : Playing to Annotate Dependency Syntax. *International Conference on Computational Linguistics (COLING)*. <https://inria.hal.science/hal-01378980>
- GUT, U., & BAYERL, P. S. (2004). Measuring the Reliability of Manual Annotations of Speech Corpora. <https://api.semanticscholar.org/CorpusID:27970161>
- HART, S. G., & STAVELAND, L. E. (1988). Development of NASA-TLX (Task Load Index) : Results of Empirical and Theoretical Research. In P. A. HANCOCK & N. MESHKATI (Éd.), *Human Mental Workload* (p. 139-183, T. 52). North-Holland. <https://www.sciencedirect.com/science/article/pii/S0166411508623869>
- HONNIBAL, M., & MONTANI, I. (2017). spaCy 2 : Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing.
- HOWE, J. (2008). *Crowdsourcing : how the power of the crowd is driving the future of business* (Random House Books). RH Business Books.
- HUANG, J., SHAO, H., & CHANG, K. C.-C. (2022). Are Large Pre-Trained Language Models Leaking Your Personal Information ? *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2205.12628>
- HUGGING FACE. (2016). *Hugging Face - the AI Community Building the Future*. <https://huggingface.co/datasets>
- IMAN, M., ARABNIA, H. R., & RASHEED, K. (2023). A Review of Deep Transfer Learning and Recent Advancements. *Technologies*, 11(2), 40. <https://doi.org/10.3390/technologies11020040>
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. (2007, février 16). *Codes for the Representation of Names of Languages – Part 3 : Alpha-3 Code for Comprehensive Coverage of Languages*. <https://www.iso.org/standard/39534.html>
- IWOZ, V. (2017, décembre 20). *Découvrir les métiers de la data science*. LinkedIn Learning. Récupérée septembre 26, 2023, à partir de <https://fr.linkedin.com/learning/dcouvrir-les-metiers-de-la-data-science/dcouvrir-la-data-science>
- JAIPURIA, N., ZHANG, X., BHASIN, R., ARAFA, M., CHAKRAVARTY, P., SHRIVASTAVA, S., MANGLANI, S., & MURALI, V. N. (2020). Deflating Dataset Bias Using Synthetic Data

- Augmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- JONES, G., HOCINE, M., SALOMON, J., DAB, W., & TEMIME, L. (2015). Demographic and occupational predictors of stress and fatigue in French intensive-care registered nurses and nurses' aides : A cross-sectional study. *International Journal of Nursing Studies*, 52(1), 250-259. <https://doi.org/10.1016/j.ijnurstu.2014.07.015>
- KAHNEMAN, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.
- KAMVAR, S. D., KLEIN, D., & MANNING, C. D. (2003). Spectral Learning. *Proceedings of the international joint conference on artificial intelligence*, 561-566.
- KEUNG, P., LU, Y., SZARVAS, G., & SMITH, N. A. (2020). The Multilingual Amazon Reviews Corpus. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4563-4568. <https://doi.org/10.18653/v1/2020.emnlp-main.369>
- KHAN, M. A., TAMIM, I., AHMED, E., & AWAL, M. A. (2012). Multiple Parameter Based Clustering (MPC) : Prospective Analysis for Effective Clustering in Wireless Sensor Network (WSN) Using K-Means Algorithm. *WSN*, 04(01), 18-24. <https://doi.org/10.4236/wsn.2012.41003>
- KIRCH, W. (2008). Pearson's Correlation Coefficient. In *Encyclopedia of Public Health* (p. 1090-1091). Springer Netherlands. Récupérée juillet 6, 2023, à partir de https://link.springer.com/10.1007/978-1-4020-5614-7_2569
- KLIE, J.-C., BUGERT, M., BOULLOSA, B., de CASTILHO, R. E., & GUREVYCH, I. (2018). The INCEpTION platform : Machine-assisted and knowledge-oriented interactive annotation. *Proceedings of the 27th International Conference on Computational Linguistics : System Demonstrations*, 5-9. <https://inception-project.github.io/>
- KOTHADIYA, D., PISE, N., & BEDEKAR, M. (2020). Different Methods Review for Speech to Text and Text to Speech Conversion. *IJCA*, 175(20), 9-12. <https://doi.org/10.5120/ijca2020920727>
- KOTSIANTIS, S. B., ZAHARAKIS, I. D., & PINTELAS, P. E. (2006). Machine learning : a review of classification and combining techniques. *Artif Intell Rev*, 26(3), 159-190. <https://doi.org/10.1007/s10462-007-9052-3>
- KRIEGEL, H.-P., KRÖGER, P., SANDER, J., & ZIMEK, A. (2011). Density-based clustering. *WIREs Data Min & Knowl*, 1(3), 231-240. <https://doi.org/10.1002/widm.30>
- KRIPPENDORFF, K. (2004). *Content analysis : an introduction to its methodology* (2nd ed). Sage.
- KRUSKAL, W., & MOSTELLER, F. (1979a). Representative Sampling, I : Non-Scientific Literature. *International Statistical Review / Revue Internationale de Statistique*, 47(1), 13. <https://doi.org/10.2307/1403202>
- KRUSKAL, W., & MOSTELLER, F. (1979b). Representative Sampling, II : Scientific Literature, Excluding Statistics. *International Statistical Review / Revue Internationale de Statistique*, 47(2), 111. <https://doi.org/10.2307/1402564>
- LAMIREL, J.-C., CUXAC, P., & HAJLAOUI, K. (2017). A Novel Approach to Feature Selection Based on Quality Estimation Metrics. In F. GUILLET, B. PINAUD & G. VENTURINI (Éd.), *Advances in Knowledge Discovery and Management* (p. 121-140, T. 665). Springer International Publishing. Récupérée novembre 23, 2018, à partir de http://link.springer.com/10.1007/978-3-319-45763-5_7
- LAMPERT, T., DAO, T.-B.-H., LAFABREGUE, B., SERRETTE, N., FORESTIER, G., CRÉMILLEUX, B., VRAIN, C., & GANÇARSKI, P. (2018). Constrained distance based clustering for time-series : a comparative and experimental study. *Data Min Knowl Disc*, 32(6), 1663-1707. <https://doi.org/10/gfbpj8>

BIBLIOGRAPHIE

- LAMPERT, T., LAFABREGUE, B., & GANCARSKI, P. (2019). Constrained Distance based K-Means Clustering for Satellite Image Time-Series. *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2419-2422. <https://doi.org/10/ggx3tj>
- LANDIS, J. R., & KOCH, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1), 159. <https://doi.org/10.2307/2529310>
- LEE, K., & SENGUPTA, S. (2022). *Introducing the Ai Research Supercluster - Meta's Cutting-Edge Ai Supercomputer for Ai Research*. Meta AI. <https://ai.meta.com/blog/ai-rsc/>
- LEECH, G. (1993). Corpus Annotation Schemes. *Literary and Linguistic Computing*, 8(4), 275-281. <https://doi.org/10.1093/lrc/8.4.275>
- LEECH, G. (2004). Adding linguistic annotation. In M. WYNNE (Ed.), *Developing linguistic corpora : a guide to good practice* (Oxbow Books, p. 17-29). AHDS : Literature, Languages, and Linguistics. <http://ahds.ac.uk/creating/guides/linguistic-corpora/chapter2.htm>
- LES ECHOS. (2023). IA : L'auteur de "Game of Thrones" et d'autres écrivains portent plainte contre le créateur de ChatGPT [newspaper]. *Les Echos : Tech-Médias*. Récupérée septembre 29, 2023, à partir de <https://www.lesechos.fr/tech-medias/intelligence-artificielle/ia-lauteur-de-game-of-thrones-et-dautres-ecrivains-portent-plainte-contre-le-createur-de-chatgpt-1980235>
- LEWIS, M., LIU, Y., GOYAL, N., GHAZVININEJAD, M., MOHAMED, A., LEVY, O., STOYANOV, V., & ZETTLEMOYER, L. (2019). BART : Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.1910.13461>
- LI, J., SUN, A., HAN, J., & LI, C. (2022). A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.*, 34(1), 50-70. <https://doi.org/10.1109/TKDE.2020.2981314>
- LIN, T.-Y., MAIRE, M., BELONGIE, S., BOURDEV, L., GIRSHICK, R., HAYS, J., PERONA, P., RAMANAN, D., ZITNICK, C. L., & DOLLÁR, P. (2014). Microsoft COCO : Common Objects in Context. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.1405.0312>
- LOIGNON, S. (2023). IA : Les médias français s'organisent face à la collecte de données par les robots [newspaper]. *Les Echos : Tech-Médias*. Récupérée octobre 4, 2023, à partir de <https://www.lesechos.fr/tech-medias/medias/ia-les-medias-francais-sorganisent-face-a-la-collecte-de-donnees-par-les-robots-1973079>
- MAALOUF, M. (2011). Logistic regression in data analysis : an overview. *IJDATS*, 3(3), 281. <https://doi.org/10.1504/IJDATS.2011.041335>
- MACQUEEN, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14), 281-297.
- MAHARANA, K., MONDAL, S., & NEMADE, B. (2022). A review : Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 3(1), 91-99. <https://doi.org/10.1016/j.gltip.2022.04.020>
- MANNING, C. D., & SCHÜTZE, H. (2000). *Foundations of statistical natural language processing* (2e éd. avec des corrections). MIT Press.
- MCCOWAN, I., MOORE, D., DINES, J., GATICA-PEREZ, D., FLYNN, M., WELLNER, P., & BOURLARD, H. (2005, mars). *On the Use of Information Retrieval Measures for Speech Recognition Evaluation* (IDIAP-RR 04-73). IDIAP Research Institute. Martigny, Switzerland.
- MICROSOFT CORPORATION. (2018). *Microsoft Excel*. <https://office.microsoft.com/excel>
- MILLER, G. A., & CHARLES, W. G. (1991). Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, 6(1), 1-28. <https://doi.org/10.1080/01690969108406936>

- MONTANI, I., & HONNIBAL, M. (2017, décembre 18). *Prodigy : A Modern and Scriptable Annotation Tool for Creating Training Data for Machine Learning Models*. <https://prodi.gy/>
- MORRIS & GOSCINNY, R. (1950). *Rodeo*. Dupuis.
- MORRIS & GOSCINNY, R. (1952). *Sous le ciel de l'ouest*. Dupuis.
- MORRIS & GOSCINNY, R. (1958). *Les Cousins Dalton*. Dupuis.
- MU, Z., YANG, X., & DONG, Y. (2021). Review of End-to-End Speech Synthesis Technology Based on Deep Learning. *ArXiv preprint*. <https://doi.org/10.48550/arXiv.2104.09995>
- MURTAGH, F., & CONTRERAS, P. (2012). Algorithms for hierarchical clustering : An overview. *Wiley Interdisc. Rew. : Data Mining and Knowledge Discovery*, 2, 86-97. <https://doi.org/10.1002/widm.53>
- NÉDELLEC, C., BESSIERES, P., BOSSY, R., & KOTOUJANSKY, A. (2006). Annotation Guidelines for Machine Learning-Based Named Entity Recognition in Microbiology.
- NELDER, J. A., & WEDDERBURN, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3), 370. <https://doi.org/10.2307/2344614>
- NG, A. Y., JORDAN, M. I., & WEISS, Y. (2002). On Spectral Clustering : Analysis and an Algorithm. In T. G. DIETTERICH, S. BECKER & Z. GHAHRAMANI (Éd.), *Advances in Neural Information Processing Systems 14* (p. 849-856). MIT Press. Récupérée octobre 22, 2020, à partir de <http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf>
- NICOLETTI, L., & BASS, D. (2023). Generative AI takes stereotypes and bias from bad to worse [newspaper]. *Bloomberg.com*. Récupérée octobre 2, 2023, à partir de <https://www.bloomberg.com/graphics/2023-generative-ai-bias/>
- NIVRE, J. (2006). *Inductive Dependency Parsing* (T. 34). Springer Netherlands. Récupérée juillet 6, 2023, à partir de <http://link.springer.com/10.1007/1-4020-4889-0>
- NOTHMAN, J., QIN, H., & YURCHAK, R. (2018). Stop Word Lists in Free Open-source Software Packages. *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, 7-12. <https://doi.org/10.18653/v1/W18-2502>
- O'NEILL, M., & CONNOR, M. (2023). Amplifying Limitations, Harms and Risks of Large Language Models. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2307.04821>
- OPENAI. (2023). *ChatGPT*. <https://chat.openai.com>
- PARNAMI, A., & LEE, M. (2022). Learning from Few Examples : A Summary of Approaches to Few-Shot Learning. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2203.04291>
- PERRIGO, B., & ZORTHIAN, J. (2023). Exclusive : OpenAI used Kenyan workers on less than \$2 per hour to make ChatGPT less toxic [newspaper]. *Time : Business, Technology*. Récupérée septembre 29, 2023, à partir de <https://time.com/6247678/openai-chatgpt-kenya-workers/>
- PERROTIN, R., NASR, A., & AUGUSTE, J. (2018). Annotation En Actes de Dialogue Pour Les Conversations d'Assistance En Ligne. *25e Conférence Sur Le Traitement Automatique Des Langues Naturelles (TALN)*. <https://hal.science/hal-01943345>
- PERRY, T. (2021). LightTag : Text Annotation Platform. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, 20-27. <https://aclanthology.org/2021.emnlp-demo.3>
- PRADHAN, S. S., LOPER, E., DLIGACH, D., & PALMER, M. (2007). SemEval-2007 task 17 : English lexical sample, SRL and all words. *Proceedings of the 4th International Workshop on Semantic Evaluations - SemEval '07*, 87-92. <https://doi.org/10.3115/1621474.1621490>
- Proposal for a Regulation of the European Parliament and the Council Laying down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union

BIBLIOGRAPHIE

- Legislative Acts (2021, avril 21). <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206>
- PURVES, D., & BRANNON, E. M. (Éd.). (2013). *Principles of cognitive neuroscience* (2. ed). Sinauer.
- PUSTEJOVSKY, J., & STUBBS, A. (2012). Natural language annotation for machine learning. <https://api.semanticscholar.org/CorpusID:60457717>
- R CORE TEAM. (2017). *R : A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. <https://www.R-project.org/>
- RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., & SUTSKEVER, I. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8), 9.
- RADOVILSKY, Z., HEGDE, V., & ACHARYA, A. (2018). Skills Requirements of Business Data Analytics and Data Science Jobs : A Comparative Analysis. *16*(1).
- RAJBAHADUR, G. K., TUCK, E., ZI, L., LIN, D., CHEN, B., MING, Z., JIANG & GERMAN, D. M. (2022). Can I use this publicly available dataset to build commercial AI software ? – A Case Study on Publicly Available Image Datasets. *ArXiv preprint*. Récupérée septembre 29, 2023, à partir de <http://arxiv.org/abs/2111.02374>
- RAMESH, A., PAVLOV, M., GOH, G., GRAY, S., VOSS, C., RADFORD, A., CHEN, M., & SUTSKEVER, I. (2021). Zero-Shot Text-to-Image Generation. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2102.12092>
- RAMOS, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the first instructional conference on machine learning*.
- RASCHKA, S., & MIRJALILI, V. (2019). *Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow 2* (Third edition). Packt.
- RE3DATA.ORG. (2013). Zenodo. <https://doi.org/10.17616/R3QP53>
- Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation) (2016, mai 4). <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679>
- ROACH, J. (2023). *How Microsoft's Bet on Azure Unlocked an AI Revolution*. Microsoft. <https://news.microsoft.com/>
- ROSENBERG, A., & HIRSCHBERG, J. (2007). V-Measure : A Conditional Entropy-Based External Cluster Evaluation Measure.
- ROWE, N. (2023). "It's destroyed me completely" : Kenyan moderators decry toll of training of AI models [newspaper]. *The Guardian : Artificial Intelligence*. Récupérée septembre 29, 2023, à partir de <https://www.theguardian.com/technology/2023/aug/02/ai-chatbot-training-human-toll-content-moderator-meta-openai>
- RUIZ, C., SPILIOPOULOU, M., & MENASALVAS, E. (2010). Density-based semi-supervised clustering. *Data Min Knowl Disc*, 21(3), 345-370. <https://doi.org/10.1007/s10618-009-0157-y>
- SASAKI, Y. (2007). The truth of the F-measure.
- SCHILD, E. (2022a, août 22). *Cognitivefactory/Interactive-Clustering*. Récupérée février 13, 2023, à partir de <https://doi.org/10.5281/zenodo.4775251>
- SCHILD, E. (2022b, novembre 5). *Cognitivefactory/Interactive-Clustering-Comparative-Study*. Récupérée février 13, 2023, à partir de <https://doi.org/10.5281/zenodo.5648255>
- SCHILD, E. (2022c, novembre 9). *French trainset for chatbots dealing with usual requests on bank cards*. Zenodo. <https://doi.org/10.5281/zenodo.4769949>
- SCHILD, E. (2023, février 16). *Cognitivefactory/Features-Maximization-Metric*. Récupérée février 16, 2023, à partir de <https://doi.org/10.5281/zenodo.7646382>

- SCHILD, E., & ADLER, M. (2023, octobre 2). *Subset of 'MLSUM : The Multilingual Summarization Corpus' for constraints annotation experiment* (Version 1.0.0 [subset : fr+train+filtered]). Zenodo. <https://doi.org/10.5281/ZENODO.8399301>
- SCHILD, E., DURANTIN, G., LAMIREL, J.-C., & MICONI, F. (2021). Conception itérative et semi-supervisée d'assistants conversationnels par regroupement interactif des questions. *RNTI E-37*. Récupérée juin 14, 2021, à partir de <https://hal.inria.fr/hal-03133007>
- SCHILD, E., DURANTIN, G., LAMIREL, J.-C., & MICONI, F. (2022). Iterative and Semi-Supervised Design of Chatbots Using Interactive Clustering. *International Journal of Data Warehousing and Mining (IJDWM)*, 18(2), 1-19. <https://doi.org/10.4018/IJDWM.298007>
- SCHILD, E., TREMBLE, T., & MISIAK, C. (2022, septembre 1). *Cognitivefactory/Interactive-Clustering-Gui*. Récupérée février 13, 2023, à partir de <https://doi.org/10.5281/zenodo.4775270>
- SCIALOM, T., DRAY, P.-A., LAMPRIER, S., PIWOWARSKI, B., & STAiano, J. (2020, avril 30). *MLSUM : The Multilingual Summarization Corpus* (arXiv :2004.14900). arXiv. Récupérée juin 7, 2023, à partir de <http://arxiv.org/abs/2004.14900>
- SEABOLD, S., & PERKTOLD, J. (2010). Statsmodels : Econometric and Statistical Modeling with Python, 92-96. <https://doi.org/10.25080/Majora-92bf1922-011>
- SETTLES, B. (2010). Active Learning Literature Survey, 67.
- SHORTEN, C., & KHOSHGOFTAAR, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *J Big Data*, 6(1), 60. <https://doi.org/10.1186/s40537-019-0197-0>
- SHORTEN, C., KHOSHGOFTAAR, T. M., & FURHT, B. (2021). Text Data Augmentation for Deep Learning. *J Big Data*, 8(1), 101. <https://doi.org/10.1186/s40537-021-00492-0>
- SINCLAIR, J. (2004). Corpus and Text : Basic Principles. In M. WYNNE (Éd.), *Developing Linguistic Corpora : A Guide to Good Practice* (Oxbow Books, p. 1-16). AHDS : Literature, Languages, and Linguistics. <http://ahds.ac.uk/creating/guides/linguistic-corpora/chapter1.htm>
- SNOW, R., O'CONNOR, B., JURAFSKY, D., & NG, A. (2008). Cheap and Fast - But is it Good ? Evaluating Non-Expert Annotations for Natural Language Tasks. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 254-263.
- SPARCK JONES, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1), 11-21. <https://doi.org/10.1108/eb026526>
- SPERANDIO, J.-C. (1978). The Regulation of Working Methods as a Function of Work-load among Air Traffic Controllers. *Ergonomics*, 21(3), 195-202. <https://doi.org/10.1080/00140137808931713>
- SPERANDIO, J.-C. (1987). *L'ergonomie Du Travail Mental*. FeniXX.
- STEINBACH, M., ERTÖZ, L., & KUMAR, V. (2004). The Challenges of Clustering High Dimensional Data. In L. T. WILLE (Éd.), *New Directions in Statistical Physics* (p. 273-309). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-08968-2_16
- STUBBS, A. C. (2013). *A Methodology for Using Professional Knowledge in Corpus* [thèse de doct., Brandeis University].
- TEAM DATASCIENTEST. (2022, septembre). *Les métiers de la data : mieux comprendre leurs différences*. Datascientest.com. Récupérée septembre 26, 2023, à partir de <https://datascientest.com/les-metiers-de-la-data>
- THORNDIKE, R. L. (1953). Who belongs in the family ? *Psychometrika*, 18(4), 267-276. <https://doi.org/10.1007/BF02289263>
- TOUVRON, H., MARTIN, L., STONE, K., ALBERT, P., ALMAHAIRI, A., BABAEI, Y., BASHLYKOV, N., BATRA, S., BHARGAVA, P., BHOSALE, S., BIKEL, D., BLECHER, L., FERRER, C. C., CHEN, M., CUCURULL, G., ESIOPU, D., FERNANDES, J., FU, J., FU, W., ... SCIALOM,

BIBLIOGRAPHIE

- T. (2023). Llama 2 : Open Foundation and Fine-Tuned Chat Models. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.2307.09288>
- TUKEY, J. W. (1949). Comparing Individual Means in the Analysis of Variance. *Biometrics*, 5(2), 99. <https://doi.org/10.2307/3001913>
- USZKOREIT, J. (2017, août 31). *Transformer : A Novel Neural Network Architecture for Language Understanding*. Google AI Blog. Récupérée juin 10, 2020, à partir de <http://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>
- VALETTTE, M. (2016). Analyse statistique des données textuelles et traitement automatique des langues. Une étude comparée. *International conference on statistical analysis of textual data (JADT2016)*, 2, 697-706. <https://inhalo.hal.science/hal-01335084>
- VAN ROSSUM, G., & DRAKE, F. L. (2009). *Python 3 Reference Manual* (CreateSpace).
- VON AHN, L. (2006). Games with a Purpose. *Computer*, 39(6), 92-94. <https://doi.org/10.1109/MC.2006.196>
- VOORMANN, H., & GUT, U. (2008). Agile Corpus Creation. <https://api.semanticscholar.org/CorpusID:56885448>
- WAGSTAFF, K., & CARDIE, C. (2000). Clustering with Instance-level Constraints. *Proceedings of the Seventeenth International Conference on Machine Learning*, 1103-1110.
- WAGSTAFF, K., CARDIE, C., ROGERS, S., & SCHRÖDL, S. (2001). Constrained K-means Clustering with Background Knowledge, 577-584.
- WALLACH, D., & GOFFINET, B. (1987). Mean Squared Error of Prediction in Models for Studying Ecological and Agronomic Systems. *Biometrics*, 561-573.
- WYNNE, M. (Éd.). (2004). *Developing linguistic corpora : a guide to good practice* (Oxbow Books). AHDS : Literature, Languages, and Linguistics. <http://www.ahds.ac.uk/creating/guides/linguistic-corpora/index.htm>
- XU, D., & TIAN, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2, 165-193.
- YANG, T.-N., & WANG, S.-D. (2004). Competitive algorithms for the clustering of noisy data. *Fuzzy Sets and Systems*, 141(2), 281-299. [https://doi.org/10.1016/S0165-0114\(02\)00525-0](https://doi.org/10.1016/S0165-0114(02)00525-0)
- ZDANIUK, B. (2014). Ordinary Least-Squares (OLS) Model. In A. C. MICHALOS (Éd.), *Encyclopedia of Quality of Life and Well-Being Research* (p. 4515-4517). Springer Netherlands. Récupérée septembre 15, 2023, à partir de http://link.springer.com/10.1007/978-94-007-0753-5_2008
- ZHANG, J., ZHAO, Y., SALEH, M., & LIU, P. J. (2019). PEGASUS : Pre-training with Extracted Gap-sentences for Abstractive Summarization. *ArXiv preprint*. <https://doi.org/10.48550/ARXIV.1912.08777>
- ZHOU, Z.-H. (2021). *Machine learning* (S. LIU, Trad.). Springer. <https://books.google.fr/books?id=Zd5hywEACAAJ>
- ZHUANG, F., QI, Z., DUAN, K., XI, D., ZHU, Y., ZHU, H., XIONG, H., & HE, Q. (2021). A Comprehensive Survey on Transfer Learning. *Proc. IEEE*, 109(1), 43-76. <https://doi.org/10.1109/JPROC.2020.3004555>

Liste des TODOs

CHAPITRE : TITRE À TROUVER : "Introduction"	1
CHAPITRE : INTRODUCTION À RÉDIGER	1
CHAPITRE : INTRODUCTION À RÉDIGER	1
SECTION : TITRE À TROUVER : "Asset centrality"	1
SECTION : À RÉDIGER :	
- Utilisation intensive de l'IA / des chatbots, et description des nombreuses opportunités liées ;	
- Mais aussi, mystification de l'IA : le grand public ne se sait pas comment ça marche, comme ça s'entraîne, ont des craintes sur les capacités des modèles,	1
SECTION : TITRE À TROUVER : "Establishing a Niche"	1
SECTION : À RÉDIGER :	
- Pour démystifier :	
- 1. Chat-oriented ou Task-oriented ;	
- 2. Conception par approche symbolique ou par approche générative ;	
- 3. Besoin d'annotation pour avoir des données de qualité.	1
SECTION : TITRE À TROUVER : "Gap"	2
SECTION : À RÉDIGER :	
- Peu de travaux sur la conception d'un jeu de données : en recherche les données sont publiques, en entreprises les données sont privées ;	
- Nombreuses pistes d'amélioration, mais peu sont exploitées en pratique ;	
- Défis d'organisation, de gestion de coûts, de complexité, de qualité, ...	
- Conception trop manuelle, experts pas à leur place,	2
SECTION : TITRE À TROUVER : "Occupying the Niche"	2
SECTION : À RÉDIGER :	
- Besoin de recentrer l'activité des experts métiers ;	
- Besoin d'assister la conception d'un jeu de données ;	
- Nous proposons donc une méthode itérative et semi-supervisée.	2
TRANSITION : Annonce du plan ?	2
SECTION : À RÉDIGER :	
- trouver une base d'apprentissage acceptable, puis corriger manuellement	
- intervention d'experts métiers sur la base de leur connaissance métiers	
- revues d'annotations basée sur leur connaissance métiers	
- aide à la modélisation par le clustering	151

SECTION À RÉDIGER :	
- bien définir l'objectif à modéliser (par action ? par objet ?)	
- laisser à la mach	
- faire référence aux maximes de LEECH, 1993 ? 1. VOIR LA DONNEE NON PRE-TRAITEE : It should always be possible to come back to initial data (example BC). Note : can be hard after normalization ("l'arbre!" vs "le arbre", etc.)	
2. ANNOTER DES DIFFERENCES VISIBLES : Annotations should be extractable from the text	
3. DOCUMENTER L'AJOUT DE CONTRAINTES : The annotation procedure should be documented (ex : Brown Corpus annotation guide, Penn Tree Bank annotation guide)	
4. DOCUMENTER LES COMPETENCES DES ANNOTATEURS : Mention should be made of the annotator(s) and the way annotation was made (manual/automatic annotation, number of annotators, manually corrected/uncorrected...)	
5. SUBJECTIVITE => 3 ANNOTATEURS : Annotation is an act of interpretation (cannot be infallible)	
6. MIEUX VAUT NE PAS LIER QUE LIER DE MANIERE AMBIGUE : Annotation schemas should be as independent as possible on formalisms	
7. PLUSIEURS VISIONS POSSIBLES, IL FAUT EN CHOISIR UNE ET S'Y TENIR : No annotation schema should consider itself a standard (it possibly becomes one)	151
SECTION À RÉDIGER :	
- Cas d'arrêt quand le clustering stagne à 5% : si change pas, alors l'annotation n'a plus d'effet...	
- utiliser le graphe de contraintes pour voir les données liées entre elles et les données isolées	
- Analyse avec résumer par LLM pour identifier facilement les thématiques qui se dégagent	
- Analyse avec FMC pour identifier le vocabulaire qui caractérise chaque thématique	
- utiliser ces analyses pour régler le nombres d clusters?	152
SECTION À RÉDIGER :	
- Architecture parallele : pour gagner du temps	
- paramétrage optimal : simple + tfidf + kmeans + closest	
- taille de batch d'annotation dépendant de la taille du jeu de données (dépend du temps de clustering)	152
SECTION À RÉDIGER :	
- rappeler les équations de temps : environ 24 x taille de dataset, sans les revues d'annotations	
- besoin de 3 annotateurs pour confronter les vision et modéliser dans de bonnes conditions. organiser les revues sur la base des différences entre cas d'usage métier	
- ajouter de la redondance si le cas d'usage est complexe, mais ça ralenti	
- avantage par rapport aux méthodes usuelles : moins abstrait/complexes, moins d'essai-erreur	152
SECTION : TITRE À TROUVER : " <i>Occupying the Niche</i> "	153
SECTION : À RÉDIGER	153
SECTION : TITRE À TROUVER : " <i>Gap</i> "	153
SECTION : À RÉDIGER	153
SECTION : TITRE À TROUVER : " <i>Establishing a Niche</i> "	153

SECTION : À RÉDIGER	153
SECTION : TITRE À TROUVER : "Asset centrality"	153
SECTION : À RÉDIGER	153
A REDIGER	159
introduction à rédiger	163
introduction à rédiger	171
à rédiger	175
à rédiger	176
à rédiger	177
à rédiger	178
à rédiger	179
à rédiger	180
à rédiger	181
ANNEXE : v-measure	181

Liste des TODOs

Liste des figures

2.1	Exemple d'annotation de l'état d'une BD (ici : MORRIS et GOSCINNY, 1950 et MORRIS et GOSCINNY, 1952). La première est en très bon état (couverture comme neuve, tranches légèrement usées, pages intactes) tandis que la seconde est en mauvais état (couverture usée, dos abimée, traces sur les pages, ...).	7
2.2	Exemple d'annotation de textes présents sur la couverture d'une bande dessinée (ici : MORRIS et GOSCINNY, 1958). Les informations essentielles telles que la collection, le numéro, le titre, l'auteur et l'éditeur y sont présentes.	8
2.3	Exemple de paroles prononcées dans un audio. Ici, la voix de <i>Lucky Luke</i> est interprétée par Jacques THEBAULT. Le texte annoté, c'est-à-dire celui prononcé dans l'audio, est « <i>Ils sont à vous chef, et j'veus s'rai reconnaissant de bien les garder cette fois.</i> ». Les phonèmes en alphabet phonétique international associés à chaque séquence de l'audio sont disponibles si besoin.	9
2.4	Cycle MATTER structurant un projet d'annotation en six étapes principales : <i>Modelize</i> , <i>Annotate</i> , <i>Train</i> , <i>Test</i> , <i>Evaluate</i> et <i>Revise</i> . Le carré bleu identifie le mini-cycle MAMA durant lequel la modélisation est adaptée en cours d'annotation, et le carré orange identifie le mini-cycle <i>Train-Test</i> lors de la conception du modèle.	11
2.5	Quatre exemples d'outils d'annotation : (1) INCEPTION pour le texte (KLIE et al., 2018), (2) prodigy pour le texte ou l'image (MONTANI et HONNIBAL, 2017), (3) Audacity pour l'audio (AUDACITY TEAM, 2000) et (4) CVAT pour l'image (CVAT.AI CORPORATION, 2019).	19
2.6	Répartitions des teintes de peaux (1) et des genres (2) par métier lors de génération de portrait avec Stable Diffusion (étude menée par NICOLETTI et BASS, 2023).	22
2.7	Exemple de bruits courant perturbant l'analyse d'une image : (1) le flou, (2) un doigt sur le capteur, (3) un problème de cadrage et (4) un problème d'angle de vue.	24
3.1	Schéma illustrant l'architecture du <i>clustering</i> interactif. La boucle principale enchaîne un échantillonnage de couples de données, une annotation de contraintes, et un <i>clustering</i> sous contraintes.	44

3.2 Exemple d'une itération de <i>clustering</i> interactif. Lors de l'initialisation, (1) correspond au jeu de données brut, et (2) correspond à une première segmentation des données en 3 <i>clusters</i> . Lors de l'itération 1 : (3) correspond à un exemple d'échantillonnage de 6 contraintes représentées par les flèches en pointillées, (4) correspond à la caractérisation de ces 6 contraintes par des liens MUST-LINK en vert et CANNOT-LINK en rouge, et (5) correspond à la nouvelle segmentation des données en 3 <i>clusters</i> respectant les 6 contraintes annotées. La prochaine itération se poursuivra par un nouvel échantillonnage de contraintes.	46
3.3 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page d'annotation d'une contrainte . Parmi les éléments importants, nous retrouvons les deux textes à annoter (disposés à gauche et droite de l'écran) et les boutons d'annotation (bouton vert pour un MUST-LINK, bouton rouge pour un CANNOT-LINK). Les autres fonctionnalités sont détaillées en ANNEXE C.4.	47
4.1 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 0/6</i>) en schématisant l'évolution de la performance (<i>accord avec la vérité terrain calculé en v-measure</i>) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (<i>nombre d'annotations par un expert métier</i>).	50
4.2 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 1/6</i>) en schématisant l'évolution de la performance (<i>accord avec la vérité terrain calculé en v-measure</i>) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (<i>nombre d'annotations par un expert métier</i>).	53
4.3 Évolution de la moyenne de la v-measure entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de <i>clustering</i> interactif, moyenne réalisée itération par itération sur l'ensemble des tentatives. Représentation des tentatives ayant été les plus rapides (<i>un prétraitement prep.simple, une vectorisation vect.tfidf, un clustering clust.hier.comp ou clust.hier.ward, et un échantillonnage samp.closest.diff</i>) et les plus lentes (<i>un prétraitement prep.no, une vectorisation vect.tfidf, un clustering clust.spec, et un échantillonnage de contraintes samp.farthest.same</i>) pour atteindre 100% de v-measure	57
4.4 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 2/6</i>) en schématisant l'évolution de la performance (<i>accord avec la vérité terrain calculé en v-measure</i>) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (<i>nombre d'annotations par un expert métier</i>).	60
4.5 Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation partielle, c'est-à-dire l'itération à laquelle elles parviennent à 90% de v-measure entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics pour simplifier l'affichage.	63
4.6 Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation suffisante, c'est-à-dire l'itération à laquelle elles parviennent à 100% de v-measure entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics pour simplifier l'affichage.	65
4.7 Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation exhaustive, c'est-à-dire l'itération à laquelle toutes les contraintes possibles entre les données ont été annotées. L'histogramme est réduit à 60 pics pour simplifier l'affichage.	66

4.8	Évolution des moyennes du nombre d’itérations nécessaire de la méthode de <i>clustering</i> interactif pour obtenir un seuil défini de v-measure entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes valeurs que peuvent prendre les facteurs analysés et affichées par facteur : (1) prétraitement, (2) vectorisation, (3) <i>clustering</i> et (4) échantillonnage. Note : <i>Le seuil d’annotation exhaustive (annoter toutes les contraintes possibles) n’étant pas exprimé en terme de v-measure, ce seuil n’est pas affiché ici.</i>	68
4.9	Évolution des moyennes du nombre d’itérations nécessaire de la méthode de <i>clustering</i> interactif pour obtenir un seuil défini de v-measure entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes seuils d’annotations étudiés : l’annotation partielle (<i>atteindre une v-measure de 90%</i>), l’annotation suffisante (<i>atteindre une v-measure de 100%</i>) et l’annotation exhaustive (<i>annoter toutes les contraintes possibles</i>).	69
4.10	Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 3/6</i>) en schématisant l’évolution de la performance (<i>accord avec la vérité terrain calculé en v-measure</i>) d’une base d’apprentissage en cours de construction en fonction du coût temporel de la méthode (<i>temps nécessaire à l’expert métier et à la machine</i>).	71
4.11	Capture d’écran de l’application web permettant utilisant notre méthodologie de <i>clustering</i> interactif : page d’annotation de contraintes . Les deux textes à annoter sont disposés à gauche et droite de l’écran. Chacun dispose d’un cache à cocher si le texte n’est pas pertinent à analyser (<i>ambigu, hors périmètre, incompréhensible, ...</i>). Les boutons à disposition permettent respectivement d’annoter un MUST-LINK si les données sont similaires (<i>bouton en vert</i>), un CANNOT-LINK si les données ne sont pas similaire (<i>bouton en rouge</i>), d’ignorer la contrainte pour laisser la main à l’algorithme de <i>clustering</i> (<i>bouton en bleu</i>), et d’ajouter un commentaire pour revoir la contrainte plus tard (<i>case à choser et champ de texte libre</i>). Deux éléments déroulant permettent d’avoir des informations supplémentaires (<i>metadata de sélection et de clustering, représentation graphique des liens entre contraintes annotées</i>). Les boutons de navigation (<i>boutons flèches et liste</i>) sont disponibles en bas de page.	74
4.12	Capture d’écran de l’application web permettant utilisant notre méthodologie de <i>clustering</i> interactif : page d’inventaire des contraintes à annoter . La partie supérieure permet d’identifier le nombre de textes et de contraintes sur le projet, ainsi que les boutons destinés à calculer les transitivités entre les contraintes et à approuver le travail réalisé si aucune transitivité n’entre en conflit avec un contrainte annotée. La partie inférieure liste l’ensemble des contraintes du projet, avec les annotations réalisées, l’itération à laquelle la contraintes a été sélectionnée et annotée, si elle est à revoir ou si une incohérence la concernant est détectée.	75
4.13	Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.	76
4.14	Étude de cas d’évolution de la vitesse d’annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d’annotations.	77
4.15	Estimation du temps nécessaire (en minutes) pour effectuer une tâche de prétraitement en fonction du nombre de données à traiter. Les paramétrages prep.simple , prep.lemma et prep.filter ayant des temps de calculs similaires, leurs modélisations n’ont pas été séparées.	85
4.16	Estimation du temps nécessaire (en minutes) pour effectuer une tâche de vectorisation en fonction du nombre de données à traiter.	86

4.17 Estimation du temps nécessaire (en minutes) pour effectuer une tâche de clustering en fonction du nombre de données à traiter.	87
4.18 Estimation du temps nécessaire (en minutes) pour effectuer une tâche d' échantillonnage de contraintes en fonction du nombre de données à traiter.	89
4.19 Estimation du nombre moyen de contraintes nécessaire à notre paramétrage favori du <i>clustering</i> interactif afin d'obtenir une annotation partielle (<i>atteindre une v-measure de 90%</i>) en fonction de la taille du jeu de données à modéliser.	93
4.20 Exemple de caractérisation exhaustive d'un jeu de données (10 données, 3 classes) en ajoutant un nombre minimal de contraintes (cf. (1)) ou en ajoutant toutes les contraintes possibles (cf. (2)).	94
4.21 Schéma comparatif des architectures du <i>clustering</i> interactif : (1) représente la version séquentielle initialement présentée en CHAPITRE 3 où le <i>clustering</i> s'adapte avec les annotation de l'itération en cours ; (2) représente l'évolution en mode <i>parallèle</i> où le <i>clustering</i> s'adapte avec les annotations de l'itération précédente (décalage d'une itération).	97
4.22 Estimation du temps total nécessaire (en heures) pour modéliser un jeu de données avec notre paramétrage favori du <i>clustering</i> interactif afin d'obtenir une annotation partielle (<i>atteindre une v-measure de 90%</i>), en fonction de plusieurs taille de jeu de données, plusieurs tailles de lots d'annotation, et mettant en opposition l'approche séquentielle (<i>annotation puis le clustering</i>) et l'approche parallèle (<i>annotation pendant le clustering</i>).	98
4.23 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 4/6</i>) en schématisant l'évolution de la pertinence (<i>valeur métier évaluée par l'expert et exprimé en nombre de clusters</i>) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (<i>temps nécessaire à l'expert métier et à la machine</i>).	100
4.24 Évolution de la pertinence métier moyenne estimée manuellement au cours des itérations du résultat du <i>clustering</i> interactif avec notre paramétrage favori. Cette pertinence, exprimée en proportion du nombre de <i>clusters</i> , est retranscrite en trois niveaux : exploitable en vert, partiellement exploitable en orange, et non exploitable en rouge.	103
4.25 Évolution de la pertinence métier moyenne en fonction du nombre d'itérations de la méthode. Cette pertinence, exprimée en proportion du nombre de <i>clusters</i> , est estimée sur la base du résumé automatique des <i>clusters</i> par un large modèle de langage et est retranscrite en trois niveaux : exploitable en vert, partiellement exploitable en orange, et non exploitable en rouge.	115
4.26 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 5/6</i>) en schématisant l'évolution de la pertinence (<i>valeur métier évaluée par l'expert et exprimé en nombre de clusters</i>) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (<i>temps nécessaire à l'expert métier et à la machine</i>), ainsi que la rentabilité de chaque itération de la méthode (<i>rapport entre le gain potentiel de pertinence et le coût à investir</i>).	119
4.27 Exemples d'accords et de désaccord entre les annotations d'une itération et le résultat du <i>clustering</i> de l'itération précédente. Des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges) sont représentées dans deux situations : (1) montre des cas d'accords (MUST-LINK dans un même <i>cluster</i> , CANNOT-LINK entre deux <i>clusters</i> différents), et (2) montre des cas de désaccords (MUST-LINK entre deux <i>clusters</i> différents, CANNOT-LINK dans un même <i>cluster</i>).	122

4.28 Évolution au cours des itérations de l'accord entre l'annotation de contraintes d'un expert et le résultat de <i>clustering</i> sur lequel est basé l'échantillonnage de contraintes. Ces accords sont exprimés grâce à des lots de 50 contraintes annotées. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori. À titre d'information, les courbes en noir représentent l'évolution de la v-measure entre le <i>clustering</i> et la vérité terrain.	122
4.29 Évolution de la différence de résultats entre deux itérations de <i>clustering</i> . Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori. À titre d'information, les courbes en noir représentent l'évolution de la v-measure entre le <i>clustering</i> et la vérité terrain.	126
4.30 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 6/6</i>) en schématisant l'évolution de la pertinence (<i>valeur métier évaluée par l'expert et exprimé en nombre de clusters</i>) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (<i>temps nécessaire à l'expert métier et à la machine</i>), ainsi que les marges d'erreurs représentant l'impact de différences d'annotation sur le nombre d'itérations nécessaire à la méthode.	129
4.31 Évolution des similitudes moyennes (calculées en terme de v-measure) des résultats de <i>clustering</i> des tentatives introduisant des erreurs d'annotation par rapport à la vérité terrain au cours des itérations. Les dégradés de couleurs des courbes représentent les déclinaisons de ces évolutions en fonction des différents taux d'annotations erronées (allant de 0% et 25%). (1) représente l'approche naïve ignorant les conflits d'annotation et (2) représente l'approche corrigéant les conflits détectés par le gestionnaire de contraintes. Toutes les courbes sont tronquées à 3 000 contraintes (nombre maximum de contraintes nécessaire à une tentatives n'introduisant pas erreurs pour converger vers la vérité terrain).	138
4.32 Exemple d'une évolution de similitudes moyennes (calculées en terme de v-measure) de résultats de <i>clustering</i> de tentatives introduisant des différences d'annotation par rapport à la vérité terrain au cours des itérations, exemple pour une vérité terrain ayant une taille de 5 000 données. Les dégradés de couleurs des courbes représentent les déclinaisons de ces évolutions en fonction des différents taux d'annotations divergentes (allant de 0% et 25%). La barre verticale indique le nombre moyen de contraintes nécessaires aux tentatives n'introduisant pas de désaccords pour obtenir un score de 90% de v-measure (ici : 16 250 contraintes).	144
B.1 Schéma illustrant les deux architecture usuelles pour implémenter un assistant conversationnel : (1) représente les approches symboliques avec un schéma d'architecture de gestion d'états de dialogue, et (2) représente les approches génératives avec un schéma d'architecture à base de <i>transformers</i> , composée d'un encodeur et d'un décodeur.	160

Liste des figures

C.1 Exemples des propriétés de transitivité des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges). (1) et (2) représente les possibilités de déduction d'une contrainte ((c)) en fonction des deux autres ((a) et (b)). (3) représente deux composants connexes définis par la transitivité des contraintes MUST-LINK. Enfin, (4) représente un cas de conflit où une contrainte ((c)) ne correspond pas à sa déduction faite à partir des autres contraintes ((a) et (b)).	166
C.2 Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (<code>samp.farhest.same</code>), et de données issues de clusters différents et étant les plus proches les unes des autres (<code>samp.closest.diff</code>).	170
C.3 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page d'accueil de l'application	173
C.4 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page de gestion des projets	174
C.5 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page d'accueil du projet en cours	175
C.6 Diagramme d'états simplifié de l'application web implémentant notre méthodologie de <i>clustering</i> interactif.	176
C.7 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page de gestion des paramètres	177
C.8 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page d'inventaire des textes	178
C.9 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page d'inventaire des contraintes	179
C.10 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : page d'annotation d'une contrainte	180
C.11 Capture d'écran de l'application web implémentant notre méthodologie de <i>clustering</i> interactif : graphe de contraintes présentant un conflit d'annotation . 180	

Liste des tableaux

2.1	Exemple d'annotation du prix de vente de bandes dessinées en fonction de leur édition, de la note de leur lecteurs et de leur état (source : https://www.bedetheque.com/serie-213-BD-Lucky-Luke.html).	6
4.1	Détails de l'évolution de la moyenne de la v-measure entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de <i>clustering</i> interactif, moyenne réalisée itération par itération sur l'ensemble des tentatives.	57
4.2	ANOVA du nombre d'itérations nécessaires pour l'obtention de 90% de v-mesure. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne Moyenne indiquent le classement des niveaux selon les analyses post-hoc.	64
4.3	ANOVA du nombre d'itérations nécessaires pour l'obtention de 100% de v-mesure. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne Moyenne indiquent le classement des niveaux selon les analyses post-hoc.	65
4.4	ANOVA du nombre d'itérations nécessaires pour annoter toutes les contraintes possibles. Les (*) dénotent le niveau de significativité ($\alpha = 0.05$). Pour les effets significatifs, les chiffres précisés entre parenthèses dans la colonne Moyenne indiquent le classement des niveaux selon les analyses post-hoc.	67
4.5	Extrait de l'analyse linguistique de <i>clusters</i> exploitables dès la première itération. Ces <i>clusters</i> représentent la thématique gestion_sans_contact entre l'itération 0 (initialisation) et l'itération 15 (atteinte de la vérité terrain). La troisième colonne expose un aperçu du contenu des <i>clusters</i> en mettant l'emphase sur les termes caractéristiques identifiés grâce à la FMC, et la deuxième colonne représente le top 10 de ces termes les plus caractéristiques pour chaque <i>cluster</i>	108
4.6	Extrait de l'analyse linguistique de <i>clusters</i> évoluant de non exploitables à exploitables. Ces <i>clusters</i> représentent la conception des thématiques gestion_carte_virtuelle et debloque_carte , entre l'itération 0 (initialisation) et l'itération 15 (atteinte de la vérité terrain). La troisième colonne expose un aperçu du contenu des <i>clusters</i> en mettant l'emphase sur les termes caractéristiques identifiés grâce à la FMC, et la deuxième colonne représente le top 10 de ces termes les plus caractéristiques pour chaque <i>cluster</i>	109
4.7	Extrait de l'analyse de résumés automatiques de <i>clusters</i> exploitables dès la première itération. Ces <i>clusters</i> représentent la thématique gestion_sans_contact entre l'itération 0 (initialisation) et l'itération 15 (atteinte de la vérité terrain). La seconde colonne expose le résumé obtenu en appelant un large modèle de langage (gpt-3.5-turbo) sur une tâche de résumé.	115

4.8 Extrait de l'analyse de résumés automatiques de <i>clusters</i> évoluant de non exploitables à exploitables. Ces <i>clusters</i> représentent la conception des thématiques <i>gestion_carte_virtuelle</i> et <i>debloque_carte</i> , entre l'itération 0 (initialisation) et l'itération 15 (atteinte de la vérité terrain). La seconde colonne expose le résumé obtenu en appelant un large modèle de langage (gpt-3.5-turbo) sur une tâche de résumé.	116
4.9 Score de corrélation r de <i>Pearson</i> entre la performance du <i>clustering</i> obtenu à l'aide d'une vérité terrain (v-measure) et le score d'accord entre annotation et <i>clustering</i>	123
4.10 Score de corrélation r de <i>Pearson</i> entre la performance du <i>clustering</i> obtenu à l'aide d'une vérité terrain (v-measure) et le score de différence entre deux <i>clusters</i> consécutifs.	127
4.11 Score d'accord avec la vérité terrain des 4 opérateurs (1 réviseur, 3 annotateurs) sur un lot commun de 400 contraintes (200 MUST-LINK, 200 CANNOT-LINK). L'accord brut représente le pourcentage de contraintes ayant la même annotation et l'accord α représente la mesure de Krippendorff. Les numéros d'opérateurs correspondent à leur identifiants leur de l'expérience.	133
4.12 Score d'accord inter-opérateurs (1 réviseur, 3 annotateurs) sur un lot commun de 400 contraintes (200 MUST-LINK, 200 CANNOT-LINK). L'accord brut représente le pourcentage de contraintes ayant la même annotation et l'accord α représente la mesure de Krippendorff. Les numéros d'opérateurs correspondent à leur identifiants leur de l'expérience.	133
4.13 Estimation de la similitude moyenne (calculée en terme de v-measure) des résultats de <i>clustering</i> des tentatives introduisant des désaccords d'annotation par rapport aux résultats de clustering de leurs tentatives de référence . Cette similitude est rapportée en fonction de la taille du jeu de données utilisé et du taux de désaccords introduits lors des tentatives. Pour chaque taille de jeu de données, les calculs sont réalisés avec un nombre de contraintes fixe, choisi comme étant le nombre de contraintes nécessaires à une tentative de référence pour atteindre une v-measure moyenne de 90% avec sa vérité terrain (ce nombre est rapporté en deuxième ligne).	145
A.1 Présentation du jeu de données Bank Cards avec quelques exemples. La version 2.0.0 contient 100 questions par intention.	156
A.2 Présentation du jeu de données échantillonné à partir de MLSUM avec quelques exemples.	157
C.1 Table de nomenclatures des noms d'algorithmes utilisés.	162
C.2 Table de nomenclatures des noms de variables et de paramètres utilisés.	163

Liste des algorithmes

3.1	<i>Description en pseudo-code de la méthode d'annotation proposée employant le clustering interactif</i>	45
4.1	<i>Description en pseudo-code du protocole expérimental de l'étude de convergence du clustering interactif vers une vérité terrain pré-établie.</i>	55
4.2	<i>Description en pseudo-code du protocole expérimental de l'étude d'optimisation de la convergence du clustering interactif vers une vérité terrain pré-établie.</i>	61
4.3	<i>Description en pseudo-code du protocole expérimental de l'étude du temps d'annotation d'un lot de contraintes par plusieurs experts métiers en situation réelle.</i>	73
4.4	<i>Description en pseudo-code du protocole expérimental de l'étude du temps d'exécution des algorithmes du clustering interactif.</i>	82
4.5	<i>Description en pseudo-code du protocole expérimental de l'étude du nombre de contraintes nécessaires pour converger vers une vérité terrain pré-établie avec notre paramétrage favori du clustering interactif.</i>	91
4.6	<i>Description en pseudo-code du protocole expérimental de l'étude de validation manuelle non assistée de la valeur métier d'une base d'apprentissage.</i>	101
4.7	<i>Description en pseudo-code du protocole expérimental de l'étude des patterns linguistiques pertinents pour vérifier la valeur métier d'une base d'apprentissage.</i>	106
4.8	<i>Description en pseudo-code du protocole expérimental de l'étude d'un résumé automatique des clusters à l'aide d'un large modèle de langage pour vérifier la valeur métier d'une base d'apprentissage.</i>	112
4.9	<i>Description en pseudo-code du protocole expérimental de l'étude de l'évolution d'accord entre l'annotation et le clustering.</i>	121
4.10	<i>Description en pseudo-code du protocole expérimental de l'étude de l'évolution de la différence entre deux clustering consécutifs.</i>	125
4.11	<i>Description en pseudo-code du protocole expérimental de l'étude du score inter-annotateurs d'annotation d'un lot de contraintes par plusieurs experts métiers en situation réelle.</i>	131
4.12	<i>Description en pseudo-code du protocole expérimental de l'étude d'impact d'une erreur d'annotation et l'intérêt de la corriger.</i>	135
4.13	<i>Description en pseudo-code du protocole expérimental de l'impact de la subjectivité de l'annotation sur la divergence des résultats.</i>	141

LISTE DES ALGORITHMES

Liste de codes informatiques

C.1	Jeu exemple pour présenter notre implémentation du <i>clustering</i> interactif.	163
C.2	Démonstration de notre implémentation du prétraitement et de la vectorisation sur le jeu d'exemple.	164
C.3	Démonstration de notre implémentation de gestion des contraintes sur le jeu d'exemple.	166
C.4	Démonstration de notre implémentation du <i>clustering</i> sous contraintes sur le jeu d'exemple.	168
C.5	Démonstration de notre implémentation de l'échantillonnage sur le jeu d'exemple.	169