

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 :

- ma femme
- ma tortue
- ma famille
- mes amis
- mes collègues
- mes encadrants
- chatGPT qui m'a aidé à corriger ce template LaTeX
- ...

*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	3
2.1.1 Définition et objectifs de l'annotation de données	4
2.1.2 Exemples de tâches d'annotations	5
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	16
2.3 Les 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	20
2.3.3 Défis concernant les différences de comportements intra- et inter- annotateurs	20
2.4 Techniques et organisations avancées d'annotation	21
2.4.1 Avancées concernant le besoin de qualité des données	21

2.4.2	Avancées concernant la diminution de la complexité de la tâche d'annotation	21
2.4.3	Avancées concernant la réduction des différences de comportements intra- et inter-annotateurs	21
2.5	(<i>conception toujours difficile en entreprise</i>)	21

3

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

3.1	Intuitions à l'origine	24
3.2	Description théorique	24
3.3	Description technique et implémentation	27
3.3.1	Gestion des données	27
3.3.2	Gestion des contraintes	29
3.3.3	Algorithme de <i>clustering</i> sous contraintes	32
3.3.4	Algorithme d'échantillonnage de contraintes	33
3.3.5	todo	34
3.4	Espoirs de la méthode proposée	35

4

Étude de six hypothèses sur le *Clustering Interactif*

4.1	Évaluation de l'hypothèse d'efficacité	41
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 .	41
4.2	Évaluation de l'hypothèse d'efficience	47
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 .	47
4.3	Évaluation de l'hypothèse sur les coûts	58
4.3.1	Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle	59
4.3.2	Étude du temps de calcul nécessaire aux algorithmes implantés en chronométrant des exécutions dans différentes situations	68
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	77
4.3.4	Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts	82
4.3.5	Ouverture vers une annotation en parallèle du <i>clustering</i>	84

4.4	Évaluation de l'hypothèse de pertinence	87
4.4.1	Étude d'une validation manuelle et non assistée de la valeur métier d'une base d'apprentissage par un expert	88
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	92
4.4.3	Étude d'un résumé automatique des <i>clusters</i> à l'aide d'un large modèle de langage	98
4.4.4	Mise en commun des stratégies d'évaluation de la pertinence métier d'un résultat de <i>clustering</i> interactif	105
4.5	Évaluation de l'hypothèse de rentabilité	106
4.5.1	Étude de l'évolution d'accord entre l'annotation et le <i>clustering</i>	107
4.5.2	Étude de l'évolution de la différence entre deux <i>clusterings</i> consécutifs	111
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.	115
4.6	Évaluation de l'hypothèse de robustesse	116
4.6.1	Étude de l'intérêt de la correction des incohérences d'annotation	117
4.6.2	Étude de l'impact des incohérences d'annotation sur les performances	121
4.6.3	Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle	121
4.6.4	Bilan concernant la robustesse du <i>clustering</i> interactif	124
4.7	Autres hypothèses non vérifiées	125
4.7.1	Etude du nombre de clusters optimal.	125
4.7.2	Etude d'autres méthodes de vectorisation	125
4.7.3	Etude d'autres méthodes d'échantillonnage	126
4.7.4	Etude ergonomique de l'interface d'annotation	126
4.8	Bilan des études réalisées	127

5

Guide d'utilisation du *Clustering Interactif*

5.1	Organisation ITTER	129
5.2	Conseils pratiques	129

6

Conclusion

6.1	(<i>Occupying the Niche</i>)	131
6.2	(<i>Gap</i>)	131

LISTE DE CODES INFORMATIQUES

6.3 (<i>Establishing a Niche</i>)	131
6.4 (<i>Asset centrality</i>)	131

Annexes

A

Jeux de données

A.1 Bank Cards : Jeu d'entraînement en français d'assistants conversationnels traitant des demandes courantes sur les cartes bancaires.	133
A.2 MLSUM : The Multilingual Summarization Corpus (échantillon)	134

Bibliographie	137
----------------------	------------

Liste des TODOs	143
------------------------	------------

Liste des figures	147
--------------------------	------------

Liste des tableaux	153
---------------------------	------------

Liste des algorithmes	155
------------------------------	------------

Liste de codes informatiques	157
-------------------------------------	------------

TITRE A REVOIR : Faciliter => Accélérer ? Améliorer l'accessibilité ? Limiter les biais et erreurs ? ...

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

CHAPITRE : TITRE À TROUVER : "*(Revue de littérature)*"

CHAPITRE : INTRODUCTION À RÉDIGER

CHAPITRE : INTRODUCTION À RÉDIGER

Sommaire

2.1	Présentation théorique de l'annotation	3
2.1.1	Définition et objectifs de l'annotation de données	4
2.1.2	Exemples de tâches d'annotations	5
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	16
2.3	Les 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	20
2.3.3	Défis concernant les différences de comportements intra- et inter- annotateurs	20
2.4	Techniques et organisations avancées d'annotation	21
2.4.1	Avancées concernant le besoin de qualité des données	21
2.4.2	Avancées concernant la diminution de la complexité de la tâche d'annotation	21
2.4.3	Avancées concernant la réduction des différences de comportements intra- et inter-annotateurs	21
2.5	(conception toujours difficile en entreprise)	21

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

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 statistique. Cette approche, très populaire en ce moment, permet 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, ...

💡 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.

Qu'est qu'un « corpus d'entraînement » ?

Pour concevoir un modèle statistique 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 apprêhender : 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).

💡 Pour information : Si vous voulez mieux comprendre cette notion de corpus, vous pouvez vous référer à SINCLAIR, 2004 issu du livre *Developing Linguistic Corpora* (WYNNE, 2004).

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** » des données dites « **annotées** » en fonction de l'absence ou de la présence d'un complément d'informations.

Les informations renseignées peuvent porter sur la donnée en entité ou 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 (en entraînant par exemple un modèle 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 pensons qu'il est préférable de compléter ces définitions par quelques exemples concrets : nous pourrons ainsi mieux dresser le portait d'une tâche d'annotation, avec ses intérêts et ses complications. Pour cela, nous allons prendre le thème de la bande dessinée et de ses dérivés, et explorer ensemble différents cas d'usage qui pourraient intéresser un auteur, un libraire ou un lecteur.

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 linéaire**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).

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

Exemples :

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

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.

❶ Pour information : De manière équivalente, il est possible de faire de la régression linéaire 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.

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 nous 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 **classification**² permettant, à partir d'une image, de d'affecter chaque BD à une catégorie prédéfinie (par exemple : "Mauvais état", "Bon état", "Très bon état", "Neuf"). Mais pour entraîner un tel modèle, il est nécessaire d'avoir une base d'apprentissage contenant des exemples d'image de BD associée 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 ??.

Exemples :

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 la catégorie d'état la 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 |

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. Mais 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'entourant sur l'image (avec un rectangle par exemple) ;
- la valeur écrite dans encadré sur l'image.

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

Exemples :

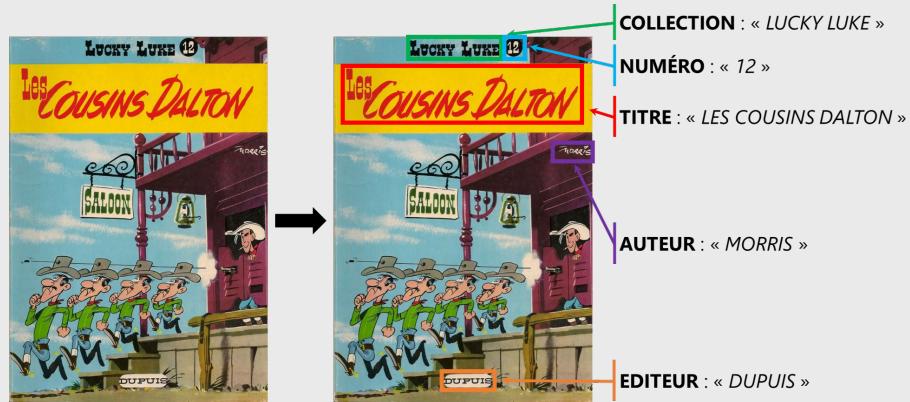


FIGURE 2.2 – Exemple d'annotation de textes présents sur la couverture d'une bande dessinée (ici : MORRIS et GOSCINNY, 1958). Les informations essentielles telles que la collection, le numéro, le titre, l'auteur et l'éditeur y sont présentes.

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)⁴ pour extraire les informations importantes citées dans un texte. Dans l'exemple ci-dessous, les types d'entités présentes sont "personnage", "métier", "argent", "lieu" et "date", et 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.

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 éducatives ou simplement disponibles pour le loisir. Mais 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. Mais 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 :

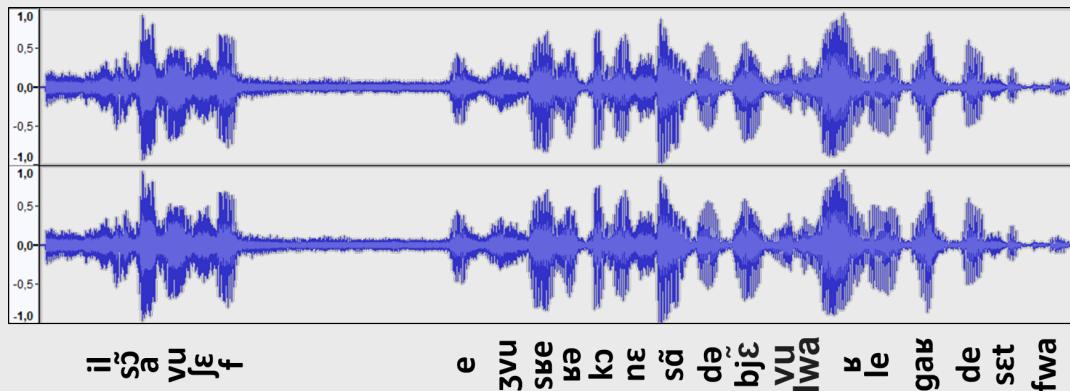


FIGURE 2.3 – Exemple de paroles prononcées dans un audio. Ici, la voix de Lucky Luke est interprétée par Jacques THEBAULT. Le texte annoté, c'est-à-dire celui prononcé dans l'audio, est « Ils sont à vous chef, et j'veus s'rapi reconnaissant de bien les garder cette fois. ». Les phonèmes en alphabet phonétique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

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

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.

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 est une tâche plutôt complexe, et chaque tâche de *Machine Learning* demande ses propres données annotées pour construire une base d'apprentissage.

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 linéaire, 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 d'un modèle de *Machine Learning*. Maintenant, nous allons 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

Une référence en matière d'organisation de projet d'annotation 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.

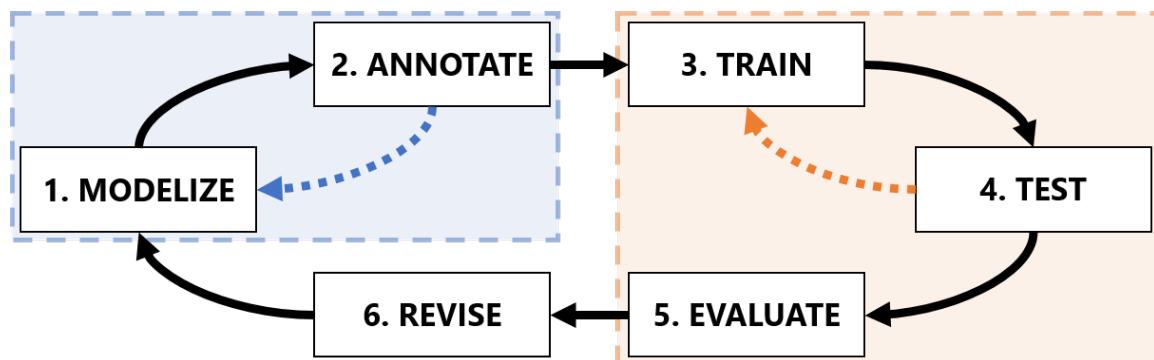


FIGURE 2.4 – Cycle **MATTER** structurant un projet d'annotation en six étapes principales : **Modelize**, **Annotate**, **Train**, **Test**, **Evaluate** et **Revise**.

Note de l'auteur : Nous conseillons 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 importants de chacune des étapes.

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

Pour obtenir un bon 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. 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 après d'utilisateurs finaux, ou encore sur les avis éclairés d'experts du problème. Ces données brutes ont ensuite 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 ainsi que les informations à annoter peuvent 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** pour clarifier les attendus et les modalités de cette annotation (voir FIGURE 2.4, étape 1. *Modelize*).

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

- la **modélisation** du problème, représentation 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 *quois* 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), 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 détail 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 qui ne soit pas trop pénible.

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

- par soucis d'*inter-opérabilité*, nous pourrions plutôt utiliser la norme ISO 639-3 (« Codes for the Representation of Names of Languages – Part 3 : Alpha-3 Code for Comprehensive Coverage of Languages », 2007), soit les code "`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*).

❶ 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 si vous avez besoin de référence pour concevoir votre propre guide.

Lorsque le guide d'annotation est rédigé, la **phase de labellisation** peut commencer (voir FIGURE 2.4, étape 2. *Annotate*). Cette tâche est traditionnellement réalisée par un groupe d'experts choisi en fonction de leur connaissance sur 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, et 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, 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*.

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 (voir FIGURE 2.4, étapes 3. *Train*, 4. *Test* et 5. *Evaluate*).

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. Il est important que ce jeu de données

Ainsi, le modèle représente la connaissance présente dans le jeu **entraînement**, il est étudié puis affiné grâce au jeu de **développement**, et est finalement jugé 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 bonne prédiction), 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 bien tenir à l'écart le jeu de test des deux autres jeux de données et qu'il ne soit pas utilisé dans la phase de développement pour éviter tout biais de sur-apprentissage⁶.

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

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 mauvaise 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 obsolète 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 ;
- ou encore, un nouvel algorithme de *Machine Learning* a priori plus performant requiert une modélisation différente pour traiter le problème.

Exemples : Pour illustrer nos propos, prenons la tâche d'estimation du prix d'une bande dessinée (cf. SECTION 2.1.2) : 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 impacté : 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), 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), certains textes du décors pourrait être extrait à tord (comme le texte de la pancarte « *Saloon* » dans la FIGURE ??). 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.

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

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 au 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 ses compétences qu'un acteur peut concevoir le guide d'annotation, estimer le taux d'accord 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. et 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 ;
 - 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. En effet, une diversité d'applications existent pour répondre aux besoins des annotateurs, mais il est important de noter que l'absence de certaines fonctionnalités essentielles risque de gêner le projet d'annotation, soit par l'intro-

duction de biais, soit à cause de son manque d'inter-opérabilité avec d'autres tâches d'analyses ou 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 à la **besoin d'annotation** : cette fonctionnalité est bien entendu obligatoire, car un logiciel ne permettant pas d'annoter vos données ne vous sera d'aucune 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 plusieurs 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 natures différentes (*en combinant texte, image et voix comme dans l'annotation des sous-titres d'une vidéo*). Les logiciels permettant ainsi de labelliser plusieurs informations et de manipuler plusieurs types de données sont donc plus facilement réutilisables et permettent de centraliser les annotations ;
- 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 apparent lors d'avis discordants ;
- permettre l'**inter-opérabilité** technique : il est toujours 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 annotation 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 pouvant assister un annotateur dans son travail (*pré-annotation, apprentissage actif, visualisation, interaction, ...*). Nous détaillerons plusieurs de ces techniques dans la SECTION 2.4.

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.

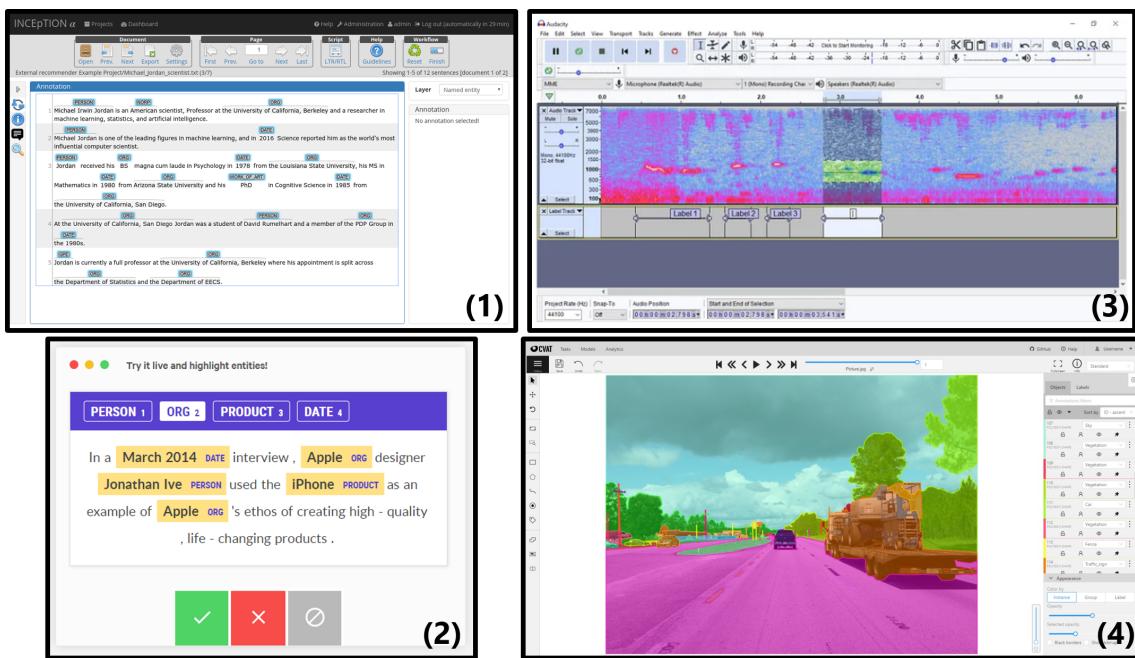


FIGURE 2.5 – Quatre exemples d'outils d'annotation : (1) INCEPTION pour le texte (KLIE et al., 2018), (2) prodigy pour le texte ou l'image (MONTANI et HONNIBAL, 2017), (3) Audacity pour l'audio (AUDACITY TEAM, 2000) et (4) CVAT pour l'image (CVAT.AI CORPORATION, 2019).

Note de l'auteur : De part notre expérience, nous constatons malheureusement que plusieurs projets industriels n'utilisent pas ou peu d'outils d'annotation dédiés, et se contentent plutôt 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 biais potentiels de l'annotation que ces outils aux fonctionnalités avancées ne sont pas employés ?

Points à retenir :

- ✓ Un projet d'annotation s'organise généralement 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**, **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 Les 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 ou des biais, explicites ou implicites, 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 constituant 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).

Nous profiterons aussi de ces points pour discuter des techniques et bonnes pratiques permettant de limiter les désagréments lors d'un projet d'annotation et identifier les freins lors de mises en application industrielles.

2.3.1 Défis concernant le besoin de qualité des données

Comme nous l'avons défini en SECTION2.1.1, 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 la création d'un jeu de données.

Problèmes de représentativité

Tout d'abord, accordons-nous sur le fait que certains phénomènes sont par essence complexes à représenter avec fidélité. C'est par exemple le cas avec :

- le traitement du langage :

SUITE A REDIGER

Exemples :

Problèmes de bruits

Problèmes de droits d'utilisation

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

SECTION : À RÉDIGER

2.3.3 Défis concernant les différences de comportements intra- et inter-annotateurs

SECTION : À RÉDIGER

Points à retenir :

- ✓ L'enjeu d'un projet d'annotation consiste à avoir des **données de qualité** qui soient représentatives du problème à traiter ;
- ✓ Or la tâche d'annotation et son exigence de qualité engendre de la **complexité**, et donc une **charge de travail élevée** ;
- ✓ Pour réguler cette charge de travail élevée, chaque opérateur va **adapter sa tâche** pour la rendre supportable, créant alors des **différences de comportement**.

2.4 Techniques et organisations avancées d'annotation

SECTION : À RÉDIGER :

- Sur les données : ...
- Sur la complexité : ...
- Sur les annotateurs : ...

2.4.1 Avancées concernant le besoin de qualité des données

SECTION : À RÉDIGER

2.4.2 Avancées concernant la diminution de la complexité de la tâche d'annotation

SECTION : À RÉDIGER

2.4.3 Avancées concernant la réduction des différences de comportements intra- et inter-annotateurs

SECTION : À RÉDIGER

Points à retenir :

☒ (TODO)

2.5 (*conception toujours difficile en entreprise*)

SECTION : TITRE À TROUVER : "REX entreprise"

SECTION : À RÉDIGER :

- Modélisation toujours compliquée ;
- Expert métier "pas à leur place" ;
- Peu de stratégies de notre revue mises en oeuvre...

Chapitre 3

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

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

- ✓ Dans un cadre industriel, le choix de l'algorithme utilisé pour l'entraînement d'un modèle est déterminé à l'avance, donc la qualité de l'assistant repose principalement sur la fiabilité et la pertinence de son jeu de données ;
- ✓ Pour concevoir ce jeu de données, il est nécessaire de faire appel à des experts maîtrisant le domaine à couvrir par l'assistant car les données sont en général spécifiques ou privées ;
- ✓ L'intervention de ces experts métiers au sein du projet est en général laborieuse : d'une part à cause de leur manque de connaissances en data science (ce n'est pas leur domaine d'expertise), d'autre part à cause de la complexité inhérente des tâches de modélisation et d'annotation des données.
- ✓ Par manque de compétences, de connaissances ou d'ergonomie, la tâche de conception d'un jeu de données reste manuelle et est encore mal assistée par ordinateur.

À REFORMULER APRÈS L'ÉCRITURE DE L'ÉTAT DE L'ART

Dans cette partie, nous proposons une alternative à l'organisation manuelle destinée à la conception d'un jeu de données. Notre proposition vise à remplir un double objectif :

- Proposer une méthode permettant d'assister la modélisation et l'annotation des données pour créer plus efficacement une base d'apprentissage pour la classification d'intention 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 les experts métiers intervenants dans le projet.

Sommaire

3.1	Intuitions à l'origine	24
3.2	Description théorique	24
3.3	Description technique et implémentation	27
3.3.1	Gestion des données	27
3.3.2	Gestion des contraintes	29
3.3.3	Algorithme de <i>clustering</i> sous contraintes	32
3.3.4	Algorithme d'échantillonnage de contraintes	33
3.3.5	todo	34
3.4	Espoirs de la méthode proposée	35

3.1 Intuitions à l'origine

La pierre angulaire de notre méthode repose 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 initiales, nécessite en contre-partie de nombreuses formations, et introduit de nombreuses erreurs d'annotations.

Remarque Gautier 20/02/2023 : erreur de routine, erreur par manque de connaissance, ... Il faudra discuter les causes de ces erreurs

De fait, il semble plus adéquat de demander à l'expert métier de discriminer deux questions sur la base de leurs réponses : une telle approche demande une charge de travail plus faible et est plus intuitive car elle est plus proche des compétences réelles de l'annotateur. Ainsi, nous basons notre méthode sur l'annotation de contraintes sur les données.

Toutefois, l'annotation de contraintes semble elle aussi fastidieuse. En effet, pour faire émerger une base d'apprentissage, il faut annoter un grand nombre de contraintes et être attentifs aux éventuelles incohérences pour ne pas introduire de contraintes contradictoires. Pour assister l'expert dans cette tâche, nous avons donc décidé de l'intégrer dans une stratégie d'apprentissage actif en essayant de tirer parti des interactions possibles avec la machine. Ce choix est motivé entre autres par l'intuition qu'il est possible de coopérer avec la machine pour obtenir plus efficacement un résultat pertinent.

C'est sur la combinaison de ces deux éléments que repose notre méthode d'annotation pour concevoir le jeu d'entraînement de notre assistant conversationnel.

3.2 Description théorique

Description détaillée. Nous proposons la méthode suivante pour transformer une collecte de données brutes en une base d'apprentissage nécessaire à l'entraînement d'un assistant conversationnel. Cette méthode, que nous appelons *Clustering Interactif* repose principalement sur l'alternance successive entre deux phases clefs (voir FIGURE 3.1) :

- une phase d'**annotation de contraintes** par un expert sur la base des connaissances qu'il détient ;

- 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 la création d'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 pour corriger le fonctionnement de la machine et ainsi obtenir une segmentation plus pertinente à l'itération suivante.

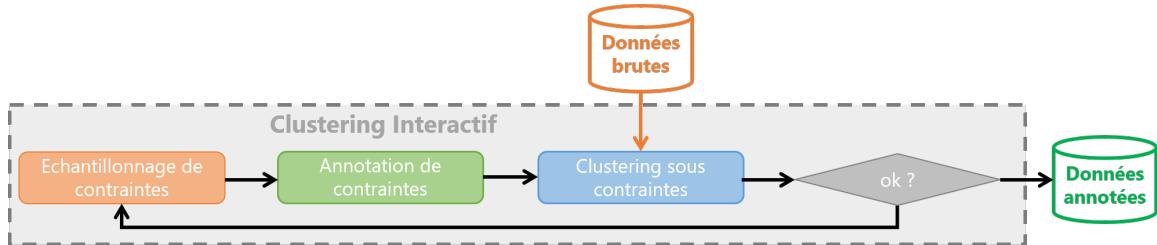


FIGURE 3.1 – Schéma illustrant l'architecture du clustering interactif. La boucle principale enchaîne un échantillonnage de couples de données, une annotation de contraintes, et un clustering sous contraintes.

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

Entrées :	données non segmentées ; 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.

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 ajouterons 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 de 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 coeur 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 possibilité croît proportionnellement au carré du nombre de données). De plus, l'intervention d'expert est chiffrée et représente en général la majeure partie des coûts à investir dans un projet. 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 aura le plus d'impact lors de sa transmission de connaissance. 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 leur relations avec d'autres données déjà liées par d'autres 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 prochain 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 concernée⁷, sur l'action désirée⁸, ou encore sur le besoin de l'utilisateur⁹. On notera que des incohérences peuvent s'introduire, ayant pour conclusions de devoir à la fois considérer comme similaires et différentes deux données (voir la gestion des conflits en SECTION 3.3.2).

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 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 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.

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

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

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

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.

3.3 Description technique et implémentation

Nous avons réalisé une implémentation en Python de notre *clustering interactif*. Celle-ci est répartie en trois librairies :

1. `cognitivefactory-interactive-clustering`, regroupant les gestions de données, de contraintes et les algorithmes de *Machine Learning* qui ont été implémentés ;
2. `cognitivefactory-features-maximization-metrics`, disposant d'une méthode de sélection des patterns linguistiques pertinents (composantes principales) d'un jeu de données ;
3. `cognitivefactory-interactive-clustering-gui`, encapsulant les algorithmes précédents et intégrant la logique de la méthode dans une interface graphique.

Pour les sections suivantes, nous suivrons l'exemple suivant (cf. CODE 3.1) 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 3.1 – Jeu exemple pour présenter notre implémentation du clustering interactif.

3.3.1 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`¹⁰ qui permet de normaliser les données. Par défaut :

- 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 blanches* 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 dépendance syntaxique (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`¹¹ 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 3.2.

```
1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.utils.preprocessing
    import preprocess
```

10. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/utils/preprocessing/

11. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/utils/vectorization/

```

3 from cognitivefactory.interactive_clustering.utils.vectorization
4     import vectorize
5
6     # Prétraitement des données.
7     dict_of_preprocess_texts = preprocess(
8         dict_of_texts=dict_of_texts,
9         apply_stopwords_deletion=False,
10        apply_parsing_filter=False,
11        apply_lemmatization=False,
12        spacy_language_model="fr_core_news_md",
13    )
14 """
15     {"0": "comment signaler un vol de carte bancaire",
16      "1": "j ai egare ma carte bancaire , que faire",
17      "2": "j ai perdu ma carte de paiement",
18      "3": "le distributeur a avale ma carte",
19      "4": "en retirant de l argent le gab a garde ma carte",
20      "5": "le distributeur ne m a pas rendu ma carte bleue",
21      "# ...",
22      "N": "pourquoi le sans contact ne fonctionne pas"}
23 """
24
25     # Vectorisation des données.
26     dict_of_vectors = vectorize(
27         dict_of_texts=dict_of_preprocess_texts,
28         vectorizer_type="tfidf",
29     )

```

CODE 3.2 – Démonstration de notre implémentation du prétraitement et de la vectorisation sur le jeu d'exemple.

3.3.2 Gestion des contraintes

En ce qui concerne la **manipulation de contraintes**, nous utilisons le module `constraints`¹² 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 3.1) et sont illustrées dans la FIGURE 3.2 ((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.

$$(\forall A, B, C) \left\{ \begin{array}{l} \text{MUST_LINK}(A, B) \wedge \text{MUST_LINK}(B, C) \Rightarrow \text{MUST_LINK}(A, C) \\ \text{MUST_LINK}(A, B) \wedge \text{CANNOT_LINK}(B, C) \Rightarrow \text{CANNOT_LINK}(A, C) \end{array} \right. \quad (3.1)$$

12. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/constraints/

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**).

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 3.2 et EQUATION 3.2). 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...

$$(\exists A, B, C) \text{ MUST_LINK}(A, B) \wedge \text{MUST_LINK}(B, C) \wedge \text{CANNOT_LINK}(A, C) \quad (3.2)$$

A 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 3.2). 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*.

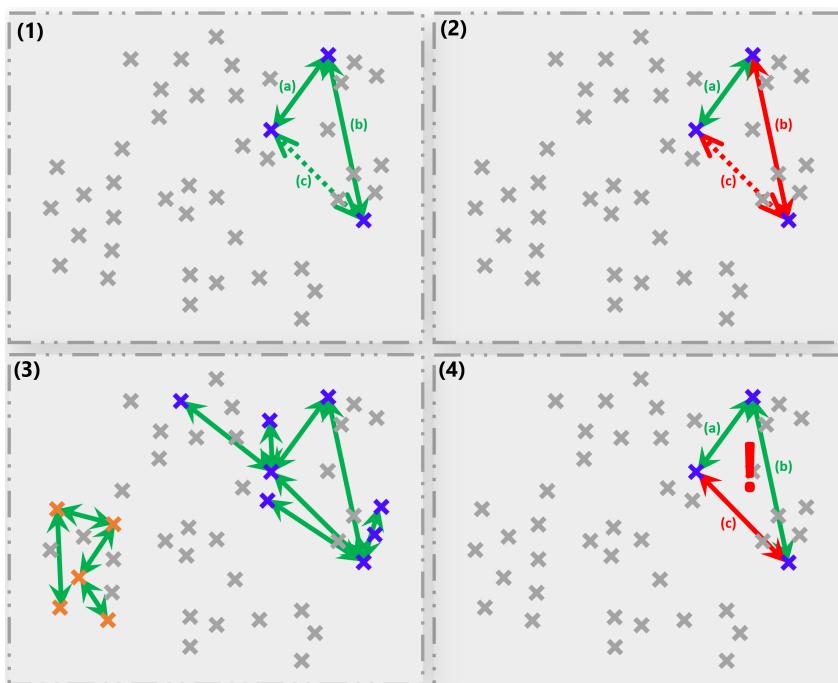


FIGURE 3.2 – Exemples des propriétés de transitivité des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges). (1) et (2) représente les possibilités de déduction d'une contrainte ((c)) en fonction des deux autres ((a) et (b)). (3) représente deux composants connexes définis par la 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 3.3.

1 `# Import des dépendances .`

```

2 from cognitivefactory.interactive_clustering.constraints.factory
3     import managing_factory
4
5 # Création du gestionnaire de contraintes.
6 constraints_manager = managing_factory(
7     manager="binary",
8     list_of_data_IDs = list(dict_of_texts.keys()), # [ "0" , "1" , "2" ,
9                               "3" , "4" , "5" , ... , "N" ]
10 )
11
12 # Ajout de contraintes.
13 constraints_manager.add_constraint(
14     data_ID1="0" , # "Comment signaler un vol de carte bancaire ?"
15     data_ID2="1" , # "J'ai égaré ma carte bancaire , que faire ?"
16     constraint_type="MUST_LINK" ,
17 )
18 constraints_manager.add_constraint(
19     data_ID1="3" , # "Le distributeur a avalé ma carte !"
20     data_ID2="4" , # "En retirant de l'argent , le GAB a gardé ma carte
21                               ...
22     constraint_type="MUST_LINK" ,
23 )
24 constraints_manager.add_constraint(
25     data_ID1="0" , # "Comment signaler un vol de carte bancaire ?"
26     data_ID2="N" , # "Pourquoi le sans contact ne fonctionne pas ?"
27     constraint_type="CANNOT_LINK" ,
28 )
29 # NB: ajouter une contrainte "MUST_LINK" entre "1" et "N" lèverait
30             une erreur .
31
32 # Récupération des composants connexes.
33 connected_components = constraints_manager.get_connected_components()
34 """
35     [[ '0' , '1' ] ,
36      [ '2' ] ,
37      [ '3' , '4' ] ,
38      [ '5' ] ,
39      [ 'N' ] ]
40 """

```

CODE 3.3 – Démonstration de notre implémentation de gestion des contraintes sur le jeu d'exemple.

3.3.3 Algorithme de *clustering* sous contraintes

En ce qui concerne le **regroupement automatique** des données par similarité, nous utilisons le module **clustering**¹³ de la librairie **cognitivefactory-interactive-clustering**.

Ce module met à disposition six algorithmes de *clustering* sous contraintes :

1. **KMeans**, dans sa version COP (WAGSTAFF et al., 2001), noté `clust.kmeans.cop`, et sa version MPC (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 3.4.

```

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# Lancer le clustering.
12clustering_result = clustering_model.cluster(
13    constraints_manager=constraints_manager, # contient les
14        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."
# ...

```

13. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/clustering/

```
25 "N": 1} # Pourquoi le sans contact ne fonctionne pas ?"
26 """
```

CODE 3.4 – 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 KMeans (MPC), 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 (KMeans (COP), hiérarchique et spectral (SPEC)) ont été implémentés au début de ce doctorat.

3.3.4 Algorithme d'échantillonnage de contraintes

En ce qui concerne l'**échantillonnage** de contraintes à annoter, nous utilisons le module `sampling`¹⁴ 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.farhest.same` (cf. FIGURE 3.3) ;
4. Un échantillonnage des données issues de **clusters différents** et étant **les plus proches** les unes des autres, noté `samp.closest.diff` (cf. FIGURE 3.3).

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 3.5.

¹⁴. https://cognitivefactory.github.io/interactive-clustering/reference/cognitivefactory/interactive_clustering/sampling/

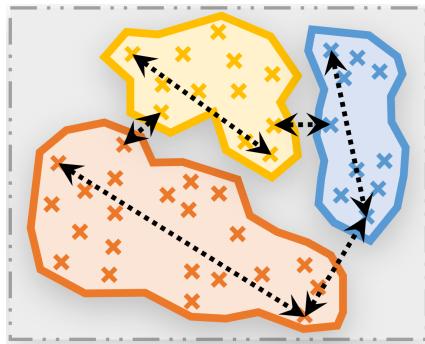


FIGURE 3.3 – Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (`samp.farhest.same`), et de données issues de clusters différents et étant les plus proches les unes des autres (`samp.closest.diff`).

```

1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.sampling.factory import
3     sampling_factory
4
5 # Initialiser un objet d'échantillonnage.
6 sampler = sampling_factory(
7     algorithm="random",
8     random_seed=42,
9 )
10
11 # Run sampling.
12 selection = sampler.sample(
13     constraints_manager=constraints_manager,
14     nb_to_select=2,
15     clustering_result=clustering_result, # optionnel pour "random"
16     vectors=dict_of_vectors, # optionnel pour "random"
17 )
18 """
19     [("0", '5'), # "Comment signaler un vol de carte bancaire ?" vs "
20      "Le distributeur ne m'a pas rendu ma carte bleue."
21     ("0", '2'), # "Comment signaler un vol de carte bancaire ?" vs "J
22      'ai perdu ma carte de paiement"
23     ("2", 'N')] # "J'ai perdu ma carte de paiement" vs "Pourquoi le
24      sans contact ne fonctionne pas ?"
25 """

```

CODE 3.5 – Démonstration de notre implémentation de l'échantillonnage sur le jeu d'exemple.

3.3.5 todo

SECTION À RÉDIGER : FMC

SECTION À RÉDIGER : IC-GUI page d'annotation

SECTION À RÉDIGER : IC-GUI gestion d'état de l'application

SECTION À RÉDIGER : IC-GUI page d'analyse (en cours)

3.4 Espoirs de la méthode proposée

SECTION À RÉDIGER

- Moins de formations, d'ateliers, ... • Se concentrer sur son domaine de compétence (i.e. pas de datascience pour les experts métiers) • Permettre de trouver la base d'apprentissage • Méthode réaliste / pas trop coûteuse • ...

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 techniques actuellement exigés à 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.
- ...

divers à compléter (technique ? méthode ? ...).

Comme nous l'avons détaillé dans le CHAPITRE 2, des procédés d'annotation similaires existent pour des données facilement visualisables, comme dans le cadre du traitement d'images. Cependant, l'application d'une telle approche dans le cadre de la classification de données textuelles est peu détaillée dans la littérature. Ainsi, dans cette partie, nous étudierons la faisabilité d'un *clustering interactif* pour des données textuelles en explorant les questions suivantes :

- Peut-on obtenir une base d'apprentissage à l'aide de notre proposition d'implémentation de la méthodologie d'*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 raffinement supplémentaire ? (cf. hypothèse de **rentabilité** en SECTION 4.5) ;
- Peut-on estimer l'influence d'une erreur ou d'une différence d'annotation dans la construction de la base d'apprentissage ? (cf. hypothèse de **robustesse** en SECTION 4.6).

Afin d'illustrer ces interrogations, nous vous proposons de considérer de la FIGURE 4.1. Dans les sections suivantes, cette figure évoluera pour résumer les études réalisées.



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).

Pour ces études, nous allons (1) faire des analyses théoriques (2) des analyses empiriques (car dans la vrai vie on n'a pas de vérité terrain). Nous utilisons aussi la vmeasure.

Table des nomenclatures.

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 parallélisé avec la librairie Python *multiprocessing* (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 script R (TEAM, 2017), sont disponibles dans SCHILD, 2022b. Enfin, les jeux de données utilisés pour ces études sont détaillés en Annexe A.

Sommaire

4.1	Évaluation de l'hypothèse d'efficacité	41
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	41
4.2	Évaluation de l'hypothèse d'efficience	47
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	47
4.3	Évaluation de l'hypothèse sur les coûts	58
4.3.1	Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle	59
4.3.2	Étude du temps de calcul nécessaire aux algorithmes implémentés en chronométrant des exécutions dans différentes situations	68
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	77
4.3.4	Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts	82
4.3.5	Ouverture vers une annotation en parallèle du <i>clustering</i>	84
4.4	Évaluation de l'hypothèse de pertinence	87
4.4.1	Étude d'une validation manuelle et non assistée de la valeur métier d'une base d'apprentissage par un expert	88
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	92
4.4.3	Étude d'un résumé automatique des <i>clusters</i> à l'aide d'un large modèle de langage	98
4.4.4	Mise en commun des stratégies d'évaluation de la pertinence métier d'un résultat de <i>clustering</i> interactif	105
4.5	Évaluation de l'hypothèse de rentabilité	106
4.5.1	Étude de l'évolution d'accord entre l'annotation et le <i>clustering</i>	107
4.5.2	Étude de l'évolution de la différence entre deux <i>clusterings</i> consécutifs	111
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.	115
4.6	Évaluation de l'hypothèse de robustesse	116
4.6.1	Étude de l'intérêt de la correction des incohérences d'annotation	117
4.6.2	Étude de l'impact des incohérences d'annotation sur les performances	121
4.6.3	Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle	121
4.6.4	Bilan concernant la robustesse du <i>clustering</i> interactif	124
4.7	Autres hypothèses non vérifiées	125

4.7.1	Etude du nombre de clusters optimal	125
4.7.2	Etude d'autres méthodes de vectorisation	125
4.7.3	Etude d'autres méthodes d'échantillonnage	126
4.7.4	Etude ergonomique de l'interface d'annotation	126
4.8	Bilan des études réalisées	127

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 aimeraisons 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.

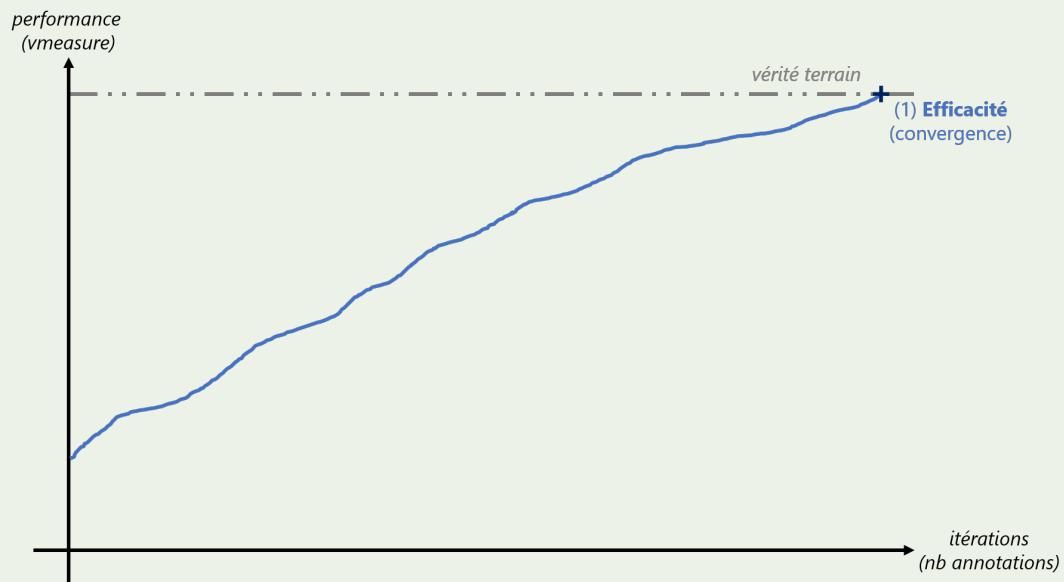


FIGURE 4.2 – Illustration des études réalisées sur le clustering interactif (étape 1/6) en schématisant l'évolution de la performance (accord avec la vérité terrain calculé en *v*-measure) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (nombre d'annotations 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 et al., 2022).¹⁵

Protocole expérimental

A 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 sera discuté en fin de partie, et nous nous y intéresserons 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.

Données : jeu de données annoté (vérité terrain)

Entrées : arrangements 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* obtenu et la vérité terrain ;

8 **répéter**

9 **échantillonnage** : sélectionner de nouvelles contraintes à annoter ;

10 **simulation d'annotation** : caractériser les contraintes grâce à 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* obtenu 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.

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 annotées par l'expert.

Pour cette étude, nous essayons une tentative pour chaque combinaison de paramètres de notre implémentation du *clustering* interactif (cf. SECTION 3.3). Cela comprend les tâches et leurs paramètres respectifs suivants :

1. le **prétraitement** des données, avec les niveaux suivants : **aucun** (noté `prep.no`), **simple** (noté `prep.simple`), avec **lemmatisation** (noté `prep.lemma`) et avec **filtres** (noté `prep.filter`) ;
2. la **vectorisation** des données, avec les niveaux suivants : **TF-IDF** (noté `vect.tfidf`) et **SpaCy** (noté `vect.frcorenewsmd`) ;
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 : **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`). Le choix de la taille d'échantillon n'est pas étudié ici, et cette taille est arbitrairement fixée à 50¹⁶.

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 la SECTION 3.3 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 proposent 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

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).

d'annotation basée sur le *clustering* interactif.

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

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'itération 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 les tentatives 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.

A l'itération 0, une tentative commence avec une moyenne de 19.05% de **v-measure** entre son *clustering* initial (sans contraintes) 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 toute 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 tentatives puisse annoter toutes les contraintes possibles sur le jeu de données. 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%).

Annotations		Performances (v-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.

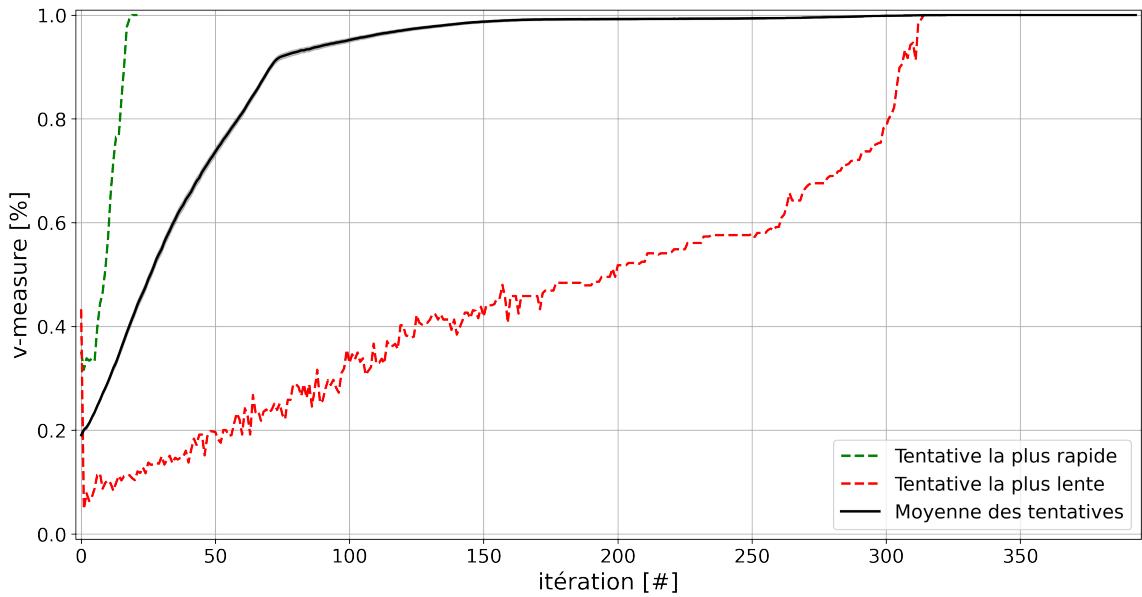


FIGURE 4.3 – Évolution de la moyenne de la *v*-measure entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de clustering interactif, moyenne réalisée itération par itération sur l'ensemble des tentatives. Représentation des tentatives ayant été les plus rapides (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.

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 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 été 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 paramétrage 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 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 connaissant parfaitement le domaine traité. Cette hypothèse forte n'est a priori pas valable en situation réelle : En effet, des erreurs 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).

4.2 Évaluation de l'hypothèse d'efficience

Suite à la validation de l'hypothèse d'efficacité (convergence de la méthode, cf. SECTION 4.1), nous déterminer les paramètres optimaux de la méthodes 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 étudierons 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'effet 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.

❶ 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 et al., 2022).¹⁷

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 sera discuté en fin de partie, et nous nous y intéresserons 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 annoté (vérité terrain)

Entrées : arrangements d'algorithmes et de paramètres à tester

```
1 pour chaque arrangement d'algorithme 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 obtenu et la vérité terrain ;
8   répéter
9     échantillonnage : sélectionner de nouvelles contraintes à annoter ;
10    simulation d'annotation : caractériser les contraintes grâce à 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 obtenu 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 : meilleurs arrangements 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 paramétrage, 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 la SECTION 3.3.

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 totale¹⁸ ;
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 métier¹⁹.

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 (TEAM, 2017), sont disponibles dans un dossier dédié de SCHILD, 2022b.

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.
- 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`,

18. 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).

19. 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.

et un échantillonnage de contraintes `samp.farthest.same`. Ces tentatives ont requis 315 itérations, soit 15 750 annotations, dont 1 032 contraintes MUST-LINK.

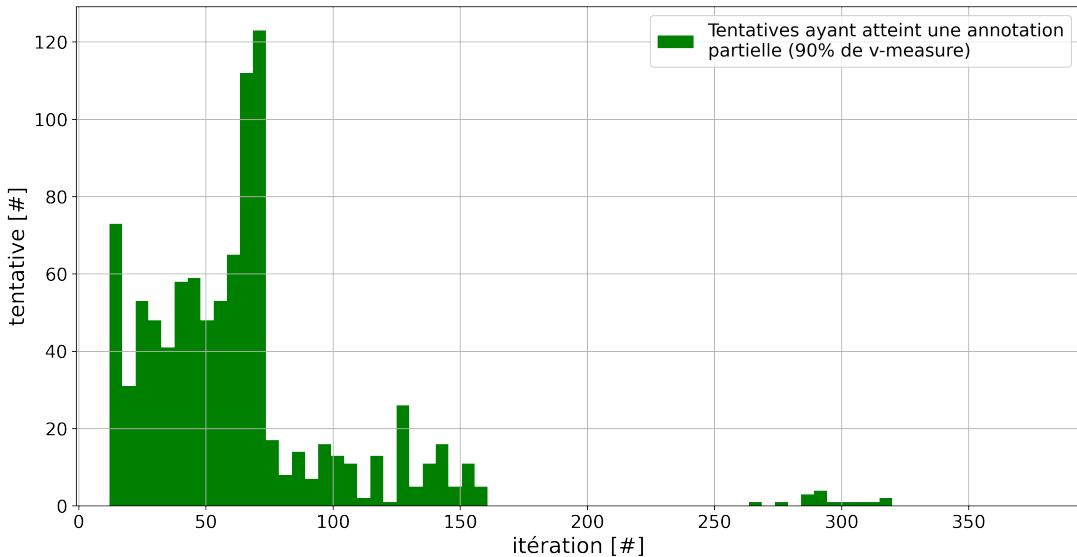


FIGURE 4.5 – Répartition des tentatives en fonction de l’itération de la méthode à laquelle elles atteignent le seuil d’une annotation partielle, c’est-à-dire l’itération à laquelle elles parviennent à 90% de *v-measure* entre un résultat obtenu et la vérité terrain. L’histogramme est réduit à 60 pics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

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.

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 la 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 suivant :

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

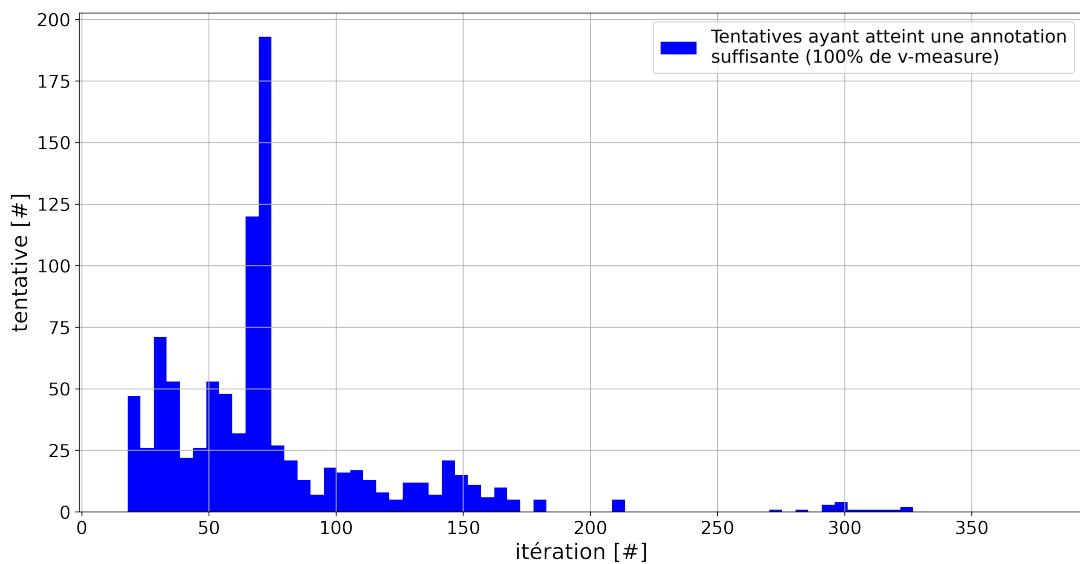


FIGURE 4.6 – Répartition des tentatives en fonction de l’itération de la méthode à laquelle elles atteignent le seuil d’une annotation suffisante, c’est-à-dire l’itération à laquelle elles parviennent à 100% de v -measure entre un résultat obtenu et la vérité terrain. L’histogramme est réduit à 60 pics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

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.

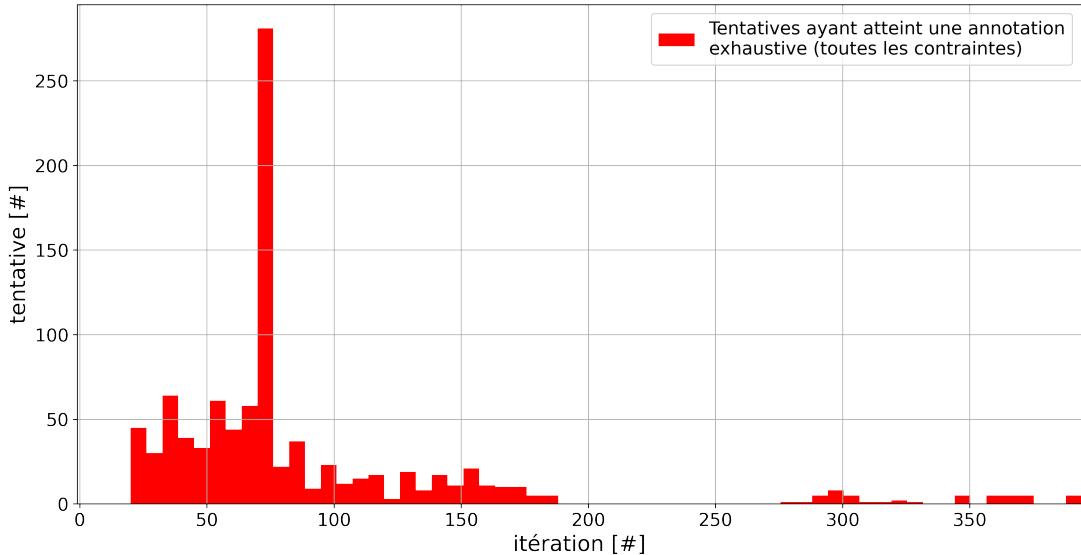


FIGURE 4.7 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation exhaustive, c'est-à-dire l'itération à laquelle toutes les contraintes possibles entre les données ont été annotées. L'histogramme est réduit à 60 pics pour simplifier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

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 :

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

4.2. Évaluation de l'hypothèse d'efficience

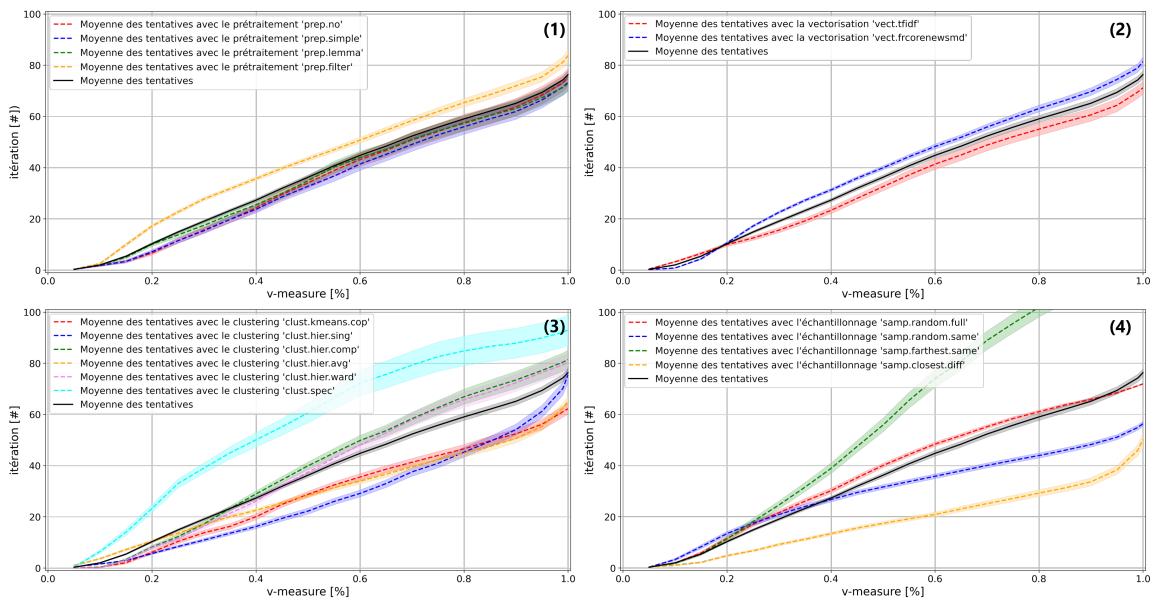


FIGURE 4.8 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de clustering interactif pour obtenir un seuil défini de *v-measure* entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes valeurs que peuvent prendre les facteurs analysés et affichées 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.

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.

Note 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 :

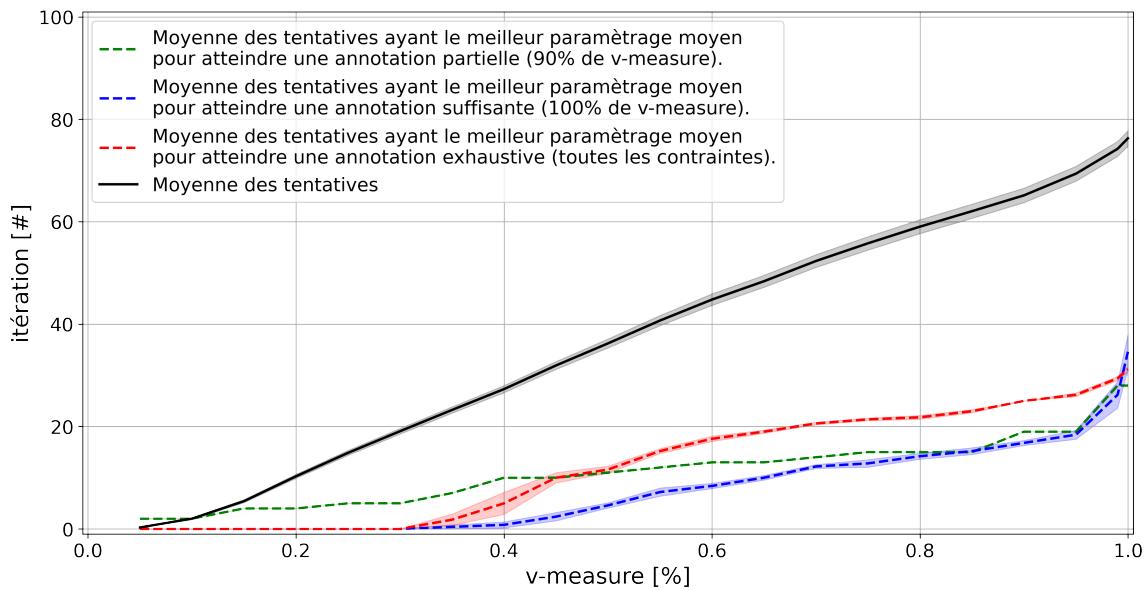


FIGURE 4.9 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de clustering interactif pour obtenir un seuil défini de *v-measure* entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes seuils d’annotations étudiés : l’annotation partielle (atteindre 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 réalisé 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 erreurs 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 étudierons 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'espérance 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 aimerions 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.

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 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 annoté (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.

A 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 4.6.3.

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 seconde²⁰ (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* » ;

20. 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 (cf.). Nous arrondissons à 1 seconde pour garder une marge d'erreur.

4.3. Évaluation de l'hypothèse sur les coûts

- **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 SECTION 3.3.

description
à faire

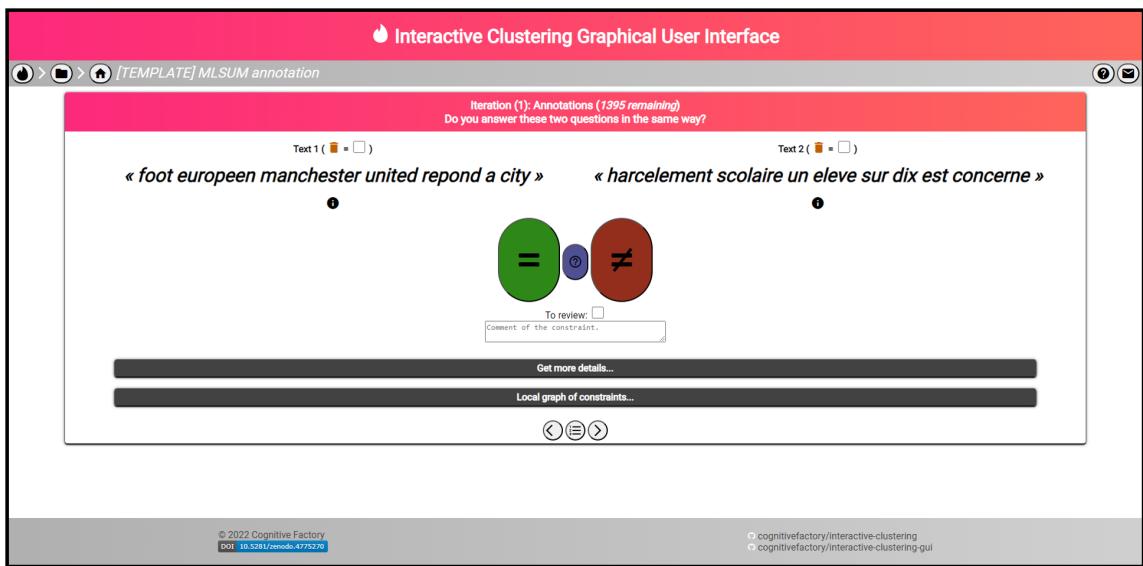


FIGURE 4.11 – Capture d'écran de l'application web permettant utilisant notre méthodologie de clustering interactif pour annoter des contraintes (page d'annotation). Les deux textes à annoter sont disposés à gauche et droite de l'écran. Chacun dispose d'un cache à cocher si le texte n'est pas pertinent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

The screenshot shows the 'Interactive Clustering Graphical User Interface' with the title '[TEMPLATE] MLSUM annotation'. At the top, there's a summary table for 'Iteration (1): Constraints synthesis':

Number of texts:	744 texts	<input type="button" value="TEXTS"/>
Number of constraints:	5 constraints and 1395 need your annotation.	<input type="button" value="ANNOTATE"/> <input type="button" value="APPROVE"/>
Modelization state:	<input checked="" type="checkbox"/> Modelization is up to date. Any modification will lead to an outdated modelization. You can approve your work if all annotations are done.	<input type="button" value="UPDATE"/>

Below this is a 'List of constraints' table:

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 »	<input checked="" type="checkbox"/> CANNI	« pour marisol touraine la liberté d installation des medecins est préservée »	1	05/06/2023, at 14:43:00.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« le bouclier fiscal c est reporter la fiscalité des plus riches vers les moins riches »	<input checked="" type="checkbox"/> CANNI	« immobilier ces voisins qui coutent cher »	1	05/06/2023, at 14:43:11.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« jeu de piste entre picasso et godard »	<input checked="" type="checkbox"/> MUST	« le centre pompidou expose wifredo lam pour la premiere fois »	1	05/06/2023, at 14:43:17.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« harclement scolaire un eleve sur dix est concerné »	<input checked="" type="checkbox"/> CANNI	« a istanbul l art a un air de defi »	1	05/06/2023, at 14:43:21.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« football luis fernandez est le nouveau sélectionneur d israel »	<input checked="" type="checkbox"/> CANNI	« logements la correction du marche neuf devrait se poursuivre »	1	05/06/2023, at 14:43:25.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« deces d egan bahr figure du spd et artisan de l ostpolitik »	<input type="checkbox"/> SKIP	« la droite s interroge sur la strategie de nicolas sarkozy »	1	Never	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« chomage les demandeurs d emploi globalement satisfaits du pole emploi »	<input type="checkbox"/> SKIP	« d autres virus h7 potentiellement dangereux pour l homme a l instar de la grippe aviaire »	1	Never	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>
« bygnallion nicolas sarkozy directement vise »	<input type="checkbox"/> SKIP	« logement la crise sans fin »	1	Never	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value=""/>

FIGURE 4.12 – Capture d'écran de l'application web permettant utilisant notre méthodologie de clustering interactif pour annoter des contraintes (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 et al., 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` et `statsmodels` (SEABOLD et PERKTOLD, 2010).

Résultats obtenus

Durant cette expérience, 14 annotateurs ont participé à l'annotation de 1 000 contraintes aléatoires sur un jeu de données. Par manque de disponibilités, 4 annotateurs n'ont que partiellement réalisé leur tâche : nous avons toutefois intégré leurs participations car elles contenaient toutes au moins 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

4.3. Évaluation de l'hypothèse sur les coûts

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, $11f$: -499.15, $11f_{null}$: -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.

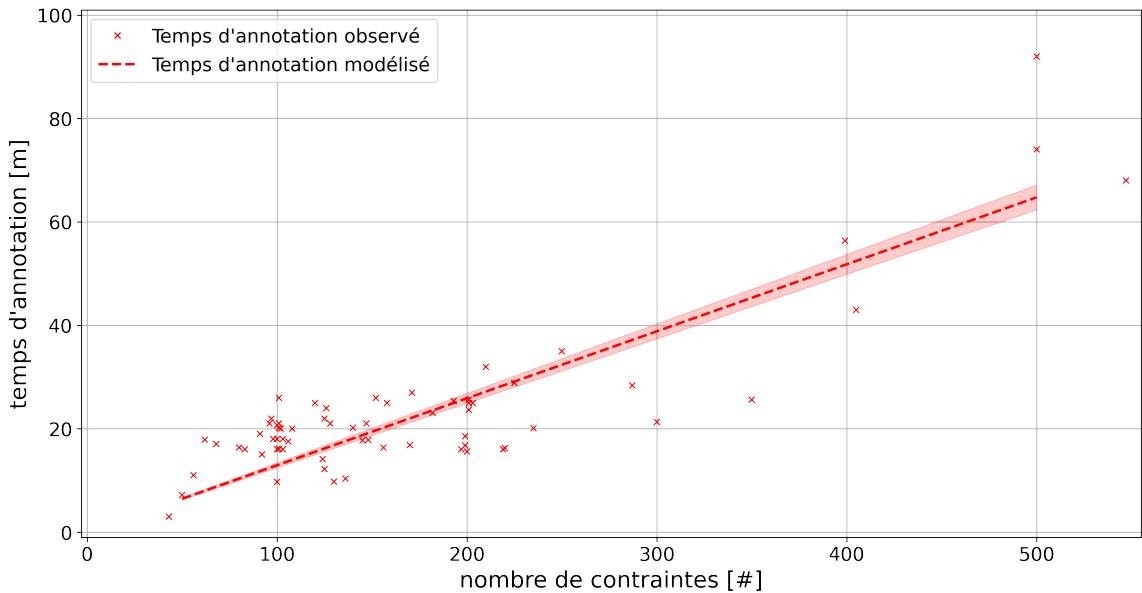


FIGURE 4.13 – Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.

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.

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).

Note 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*).

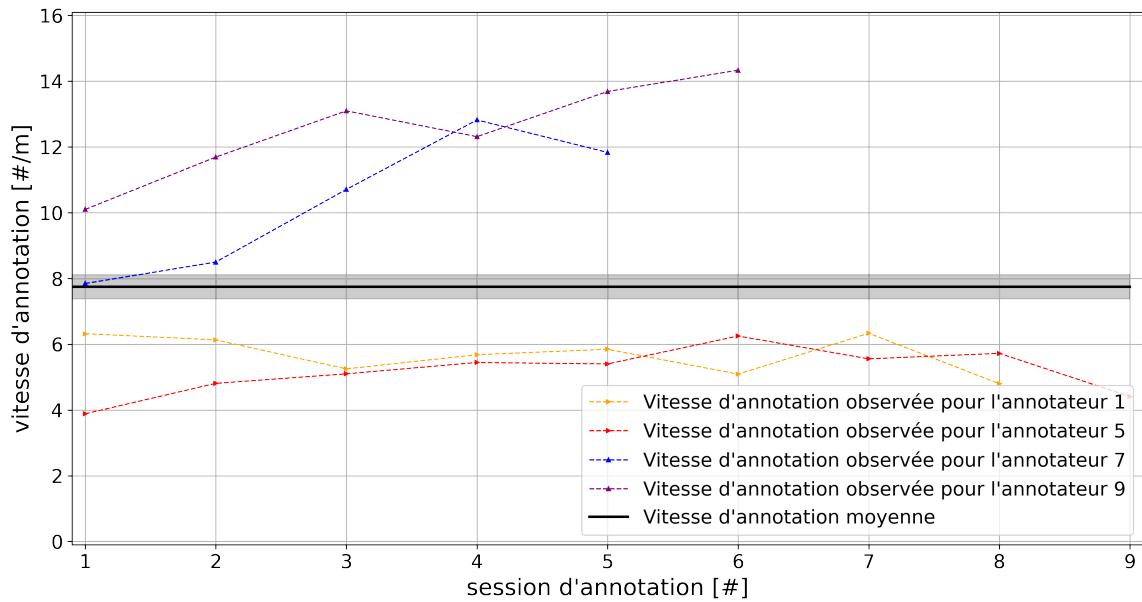


FIGURE 4.14 – Etude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations.

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 déléguent quelques tâches d'annotation à Amazon Mechanical Turk :

i Pour information : Amazon Mechanical Turk est une plateforme de travail collaboratif (*crowdsourcing*) sur laquelle n'importe quel internaute peut contribuer à une tâche plus ou moins complexe en échange d'une rémunération. Certaines entreprises utilisent ce service pour réaliser de l'analyse de données, comme l'étiquetage de données, la traduction de textes, la rédaction d'avis et de critiques, la modération de contenu, ... Cette approche comporte quelques inconvénients : d'une part, les travailleurs ne sont généralement pas experts du domaine traité (ce sont n'importe quel internaute ayant du temps à offrir) ; d'autre part, le travail est parfois réalisé à la hâte à cause de la faible taux de rémunération des opérateurs (environ de 2 \$ l'heure). Afin garantir la qualité des réalisations, il est donc commun de demander plusieurs exécution d'une même tâche par différents opérateurs et de récupérer le résultat faisant le plus grand consensus.

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 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.

Note 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 les 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. A cet effet, une option e tri a 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.

❶ Pour information : Ces différentes évolutions sont en cours d'analyse ou ont déjà été intégrées dans l'application que nous proposons (SCHILD et al., 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 employés 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.

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. SECTION 3.3) avec différentes variations de contexte d'utilisation. Cela comprend les tâches, algorithmes et contextes d'utilisation suivants :

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

Données : jeux de données annotés (vérité terrain) de tailles différentes

Entrées : arrangements d'algorithmes et de paramètres à tester

```

1 pour chaque arrangement 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 : caractériser les contraintes grâce à 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 : caractériser les contraintes grâce à 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.

- 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 al-

gorithme 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 (« Pearson's Correlation Coefficient », 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` et `statsmodels` (SEABOLD et PERKTOLD, 2010).

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 :

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

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 :

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

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 :

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

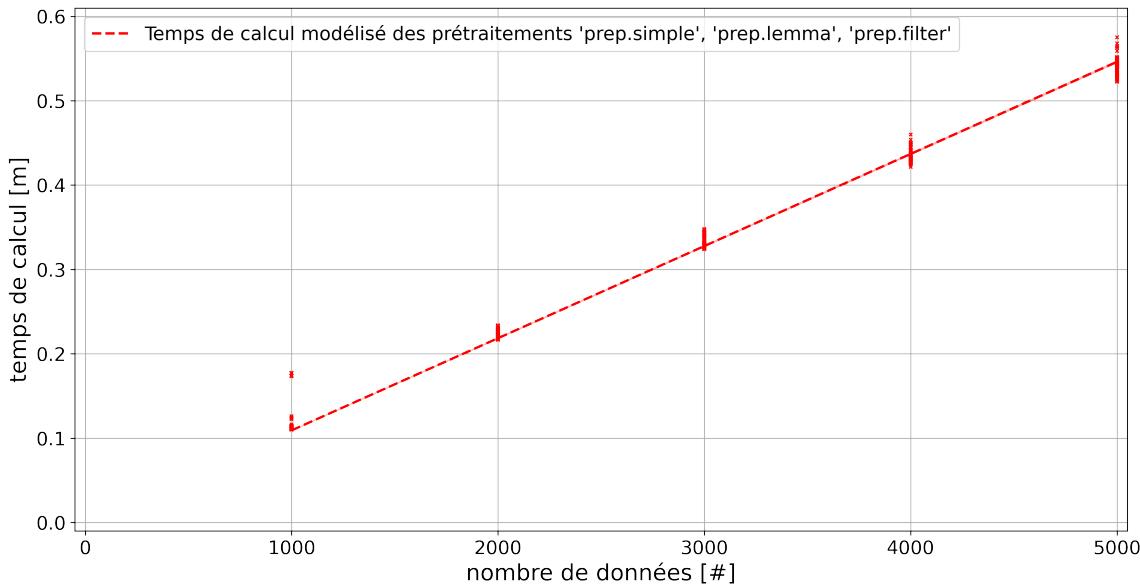


FIGURE 4.15 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de prétraitement en fonction du nombre de données à traiter. Les paramétrages `prep.simple`, `prep.lemma` et `prep.filter` ayant des temps de calculs similaires, leurs modélisations n'ont pas été séparées.

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.

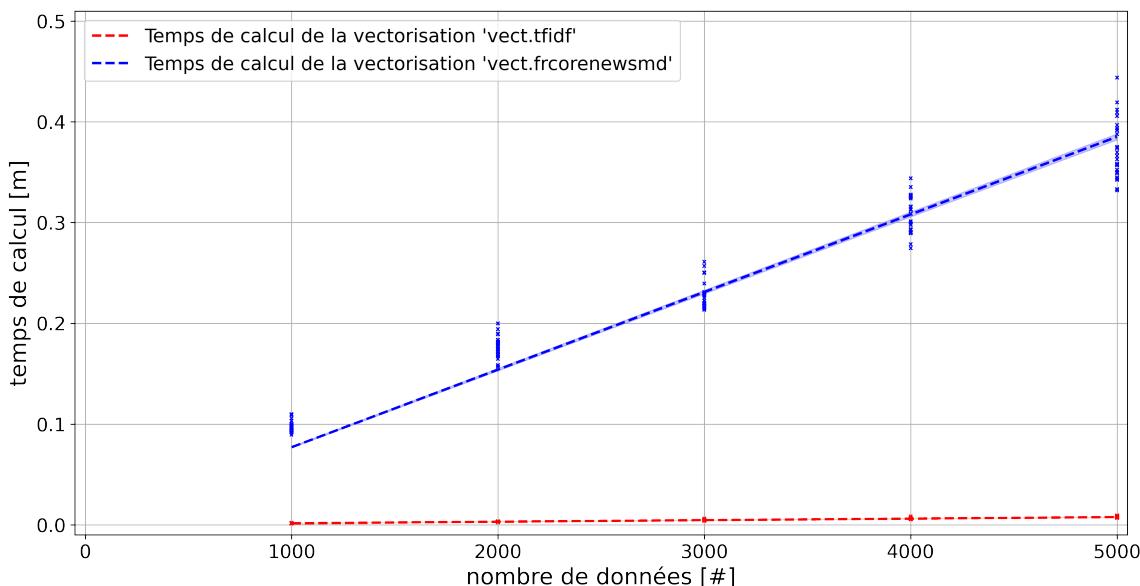


FIGURE 4.16 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de vectorisation en fonction du nombre de données à traiter.

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 excéde 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` ($r : 0.545$). Le modèle linéaire généralisé retenu ($R^2 : 0.802$, $llf : -9.37 \cdot 10^4$, $llf_{null} : -1.00 \cdot 10^5$) 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 ($R^2 : 0.987$, $llf : -5.54 \cdot 10^4$, $llf_{null} : -6.10 \cdot 10^4$) 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 ($R^2 : 0.984$, $llf : -5.56 \cdot 10^4$, $llf_{null} : -6.11 \cdot 10^4$) 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 ($R^2 : 0.981$, $llf : -5.90 \cdot 10^4$, $llf_{null} : -6.45 \cdot 10^4$) nous permet de déduire l'équation suivante :

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

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.

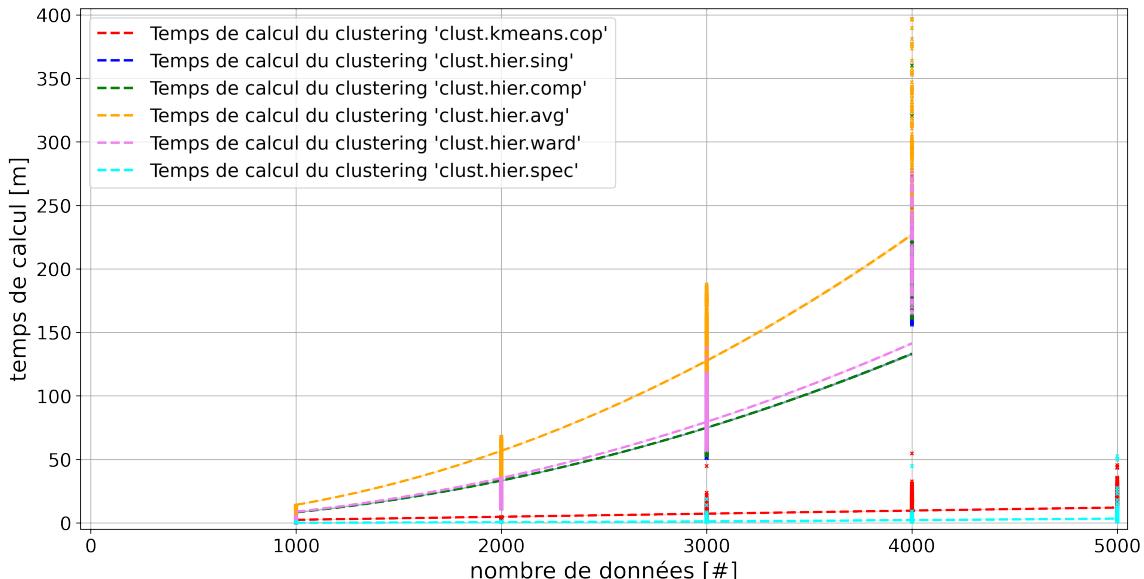


FIGURE 4.17 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de *clustering* en fonction du nombre de données à traiter.

En ce qui concerne la tâche d'**échantillonnage de contraintes**, une première analyse montre que les modélisations des quatre implémentations sont différenciables ($p\text{-valeur} : < 10^{-3}$). 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.

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.

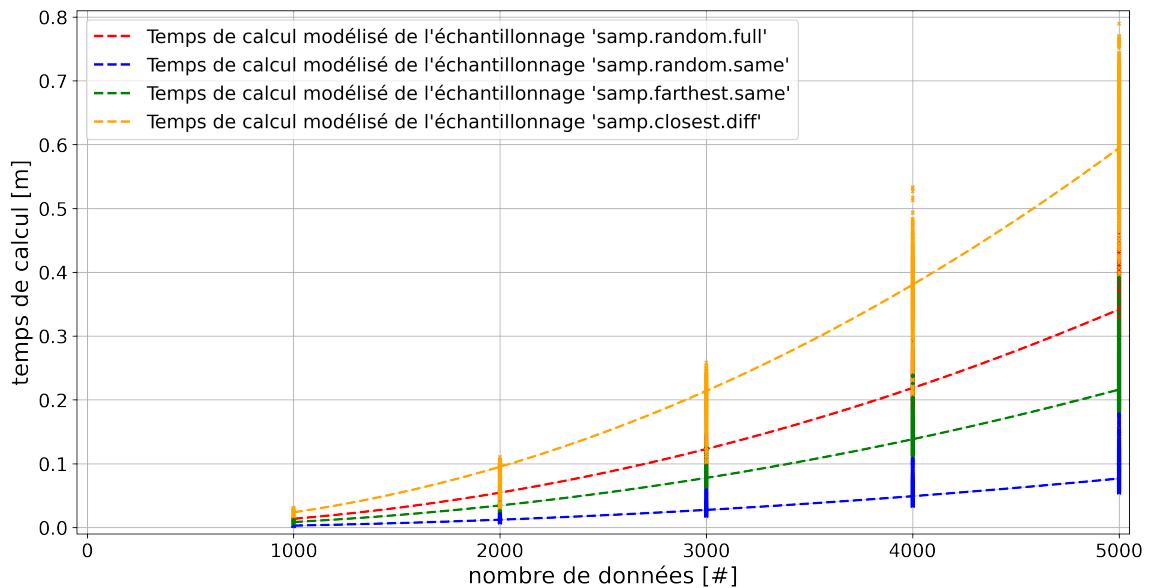


FIGURE 4.18 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche d'échantillonage de contraintes en fonction du nombre de données à traiter.

Note 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. SECTION 3.3.2). 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 résonnement 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*²¹ 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 suivante²² :

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

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**²³ sur des jeux de données de différentes tailles et mesurons le nombre de contraintes nécessaires pour converger vers la vérité terrain.

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 annotés (vérité terrain) 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 `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* obtenu et la vérité terrain ;

8 **répéter**

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

10 **simulation d'annotation** : caractériser les contraintes grâce à la vérité terrain ;

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

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

13 **évaluation** : estimer l'équivalence entre le *clustering* obtenu 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 :

- 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 ;

23. 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 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, noté `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 complète²⁴ 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 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 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` (SEABOLD et PERKTOLD, 2010).

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 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.

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

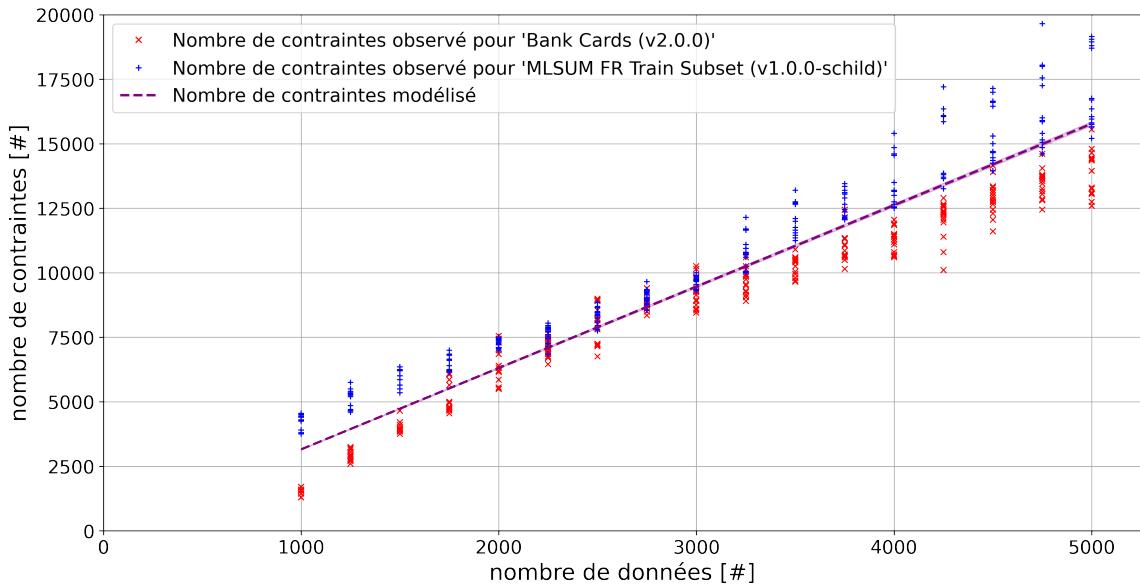


FIGURE 4.19 – Estimation du nombre moyen de contraintes nécessaire à notre paramétrage favori du clustering interactif afin d’obtenir une annotation partielle (atteindre une v -measure de 90%) en fonction de la taille du jeu de données à modéliser.

-note 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 minimux 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.

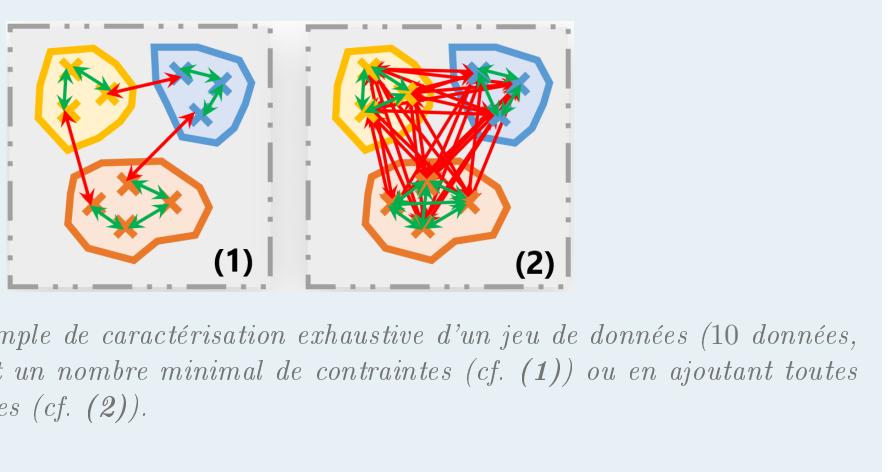


FIGURE 4.20 – Exemple de caractérisation exhaustive d'un jeu de données (10 données, 3 classes) en ajoutant un nombre minimal de contraintes (cf. (1)) ou en ajoutant toutes les contraintes possibles (cf. (2)).

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.

Note 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**²⁵. 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.

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** :

$$\left\{ \begin{array}{l} \text{total_time(sequential) [s]} \propto \text{iteration_time} \cdot \text{iterations_needed} \end{array} \right. \quad (4.19)$$

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.

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 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

25. Paramétrage favori (atteindre 90% de **v-measure** avec un coût minimal).

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 (`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 échantillonnage 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 travail à un expert métier pour obtenir une base d'apprentissage avec une méthodologie 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.

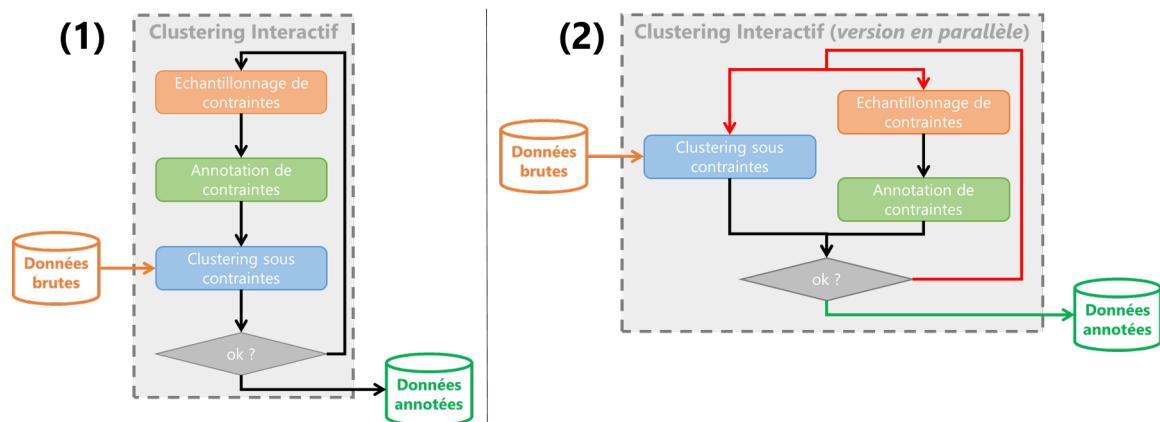


FIGURE 4.21 – Schéma comparatif des architectures du clustering interactif : (1) représente la version séquentielle initialement présentée en CHAPITRE 3 où le clustering s'adapte avec les annotations de l'itération en cours ; (2) représente l'évolution en mode parallèle où le clustering s'adapte avec 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 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 rendre les deux durées équivalentes. Nous déduisons donc les changements suivants dans l'EQUATION 4.18 :

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

4.3. Évaluation de l'hypothèse sur les coûts

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.

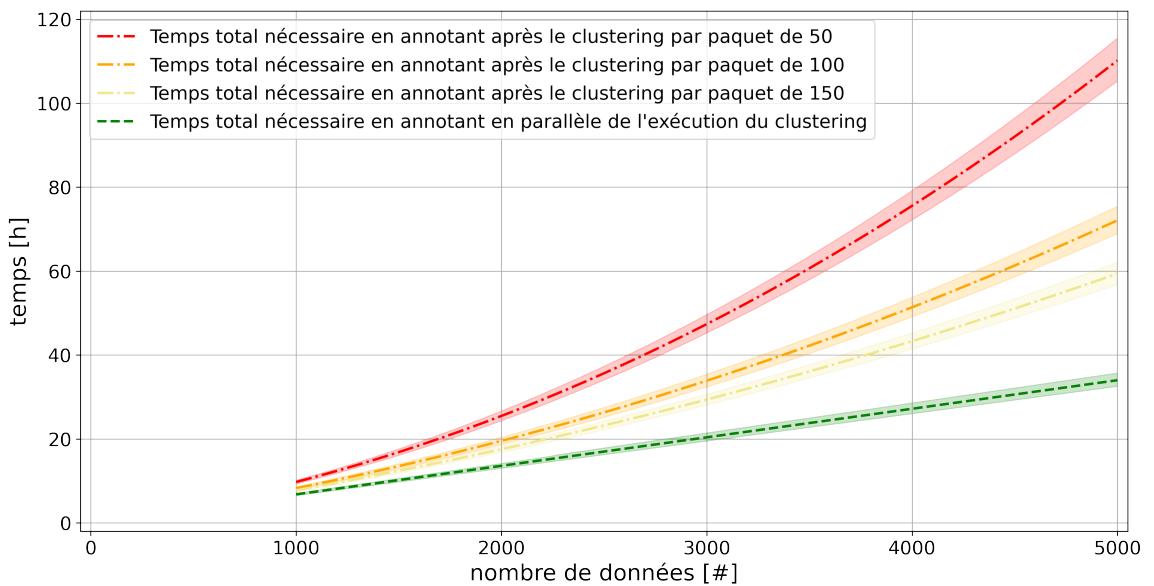


FIGURE 4.22 – Estimation du temps total nécessaire (en heures) pour modéliser un jeu de données avec notre paramétrage favori du clustering interactif afin d'obtenir une annotation partielle (atteindre une *v-measure* de 90%), en fonction de plusieurs taille de jeu de données, plusieurs tailles de lots 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*²⁶, cela représente 4.8 (+2) jours de travail, une 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.

26. voir. ALGORITHME 3.1, ligne 13.

 **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).

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 :

❶ Hypothèse de pertinence ❶

« 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.

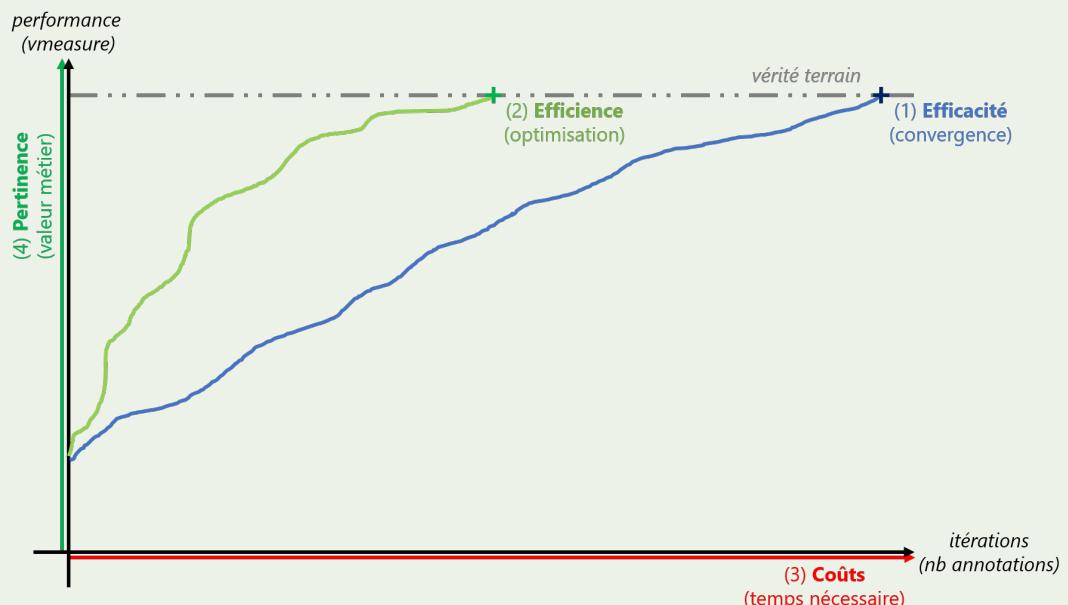


FIGURE 4.23 – Illustration des études réalisées sur le *clustering* interactif (étape 4/6) en schématisant l'évolution de la pertinence (valeur métier évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode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

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 annotés (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 : caractériser les contraintes grâce à 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 complète²⁷ de la méthode du *clustering* interactif en utilisant notre paramétrage favori (voir SECTION 4.4.3), 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. 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 placé 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.

²⁷. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

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).

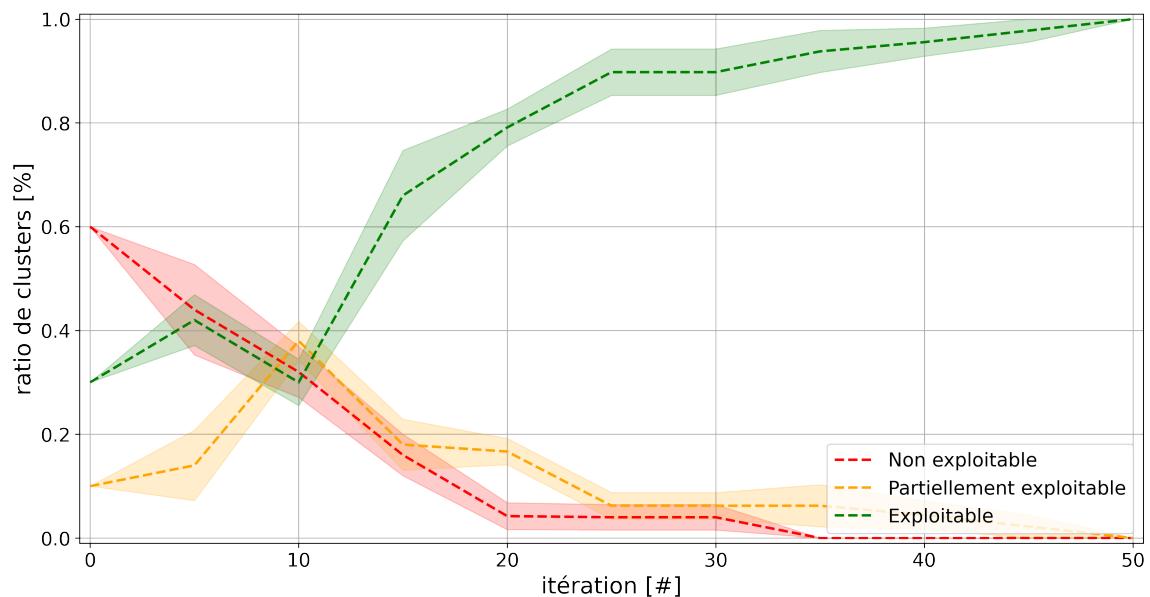


FIGURE 4.24 – Évolution de la pertinence métier moyenne estimée manuellement au cours des itérations du résultat du clustering interactif avec notre paramétrage favori. Cette pertinence, exprimée en proportion du nombre de clusters, est retranscrite en trois niveaux : *exploitable* en vert, *partiellement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

Discussion

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

Comme l'analyse n'a pu être faite que par un seul opérateur, nous ne nous attarderons 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.

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, 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, 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 ?*)

Données : jeu de données annotés (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   analyse linguistique : caractériser les clusters grâce à la FMC ;
8   évaluation assistée : juger de l'exploitabilité de chaque cluster ;
9   labellisation assistée : nommer chaque cluster exploitable ;
10  répéter
11    échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
12    simulation d'annotation : caractériser les contraintes grâce à la vérité terrain ;
13    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
14    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
15    analyse linguistique : caractériser les clusters grâce à la FMC ;
16    évaluation assistée : juger de l'exploitabilité de chaque cluster ;
17    labellisation assistée : nommer chaque cluster exploitable ;
18  jusqu'à annotation de toutes les contraintes possibles ;
19 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.

💡 Idée : 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.

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'implémentation de la maximisation des traits est accessible ici dans SCHILD, 2023.

Résultats obtenus

⚠️ Attention : Par manque de moyen, la relecture 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 linguistique 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. A 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 même thématique, et les questions semblent toutes traiter de sujets différents. A 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 présents, mais ce *cluster* à l'itération 10 peut donc être associé aux classes *gestion_carte_virtuelle* et *debloqueage_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.

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 représentatifs et discriminants de chaque groupe de questions. Nous discutons ici de l'intérêt

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 : 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.

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_soldé* avec "solde" et "compte"; *gestion_carte_virtuelle* avec "carte virtuelle" et "numero"), mais certains termes quelconque sont pourtant sont présents 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

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 aimeraai 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	- virtuelle - carte virtuelle - un numero - numero - de carte virtuelle - un numero de - numero de carte - numero de - numeros (Total : 34)	- 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 aimeraai utiliser une carte virtuelle - ou peut on gerer ses numeros virtuels - supprimer un numero de carte virtuel (Total : 49)
Tentative : 1 Itération : 15 Cluster : 4 Avis initial : Exploitable	- reactiver - debloquer - debloquer ma - debloquer ma carte - bloquee - reactiver sa - reactiver ma - deverrouiller - reactiver sa carte (Total : 24)	- - - - - - 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_carte_virtuelle* et *deblocage_carte*, 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.

"*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 nuances 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*.

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, ...

Note 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 métier 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 adapter²⁸ et étudier ici.

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 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, 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 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 **OpenAI**²⁹. 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.* » ;

28. 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**.

29. 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)

Données : jeu de données annotés (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   synthèse automatique : résumer les thématiques des clusters par un LLM ;
8   évaluation assistée : juger de l'exploitabilité de chaque cluster ;
9   labellisation assistée : nommer chaque cluster exploitable ;
10  répéter
11    échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
12    simulation d'annotation : caractériser les contraintes grâce à la vérité terrain ;
13    intégration : ajouter les nouvelles contraintes au gestionnaire de contraintes ;
14    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
15    synthèse automatique : résumer les thématiques des clusters par un LLM ;
16    évaluation assistée : juger de l'exploitabilité de chaque cluster ;
17    labellisation assistée : nommer chaque cluster exploitable ;
18  jusqu'à annotation de toutes les contraintes possibles ;
19 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 4.8 – 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.

- 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 ?*)

💡 Idée : 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.

⚠️ 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. 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 placé pour juger la pertinence d'un *clustering* par rapport à sa propre modélisation du problème. Les problèmes d'accords inter-annotateurs seront plutôt discutés en SECTION 4.6. Nous réalisons toutefois 5 tentatives différentes de la méthode pour limiter les biais intra-individuels.

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. L'appel à un large modèle de langage, sous la forme d'un appel API, se fait grâce à la librairie openai.

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.

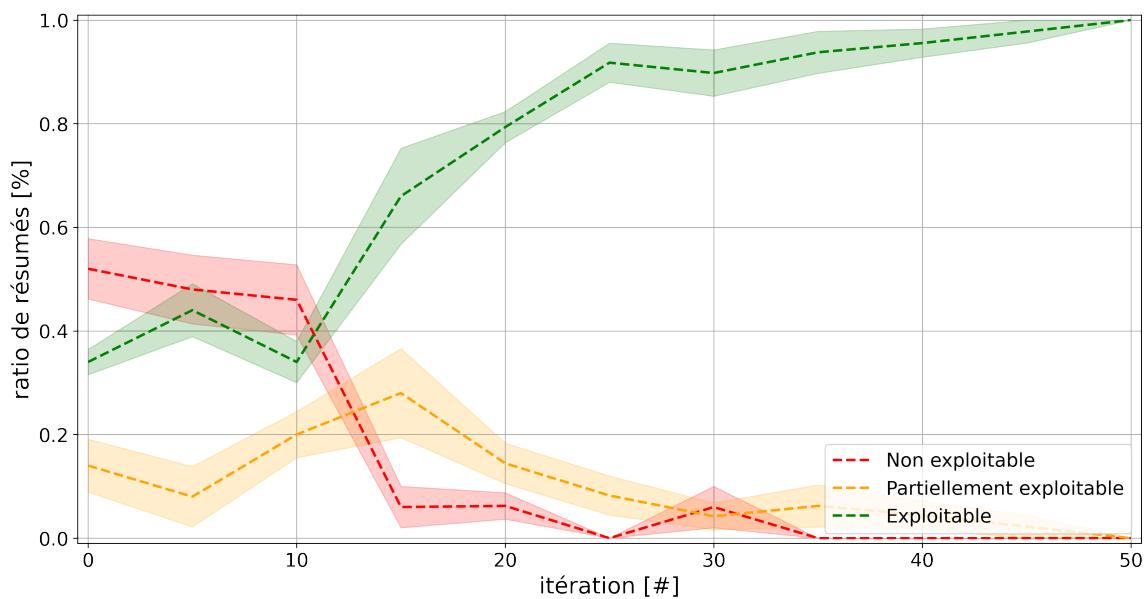


FIGURE 4.25 – Évolution de la pertinence métier moyenne en fonction du nombre d'itérations de la méthode. Cette pertinence, exprimée en proportion du nombre de *clusters*, est estimée sur la base du résumé automatique des *clusters* par un large modèle de langage et est retranscrite en trois niveaux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

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.

commentaire
Gau-
tier :
décrire
l'opé-
rateur,
son ex-
pertise,
...

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.

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.

- 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).

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,

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	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.
Tentative : 1 Itération : 15 Cluster : 4 Avis initial : Exploitable	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 *deblocage_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.

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 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 centré 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 approche 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 si 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 peuvent 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))³⁰.

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 drastique-

30. A 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.

ment 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 étudierons cet aspect dans la prochaine section (cf. hypothèse de rentabilité en SECTION 4.5).

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.

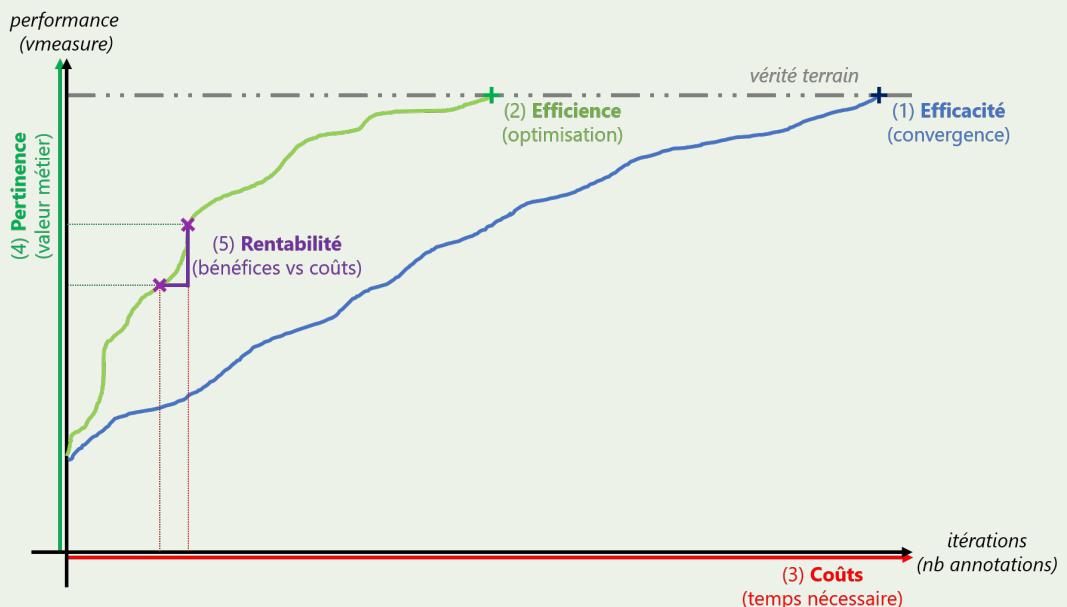


FIGURE 4.26 – Illustration des études réalisées sur le clustering interactif (étape 5/6) en schématisant l'évolution de la pertinence (valeur métier évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps 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.

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 complète³¹ de la méthode du *clustering* interactif en utilisant notre paramétrage favori, 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)) ;
- 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)).

³¹. Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

Utiliser 'foot-misc' et 'footref' pour faire des notes de bas de pages communes ? ou lien vers des conclusions ?

Données : jeu de données annotés (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 : caractériser les contraintes grâce à 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.

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 (« Pearson's Correlation Coefficient », 2008) est utilisée).

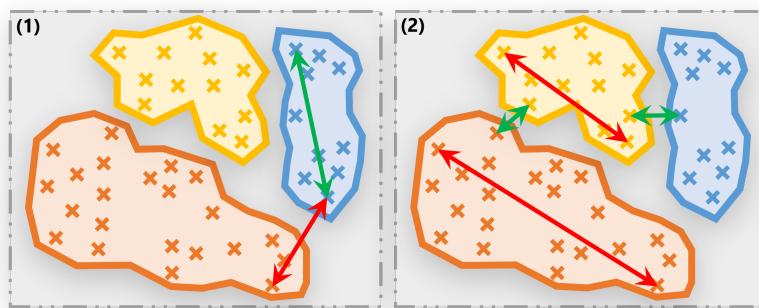


FIGURE 4.27 – Exemples d'accords et de désaccord entre les annotations d'une itération et le résultat du clustering de l'itération précédente. Des contraintes *MUST-LINK* (flèches vertes) et *CANNOT-LINK* (flèches rouges) sont représentées dans deux situations : (1) montre des cas d'accords (*MUST-LINK* 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).

Idée : 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).

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.

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.

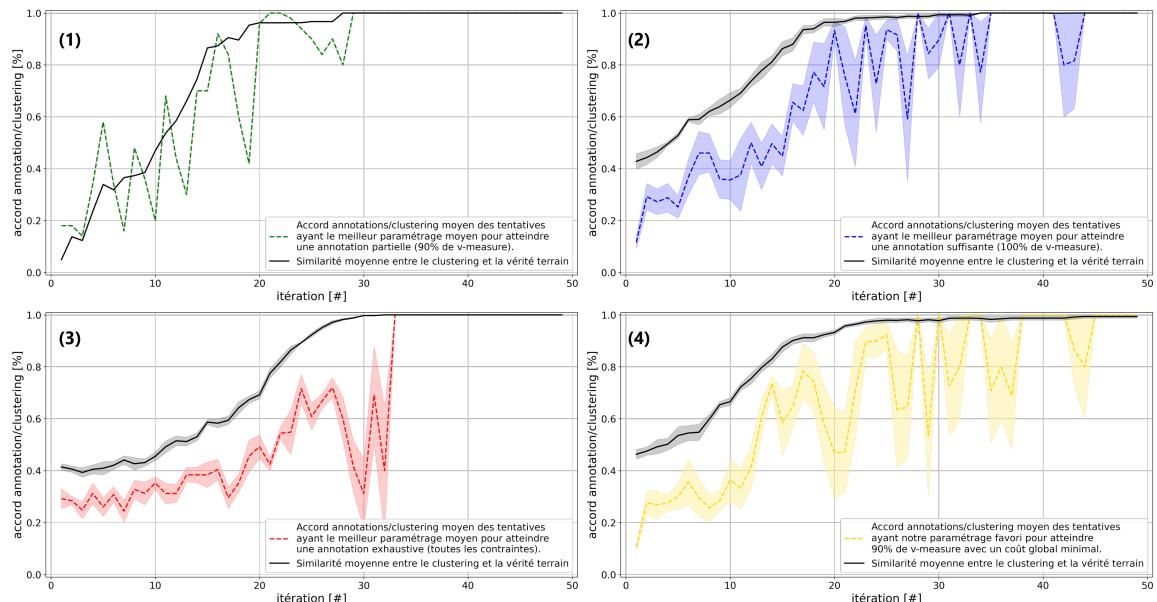


FIGURE 4.28 – Évolution au cours des itérations de l'accord entre l'annotation de contraintes d'un expert et le résultat de clustering sur lequel est basé l'échantillonnage de contraintes. Ces accords sont exprimés grâce à des lots de 50 contraintes annotées. Les évolutions moyennes de différents paramétrages 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, ...).

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.

Note 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*.

Note 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, si une analyse de l'annotation réalisée n'est pas fructueuse, nous nous tournerons vers l'analyse plus abstraite des 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*.

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 annotés (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 : caractériser les contraintes grâce à 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, 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* (« Pearson's Correlation Coefficient », 2008) est utilisée).

💡 Idée : 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, notamment

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.

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 enfin en oscillant très légèrement ($\pm 1\%$) autour de 0% jusqu'à la fin des annotations.

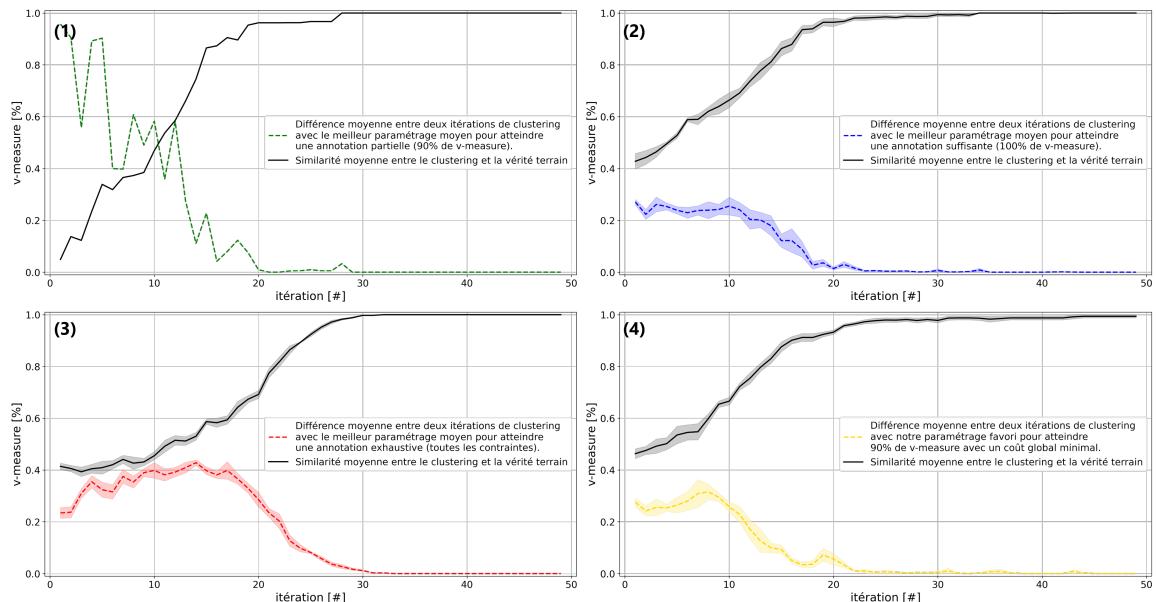


FIGURE 4.29 – Évolution de la différence de résultats entre deux itérations de clustering. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation 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, ...).

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 a 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'en 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.

💡 Idée : 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. (cf. SECTION 4.5.2) et SECTION 4.4.4).

Pour terminer nos différentes analyses, il convient maintenant d'anticiper la présence d'erreurs d'annotation. En effet, nous avons fait jusqu'à présent l'hypothèse que l'annotateur ne se trompe jamais, mais cette hypothèse forte n'est pas toujours vérifiée en pratique. Pour estimer l'impact de ces erreurs ou incohérences d'annotation, nous devons donc réaliser une analyse de robustesse de notre méthode d'annotation : celle-ci sera réalisé 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 intéresserons 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 performances de la méthode. »

La FIGURE 4.30 illustre cette hypothèse et l'espoir de estimer l'impact différences ou d'erreurs d'annotations sur le nombre d'itérations de la méthode.

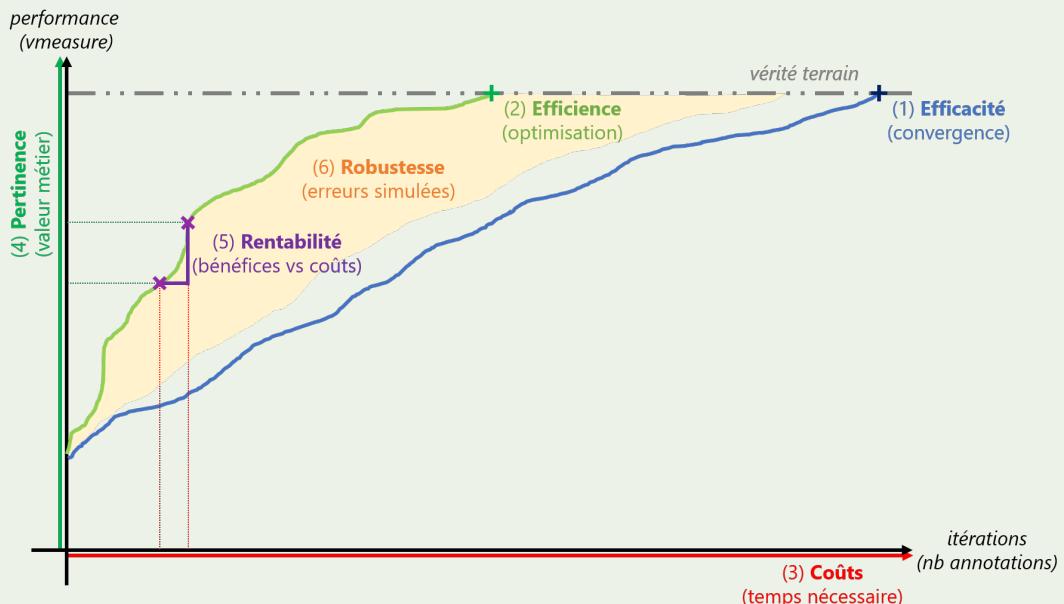


FIGURE 4.30 – Illustration des études réalisées sur le clustering interactif (étape 6/6) en schématisant l'évolution de la pertinence (valeur métier évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (temps 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 sur la **correction des incohérences d'annotation** (cf. SECTION 4.6.1) ;
 - une simulation de l'**impact des incohérences d'annotation** sur les performances de la méthode (cf. SECTION 4.6.2) ;
 - une étude de cas sur le **score inter-annotateurs** obtenu lors d'une annotation de contraintes en situation réelle avec plusieurs opérateurs (cf. SECTION 4.6.3).

4.6.1 Étude de l'intérêt de la correction des incohérences d'annotation

Nous cherchons à estimer la robustesse du *clustering* interactif face aux incohérences d'annotation. Dans cette étude, nous nous intéressons plus particulièrement à l'intérêt de la détection et de la correction des conflits présents dans les contraintes. En effet, deux approches de travail s'opposent :

- une approche naïve ignorant simplement les conflits : celle-ci n'engendre pas de coûts supplémentaires, mais elle s'expose aux risques de dérives de fonctionnement ;
- une seconde approche avec correction des conflits : celle-ci nécessite de revoir ou de ré-annoter des contraintes, impliquant un coût supplémentaire, mais permet de limiter l'impact de dérives potentielles.

Pour trancher entre ces deux options, nous simulons ces deux approches afin d'estimer si l'absence de correction induit une régression significative des performances de notre méthode d'annotation.

Protocole expérimental

Pour résumer le protocole expérimental que nous décrivons ci-dessous, vous pouvez vous référer au pseudo-code décrit dans ALGORITHME 4.11.

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 complète³² de la méthode du *clustering* interactif en utilisant notre paramétrage favori (voir SECTION 4.4.3). Toutefois, contrairement aux précédents expériences, nous allons ajouter un pourcentage de contraintes erronées à chaque itération :

- Le taux d'erreurs insérées, variant de 0% à 50% 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 performances au courant des itérations ;
- Les contraintes erronées à insérer sont tirées aléatoirement parmi le lot de contraintes qui aurait été échantillonné au cours d'une tentative sans erreur : 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 vont apparaître dans le gestionnaire de contraintes. Pour rappel, un conflit est détecté dans le cas où l'ajout d'une nouvelle

³². Tentative complète : itérations d'échantillonnage, d'annotation et de *clustering* jusqu'à annotation de toutes les contraintes possibles.

Données : jeu de données annoté (vérité terrain)

Entrées : taux d'erreurs à tester

```
1 pour chaque arrangement de taux d'erreur 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 : estimer l'équivalence entre le clustering obtenu et la vérité terrain ;
8   répéter
9     échantillonnage : sélectionner des contraintes avec samp.closest.diff ;
10    échantillonnage d'erreurs : définir les contraintes qui seront erronées ;
11    simulation d'annotation : caractériser les contraintes grâce à la vérité terrain ;
12    si absence de correction des conflits d'annotation alors
13      intégration naïve : ignorer les conflits avec le gestionnaire de contraintes ;
14    sinon si détection et correction des conflits d'annotation alors
15      intégration corrective : changer les annotations erronées en conflit ;
16    clustering : regrouper les données par similarité avec clust.kmeans.cop ;
17    évaluation : estimer l'équivalence entre le clustering obtenu et la vérité terrain ;
18  jusqu'à annotation de toutes les contraintes possibles;
19 analyse : déterminer l'impact par itération et par taux d'erreurs de la correction ;
Résultat : discussion sur l'impact de la correction des incohérences
```

ALGORITHME 4.11 – Description en pseudo-code du protocole expérimental de l'étude d'intérêt de la correction des incohérences d'annotation.

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 3.2 en SECTION 3.3.2). Pour les traiter, nous testons deux approches :

- l'approche naïve ignorant simplement 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 ;
- l'approche avec correction des conflits : pour simuler la correction d'un expert, nous recréons 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 11 taux d'erreurs différents à simuler, chacun suivant 2 approches de gestion de conflits différentes, et chacune de ces simulations d'erreurs seront répétées 10 fois sur chaque tentative complète de la méthode pour limiter contrer les aléas statistiques des tirages de contraintes erronées, ce qui représente 220 simulation 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 1100 tentatives complètes à simuler.

Enfin, nous afficherons l'évolution de la performance moyenne du *clustering* obtenu en fonction des divers taux d'erreurs simulés, et discuterons de l'impact au cours des itérations de la présence ou de l'absence de corrections des conflits d'annotations détectés.

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.

Résultats obtenus

La FIGURE 4.31 représente l'évolution moyenne de la `v-measure` du *clustering* en fonction du nombre d'itération de la méthode, déclinée avec les 11 taux d'erreurs simulés et les 2 approches de gestion des conflits. Les contraintes utilisées sont basées sur les échantillonnages réalisées au cours des tentatives sans erreurs : 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). Ainsi, toutes les courbes simulant les différents taux d'erreurs sont tronquées à 3 000 contraintes, que les tentatives aient convergé ou non. Nous serons sensible à cette information pour ne pas faire de mauvaises interprétations, car le dernier point des différentes courbes ne représente pas forcément le point de convergence des tentatives associées.

A REDIGER : description de la figure

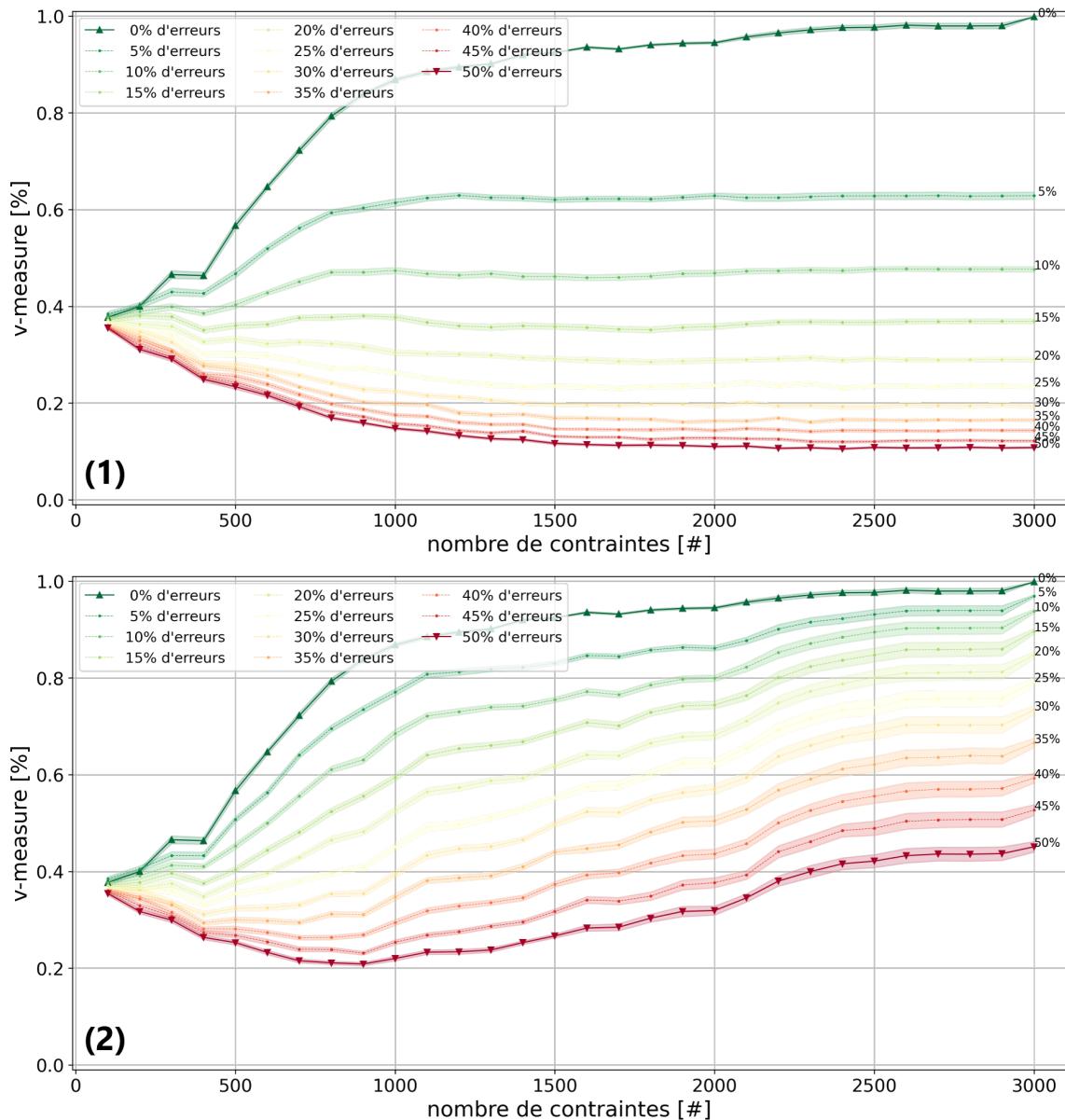


FIGURE 4.31 – Évolutions de la moyenne de la *v-measure* entre un résultat obtenu et la vérité terrain en fonction du nombre de contraintes annotées au cours des itérations du clustering interactif. Les courbes en dégradé de couleur représentent les déclinaisons de cette évolution en intégrant un pourcentage d'annotations erronées (allant de 0% et 50%). (1) et (2) représente respectivement l'approche ignorant les conflits dans les contraintes et l'approche corrigeant les conflits détectés par le gestionnaire de contraintes. Toutes les courbes sont tronquées à 3 000 contraintes.

Discussion

A REDIGER : rappel de l'objectif

A REDIGER : Super important de corriger !

A REDIGER : Besoin de mécanisme pour prédire et mettre de la redondance

A REDIGER : ouverture sur l'impact des incohérences

4.6.2 Étude de l'impact des incohérences d'annotation sur les performances

A REDIGER : objectif de l'expérience

Protocole expérimental

A REDIGER

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.

Résultats obtenus

A REDIGER

A REDIGER :

La TABLE 4.11

TABLE 4.11 – *Simulation ...*

Discussion

A REDIGER : rappel de l'objectif

A REDIGER : prédition de retard

FIN

4.6.3 Étude du score inter-annotateurs obtenu avec des opérateurs en situation réelle

Nous voulons étudier le score d'accord inter-annotateurs calculé lors d'une annotation de contraintes par plusieurs expert 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 adaptons son protocole expérimental pour estimer l'accord inter-annotateurs.

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 ??.

Données : jeu de données annoté (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 4.12 – 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.

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 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.

A partir de ces données, nous sélectionnons un lot de 400 contraintes à annoter. Pour faciliter l'analyse, l'échantillonnage sera un aléatoire équilibré d'après la vérité terrain en 200 **MUST-LINK** et en 200 **CANNOT-LINK**.

Ensuite, un groupe de 3 annotateurs vont annoter la sélection de 400 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* » ;
- **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 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 SECTION 3.3.

A REDIGER / A COMPLETER

description
à faire

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.

Résultats obtenus

La TABLE 4.12 expose les scores inter-annotateurs sur les 3 opérateurs et le réviseur de cette expérience, ainsi que l'accord avec la vérité terrain. Le score d'accord moyen avec la vérité terrain est de 0.86 (écart-type : 0.01) ; ce score est de 0.81 (écart-type : 0.05) pour les contraintes de types MUST-LINK et est de 0.92 (écart-type : 0.03) pour les contraintes de types CANNOT-LINK. Le score inter-annotateurs moyen (sans le réviseur) est de 0.84 (écart-type : 0.03).

	1 (Relecteur)	7 (Annotateur)	9 (Annotateur)	12 (Annotateur)
Vérité terrain	0.95	0.86	0.84	0.87
1 (Relecteur)		0.91	0.86	0.89
7 (Annotateur)			0.86	0.85
9 (Annotateur)				0.80

TABLE 4.12 – Score d'accord inter-annotateurs obtenu avec 1 réviseur et 3 annotateurs sur un lot commun de 400 contraintes (200 MUST-LINK, 200 CANNOT-LINK).

i Pour information : Dans une autre expérience, où 14 opérateurs devaient annoter une base de 1 000 contraintes aléatoires, nous obtenons un accord moyen avec la vérité terrain de 0.93 (écart-type : 0.02) et un score inter-annotateurs moyen de 0.91 (écart-type : 0.03). Toutefois, nous ne mettons pas en avant ces résultats car le lot de contraintes à annoter est déséquilibré à cause de l'utilisation de l'échantillonnage aléatoire (932 CANNOT-LINK, 68 MUST-LINK).

Discussion

A REDIGER : rappel de l'objectif

A REDIGER : Peu d'erreurs (environ 16% d'erreurs sans concertations)

A REDIGER : CANNOT-LINK légèrement plus facile que les MUST-LINK

A REDIGER : ouverture sur l'impact des non-corrections

4.6.4 Bilan concernant la robustesse du *clustering* interactif

A REDIGER

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

...

...

...

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 profondeur 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 Etude 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**, 2022a (*Pull Request en attente pour une version 0.6.0*).

4.7.2 Etude 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 Etude 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 Etude ergonomique de l'interface d'annotation

L'application web développée au cours de ce doctorat (SCHILD et al., 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). Toutes ces idées pourraient être l'objet d'études dédiées avec des groupes d'annotateurs différentes pour voir l'impact sur les performances et les biais de conception de modèles.

4.8 Bilan des études réalisées

SECTION À RÉDIGER

Chapitre 5

Guide d'utilisation du *Clustering Interactif*

5.1 Organisation ITTER

SECTION À RÉDIGER

5.2 Conseils pratiques

SECTION À RÉDIGER

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

Jeux de données

Pour les différentes études réalisées au cours de ce doctorant (cf. SECTION 4), nous avons utilisé les deux jeux de données suivants.

Sommaire

A.1	Bank Cards : Jeu d'entraînement en français d'assistants conversationnels traitant des demandes courantes sur les cartes bancaires.	133
A.2	MLSUM : The Multilingual Summarization Corpus (échantillon)	134

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, 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, 2022

Intention	Définition	Exemple
alerte_perte_volee	Affichage de la procédure de blocage d'une carte perdue ou volée	Comment signaler une perte de carte de paiement ?
carte_avalée	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)

Description : Cet ensemble de données est constitué d'articles de journaux avec leur titre 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 relecteurs ont travaillé sur cette tâche afin de limiter la subjectivité du filtrage. L'échantillon final contient 744 articles après relecture.

Disponibilité : Le jeu de donnée original est archivé sur arXiv et est accessible ici : SCIALOM et al., 2020. L'échantillon réalisé par nos soin est archivé sur la plateforme Zenodo et est accessible ici : SCHILD et ADLER, 2023.

ANNEXE : v-measure, fmc

ANNEXE : packages python, captures écran application

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
ecologie	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.

Bibliographie

- 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>
- 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>
- 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).
- BALEDENT, A. (2023, décembre 1). *De la complexité de l'annotation manuelle : méthodologie, biais et recommandations*. <https://theses.hal.science/tel-04011353>
- 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>
- BIBER, D. (1993). Representativeness in Corpus Design. *Literary and Linguistic Computing*, 8(4). <https://doi.org/10.1093/lrc/8.4.243>
- BLEI, D. M., NG, A. Y., & JORDAN, M. I. (2003). Latent Dirichlet Allocation. *Journal of machine Learning research*, (3), 993-1022.
- 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. <https://doi.org/10.48550/ARXIV.2005.14165>
- BRYSBAAERT, 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>
- Codes for the Representation of Names of Languages – Part 3 : Alpha-3 Code for Comprehensive Coverage of Languages*. (2007, février). <https://www.iso.org/standard/39534.html>
- 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>
- CVAT.AI CORPORATION. (2019, octobre 17). *Computer Vision Annotation Tool (CVAT)*. Récupérée septembre 27, 2023, à partir de <https://www.cvat.ai/>

BIBLIOGRAPHIE

- 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
- 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. TORG, 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, mai 24). *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding*. 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>
- 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>
- 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, février 18). *Overview of Annotation Creation : Processes & Tools*. Récupérée juin 14, 2021, à partir de <http://arxiv.org/abs/1602.05753>
- 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>
- 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.
- 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>
- 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.
- HUANG, J., SHAO, H., & CHANG, K. C.-C. (2022). Are Large Pre-Trained Language Models Leaking Your Personal Information ? <https://doi.org/10.48550/ARXIV.2205.12628>

- Isoz, 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>
- 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.
- 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>
- 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>
- 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
- 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. (2004). Adding linguistic annotation. In M. WYNNE (Éd.), *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>
- 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. <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>
- MAALOUF, M. (2011). Logistic regression in data analysis : an overview. *IJDATS*, 3(3), 281. <https://doi.org/10.1504/IJDATS.2011.041335>
- 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*

BIBLIOGRAPHIE

- 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, avril 20). *Review of End-to-End Speech Synthesis Technology Based on Deep Learning*. arXiv : 2104.09995. <https://doi.org/10.48550/arXiv.2104.09995>
- 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>
- 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. <https://doi.org/10.48550/ARXIV.2307.04821>
- OPENAI. (2023). *ChatGPT*. <https://chat.openai.com>
- Pearson's Correlation Coefficient. (2008). In W. KIRCH (Éd.), *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
- 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>
- 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>
- 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>
- 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).
- RAMESH, A., PAVLOV, M., GOH, G., GRAY, S., VOSS, C., RADFORD, A., CHEN, M., & SUTSKEVER, I. (2021). Zero-Shot Text-to-Image Generation. <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.

- 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.
- 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* (Version 0.5.2). Récupérée février 13, 2023, à partir de <https://zenodo.org/record/4775251>
- SCHILD, E. (2022b, novembre 5). *Cognitivefactory/Interactive-Clustering-Comparative-Study* (Version 0.1.0). Récupérée février 13, 2023, à partir de <https://zenodo.org/record/5648255>
- SCHILD, E. (2023, février 16). *Cognitivefactory/Features-Maximization-Metric* (Version 0.1.1). Récupérée février 16, 2023, à partir de <https://zenodo.org/record/7646382>
- SCHILD, E. (2022, novembre 9). *French trainset for chatbots dealing with usual requests on bank cards*. Zenodo. <https://doi.org/10.5281/zenodo.4769949>
- 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., TTREMBLE & CLEMENTINE-MSK. (2022, septembre 1). *Cognitivefactory/Interactive-Clustering-Gui* (Version 0.4.0). Récupérée février 13, 2023, à partir de <https://zenodo.org/record/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>
- 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>
- STUBBS, A. C. (2013). *A Methodology for Using Professional Knowledge in Corpus* [thèse de doct., Brandeis University].
- TEAM, R. C. (2017). *R : A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. <https://www.R-project.org/>

BIBLIOGRAPHIE

- 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., ALMAHAIROU, A., BABAIEI, Y., BASHLYKOV, N., BATRA, S., BHARGAVA, P., BHOSALE, S., BIKE, D., BLECHER, L., FERRER, C. C., CHEN, M., CUCURULL, G., ESIOMU, D., FERNANDES, J., FU, J., FU, W., ... SCIALOM, T. (2023). Llama 2 : Open Foundation and Fine-Tuned Chat Models. <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>
- VAN ROSSUM, G., & DRAKE, F. L. (2009). *Python 3 Reference Manual* (CreateSpace).
- 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>
- 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. <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>

Liste des TODOs

TITRE A REVOIR : Faciliter => Accélérer ? Améliorer l'accessibilité ? Limiter les biais et erreurs ?	x
CHAPITRE : TITRE À TROUVER : " <i>Introduction</i> "	1
CHAPITRE : INTRODUCTION À RÉDIGER	1
CHAPITRE : INTRODUCTION À RÉDIGER	1
SECTION : TITRE À TROUVER : " <i>Asset centrality</i> "	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 : " <i>Establishing a Niche</i> "	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 : " <i>Gap</i> "	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 : " <i>Occupying the Niche</i> "	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
CHAPITRE : TITRE À TROUVER : " <i>(Revue de littérature)</i> "	3
CHAPITRE : INTRODUCTION À RÉDIGER	3
CHAPITRE : INTRODUCTION À RÉDIGER	3
SUITE A REDIGER	20
SECTION : À RÉDIGER	20
SECTION : À RÉDIGER	20

Liste des *TODOs*

SECTION : À RÉDIGER :	
- Sur les données : ...	
- Sur la complexité : ...	
- Sur les annotateurs :	21
SECTION : À RÉDIGER	21
SECTION : À RÉDIGER	21
SECTION : À RÉDIGER	21
SECTION : TITRE À TROUVER : "REX <i>entreprise</i> "	21
SECTION : À RÉDIGER :	
- Modélisation toujours compliquée ;	
- Expert métier "pas à leur place" ;	
- Peu de stratégies de notre revue mises en oeuvre...	21
À REFORMULER APRÈS L'ÉCRITURE DE L'ÉTAT DE L'ART	23
citation	24
Remarque Gautier 20/02/2023 : erreur de routine, erreur par manque de connaissance, ...	
Il faudra discuter les causes de ces erreurs	24
citation	24
citation	24
à reformuler plus tard.	24
citation	26
SECTION À RÉDIGER : FMC	34
SECTION À RÉDIGER : IC-GUI page d'annotation	34
SECTION À RÉDIGER : IC-GUI gestion d'état de l'application	35
SECTION À RÉDIGER : IC-GUI page d'analyse (en cours)	35
SECTION À RÉDIGER	35
divers à compléter (technique ? méthode ? ...).	37
a revoir	37
Pour ces études, nous allons (1) faire des analyses théoriques (2) des analyses empiriques (car dans la vrai vie on n'a pas de vérité terrain). Nous utilisons aussi la vmeasure.	38
Table des nomenclatures	38
description à faire	61
commentaire Gautier : décrire l'opérateur, son expertise,	101
Utiliser 'footmisc' et 'footref' pour faire des notes de bas de pages communes ? ou lien vers des conclusions ?	107
A REDIGER : description de la figure	119
A REDIGER : rappel de l'objectif	120
A REDIGER : Super important de corriger !	120
A REDIGER : Besoin de mécanisme pour prédire et mettre de la redondance	120
A REDIGER : ouverture sur l'impact des incohérences	121
A REDIGER : objectif de l'expérience	121
A REDIGER	121
A REDIGER	121
A REDIGER :	121
A REDIGER : rappel de l'objectif	121
A REDIGER : prédition de retard	121
FIN	121
description à faire	123
A REDIGER / A COMPLETER	123

A REDIGER : rappel de l'objectif	124
A REDIGER : Peu d'erreurs (environ 16% d'erreurs sans concertations)	124
A REDIGER : CANNOT-LINK légèrement plus facile que les MUST-LINK	124
A REDIGER : ouverture sur l'impact des non-corrections	124
A REDIGER	124
SECTION À RÉDIGER	127
SECTION À RÉDIGER	129
SECTION À RÉDIGER	129
SECTION : TITRE À TROUVER : " <i>Occupying the Niche</i> "	131
SECTION : À RÉDIGER	131
SECTION : TITRE À TROUVER : " <i>Gap</i> "	131
SECTION : À RÉDIGER	131
SECTION : TITRE À TROUVER : " <i>Establishing a Niche</i> "	131
SECTION : À RÉDIGER	131
SECTION : TITRE À TROUVER : " <i>Asset centrality</i> "	131
SECTION : À RÉDIGER	131
ANNEXE : v-measure, fmc	134
ANNEXE : packages python, captures écran application	134
Style d'écriture : "je" ou "nous" ou "on" ?	145
Style d'écriture : "je" ou "nous" ou "on" ?	

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 « Ils sont à vous chef, et j'veus s'rai reconnaissant de bien les garder cette fois. ». 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 : Modelize , Annotate , Train , Test , Evaluate et Revise	10
2.5	Quatre exemples d'outils d'annotation : (1) INCEPTION pour le texte (KLINE 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).	18
3.1	Schéma illustrant l'architecture architectures 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.	25
3.2	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)).	30
3.3	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>).	34
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>).	38

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>).	41
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	45
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>).	47
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.	50
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.	52
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.	53
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>	55
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>).	56
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>).	58

4.11 Capture d'écran de l'application web permettant utilisant notre méthodologie de <i>clustering</i> interactif pour annoter des contraintes (page d'annotation). 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 chosir 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.	61
4.12 Capture d'écran de l'application web permettant utilisant notre méthodologie de <i>clustering</i> interactif pour annoter des contraintes (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.	62
4.13 Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.	63
4.14 Etude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations.	64
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.	72
4.16 Estimation du temps nécessaire (en minutes) pour effectuer une tâche de vectorisation en fonction du nombre de données à traiter.	72
4.17 Estimation du temps nécessaire (en minutes) pour effectuer une tâche de clustering en fonction du nombre de données à traiter.	74
4.18 Estimation du temps nécessaire (en minutes) pour effectuer une tâche d' échantillonnage de contraintes en fonction du nombre de données à traiter.	76
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.	80
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)).	81
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).	84

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>).	85
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>).	87
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.	90
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.	101
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>).	106
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>).	108
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.	109

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.	113
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.	116
4.31 Évolutions de la moyenne de la v-measure entre un résultat obtenu et la vérité terrain en fonction du nombre de contraintes annotées au cours des itérations du <i>clustering</i> interactif. Les courbes en dégradé de couleur représentent les déclinaisons de cette évolution en intégrant un pourcentage d'annotations erronées (allant de 0% et 50%). (1) et (2) représente respectivement l'approche ignorant les conflits dans les contraintes et l'approche corrigeant les conflits détectés par le gestionnaire de contraintes. Toutes les courbes sont tronquées à 3 000 contraintes.	120

Liste des figures

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.	44
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.	51
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.	52
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.	54
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>	95
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 deblocage_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>	96
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.	102

Liste des tableaux

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 <code>gestion_carte_virtuelle</code> et <code>debloque_carte</code> , 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 (<code>gpt-3.5-turbo</code>) sur une tâche de résumé.	103
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 (<code>v-measure</code>) et le score d'accord entre annotation et <i>clustering</i>	110
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 (<code>v-measure</code>) et le score de différence entre deux <i>clustering rings</i> consécutifs.	114
4.11	Simulation	121
4.12	Score d'accord inter-annotateurs obtenu avec 1 réviseur et 3 annotateurs sur un lot commun de 400 contraintes (200 MUST-LINK, 200 CANNOT-LINK).	123
A.1	Présentation du jeu de données <code>Bank Cards</code> avec quelques exemples. La version 2.0.0 contient 100 questions par intention.	134
A.2	Présentation du jeu de données échantillonné à partir de <code>MLSUM</code> avec quelques exemples.	135

Liste des algorithmes

3.1	<i>Description en pseudo-code de la méthode d'annotation proposée employant le clustering interactif</i>	25
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>	42
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>	48
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>	60
4.4	<i>Description en pseudo-code du protocole expérimental de l'étude du temps d'exécution des algorithmes du clustering interactif.</i>	69
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>	78
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>	88
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>	93
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>	99
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>	108
4.10	<i>Description en pseudo-code du protocole expérimental de l'étude de l'évolution de la différence entre deux clustering consécutifs.</i>	112
4.11	<i>Description en pseudo-code du protocole expérimental de l'étude d'intérêt de la correction des incohérences d'annotation.</i>	118
4.12	<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>	122

LISTE DES ALGORITHMES

Liste de codes informatiques

3.1	Jeu exemple pour présenter notre implémentation du <i>clustering</i> interactif.	27
3.2	Démonstration de notre implémentation du prétraitement et de la vectorisation sur le jeu d'exemple.	28
3.3	Démonstration de notre implémentation de gestion des contraintes sur le jeu d'exemple.	30
3.4	Démonstration de notre implémentation du <i>clustering</i> sous contraintes sur le jeu d'exemple.	32
3.5	Démonstration de notre implémentation de l'échantillonnage sur le jeu d'exemple.	34