

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

THÈSE

présentée et soutenue publiquement le 01 septembre 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 "Asset centrality"	1
1.2 "Establishing a Niche"	1
1.3 "Occupying the Niche"	2
2	
État de l'art : concevons un jeu de données	
2.1 Rappel sur le fonctionnement d'un chatbot	3
2.2 Les étapes usuelles de conception d'un chatbot	4
2.2.1 Définition des acteurs	4
2.2.2 Cadrage du projet	5
2.2.3 Collecte des données	5
2.2.4 Modélisation d'une structure et Labellisation des données	5
2.2.5 Entraînement et tests	6
2.2.6 Déploiement de la première version	6
2.2.7 Amélioration continue	6
2.3 Zoom sur la partie Modélisation et Labellisation de la base d'apprentissage	6
2.3.1 Création « manuelle »	6
2.3.2 Création assistée par des regroupements non-supervisés	7
2.3.3 Conception assistée par des regroupements semi-supervisés	7
2.3.4 Conception basée sur des méthodes d'apprentissage actif	7

3

Proposition d'un Clustering Interactif

3.1	Intuitions à l'origine de notre méthode	10
3.2	Description théorique de notre clustering interactif	10
3.3	Description technique et implémentation	12
3.3.1	Gestion des données	13
3.3.2	Gestion des contraintes	15
3.3.3	Algorithme de clustering sous contraintes	17
3.3.4	Algorithme d'échantillonnage de contraintes	18
3.3.5	todo	20
3.4	Espoirs de la méthode proposée	20
3.5	Protocole d'utilisation : Mode d'emploi associé (??CONCLUSION ??)	20

4

Étude de la méthode

4.1	Évaluation de l'hypothèse d'efficacité	24
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	24
4.2	Évaluation de l'hypothèse d'efficience	30
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 .	31
4.3	Évaluation de l'hypothèse sur les coûts	41
4.3.1	Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle	42
4.3.2	Étude du temps de calcul nécessaire aux algorithmes implementés en chronométrant des exécutions dans différentes situations	51
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	60
4.3.4	Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts	64
4.3.5	Ouverture vers une annotation en parallèle du <i>clustering</i>	66
4.4	Évaluation de l'hypothèse de pertinence	70
4.4.1	Étude d'une vérification manuelle de la valeur métier d'une base d'apprentissage par un expert	71
4.4.2	Étude des patterns linguistiques pertinents à l'aide de la <i>Features Maximization</i>	75

4.4.3	Étude d'un résumé automatique des <i>clusters</i> à l'aide d'un modèle de langue	77
4.5	Évaluation de l'hypothèse de rentabilité	81
4.5.1	Etude de l'accord entre annotation et <i>clustering</i>	82
4.5.2	Étude de la différence de résultats de <i>clustering</i> entre deux itérations .	83
4.6	Évaluation de l'hypothèse de robustesse	85
4.6.1	Étude de simulation d'erreurs d'annotations	86
4.7	Autres études à réaliser	87
4.7.1	Choix du nombre de clusters ==> problème de recherche complexe .	87
4.7.2	Impact d'un modèle de langage ==> nécessite de nombreuses données spécifiques au domaine	87
4.7.3	Paradigme d'annotation (intention vs dialogue) ==> problème d'UX + objectif métier	87
4.7.4	(et plein d'autres que j'ajouterai au fur et à mesure de ma rédaction) .	87

5
Conclusion

5.1	Rappel de la problématique ??	89
5.2	Avantage et limites de la méthodes ??	89
5.3	Ouverture ??	89

Annexes

A
Annexe théorique

A.1	Les algorithmes de clustering	91
A.1.1	Kmeans	91
A.1.2	Hierarchique	91
A.1.3	Spectral	91
A.1.4	DBScan	91
A.1.5	Affinity Propagation	91
A.2	Evaluation d'une clustering	92
A.2.1	Homogénéité – Complétude – Vmeasure	92
A.2.2	FMC	92

LISTE DE CODES

B

Annexe technique

B.1	package pypi interactive-clustering	93
B.2	package pypi interactive-clustering-gui	93
B.3	package pypi features-maximization-metrics	93
B.4	experimentations jupyter notebook	93

C

Annexe des jeux de données

C.1	BANK CARDS : french bank cards	95
C.2	MLSUM : press titles	95

Bibliographie	97
----------------------	-----------

Liste des TODOs	101
------------------------	------------

Liste des figures	105
--------------------------	------------

Liste des tableaux	109
---------------------------	------------

Liste des algorithmes	111
------------------------------	------------

Liste de codes	113
-----------------------	------------

Glossaire	115
------------------	------------

Index	117
--------------	------------

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

Chapitre 1

Introduction

CHAPITRE À REFORMULER FAÇON SWALES

1.1 "Asset centrality"

SECTION À RÉDIGER

- Des enjeux ou problèmes actuels
 - Accessibilité à l'information : o Grosses bases documentaires, pas toujours ordonnées ;
 - Relations client à distance o Besoin d'un accessibilité h24 ;
- Utilisation de plus en plus fréquente des chatbots
 - Description succincte ;
 - Cas d'usage usuels ;
 - Tous les canaux d'utilisation ;
 - Avantages et Dérives potentiels de l'utilisation (emploi, biais, pertinence, ergonomie, ...);
- Révolution techniques fréquentes (règles, classification, modèles)
 - Moteurs de règles : o Basé sur la détection de mots clés, o (+) facile à mettre en œuvre, o (-) peu robuste au langage naturel, o Paramétrage des réponses ;
 - Paramétrage intentions-entités : o Classification d'intention et/ou détection d'entités, o (+) plus robuste au langage naturel, facile à paramétrier, réponses contrôlées, o (-) demande de l'entraînement, des données, ..., o Paramétrage des réponses ;
 - Génération de réponse : o Réseau de neurones avec attention, o Transformers, o (+) plus robuste, o (-) plus complexe à mettre en œuvre, réponses non contrôlées, o Réponses non paramétrées ;
 - Approche hybride : o Cumul des trois approches pour cumuler certains avantages suivant les besoins ;

1.2 "Establishing a Niche"

SECTION À RÉDIGER

- Cadre industriel
 - Algorithme fixe
 - Données spécifiques
 - GAP : Besoins de données
 - Collecte de données spécifiques au domaine traité : o extraction de base de données (solution simple), o collecte manuelle (organisation complexe, biais de collecte), o scraping (pas toujours fiable) ;
- Remarque Gautier 20/02/2023 : utilité du travail Un aspect à réfléchir ici : on a besoin de données, en effet, et par conséquent on génère une industrie de l'annotation. Tout se passe un peu comme si on déportait tout le travail nécessaire pour accompagner les clients qui utilisent le chatbot sur les phases d'annotation. Ca pose une question importante de l'utilité du travail : travaille-t-on pour l'humain ou pour la machine ? (ça permet d'aborder la question des débats anti-IA aussi) Pour éviter la déshumanisation du travail, c'est donc très important de réduire l'adhérence aux données et le besoin d'annotation.
- Nombreux biais : o Biais,, o Réglementation, o Compétences (NOTRE COEUR DU SUJET), o ...

1.3 "Occupying the Niche"

SECTION À RÉDIGER

- Etude de l'organisation d'une entreprise pour concevoir ses jeux de données
- Etude de l'état de l'art pour concevoir des jeux de données
- proposition/contribution : une méthode adaptée pour un cadre industriel

Chapitre 2

État de l'art : concevons un jeu de données

Dans cette partie, nous allons faire un état des lieux des méthodes pour créer le premier jeu de données nécessaire à l'entraînement d'un assistant conversationnel. Cela comprend une description des acteurs du projet, un rappel de l'organisation usuelle en fonction de leur compétence, et une énumération des problèmes et solutions les plus communs.

TRANSITION À COMPLÉTER

Sommaire

2.1	Rappel sur le fonctionnement d'un chatbot	3	Rappel des contraintes industrielles
2.2	Les étapes usuelles de conception d'un chatbot	4	
2.2.1	Définition des acteurs	4	
2.2.2	Cadrage du projet	5	
2.2.3	Collecte des données	5	
2.2.4	Modélisation d'une structure et Labellisation des données	5	
2.2.5	Entraînement et tests	6	
2.2.6	Déploiement de la première version	6	
2.2.7	Amélioration continue	6	
2.3	Zoom sur la partie Modélisation et Labellisation de la base d'apprentissage	6	
2.3.1	Création « manuelle »	6	
2.3.2	Création assistée par des regroupements non-supervisés	7	
2.3.3	Conception assistée par des regroupements semi-supervisés	7	
2.3.4	Conception basée sur des méthodes d'apprentissage actif	7	

2.1 Rappel sur le fonctionnement d'un chatbot

SECTION À RÉDIGER

Remarque Gautier 20/02/2023 : Le "usuel" est clairement à discuter ici. Il y a deux approches à la connaissance, qui sont ici à discuter, je pense : - une approche statistique, qui cherche DIRECTEMENT à générer la connaissance à partir de la masse de données ingérée (on y retrouve les approches génératives, par exemple) - une approche symbolique, dans laquelle on décide de passer par des représentations symboliques intermédiaires (les intentions et entités) comme médiateur de la réponse qu'on apporte au client Il n'y a pas d'approche qui soit "usuelle", à mon sens, mais uniquement deux approches de la connaissance différentes, chacune à ses avantages, et en l'occurrence on peut apprécier le pragmatisme de l'approche symbolique, puisque ça a un côté très efficace et ça permet de garder le contrôle sur le vocabulaire (les symboles) qu'on souhaite couvrir. Quelle que soit ta position sur le sujet, je ne pense pas que tu puisses directement parler de fonctionnement usuel sans passer d'abord en revue les différentes approches qu'on peut choisir pour concevoir un chatbot

- Description du cas d'un chatbot supervisé / à base d'intention et d'entités o On se concentre sur ces implémentations car on peut y contrôler les réponses (image de marque en jeu) o Classification d'intention (règles, classification supervisée, ...) o Extraction d'entités (règles, ner, ...) o Mapping des réponses sur la base du couple (*intention, entites*) o
- Description du cas d'un chatbot non-supervisé / à base d'un modèle de langage o

2.2 Les étapes usuelles de conception d'un chatbot

SECTION À RÉDIGER

Préambule : l'organisation peut bien entendu varier suivant les contextes, mais la description qui suit est représentative des organisations principales

a distinguer suivant l'approche statistique et l'approche symbolique

2.2.1 Définition des acteurs

Remarque Gautier 20/02/2023 : Vu le chaos du monde du travail concernant la définition du data scientist, et en quoi il est différent d'un data engineer, analyst, etc..., ce sera important que tu livres ta définition et ton point de vue sur ce qu'est un DS. En fait on pourrait imaginer trouver des experts métiers et des chefs de projets qui connaissent l'IA. On peut même les y former (c'est une des approches qu'on suit souvent). Mais c'est juste pas pratique à faire. Je me demande, à la lecture de cette section, si le problème n'est pas plutôt un problème de division des compétences ici, plutôt que de acteurs. On divise les compétences (connaissance des algorithmes, des données, du métier, de l'organisation d'un projet), et c'est de cette division que naissent les différents acteurs d'un projet. Ca serait intéressant de trouver un exemple d'un chatbot conçu par une seule personne qui prend en charge tous les aspects.

reformuler cette section par "compétences nécessaires" et montrer qu'elles sont en général réparties entre plusieurs acteurs

- Data scientifiques : o Experts en IA o Peu de connaissance métier, i.e. peu de regard critique sur la pertinence des résultats (autre que statistique)
- Expert métier : o Peu de connaissance en IA, i.e. nécessitent des formations o Connaissance métier forte, i.e. peuvent décrire la pertinence d'un résultat

- Chef de projet o Peu de connaissance en IA o Peu de connaissance métier o Connaissance du besoin (hypothèse non vérifiée car parfois ils ne savent pas ce qu'ils veulent dû à la méconnaissance des capacités de l'IA)

2.2.2 Cadrage du projet

- Objectifs : o Clarification du besoin, o Définition du périmètre couvert (i.e. les fonctionnalités et réponses à proposer),
- Livrable : un cahier des charges

Remarque Gautier 20/02/2023 : La aussi, ça mérite presque une digression (et ton point de vue perso) sur les méthodes de travail et l'agilité en particulier. Le cahier des charges et la spécification ont l'avantage de contractualiser le travail à faire, et lorsque le travail est très divisé c'est important. Mais dans la pratique, aujourd'hui tout le monde dit qu'il est Agile. hors, dans l'agilité, on n'est pas sensé avoir de contractualisation. Pourquoi en faire une ici ?

2.2.3 Collecte des données

- Souvent pas de données à disposition : o En R&D, "80%" sur la recherche d'algo sur des données publiques, d'où le besoin de data-scientists, o En entreprise, "80%" sur la gestion des données privées/spécifiques sur des algo connus, d'où le besoin d'experts métiers ;
- Risque de biais dans les données : o Biais d'échantillon : la collecte ne représente pas la réalité, o Biais de sélection : le tri de la collecte ne représente plus la réalité, o Biais de confirmation : on garde les données qui nous arrangent, o Biais de valeur : les données ne sont pas éthiquement représentatives, o Biais de contexte : les données d'un cas d'usage ne sont pas toujours réutilisables pour un autre cas d'usage (ex : différence entre les jargons des AV clients et celui des AV conseillers) ; o **A COMPLETER**
- Livrable : une collecte de données brutes

2.2.4 Modélisation d'une structure et Labellisation des données

Remarque Gautier 20/02/2023 : Au delà de ce que tu écris (avec lequel je suis d'accord), on a aussi un problème plus large. En choisissant une approche symbolique (cf mon commentaire plus haut), ça implique que la création et l'utilisation des chatbots fait se rencontrer deux mondes symboliques : - le monde symbolique des experts travaillant dans le métier (i.e. les banquiers) - le monde symbolique des utilisateurs (i.e. les clients). Il serait intéressant de discuter les raisons pour lesquels ces mondes symboliques peuvent converger (objectifs identiques et partagés, caractère humain...) et diverger (compétences et connaissances très inégales). Ca permet d'avoir un regard critique sur l'organisation du travail, et justement de prôner l'idée que l'on doit retirer le plus possible les facteurs de divergence durant la symbolisation de la connaissance.

- Le coeur "métier" de la création du projet ;
- Objectif : Définition d'une modélisation sur la base des besoins attendus restreints au périmètre à couvrir ;
- En théorie : o Intention : verbe d'actions, o Entités : informations complémentaires, personnes, date, lieux, montants, noms de produits, ... ;
- Complexité de la tâche : o Intention abstraite : définition difficile voir subjective, ... o Annotation difficile : différence entre théorie et pratique, données ambiguës, ... o Plusieurs itérations

car modélisation trop théorique / pas pratique o Besoins de beaucoup de formation (pour donner la compétence aux experts) et d'atelier (pour se mettre d'accord)

- Livrable : un jeu de données annotées

2.2.5 Entrainement et tests

- Le cœur "technique" de la création du projet ;
- Objectif : avoir un modèle qui soit adapté à son utilisation en production
- En théorie : o Split en train et tests o Entraînement et tests o Association des réponses
- Complexité de la tâche : o Modélisation précédente pas toujours adaptée : OK pour un métier, mais pas possible à entraîner à cause de déséquilibre, de manque de données, ... o Algorithme fixe mais données variables : savoir quelle modélisation est la plus adaptée est compliqué à deviner o Réponses pas toujours adaptées aux questions : décalage entre entraînement (modélisation théorique) et réponse (modélisation pratique)

2.2.6 Déploiement de la première version

- RAS
- Parfois la modélisation est décalée par rapport à l'utilisation en production o Comportement en moteur de recherche avec des questions courtes o Vocabulaire non maîtrisé par les utilisateurs o problème d'ergonomie ou d'expérience utilisateur

Remarque Gautier 20/02/2023 : oui, cf mon commentaire plus haut sur la rencontre des mondes symboliques. C'est pour moi un désavantage de cette approche, et ça explique peut-être en partie le succès des approches non supervisées style ChatGPT

2.2.7 Amélioration continue

- Vérification du comportement ;
- Ajustement du modèle ;
- Déploiement des versions suivantes.

Remarque Gautier 20/02/2023 : Quels sont les objectifs de l'AC ? C'est seulement d'améliorer le flux de bonnes réponses ? Ou c'est plus large que ça ? (corriger les erreurs d'interprétation, faire converger les conceptions symboliques, éduquer les équipes, etc...)

2.3 Zoom sur la partie Modélisation et Labellisation de la base d'apprentissage

SECTION À RÉDIGER

2.3.1 Création « manuelle »

• Enchaînement de plusieurs ateliers/cycles : o Définition d'une structure en atelier et Annotation des données o Premier conflit : La structure est trop théorique o Redéfinition et Ré-annotation o Second conflit : Les structures ou les données ne sont pas adaptées o Collecte complémentaire, Redéfinition et Ré-annotation

- Avantages : o Transmission progressive du savoir aux data scientists o Test des modélisations potentielles

2.3. Zoom sur la partie Modélisation et Labellisation de la base d'apprentissage

- Inconvénients :
 - Nombreux ateliers
 - Nombreuses remises en questions / aller-retour de conception
 - L'avis initiale sur le périmètre à couvrir est flou quand cela concerne une centaine de demandes clients
 - Se base sur de la connaissance que les experts métiers n'ont pas
 - Comment les aider dans ce problème d'organisation ?

2.3.2 Crédation assistée par des regroupements non-supervisés

- Constat :
 - Pour des jeux de données à taille humaine (moins de 20.000 données), le premier tri est parfois "optimisé" manuellement sur la base des patterns commun (ordonnancement alphabétique)
- Solution :
 - Un clustering pourrait simplifier cette tâche !
 - Rappel : grandes lignes du fonctionnement d'un algorithme de clustering ?
 - NB : une section ou une annexe détaillera les algorithmes de les plus utilisés
 - KMeans : Classique, Incontournable, Rapide, Efficace
 - Hiérarchique : Lent mais facile à implémenter
 - Spectral : Permet des topologies complexes
 - DBScan : Classique, Incontournable, Rapide, Efficace, Peu d'hyperparamètre
 - Affinity propagation : Metric learning
 - Metric learning : Lent mais plus adapté au corpus
 - ...
- Avantages :
 - Regroupement automatique
 - Découverte de la structure
- Inconvénients :
 - Les résultats sont souvent peu pertinents
 - Similarité par entités, et pas par intentions
 - Nuances métiers non comprises
 - Plusieurs soucis si le jeu de données est déséquilibré ou spécifique
 - Absence d'un modèle de langue spécifique au contexte...
 - parfois besoin d'hyperparamètres complexes à déterminer

clustering
topic
modelling, ...

2.3.3 Conception assistée par des regroupements semi-supervisés

- Solution :
 - On peut envisager ainsi de corriger le clustering en y insérant des contraintes métiers LAMPERT et al., 2018
 - Méthodes semi-supervisée
 - NB : une section ou une annexe détaillera les algorithmes de clustering sous contraintes
 - KMeans : Classique, Incontournable, Rapide, Efficace
 - Hiérarchique : Lent mais facile à implémenter
 - Spectral : Permet des topologies complexes
 - DBScan : Classique, Incontournable, Rapide, Efficace, Peu d'hyperparamètre
 - Affinity propagation : Metric learning
 - Metric learning : Lent mais plus adapté au corpus
- Interactions possibles avec le clustering (sur la base de proposition de l'humain)
 - Sur les données / sur le résultat : ajouts de contraintes sur les données, suppressions ou modifications manuelles de données, réorganisation manuelles des clusters, ...
 - Sur les paramètres : modifier les hyper-paramètres, modifier le nombre de clusters, modifier les embeddings, utiliser d'autres algorithmes, ...
 - Besoin de visualisation : vue des contraintes, de la représentation vectorielle,
 - ...
- Avantage :
 - On a réglé les problèmes de pertinence en ajoutant des contraintes
- Inconvénients :
 - Choisir comment modéliser ces contraintes peut être complexe
 - Surtout énorme en ajoutant des contraintes
 - Choisir les contraintes pertinentes est une tâche difficile

2.3.4 Conception basée sur des méthodes d'apprentissage actif

- Solution :
 - On peut demander à la machine de définir les contraintes dont elle a besoin pour s'améliorer / confirmer son comportement
 - On peut séparer et cibler les tâches pour que le clustering se nourrissent des commentaires de l'expert et que l'expert corrige ce qui semble utile au clustering
 - Sous-entendu : Préférer la collaboration à la supériorité (que ce soit celle de la machine ou celle de l'expert)
 - NB : une section ou une annexe détaillera les interactions possibles entre homme et machine

- Interactions possibles avec le clustering (sur la base de propositions de la machine)
 - Sur les données / sur le résultat : proposition de suppression de données aberrantes, proposition d'ajout de contraintes à des endroits stratégiques, ...
 - Sur les paramètres : réévaluation des paramètres, combiner plusieurs algorithmes et synthétiser le résultat, ...
- Avantage :
 - On a réglé les problèmes de pertinence et de coûts en ajoutant des contraintes
- Inconvénients / problème à résoudre :
 - Accepter de collaborer avec la machine (problème UX, ergo, accompagnement au changement)
 - Il faut prouver cette méthode

Chapitre 3

Proposition d'un Clustering Interactif

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 plus tard.

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 de notre méthode	10
3.2	Description théorique de notre clustering interactif	10
3.3	Description technique et implémentation	12
3.3.1	Gestion des données	13
3.3.2	Gestion des contraintes	15
3.3.3	Algorithme de clustering sous contraintes	17
3.3.4	Algorithme d'échantillonnage de contraintes	18
3.3.5	todo	20
3.4	Espoirs de la méthode proposée	20
3.5	Protocole d'utilisation : Mode d'emploi associé (??CONCLUSION ??)	20

3.1 Intuitions à l'origine de notre méthode

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 autre 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 de notre clustering interactif

Nous proposons la méthode suivante pour transformer une collecte de données brut en une base d'apprentissage nécessaire à l'entraînement d'un assistant conversationnel. Cette méthode, que nous appelons "*clustering interactif*", est décrite formellement à l'aide du pseudo-code figurant dans Alg. 3.1.

AJOUTER SCHEMA : Diagramme d'état ? du point de vue de l'utilisateur ?

La méthode repose principalement sur l'alternance successive entre deux phases clefs :

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

Algorithme 3.1 Description en pseudo-code de la méthode d'annotation proposée employant le clustering interactif

Entrée(s): 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: **évaluation (1)** : estimer la pertinence et la stabilité de la segmentation
- 11: **évaluation (2)** : estimer le budget restant et les coûts restant à investir
- 12: **jusqu'à** segmentation satisfaisante OU budget épuisé

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

utiliser
l'appel-
lation
cluste-
ring ou
segmen-
tation ?

- 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.

Pour l'**initialisation** de la méthode (cf. Alg. 3.1, *lignes 1 à 3*), nous définissons une liste vide de contraintes : tout au long du processus, cette liste contiendra l'ensemble de la connaissance que l'expert transmettra au système sous la forme de contraintes simples sur les données (nous entrerons en détails en décrivant la phase d'*annotation*). De plus, il faut 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 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 segmentation des données est réalisée sans bénéficier de la connaissance de l'expert, il est donc peu probable que le résultat soit pertinent à ce stade.

Nous entrons dans le cœur de la boucle itérative par la phase d'**échantillonnage** (cf. Alg. 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. par-
tie
étude

citation

descrip-
tech-
niqe
plus
tard ?
ref
subsec-
tion :3.3.4

(cf. Alg. 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 concernée¹, sur l'action désirée², ou encore sur le besoin de l'utilisateur³. On notera que des incohérences peuvent s'introduire, ayant pour conclusions de devoir à la fois considérer comme similaire et différentes deux données.

Pour finir, la dernière phase de cette boucle est composée d'une nouvelle **segmentation** des données (cf. Alg. 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. Alg. 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 plus techniques peuvent être envisagés en fonction de la qualité de la segmentation. D'une part, nous pouvons comparer l'évolution de la segmentation des données : si les segmentations sont similaires sur plusieurs itérations, il est possible que la modélisation atteint un optimum local ou un palier de performance. D'autre part, nous pouvons aussi comparer l'évolution de l'accord entre la segmentation obtenue et l'annotation de l'expert : en effet, si l'expert ne contredit plus la répartition proposée des données, il est probable que sa vision et la vision de la machine aient convergé. Dans les deux cas, l'analyse de l'expert métier reste nécessaire pour valider si la modélisation des données est pertinente ou si elle comporte encore des incohérences à corriger.

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 ;

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

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

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

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.

Code 3.1 – Jeu exemple pour présenter notre implémentation du clustering interactif.

```

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 }
```

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`⁴ 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 blancs* sont convertis en un unique espace simple (de "au revoir" à "au revoir").

Si besoin, trois options "avancées" sont disponibles pour réaliser un prétraitement plus destructif :

- la suppression des mots vides (NOTHMAN et al., 2018),
- la conversion des mots vers leur forme racine (MANNING et SCHÜTZE, 2000),
- et la suppression des mots en fonction de leur profondeur dans l'arbre de 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 :

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

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`⁵ permet de transformer les données en une représentation exploitable pour la machine. Deux modes de vectorisation sont mis à disposition :

1. **TF-IDF** (SPARCK JONES, 1972), 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.

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

```

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

24 """

```

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

```

23
24 # Vectorisation des données.
25 dict_of_vectors = vectorize(
26     dict_of_texts=dict_of_preprocess_texts,
27     vectorizer_type="tfidf",
28 )

```

3.3.2 Gestion des contraintes

En ce qui concerne la **manipulation de contraintes**, nous utilisons le module `constraints`⁶ 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 suivantes (cf. figure. 3.1) :

$$(\forall D_1, D_2, D_3) \text{MUSTLINK}(D_1, D_2) \wedge \text{MUSTLINK}(D_2, D_3) \Rightarrow \text{MUSTLINK}(D_1, D_3) \quad (3.1)$$

$$(\forall D_1, D_2, D_3) \text{MUSTLINK}(D_1, D_2) \wedge \text{CANNOTLINK}(D_2, D_3) \Rightarrow \text{CANNOTLINK}(D_1, D_3) \quad (3.2)$$

Pour respecter ces propriétés, le gestionnaire de contraintes doit 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 déduites : nous parlons alors d'incohérence ou de conflit (cf. Fig. 3.1). 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 .

$$\begin{cases} \text{CANNOTLINK}(D_0, D_n) \\ (\exists\{D_i|i \in [0, n]\}) \wedge_{i \in [0, n-1]} \text{MUSTLINK}(D_i, D_{i+1}) \end{cases} \quad (3.3)$$

$$\begin{cases} \text{MUSTLINK}(D_0, D'_0) \\ (\exists\{D_i|i \in [0, n]\}) \wedge_{i \in [0, n-1]} \text{MUSTLINK}(D_i, D_{i+1}) \\ (\exists\{D'_j|j \in [0, m]\}) \wedge_{j \in [0, m-1]} \text{MUSTLINK}(D'_j, D'_{j+1}) \\ \text{CANNOTLINK}(D_n, D'_m) \end{cases} \quad (3.4)$$

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. Fig. 3.1). Ce composant peut être vu comme un noyau de *cluster*. Il pourra être associé à d'autres noyaux par similarité pour former un plus *cluster* plus conséquent, ou être distingué d'autres noyaux pour former plusieurs *clusters*.

Un exemple d'utilisation du module de gestion de contraintes est consultable dans Code 3.3.

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

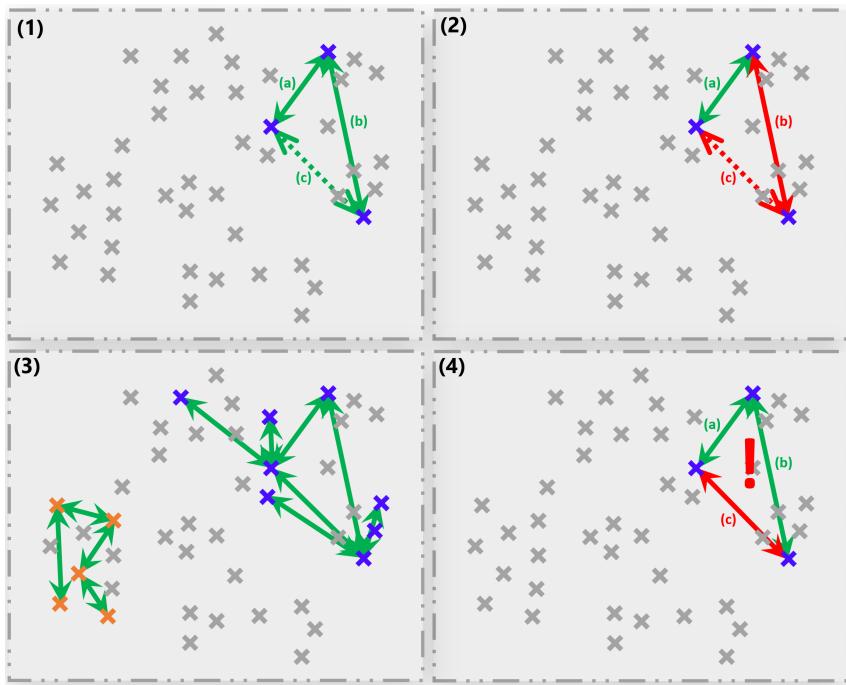


FIGURE 3.1 – Exemples des propriétés de transitivité des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges). (1) et (2) représente les possibilités de déduction d'une contrainte ((c)) en fonction des deux autres ((a) et (b)). (3) représente deux composants connexes définis par la transitivité des contraintes MUST-LINK. Enfin, (4) représente un cas de conflit où une contrainte ((c)) ne correspond pas à sa déduction faite à partir des autres contraintes ((a) et (b)).

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

```

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(

```

```

17     data_ID1=="3" , # "Le distributeur a avalé ma carte !"
18     data_ID2=="4" , # "En retirant de l'argent, le GAB a gardé ma
19     carte...""
20   )
21 constraints_manager.add_constraint(
22     data_ID1=="0" , # "Comment signaler un vol de carte bancaire ?"
23     data_ID2=="N" , # "Pourquoi le sans contact ne fonctionne pas ?"
24     constraint_type="CANNOT_LINK",
25   )
26   # NB: ajouter une contrainte "MUST_LINK" entre "1" et "N" lèverait
27   # une erreur.
27
28 # Récupération des composants connexes.
29 connected_components = constraints_manager.get_connected_components()
30 """
31   [[ '0' , '1' ] ,
32    [ '2' ] ,
33    [ '3' , '4' ] ,
34    [ '5' ] ,
35    [ 'N' ]]
36 """

```

3.3.3 Algorithme de clustering sous contraintes

En ce qui concerne le **regroupement automatique** des données par similarité, nous utilisons le module **clustering**⁷ de la librairie **cognitivefactory-interactive-clustering**.

Ce module met à disposition six algorithmes de *clustering* sous contraintes (référez-vous à la section pour les détails de fonctionnement des algorithmes non contraints) :

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 implementés (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.

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

ref :section2 :clustering

Code 3.4 – Démonstration de notre implémentation du clustering sous contraintes sur le jeu d'exemple.

```

1 # Import des dépendances .
2 from cognitivefactory.interactive_clustering.clustering.factory
3     import clustering_factory
4
5 # Initialiser un objet de clustering .
6 clustering_model = clustering_factory(
7     algorithm="kmeans",
8     model="COP",
9     random_seed=42,
10)
11
12 # Lancer le clustering .
13 clustering_result = clustering_model.cluster(
14     constraints_manager=constraints_manager, # contient les
15         constraints
16     nb_clusters=2,
17     vectors=dict_of_vectors,
18 )
19 """
20     {"0": 0, # "Comment signaler un vol de carte bancaire ?"
21      "1": 0, # "J'ai égaré ma carte bancaire , que faire ?"
22      "2": 0, # "J'ai perdu ma carte de paiement"
23      "3": 1, # "Le distributeur a avalé ma carte !"
24      "4": 1, # "En retirant de l'argent , le GAB a gardé ma carte ... "
25      "5": 1, # "Le distributeur ne m'a pas rendu ma carte bleue."
26      "# ..."
27      "N": 1} # "Pourquoi le sans contact ne fonctionne pas ?"
28 """

```

❶ Pour information : Dans le cadre d'un projet étudiant au sein de l'école 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`⁸ 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 :

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

- une restriction sur la *distance* pouvant imposer aux données d'être les plus proches ou les plus éloignées du corpus ;
- une restriction sur le *résultat du clustering* pouvant imposer aux données d'être issues d'un même cluster ou de clusters différents,
- une restriction pour exclure les contraintes *déjà annotées*,
- et enfin une restriction pour exclure les contraintes *déjà déduites* par transitivité.

Sur cette base, nous définissons quatre niveaux d'échantillonnage facilement identifiables pour nos études :

1. Un échantillonnage **purement aléatoire** en excluant toutes les contraintes déjà annotées ou déduites, noté `samp.random.full` ;
2. Un échantillonnage **pseudo-aléatoire** de données issues d'un **même cluster**, en excluant toutes les contraintes déjà annotées ou déduites, noté `samp.random.same` ;
3. Un échantillonnage des données issues d'un **même cluster** et étant **les plus éloignées** les unes des autres, noté `samp.farhtest.same` (cf. Fig 3.2) ;
4. Un échantillonnage des données issues de **clusters différents** et étant **les plus proches** les unes des autres, noté `samp.closest.diff` (cf. Fig 3.2).

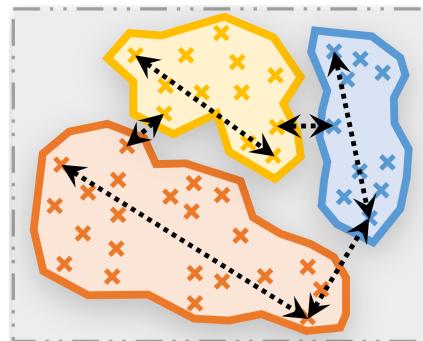


FIGURE 3.2 – Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (`samp.farhtest.same`), et de données issues de clusters différents et étant les plus proches les unes des autres (`samp.closest.diff`).

Une classe abstraite définit les prérequis des algorithmes implementés (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.

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

```

1 # Import des dépendances.
2 from cognitivefactory.interactive_clustering.sampling.factory import
    sampling_factory
3
4 # Initialiser un objet d'échantillonnage.
5 sampler = sampling_factory(

```

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

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
- ...

3.5 Protocole d'utilisation : Mode d'emploi associé (??CONCLUSION ??)

SECTION À RÉDIGER

- Collecte des données
- Itération de clustering > échantillonnage > annotation
- A chaque conflit : correction nécessaire
- A la fin d'un clustering : caractériser la pertinence métier avec FMC
- A chaque itération : voir l'évolution par rapport à la précédente NB : la démonstration de cette proposition de protocole sera démontrée dans la partie 3.

Chapitre 4

Étude de la méthode

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*).

❶ 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, 2022. Enfin, les jeux de données utilisés pour ces études sont détaillés en Annexe C.

Sommaire

4.1	Évaluation de l'hypothèse d'efficacité	24
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	24
4.2	Évaluation de l'hypothèse d'efficience	30
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	31
4.3	Évaluation de l'hypothèse sur les coûts	41
4.3.1	Étude du temps d'annotation nécessaire pour traiter un lot de contraintes en chronométrant des opérateurs en situation réelle	42
4.3.2	Étude du temps de calcul nécessaire aux algorithmes implémentés en chronométrant des exécutions dans différentes situations	51
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	60
4.3.4	Estimation du temps total d'un projet d'annotation en combinant les précédentes études de coûts	64
4.3.5	Ouverture vers une annotation en parallèle du <i>clustering</i>	66
4.4	Évaluation de l'hypothèse de pertinence	70
4.4.1	Étude d'une vérification manuelle de la valeur métier d'une base d'apprentissage par un expert	71
4.4.2	Étude des patterns linguistiques pertinents à l'aide de la <i>Features Maximization</i>	75
4.4.3	Étude d'un résumé automatique des <i>clusters</i> à l'aide d'un modèle de langue	77
4.5	Évaluation de l'hypothèse de rentabilité	81
4.5.1	Etude de l'accord entre annotation et <i>clustering</i>	82
4.5.2	Étude de la différence de résultats de <i>clustering</i> entre deux itérations	83
4.6	Évaluation de l'hypothèse de robustesse	85
4.6.1	Étude de simulation d'erreurs d'annotations	86
4.7	Autres études à réaliser	87
4.7.1	Choix du nombre de clusters ==> problème de recherche complexe .	87
4.7.2	Impact d'un modèle de langage ==> nécessite de nombreuses données spécifiques au domaine	87
4.7.3	Paradigme d'annotation (intention vs dialogue) ==> problème d'UX + objectif métier	87
4.7.4	(et plein d'autres que j'ajouterai au fur et à mesure de ma rédaction)	87

4.1 Évaluation de l'hypothèse d'efficacité

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

❖ Hypothèse d'efficacité ❖

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

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

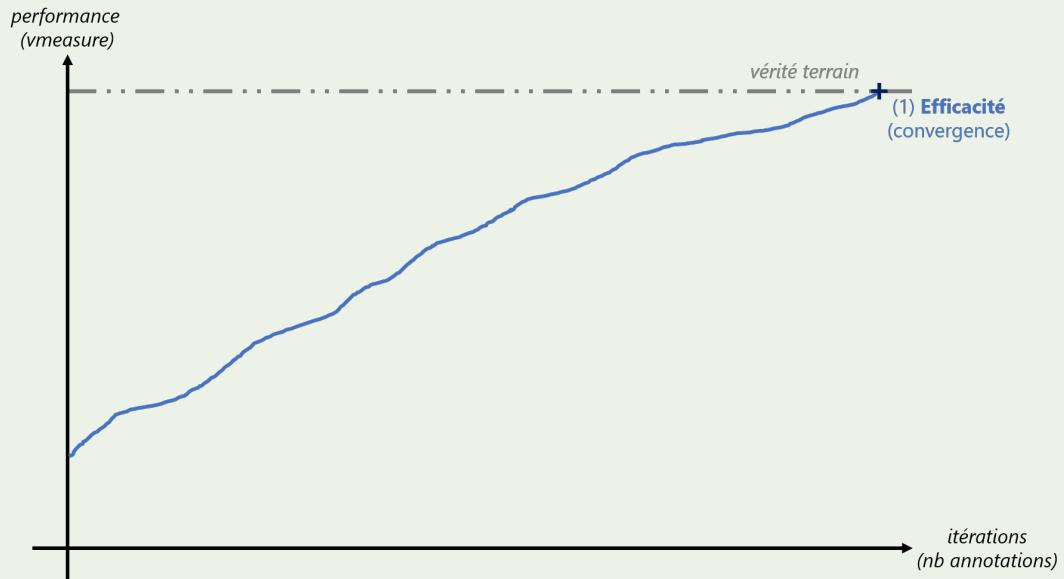


FIGURE 4.2 – Illustration des études réalisées sur le *clustering* interactif (étape 1/6) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre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

Afin de vérifier cette hypothèse, nous mettons en place une **expérience de ré-annotation** d'une base d'apprentissage (qui servira ici de vérité terrain) à l'aide de notre méthode, en simulant l'annotation d'un expert, et nous critiquons l'évolution de la nouvelle base d'apprentissage obtenue ainsi que sa similitude avec la base d'apprentissage initiale (cf. sous-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).⁹

Protocole expérimental

⚠ Attention : Dans le cadre de cette étude, nous supposons que l'expert métier connaît parfaitement le domaine traité dans ce jeu de données, et qu'il est capable de caractériser sans ambiguïté la similitude entre deux données issues de cet ensemble. Cependant, cette hypothèse forte n'est pas toujours vérifiée en pratique, surtout lorsque l'on manipule des données non structurées. L'impact de ce point sur les résultats obtenus 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 Alg. 4.1.

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

Entrée(s): jeu de données annoté (vérité terrain)

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

Sortie(s): arrangements d'algorithmes et de paramètres ayant un score d'équivalence de 100%

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

Pour cette étude, nous essayons une tentative pour chaque combinaison de paramètre 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.farthest.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 50^{10} .

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 propose une segmentation identique des données, donc que la vérité terrain a pu être retrouvée, et donc qu'il est possible d'obtenir une base d'apprentissage pour un assistant conversationnel à l'aide d'une méthodologie d'annotation basée sur le *clustering* interactif.

10. 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 en fin de partie, et nous nous y intéresserons davantage dans la section 4.3 (hypothèse sur les coûts).

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

Résultats obtenus

La figure 4.3 et le tableau 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. tableau 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 326.

La courbe se prolonge jusqu'à l'itération 394 pour que toutes les tentatives puissent 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%

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.

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.

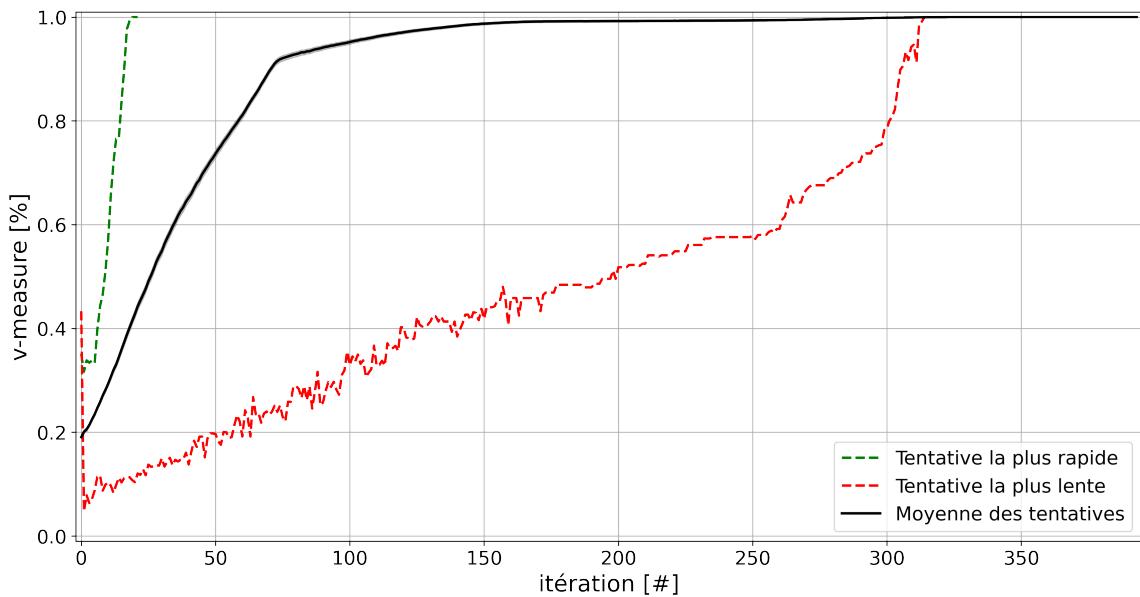


FIGURE 4.3 – Évolution de la moyenne de la **v-measure** entre un résultat obtenu et la vérité terrain en fonction du nombre d’itération de la méthode de *clustering* interactif, moyenne réalisée itération par itération sur l’ensemble des tentatives. Représentation des tentatives ayant été les plus rapides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

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 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 (ambiguités sur les données, méconnaissance du domaine, erreurs d'inattention, différence d'opinions entre annotateurs, ...), ce qui peut entraîner des divergences ou des incohérences dans la construction de la base d'apprentissage. Il semble donc nécessaire d'étudier les impacts de ces incohérences, ainsi que de proposer une méthode pour les prévenir ou les corriger. Cet aspect sera traité à la fin de ce chapitre dans la section 4.6 (hypothèse de robustesse).

4.2 Évaluation de l'hypothèse d'efficience

Suite à la validation de l'hypothèse d'efficacité (convergence de la méthode, cf. section 4.1), nous 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'espoir 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. sous-section 4.2.1).

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

Nous voulons étudier l'influence des paramètres de notre implémentation du *clustering* interactif sur la vitesse de création d'une base d'apprentissage pour un assistant conversationnel. Nous allons donc compléter le protocole expérimental de l'étude de convergence en section 4.1.1 visant à simuler la création d'une base d'apprentissage.

i Pour information : Cette étude a été l'objet d'une présentation à la conférence EGC (Extraction et Gestion des Connaissances) (SCHILD et al., 2021), et d'une extension dans le journal IJDWM (International Journal of Data Warehousing and Mining) (SCHILD et al., 2022).¹¹

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 Alg. 4.2.

Algorithm 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.

Entrée(s): jeu de données annoté (vérité terrain)

```

1: pour tout arrangement d'algorithmes et de paramètres à tester faire
2:   initialisation (données) : récupérer les données et la vérité terrain
3:   initialisation (contraintes) : créer une liste vide de contraintes
4:   prétraitement : supprimer le bruit dans les données
5:   vectorisation : transformer les données en vecteurs
6:   clustering initial : regrouper les données par similarité
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 : ajouter des contraintes en utilisant la vérité terrain
11:    clustering : regrouper les données par similarité avec les contraintes
12:    évaluation : estimer l'équivalence entre le clustering obtenu et la vérité terrain
13:   jusqu'à annotation de toutes les contraintes possibles
14: fin pour
15: analyse : déterminer les tailles d'effets des algorithmes et paramètres

```

Sortie(s): meilleurs arrangements d'algorithmes et de paramètres

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 C.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 totale¹² ;
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 métier¹³.

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, 2022.

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**.

12. 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).

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

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

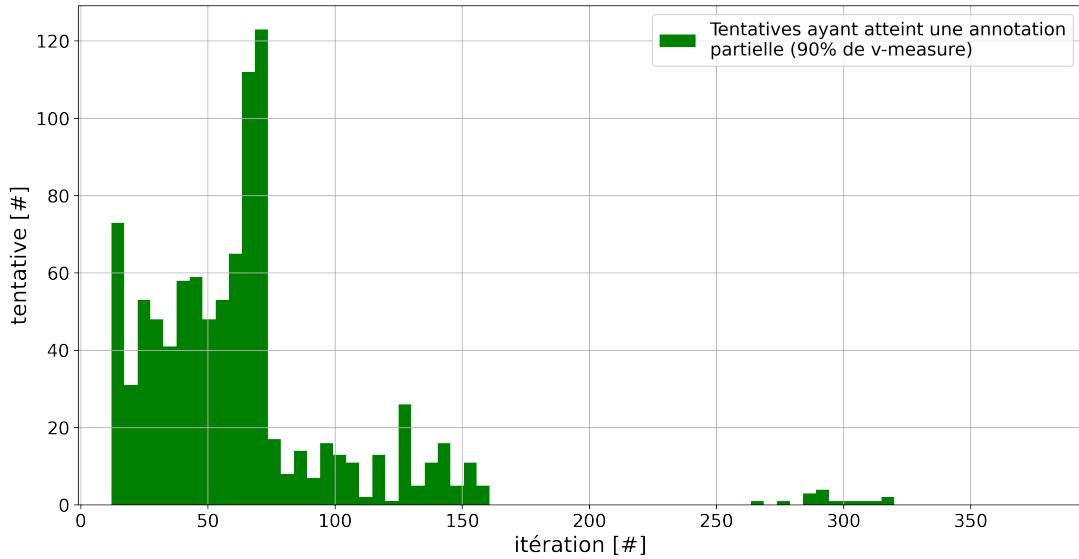


FIGURE 4.5 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation partielle, c'est-à-dire l'itération à laquelle elles parviennent à 90% de `v-measure` entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60 pics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

Le tableau 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`, la vectorisation `vect.tfidf`, le *clustering* `clust.hier.avg`, et l'échantillonnage `samp.closest.diff`. La moyenne du nombre d'itération requis pour ce paramétrage est de 19.00 (écart-type : 0.79), soit 950 annotations (écart-type : 39.34).

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

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

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

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

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

Le tableau 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

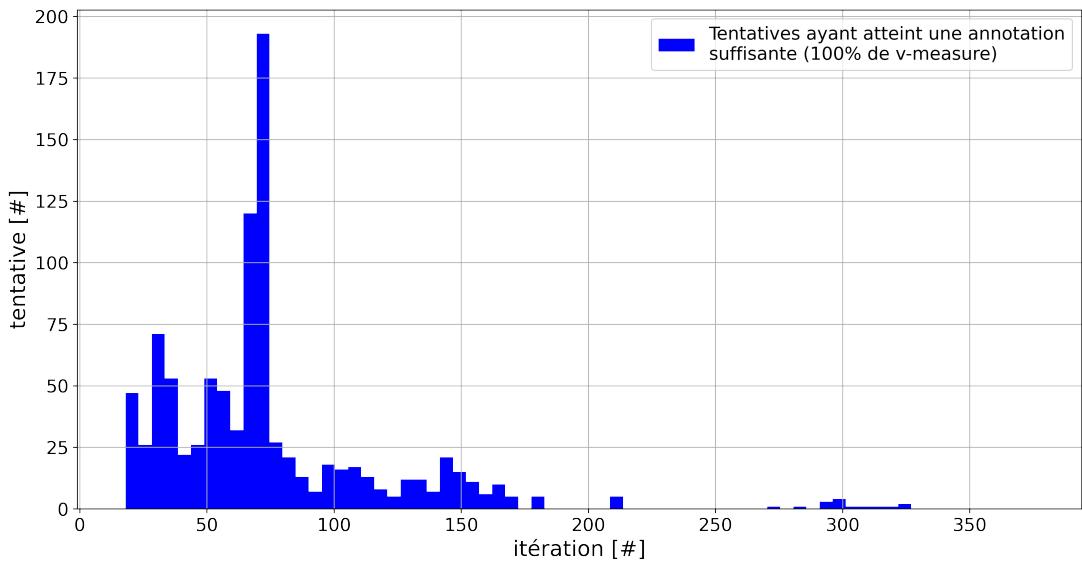


FIGURE 4.6 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation suffisante, c'est-à-dire l'itération à laquelle elles parviennent à 100% de **v-measure** entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60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

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

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

contraintes `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 20 itérations, soit 1 000 annotations, dont 653 (respectivement 668) contraintes MUST-LINK.

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

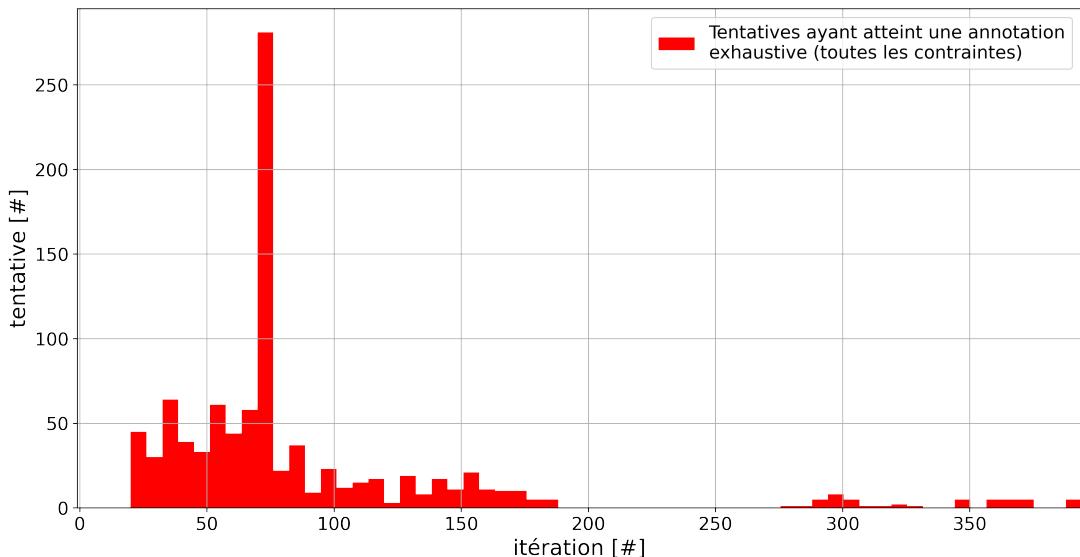


FIGURE 4.7 – Répartition des tentatives en fonction de l’itération de la méthode à laquelle elles atteignent le seuil d’une annotation exhaustive, c’est-à-dire l’itération à laquelle toutes les contraintes possibles entre les données ont été annotées. L’histogramme est réduit à 60 pics pour simplifier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

Le tableau 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).

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 :

- pour une **annotation partielle** (90% de v-measure), le meilleur paramétrage moyen est constitué du prétraitement simple (prep.simple), de la vectorisation TF-IDF (vect.tfidf), du clustering hiérarchique à lien moyen (clust.hier.avg) et de l'échantillonnage des don-

remarque sur la valeur de eta2
remarque sur la valeur de eta2

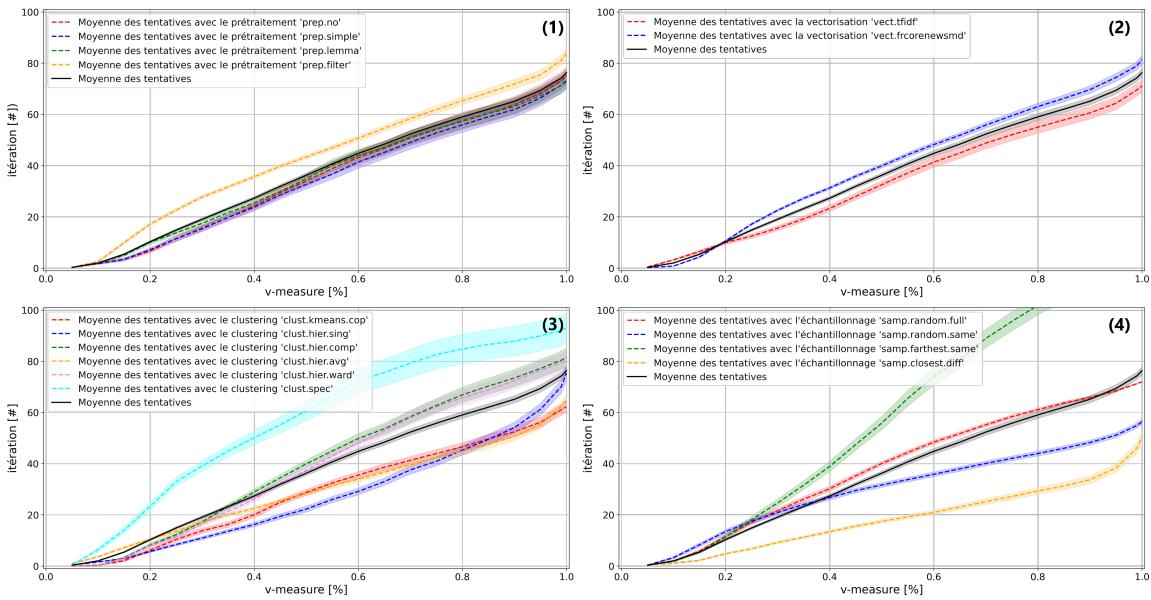


FIGURE 4.8 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de *clustering* interactif pour obtenir un seuil défini de **v-measure** entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes valeurs que peuvent prendre les facteurs analysés et affichées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

nées 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.

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 le tableau 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.

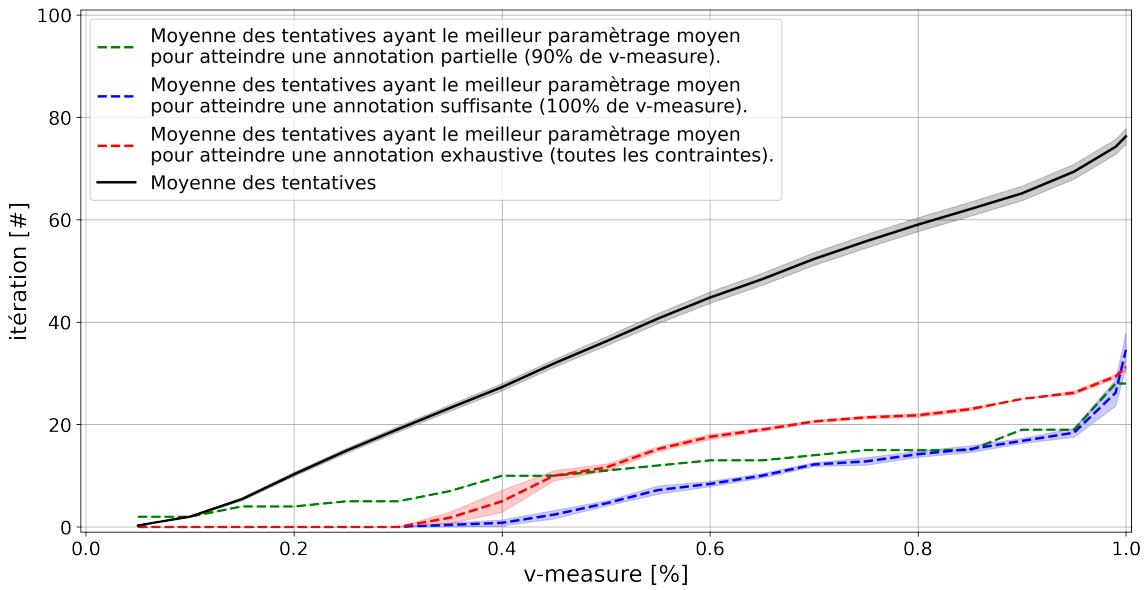


FIGURE 4.9 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de *clustering* interactif pour obtenu un seuil défini de *v-measure* entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes seuils d’annotations étudiés : l’annotation partielle (*atteindre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

Cette 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.

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*

Chapitre 4. Étude de la méthode

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'espoir de pouvoir caractériser la qualité de la base d'apprentissage en cours de construction en fonction d'un coût temporel au lieu d'un nombre abstrait d'itérations de la méthode.



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

Afin de vérifier cette hypothèse, nous organisons plusieurs expériences pour simuler ou 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. sous-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. sous-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. sous-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 sous-section 4.3.4.

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

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

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 Alg. 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 compétence 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 C.2.

Algorithme 4.3 Description en pseudo-code du protocole expérimental de l'étude du temps d'annotation d'un lot de contraintes par un expert métier.

Entrée(s): jeu de données annoté (vérité terrain)

Entrée(s): 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 tout 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:   fin tant que
12: fin pour
13: modélisation : entraîner un modèle linéaire généralisé du temps d'annotation
14: simulation : écrire l'équation du temps d'annotation d'un lot de contraintes

```

Sortie(s): modélisation du temps d'annotation d'un lot de contraintes

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**).

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 (BRYSBART, 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 seconde¹⁴ (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* » ;

14. 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.

- **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 figures 4.11 et 4.12. Une description plus détaillée de l'application et de ses fonctionnalités est disponible en section 3.3.

description
à faire

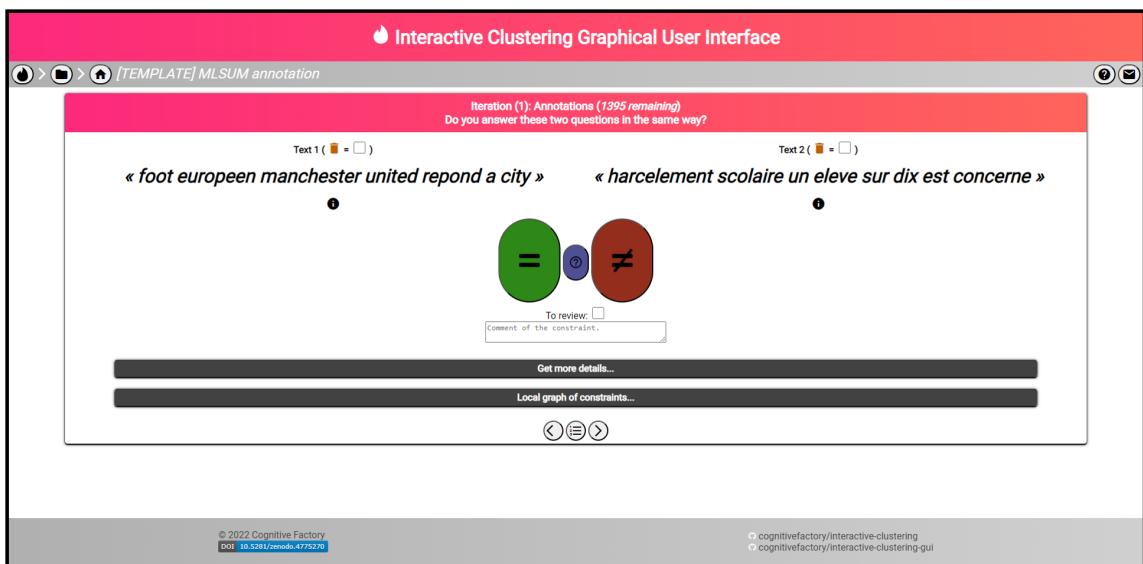


FIGURE 4.11 – Capture d'écran de l'application web permettant d'utiliser notre méthodologie de clustering interactif pour annoter des contraintes (page d'annotation). Les deux textes à annoter sont disposés à gauche et droite de l'écran. Chacun dispose d'un cache à cocher si le texte n'est pas pertinent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

Une fois les sessions d'annotations terminées, nous entraînons un modèle linéaire généralisé

4.3. Évaluation de l'hypothèse sur les coûts

List of constraints								
Sort by: iteration of san ▾ ascending								
Text 1	Constraint	Text 2	Sampling iteration	Last update	To annotate	To review	To fix	Go to
« comment lyon a banni les pesticides de ses parcs et jardins »	CANON	« pour marisol touraine la liberté d installation des medecins est préservée »	1	05/06/2023, at 16:43:20.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	✓	(>)
« le bouclier fiscal c est reporter la fiscalité des plus riches vers les moins riches »	CANON	« immobilier ces voisins qui coutent cher »	1	05/06/2023, at 16:43:11.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	✓	(>)
« jeu de piste entre picasso et godard »	MUST	« le centre pompidou expose wifredo lam pour la première fois »	1	05/06/2023, at 16:43:17.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	✓	(>)
« harclement scolaire un eleve sur dix est concerne »	CANON	« a istanbul l art a un air de defi »	1	05/06/2023, at 16:43:21.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	✓	(>)
« football luis fernandez est le nouveau selectionneur d israel »	CANON	« logements la correction du marche neuf devrait se poursuivre »	1	05/06/2023, at 16:43:25.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	✓	(>)
« deces d egon bahr figure du spd et artisan de l ostpolitik »	SKIP	« la droite s interroge sur la stratégie de nicolas sarkozy »	1	Never	<input type="checkbox"/>	<input type="checkbox"/>	? ()	(>)
« chomage les demandeurs d emploi globalement satisfaits du pole emploi »	SKIP	« d autres virus h2 potentiellement dangereux pour l homme a l instar de la grippe aviaire »	1	Never	<input type="checkbox"/>	<input type="checkbox"/>	? ()	(>)
« bygmalion nicolas sarkozy directement vise »	SKIP	« logement la crise sans fin »	1	Never	<input type="checkbox"/>	<input type="checkbox"/>	? ()	(>)

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.

(*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, 2022. Nous utilisons entre autre 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

moyenne 23.1 minutes (min : 3.0, max : 92.0, écart-type : 14.4). De plus, la vitesse d'annotation moyenne était de 7.7 contraintes par minute (min : 3.5, max : 14.3, écart-type : 2.9).

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

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

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

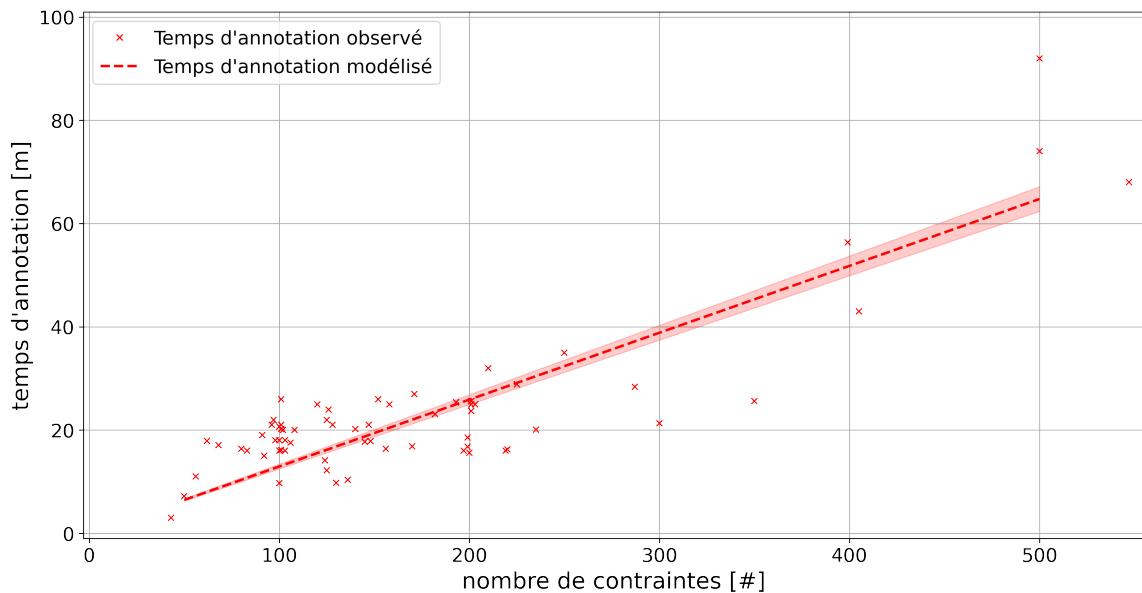


FIGURE 4.13 – Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.

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

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 autre 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*

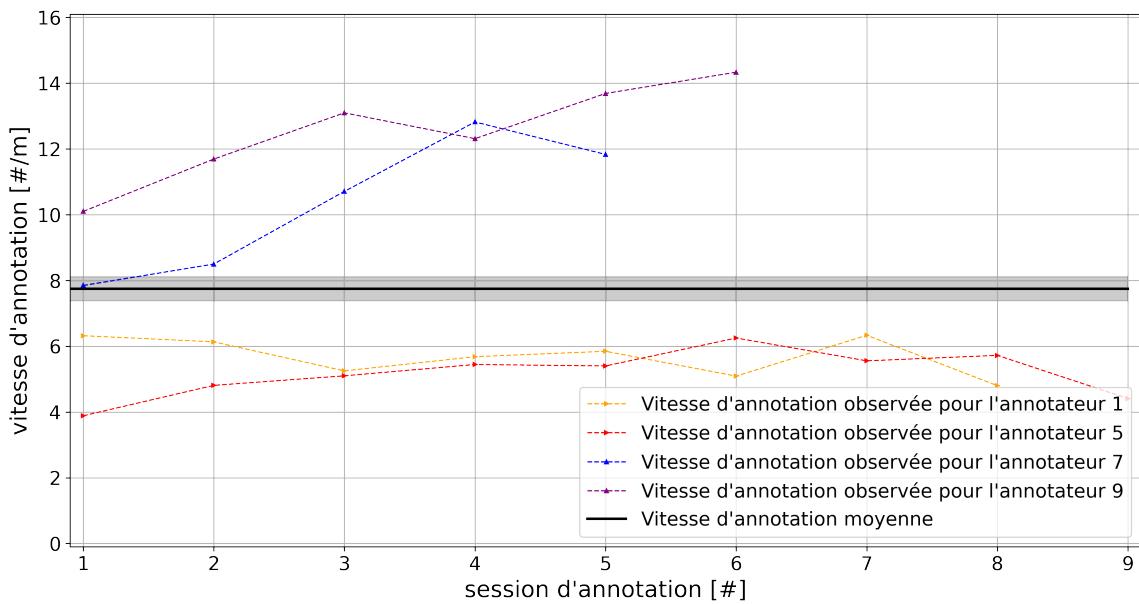


FIGURE 4.14 – Etude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations

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

Pour nous permettre de discuter du temps d'annotation mesuré et de son approximation théorique, nous utilisons utiliser plusieurs points de référence extraits de (SNOW et al., 2008). Dans cette étude, les auteurs délèguent quelques tâches d'annotation à Amazon Mechanical 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 avons choisi d'afficher des données normalisées à l'écran pour masquer le bruit provoqué par les accents, les majuscules, la ponctuation. Bien que cette fonctionnalité peut servir pour des textes bruts (issues de conversation clients ou de forum par exemple), cela a plutôt nuit à la compréhension des données par les opérateurs. Nous avons donc envisager de laisser la données brutes à disposition, dans une infobulle ou grâce à une option permettant d'interchanger le format des données.

Une seconde proposition concerne l'ordre des contraintes à annoter. Nous pouvons facilement admettre que la compréhension rapide d'un grand nombre de texte est une 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.

i 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.

Pour cela, nous allons chronométrier plusieurs exécutions des algorithmes intervenant dans notre implémentation du *clustering* interactif, et nous évaluons 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 Alg. 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 C.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...
 - 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`);

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

Entrée(s): jeux de données annotés (vérité terrain) de tailles différentes

```
1: pour tout arrangement d'algorithmes et de paramètres à tester faire
2:   initialisation : récupérer ou générer le jeu de données
3:   si estimation de la tâche de prétraitement alors
4:     chronomètre : START
5:     prétraitement (à étudier) : supprimer le bruit dans les données
6:     chronomètre : STOP
7:   sinon si estimation de la tâche de vectorisation alors
8:     prétraitement : supprimer le bruit dans les données avec prep.simple
9:     chronomètre : START
10:    vectorisation (à étudier) : transformer les données en vecteurs
11:    chronomètre : STOP
12:   sinon si estimation de la tâche de clustering alors
13:     prétraitement : supprimer le bruit dans les données avec prep.simple
14:     vectorisation : transformer les données en vecteurs avec vect.tfidf
15:     échantillonnage initial : sélectionner une base de contraintes avec samp.rand.full
16:     simulation d'annotation : ajouter des contraintes en utilisant la vérité terrain
17:     chronomètre : START
18:     clustering (à étudier) : regrouper les données par similarité
19:     chronomètre : STOP
20:   sinon si estimation de la tâche d'échantillonnage alors
21:     prétraitement : supprimer le bruit dans les données avec prep.simple
22:     vectorisation : transformer les données en vecteurs avec vect.tfidf
23:     échantillonnage initial : sélectionner une base de contraintes avec samp.rand.full
24:     simulation d'annotation : ajouter des contraintes en utilisant la vérité terrain
25:     clustering initial : regrouper les données par similarité avec clust.kmeans.cop
26:     chronomètre : START
27:     échantillonnage (à étudier) : sélectionner de nouvelles contraintes à annoter
28:     chronomètre : STOP
29:   fin si
30:   mesure : estimer la différence de chronomètre pour cet algorithme
31: fin pour
32: pour tout 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:   modélisation : entraîner un modèle linéaire généralisé avec les facteurs retenus
36:   simulation : écrire l'équation du temps d'exécution avec des paramètres obtenus
37: fin pour
```

Sortie(s): modélisation du temps d'exécution des différents algorithmes

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

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

- précédé par un prétraitement **simple** ;
3. le **clustering sous contraintes** des données...
 - avec les algorithmes suivants : **KMeans** (modèle *COP* noté `clust.kmeans.cop`), **Hiérarchique** (lien *single* noté `clust.hier.sing` ; lien *complete* noté `clust.hier.comp` ; lien *average* noté `clust.hier.avg` ; lien *ward* noté `clust.hier.ward`) et **Spectral** (modèle *SPEC* noté `clust.spec`) ;
 - avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`), le **nombre de contraintes annotés** (variant de 0 à 5 000 par pas de 500, noté `previous_nb_constraints`) et le **nombre de clusters à trouver** (variant de 5 à 50 par pas de 5, noté `algorithm_nb_clusters`) ;
 - précédé par un prétraitement **simple** et une vectorisation **TF-IDF** et un échantillonnage initial **purement aléatoire** ;
 4. l'**échantillonnage** des contraintes à annoter...
 - avec les algorithmes suivants : **purement aléatoire** (noté `samp.random.full`), **pseudo-aléatoire** (noté `samp.random.same`), **même cluster et étant les plus éloignées** (noté `samp.farhest.same`) et **clusters différents et étant les plus proches** (noté `samp.closest.diff`) ;
 - avec les contextes d'utilisation suivants : **nombre de données** (variant de 1 000 à 5 000 par pas de 1 000, noté `dataset_size`), le **nombre de contraintes annotés** (variant de 0 à 5 000 par pas de 500, noté `previous_nb_constraints`), le **nombre de clusters existant** (variant de 10 à 50 par pas de 10, noté `previous_nb_clusters`) et le **nombre de contraintes à sélectionner** (variant de 50 à 250 par pas de 50, noté `algorithm_nb_constraints`) ;
 - précédé par un prétraitement **simple**, une vectorisation **TF-IDF**, un *clustering* initial **KMeans** (modèle *COP*) et un échantillonnage initial **purement aléatoire** ;

Il y a donc 8 825 combinaisons d'algorithmes (15 pour le prétraitement, 10 pour la vectorisation, 3 330 pour le *clustering*, 5 550 pour l'échantillonnage), et chaque combinaison est répétée 5 fois pour contrer les aléas statistiques des exécutions. De plus, chaque jeu de données est généré 5 fois pour contrer les aléas statistiques de création, donc il y a 220 625 exécutions d'algorithmes (375 pour le prétraitement, 250 pour la vectorisation, 82 500 pour le *clustering*, 137 500 pour l'échantillonnage).

Sur la base de ces mesures, nous cherchons à modéliser le temps d'exécution de chaque algorithme en fonction de son contexte d'utilisation (dépendant de ses arguments d'entrée), et les interactions doubles entre paramètres sont envisagées. Afin de réduire la complexité des modélisations, nous ordonnons les interactions de facteurs possibles en fonction de leur corrélation avec le temps mesuré (la corrélation **r** de *Pearson* (KIRCH, 2008) est utilisée) et nous nous limitons aux variables responsables d'un maximum de la variance des mesures (la méthode d'*Elbow* (THORNDIKE, 1953) est utilisée pour choisir les facteurs pertinents). Sur cette base, nous entraînons un modèle linéaire généralisé (*GLM*, cf. NELDER et WEDDERBURN, 1972) pour représenter le temps d'exécution moyen de l'algorithme : ce modèle sera caractérisé par le coefficient de détermination généralisé **R²** de *Cox et Snell* (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, 2022. Nous utilisons entre autre 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.

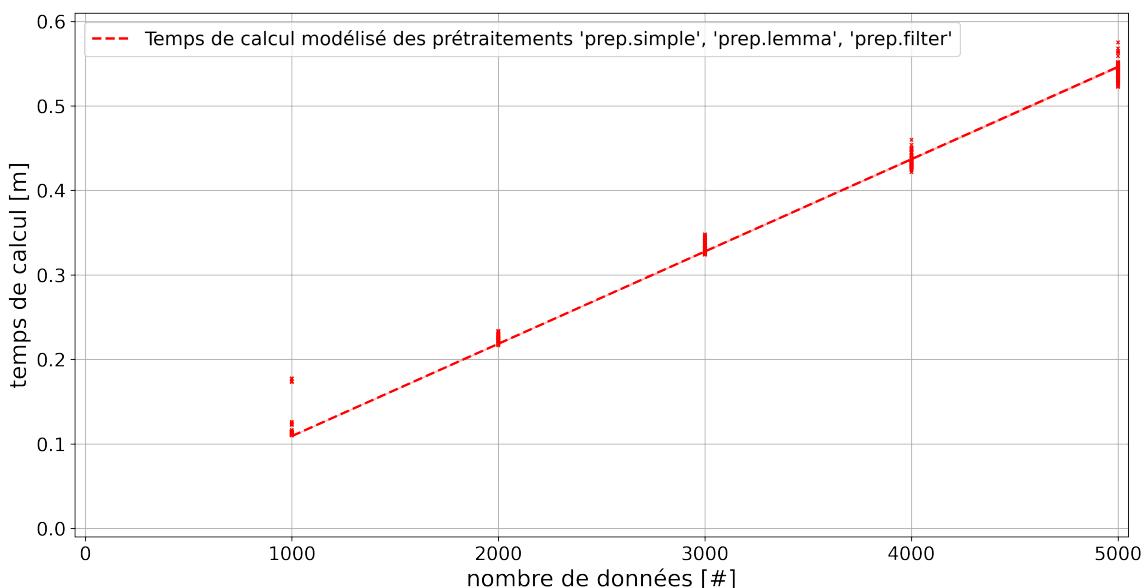


FIGURE 4.15 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de **prétraitement** en fonction du nombre de données à traiter. Les paramétrages `prep.simple`, `prep.lemma` et `prep.filter` ayant des temps de calculs similaires, leurs modélisations n'ont pas été séparées.

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.

4.3. Évaluation de l'hypothèse sur les coûts

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

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

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

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

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

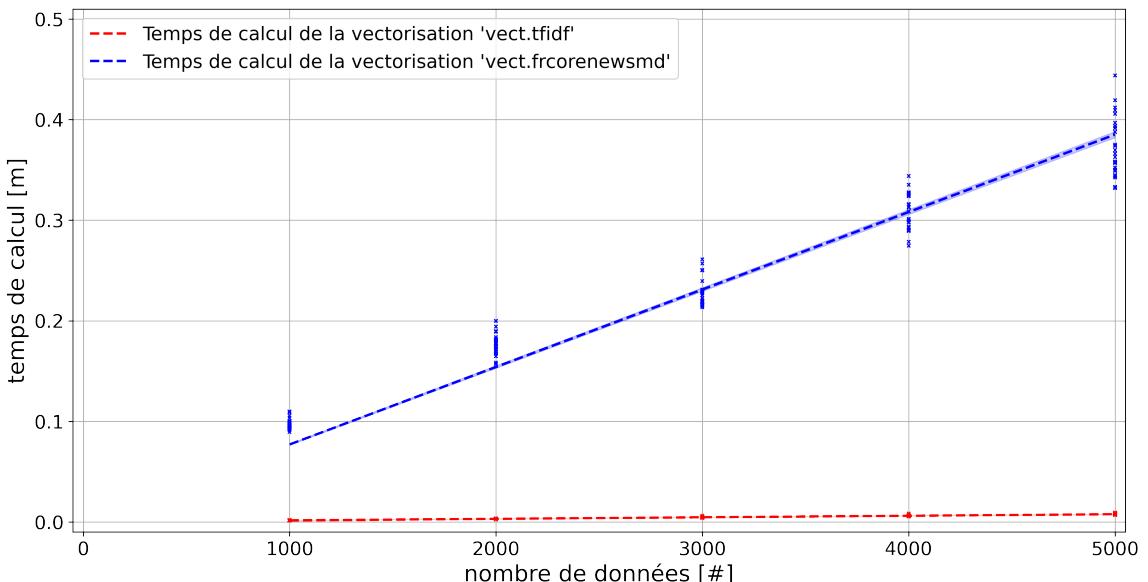


FIGURE 4.16 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de **vectorisation** en fonction du nombre de données à traiter.

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

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

ref complexité théorique algo en annexe

ref complexité théorique algo en annexe

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.

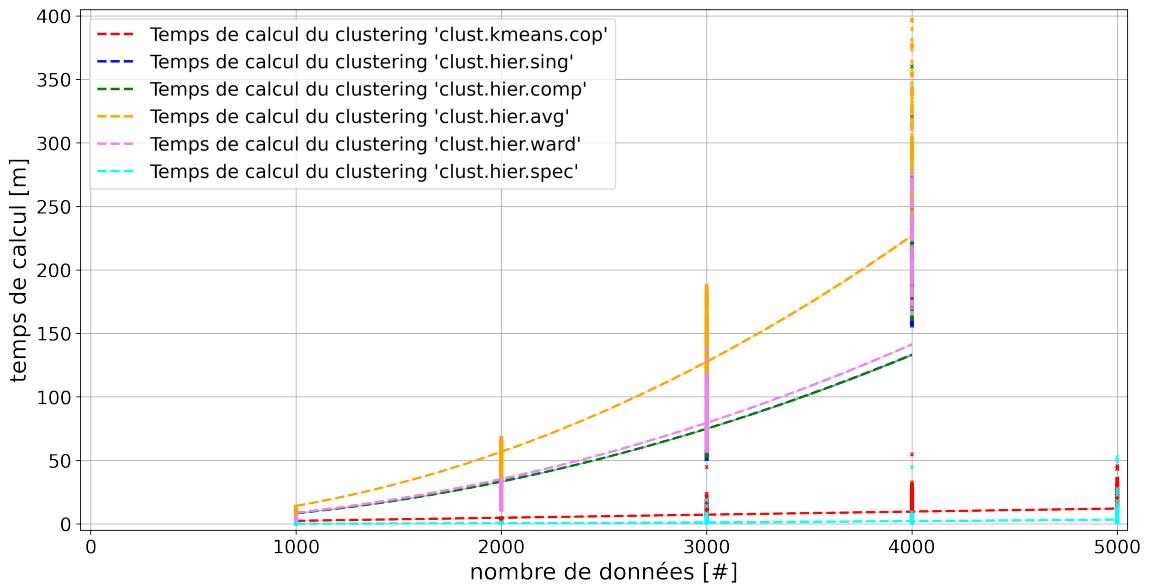


FIGURE 4.17 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche de **clustering** en fonction du nombre de données à traiter.

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

Pour les algorithmes de l'échantillonnage de contraintes `samp.rand.full`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` (*r* : 0.993). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · previous_nb_clusters` (*r* : 0.791). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, `llf` : $-4.52 \cdot 10^4$, `llf_null` : $-1.17 \cdot 10^5$) nous permet de déduire l'équation suivante :

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

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

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

Pour les algorithmes de l'échantillonnage de contraintes `samp.farhtest.same`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation

ref complexité théorique algo en annexe

ref complexité théorique algo en annexe

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.

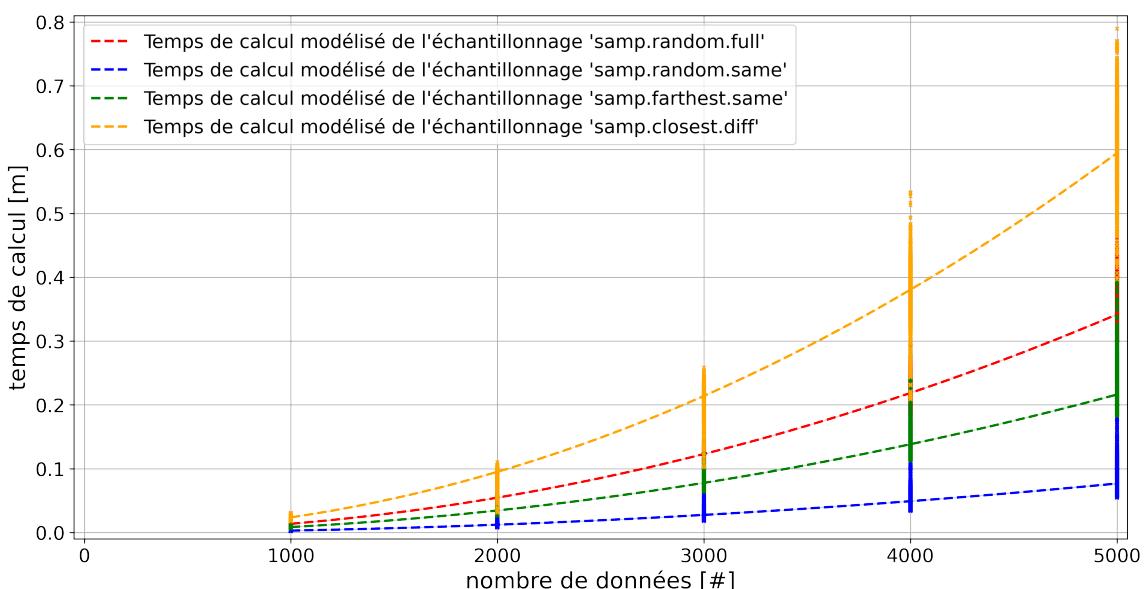


FIGURE 4.18 – Estimation du temps nécessaire (en minutes) pour effectuer une tâche d'échantillonnage de contraintes en fonction du nombre de données à traiter.

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.

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

i Pour information : Dans le cadre du projet étudiant avec l'école Télécom Physique Strasbourg visant à implémenter d'autres algorithmes de *clustering* sous contraintes, un raisonnement similaire a été utilisé pour filtrer les algorithmes. Ainsi, l'implémentation de KMeans (modèle MPC) a été exclu (complexité 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 théorique en $\mathcal{O}(\text{dataset_size})$.

En ce qui concerne les tâches de prétraitements (figure 4.15), de vectorisation (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`¹⁵ 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 suivante¹⁶ :

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

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**¹⁷ sur des jeux de données de différentes tailles et mesurons le nombre de contraintes nécessaires pour converger vers la vérité terrain.

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

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 Alg. 4.5.

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.

Entrée(s): jeux de données annotés (vérité terrain) de tailles différentes

- 1: **pour tout** 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 de nouvelles contraintes à annoter
- 10: **simulation d'annotation** : ajouter des contraintes avec `samp.closest.diff`
- 11: **clustering** : regrouper les données par similarité avec `clust.kmeans.cop`
- 12: **évaluation** : estimer l'équivalence entre le *clustering* obtenu et la vérité terrain
- 13: **jusqu'à** annotation de toutes les contraintes possibles
- 14: **fin pour**
- 15: **analyse** : entraîner un modèle linéaire généralisé du nombre de contraintes nécessaires

Sortie(s): modélisation du nombre de contraintes nécessaires pour un jeu de données

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 C.1 ;
- le jeu de données **MLSUM FR Train Subset (v1.0.0-schild)** : ce dernier concerne les titres d'articles de journaux issus des catégories de publication les plus communes. Il est composé de 744 titres d'articles rédigés et répartis en 14 classes (**économie, sport, ...**). Pour plus de détails, consultez l'annexe C.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 complète¹⁸ de la méthode du *clustering* interactif utilisant notre paramétrage favori est exécuté, et chaque tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions. Il y a donc 510 tentatives de *clustering* interactif réalisées.

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

ℹ️ Pour information : Les scripts de l'expérience, réalisés avec des *notebooks* Python (VAN ROSSUM et DRAKE, 2009), sont disponibles dans un dossier dédié de SCHILD, 2022. Nous utilisons entre autre 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** (**R²** : > 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.

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

- le nombre de contraintes possibles (avec doublons) est de **dataset_size²** (*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*) ;

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

4.3. Évaluation de l'hypothèse sur les coûts

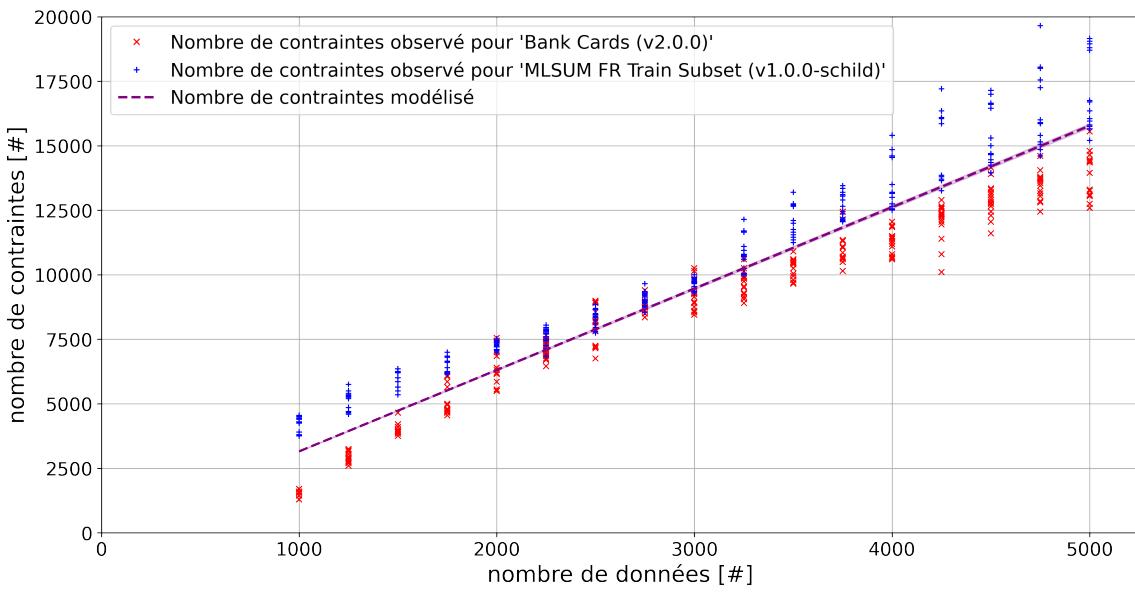


FIGURE 4.19 – Estimation du nombre moyen de contraintes nécessaire à notre **paramétrage favori** du *clustering* interactif afin d'obtenir une annotation partielle (*atteindre une v-measure de 90%*) en fonction de la taille du jeu de données à modéliser.

— le nombre minimal de contraintes à annoter sur une partition en k clusters $\{K_1, K_2, \dots, K_k\}$ du jeu de données et en étant exhaustif est estimé à $\sum_{1 \leq i \leq k} (\|K_i\| - 1) + \sum_{1 \leq i \leq k} (k - i)$ (*considérer d'abord le chemin minimal pour créer des composants connexes avec des contraintes MUST-LINK, puis ajouter le nombre minimal des contraintes CANNOT-LINK pour distinguer les composants connexes en cluster*).

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.

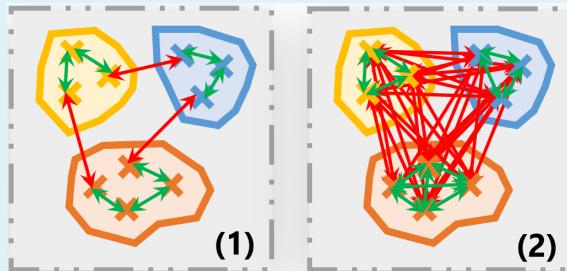


FIGURE 4.20 – Exemple de caractérisation exhaustive d'un jeu de données (10 données, 3 classes) en ajoutant un nombre minimal de contraintes (cf. (1)) ou en ajoutant toutes les contraintes possibles (cf. (2)).

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'équation 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'équation 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'équation 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'équation 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'équation 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**¹⁹. 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

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

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 les deux équations 4.17 et 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. sous-section 4.3.1), estimé le temps de calcul de chaque algorithme implémenté (cf. sous-section 4.3.2) et trouvé le moyen de prédire le nombre de contraintes à annoter sur un jeu de données (cf. sous-section 4.3.3). Nous avons pu montré qu'une annotation de contraintes est plus rapide qu'une annotation par label, et nous conseillons des sessions d'annotation de moins de 150 contraintes pour ne pas épuiser l'annotateur. Sur le paramétrage de la méthode, nous avons rejeté l'usage d'algorithmes de *clustering* de type hiérarchiques à cause de leur lenteur, au profit du KMeans (`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 les équations 4.17, 4.18 et 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 corrigéant manuellement le résultat obtenu, et 1 jour supplémentaire pour interpréter et nommer chaque *cluster*. 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 (REFERENCE) et 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.

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

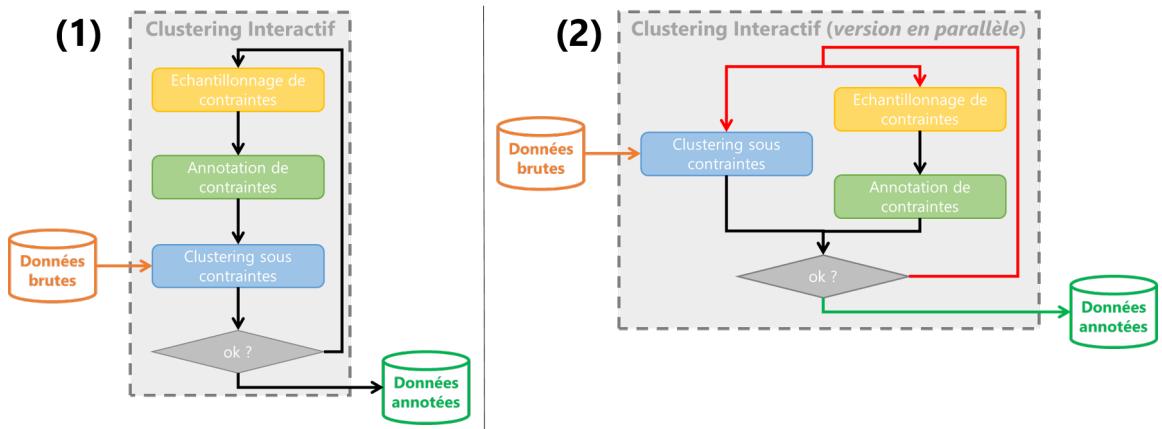


FIGURE 4.21 – Schéma comparatif des architectures du *clustering* interactif : (1) représente la version séquentielle initialement présentée en chapitre 3 où le *clustering* s'adapte avec les annotations de l'itération en cours ; (2) représente l'évolution en mode *parallèle* où le *clustering* s'adapte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

l'équation 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'équation 4.18 :

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

Enfin, il suffit de combiner les deux équations 4.20 et 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 tailles de jeu de données, et permettent de faire la comparaison avec la version séquentielle initialement présentée.

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*, 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

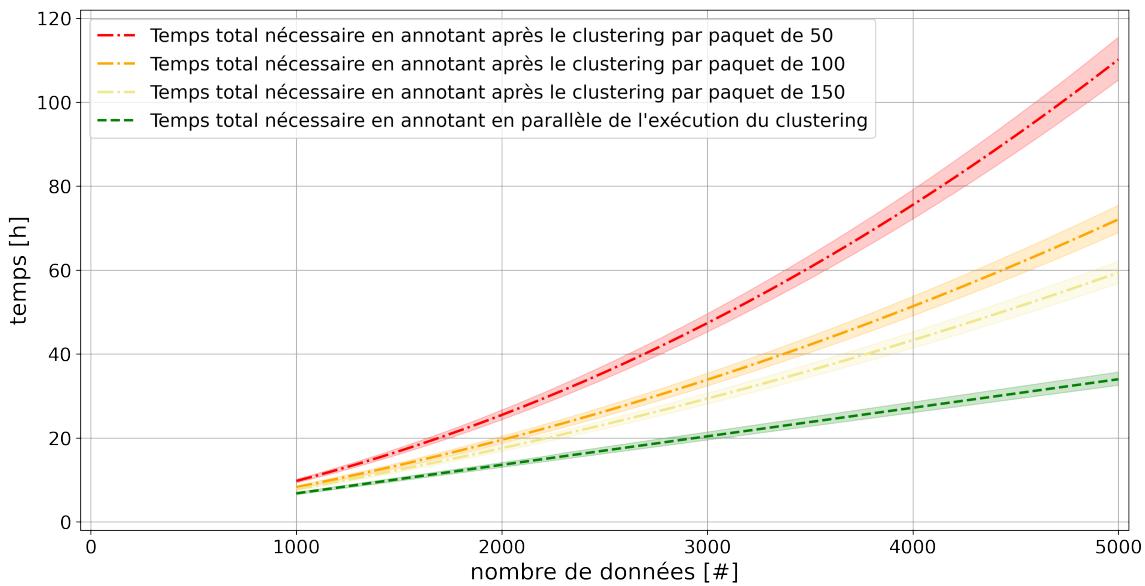


FIGURE 4.22 – Estimation du temps total nécessaire (en heures) pour modéliser un jeu de données avec notre **paramétrage favori** du *clustering* interactif afin d’obtenir une annotation partielle (*atteindre une v-measure de 90 %*), en fonction de plusieurs taille de jeu de données, plusieurs tailles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

très encourageante, car cela confirme qu’une méthodologie basée sur notre implémentation du *clustering* interactif **permet d’obtenir une base d’apprentissage avec un coût temporel équivalent à un projet traditionnel** utilisant une annotation par label. Cette méthode est d’autant plus intéressante qu’elle fait intervenir un mécanisme d’annotation rapide et intuitif pour un expert métier.

- Points à retenir :** Dans cette section, nous avons pu déduire que :
- L’annotation d’une contrainte nécessite en moyenne 8 secondes : cette tâche est rapide et intuitive (cf. sous-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. sous-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. sous-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**. Dans les prochaines sections, 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 à l'aide d'une vérité terrain (cf. calcul de *v-measure*). Cependant, une telle référence n'est pas accessible en situation réelle (l'objectif de notre méthode est précisément de la construire). Nous devons donc nous intéresser à d'autres moyens d'estimer la pertinence des bases d'apprentissages obtenus et de définir comment définir l'exploitabilité d'un résultat. Ainsi, nous aimerions vérifier l'hypothèse suivante :

💡 Hypothèse de pertinence 💡

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

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



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

Afin de vérifier cette hypothèse, nous explorons trois approches :

- une **vérification 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. sous-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. sous-section 4.4.2),
- et une approche utilisant un **résumé automatique de thématique** par un modèle de langue, permettant de décrire succinctement le contenu des *clusters* en une phrase. (cf. sous-section 4.4.3).

4.4.1 Étude d'une vérification manuelle 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 Alg. 4.6.

Algorithme 4.6 Description en pseudo-code du protocole expérimental de l'étude de vérification manuelle de la valeur métier d'une base d'apprentissage.

Entrée(s): jeux de données annotés (vérité terrain) de tailles différentes

- 1: **initialisation (données)** : récupérer ou générer les données et la vérité terrain
- 2: **initialisation (contraintes)** : créer une liste vide de contraintes
- 3: **prétraitement** : supprimer le bruit dans les données avec `prep.simple`
- 4: **vectorisation** : transformer les données en vecteurs avec `vect.tfidf`
- 5: **clustering initial** : regrouper les données par similarité avec `clust.kmeans.cop`
- 6: **évaluation manuelle** : juger de l'exploitabilité de chaque *cluster*
- 7: **répéter**
- 8: **échantillonnage** : sélectionner de nouvelles contraintes à annoter
- 9: **simulation d'annotation** : ajouter des contraintes avec `samp.closest.diff`
- 10: **clustering** : regrouper les données par similarité avec `clust.kmeans.cop`
- 11: **évaluation manuelle** : juger de l'exploitabilité de chaque *cluster*
- 12: **labellisation manuelle** : nommer chaque *cluster* exploitable
- 13: **jusqu'à** annotation de toutes les contraintes possibles
- 14: **analyse** : afficher l'évolution de l'exploitabilité de chaque itération de *clustering*

Sortie(s): discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

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 C.1.

Sur ce jeu de données, nous exécutons une tentative complète²⁰ de la méthode du *clustering* interactif utilisant notre paramétrage favori, et cette tentative est répétée 5 fois pour contrer les aléas statistiques des exécutions.

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

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* poubelle, 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.

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, 2022.

Résultats obtenus

⚠️ Attention : Par manque de personnes aptes à qualifier le jeu de données utilisé, les annotations réalisées dans cette étude n'ont pu être faites que par un seul annotateur. Malgré cette contrainte, nous supposons que la réalisation de l'analyse sur 5 tentatives différentes de la méthode permet de limiter les biais et de discuter des tendances générales.

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_soldé` (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).

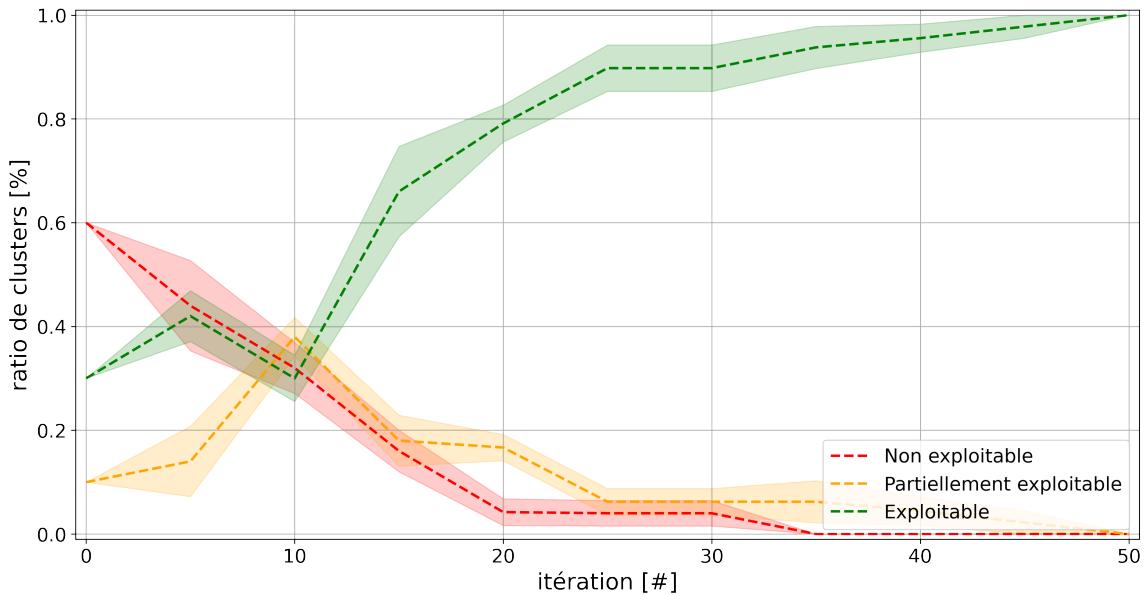


FIGURE 4.24 – Évolution de la pertinence métier moyenne estimée manuellement au cours des itérations du résultat du *clustering* interactif avec notre paramétrage favori. Cette pertinence, exprimée en proportion du nombre de *clusters*, est retranscrite en trois niveaux : **exploitable** en vert, **partiellement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

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 vérifications, 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, alterant 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 n'est pas séduisante. 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 étudier 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* ;
- prémaîcher 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. sous-section 4.4.2) ;
- automatiser une partie du travail d'analyse en utilisant les capacités des larges modèles de langue, afin d'alléger la charge de travail demandée aux experts métiers (cf. sous-section 4.4.3).

4.4.2 Étude des patterns linguistiques pertinents à l'aide de la *Features Maximization*

Nous venons de conclure que la vérification 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 Alg. 4.7.

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.

Entrée(s): jeux de données annotés (vérité terrain) de tailles différentes

- 1: **initialisation (données)** : récupérer ou générer les données et la vérité terrain
- 2: **initialisation (contraintes)** : créer une liste vide de contraintes
- 3: **prétraitement** : supprimer le bruit dans les données avec `prep.simple`
- 4: **vectorisation** : transformer les données en vecteurs avec `vect.tfidf`
- 5: **clustering initial** : regrouper les données par similarité avec `clust.kmeans.cop`
- 6: **évaluation manuelle** : juger de l'exploitabilité de chaque *cluster*
- 7: **répéter**
- 8: **échantillonnage** : sélectionner de nouvelles contraintes à annoter
- 9: **simulation d'annotation** : ajouter des contraintes avec `samp.closest.diff`
- 10: **clustering** : regrouper les données par similarité avec `clust.kmeans.cop`
- 11: **analyse linguistique** : déterminer le vocabulaire caractéristique de chaque cluster
- 12: **évaluation semi-manuelle** : juger de l'exploitabilité de chaque *cluster*
- 13: **labellisation manuelle** : nommer chaque *cluster* exploitable
- 14: **jusqu'à** annotation de toutes les contraintes possibles
- 15: **analyse** : afficher l'évolution de l'exploitabilité de chaque itération de *clustering*

Sortie(s): discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

Nous nous appuyons sur le même protocole que l'expérience précédente (sous-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 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 des composantes caractéristiques et discriminantes de chaque *cluster* en attribuant un score à chaque couple (*cluster, mot*). À l'aide de cette analyse statistique, nous pouvons ainsi mettre en avant des mots clés du *cluster* et surligner ces mots dans les questions à parcourir.

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?)

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

💡 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, 2022. 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 linguistiques au cours des itérations. Nous nous référons aux tableaux ci-contre pour afin le top 10 des termes caractéristiques des *clusters* ainsi qu'un extrait de questions issus de ces *clusters* avec une mise en évidence des termes caractéristiques dans le texte.

D'abord, prenons l'exemple suivi dans le tableau 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 le tableau 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 présent, mais ce *cluster* à l'itération 10 peut donc être associé aux classes *gestion_carte_virtuelle* et *deblocage_carte*. Dès l'itération 15, ces thématiques se séparent en deux clusters (2 et 3) : nous pouvons les identifier à l'aide de leurs listes de termes bien fournies.

4.4. Évaluation de l'hypothèse de pertinence

Identification du cluster	Analyse linguistique (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'évolution de l'analyse linguistique du *cluster* ...

Discussion

A REDIGER

A REDIGER : C'est pas adapté pour un expert métier, mais ça peut être utilisé pour de l'affichage

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

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. En effet, plusieurs de ces modèles ont montré leur efficacité sur les tâches de résumé de documents, une fonctionnalité que nous allons adapter et étudier ici.

Protocole expérimental

A REDIGER

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

Nous nous appuyons sur le même protocole que l'expérience précédente (sous-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 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**).

Chapitre 4. Étude de la méthode

Identification du cluster	Analyse linguistique (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 aimeraï 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 : 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 – Détails ...

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

- un **contexte d'utilisation**, destiné à centrer les réponses sur le domaine général traité : « *Tu es un expert des secteurs banque, assurance et finance.* » ;
- une **description de la tâche** avec les contraintes de restitution : « *Résume-moi en une phrase la thématique traitée dans les textes suivants* : » ;
- les **données** du *cluster*, sous la forme d'une énumération. 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.

À l'aide de ces résumés, nous pouvons ainsi prémâcher 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**

footenote avec détails sur OpenAI ?

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

Entrée(s): jeux de données annotés (vérité terrain) de tailles différentes

- 1: **initialisation (données)** : récupérer ou générer les données et la vérité terrain
- 2: **initialisation (contraintes)** : créer une liste vide de contraintes
- 3: **prétraitement** : supprimer le bruit dans les données avec `prep.simple`
- 4: **vectorisation** : transformer les données en vecteurs avec `vect.tfidf`
- 5: **clustering initial** : regrouper les données par similarité avec `clust.kmeans.cop`
- 6: **évaluation manuelle** : juger de l'exploitabilité de chaque *cluster*
- 7: **répéter**
- 8: **échantillonnage** : sélectionner de nouvelles contraintes à annoter
- 9: **simulation d'annotation** : ajouter des contraintes avec `samp.closest.diff`
- 10: **clustering** : regrouper les données par similarité avec `clust.kmeans.cop`
- 11: **résumé automatique** : utiliser un modèle de langue pour identifier les thématiques
- 12: **évaluation semi-manuelle** : juger de l'exploitabilité de chaque *cluster*
- 13: **labellisation manuelle** : nommer chaque *cluster* exploitable
- 14: **jusqu'à** annotation de toutes les contraintes possibles
- 15: **analyse** : afficher l'évolution de l'exploitabilité de chaque itération de *clustering*

Sortie(s): discussion sur la complexité de la tâche et sur l'évolution de l'exploitabilité

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 ?*)

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, 2022. L'appel à un modèle de langue, sous la forme d'un appel API, se fait grâce à la librairie `openai`.

Résultats obtenus

⚠️ Attention : Par manque de personnes aptes à qualifier le jeu de données utilisé, les annotations réalisées dans cette étude n'ont pu être faites que par un seul annotateur. Malgré cette contrainte, nous supposons que la réalisation de l'analyse sur 5 tentatives différentes de la méthode permet de limiter les biais et de discuter des tendances générales.

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 dans l'analyse manuelle (cf. sous-section 4.4.1), nous retrouvons trois phases se distinguant.

A REDIGER : On peut voir 3 phases : INEXPLOITABLE / DE PLUS EN PLUS EXPLOITABLE / EXPLOITABLE ; les partiellement exploitables sont pas très présents.

A REDIGER : tableau avec exemples ?

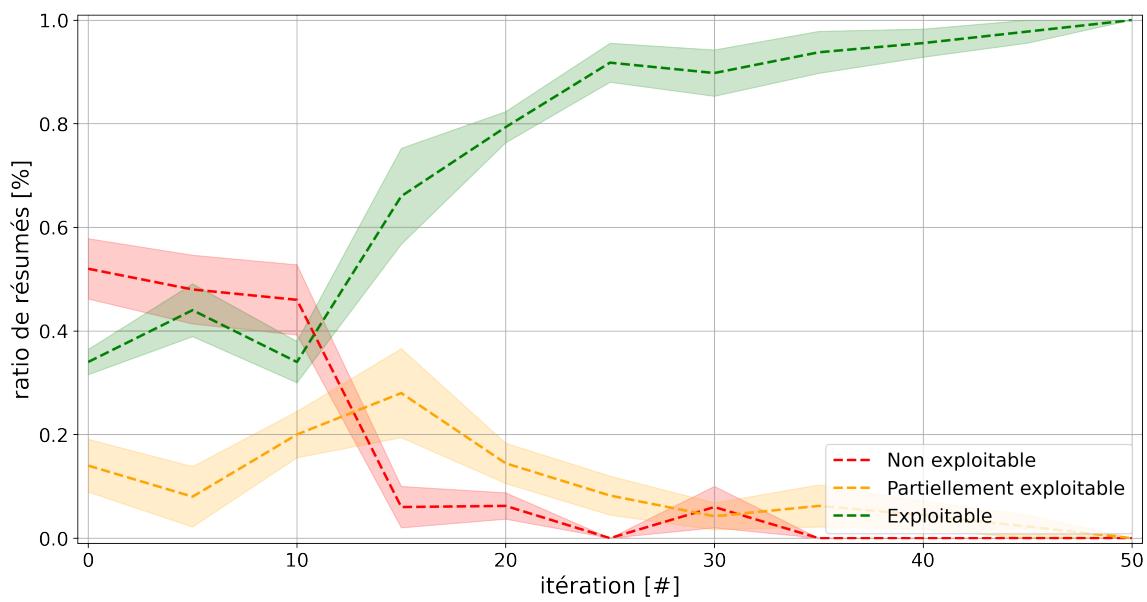


FIGURE 4.25 – Évolution de la pertinence métier moyenne en fonction du nombre d’itérations de la méthode. Cette pertinence, exprimée en proportion du nombre de *clusters*, est estimée sur la base du résumé automatique des *clusters* par un modèle de langue et est retranscrite en trois niveaux : **exploitable**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

Discussion

A REDIGER

A REDIGER : C'est super pratique, super accessible

A REDIGER : C'est parfois un peu ambigu...

4.5 Évaluation de l'hypothèse de rentabilité

Jusqu'à présent, nous avons analysé le cas d'arrêt de notre méthodologie d'annotation basée sur le *clustering* interactif à l'aide d'une comparaison à la vérité terrain. En effet, nous utilisons un seuil de 90% de **v-measure**, caractérisant une annotation dite "partielle". Cependant, une telle référence n'est pas accessible en situation réelle (l'objectif de notre méthode est précisément de la construire). Nous devons donc nous intéresser à d'autres moyens d'estimer ce cas d'arrêt de notre méthode. Ainsi, 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'espoir de pouvoir trouver des cas d'arrêt de la méthode.

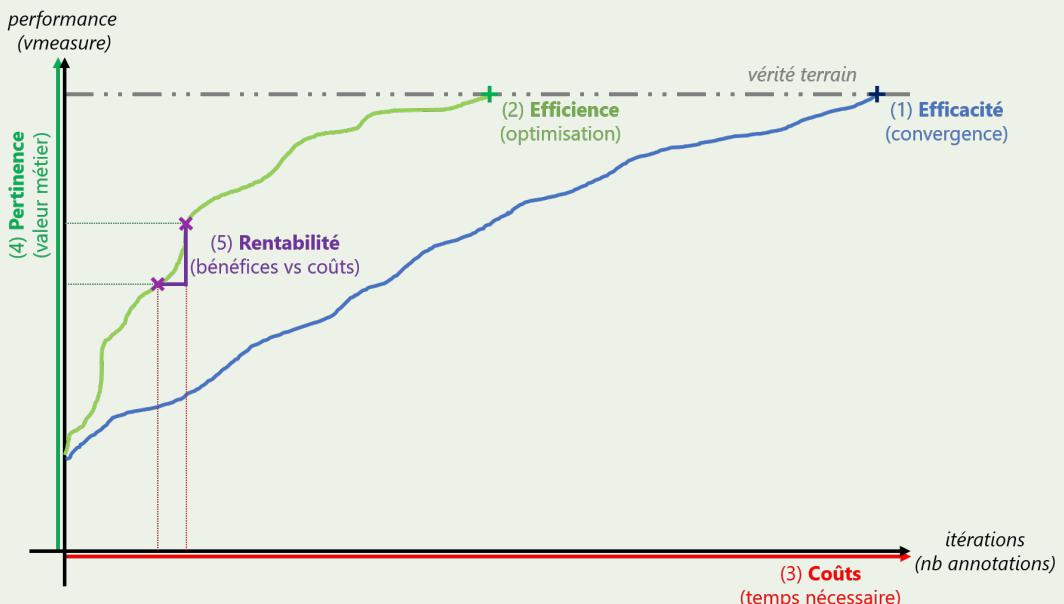


FIGURE 4.26 – Illustration des études réalisées sur le *clustering* interactif (étape 5/6) en schématisant l'évolution de la pertinence (*valeur métier* évaluée par l'expert et exprimé en *nombre de clusters*) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (*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. sous-section 4.5.1) ;

- et l'évolution de la **différence entre deux clustering** successifs, permettant de mesurer s'il y a eu des changements visibles dans le partitionnement des données après l'ajout des dernières contraintes (cf. sous-section 4.5.2).

4.5.1 Etude de l'accord entre annotation et clustering

A REDIGER : objectif de l'expérience

Protocole expérimental

A REDIGER

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

Résultats obtenus

A REDIGER

A REDIGER

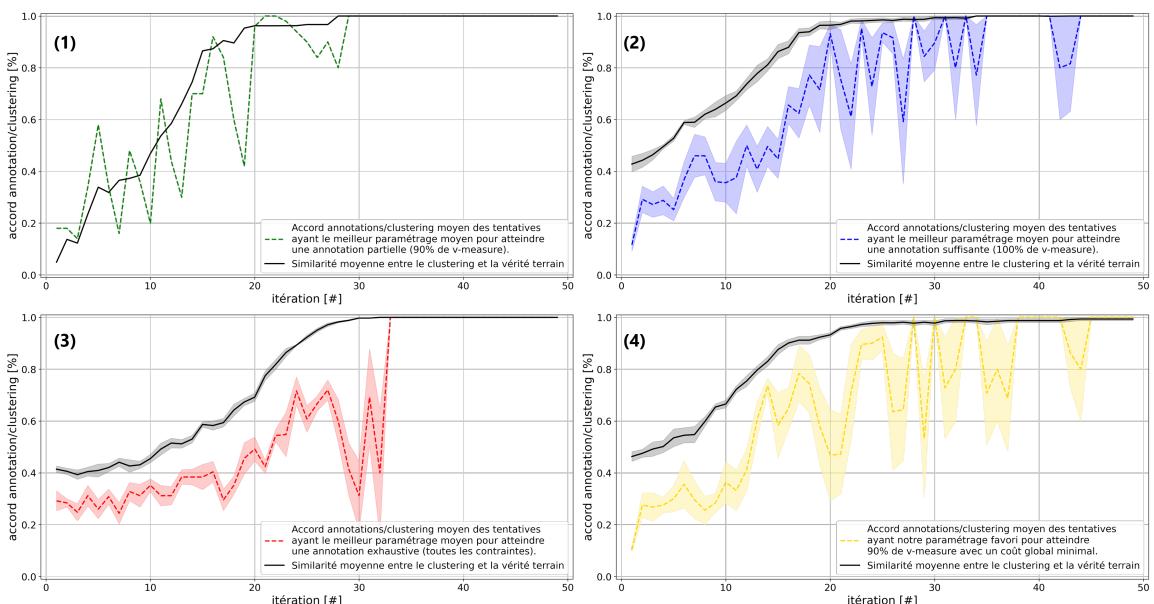


FIGURE 4.27 – Évolution de l'accord entre annotation de contraintes d'un expert et le résultat de *clustering* sur lequel est basé l'échantillonnage de contraintes. Les évolutions moyenne de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori.

Discussion

A REDIGER

A REDIGER : C'est assez instable, cf. échantillonnage optimisé

4.5.2 Étude de la différence de résultats de *clustering* entre deux itérations

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, 2022.

Résultats obtenus

A REDIGER

A REDIGER

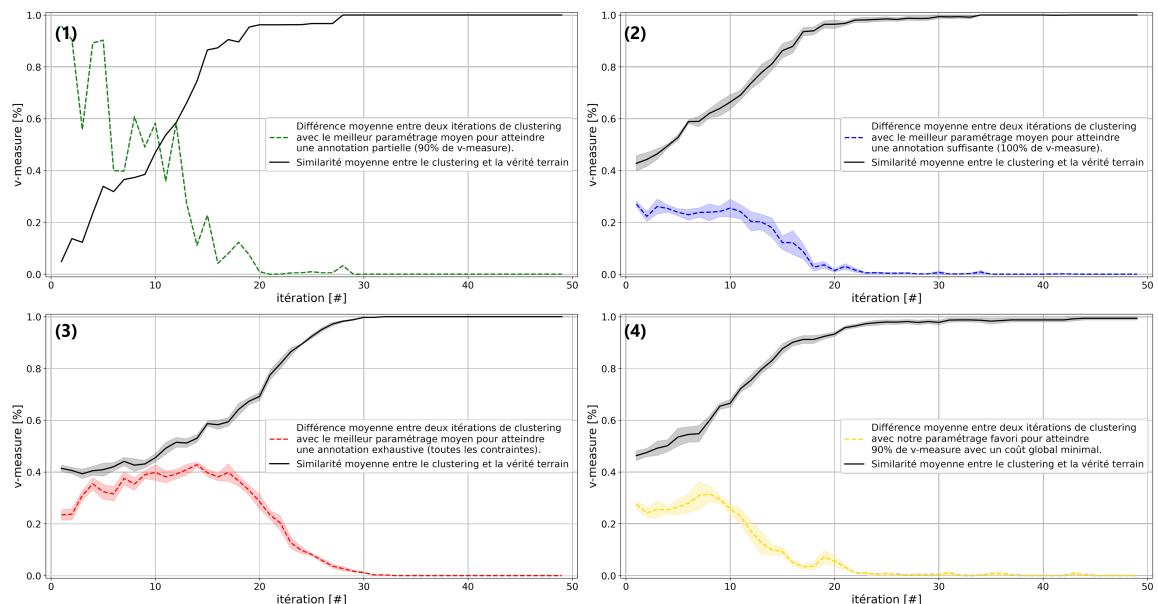


FIGURE 4.28 – Évolution de la différence de résultats entre deux itérations de *clustering*. Les évolutions moyennes de différents paramétrages de la méthode sont exposées : (1) meilleur paramétrage moyen pour atteindre une annotation partielle ; (2) meilleur paramétrage moyen pour atteindre une annotation suffisante ; (3) meilleur paramétrage moyen pour atteindre une annotation exhaustive ; et (4) paramétrage favori.

Discussion

A REDIGER

A REDIGER : Ca montre quand plus rien bouge, fixer un seuil

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. Dans cette section, nous nous intéresserons ainsi à la robustesse de notre méthode en présence d'erreurs d'annotation et d'incohérence dans les contraintes. Pour cela, nous aimerions donc vérifier l'hypothèse suivante :

❖ Hypothèse de robustesse ❖

« Il est possible d'estimer l'influence d'une différence d'annotation lors d'une méthodologie d'annotation basée sur le *clustering* interactif. »

La figure 4.29 illustre cette hypothèse et l'espoir de estimer l'impact d'erreurs ou de différences d'annotations sur le nombre d'itérations de la méthode.

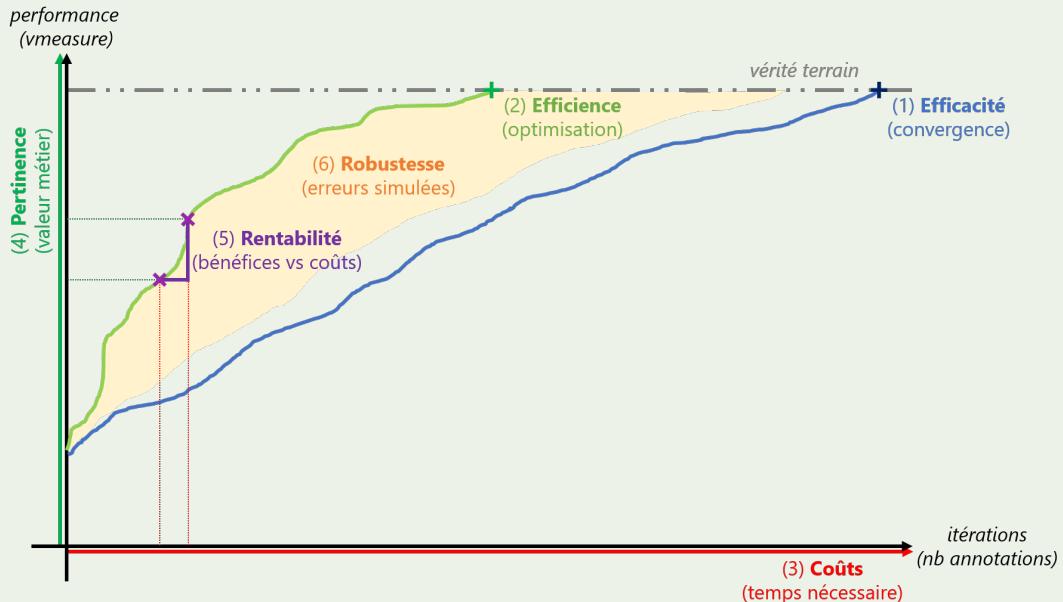


FIGURE 4.29 – Illustration des études réalisées sur le *clustering* interactif (étape 6/6) en schématisant l'évolution de la pertinence (*valeur métier* évaluée par l'expert et exprimé en nombre de clusters) d'une base d'apprentissage en cours de construction en fonction du coût temporel de la méthode (*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 analysons l'impact d'une différence d'annotation sur les performances en simulant l'ajout de contraintes erronées, et nous discuterons de l'intérêt de prédire ou corriger les conflits d'annotation (cf. sous-section 4.6.1).

4.6.1 Étude de simulation d'erreurs d'annotations

A REDIGER : objectif de l'expérience

Protocole expérimental

A REDIGER

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

Résultats obtenus

A REDIGER

A REDIGER : Avec et sans corrections, avec sans sans closest.

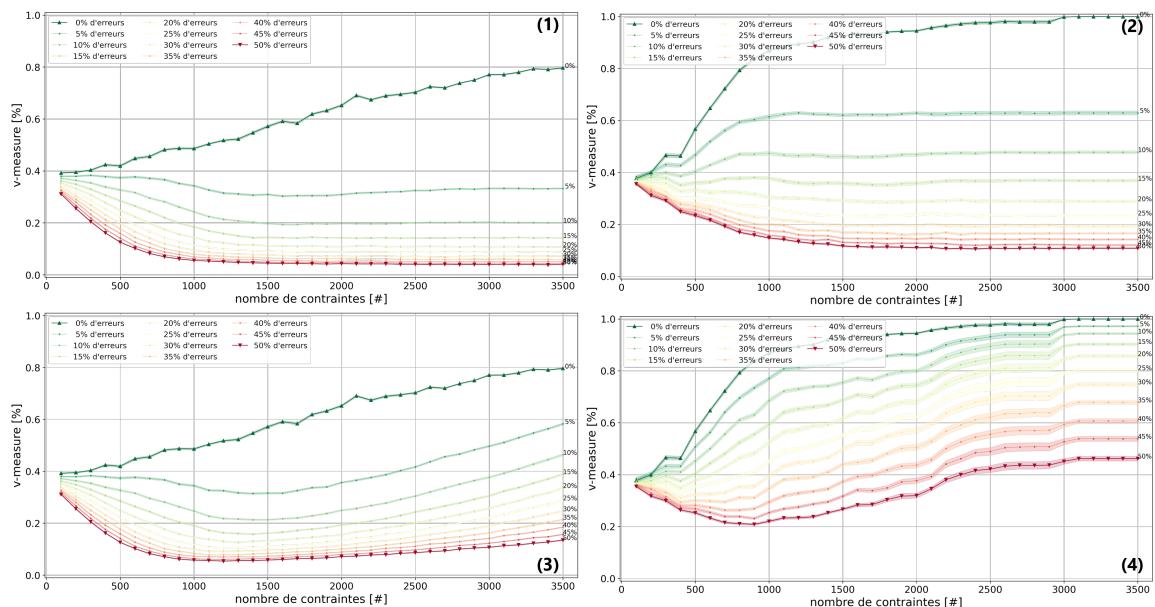


FIGURE 4.30 – Évolutions de la moyenne de la v-measure entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de clustering interactif, moyenne prenant en compte différents taux d'erreurs d'annotations entre 0 et 50%.

Discussion

A REDIGER

A REDIGER : Super important de corriger

4.7 Autres études à réaliser

SECTION À RÉDIGER

4.7.1 Choix du nombre de clusters ==> problème de recherche complexe

- o Piste de résolution : plusieurs clusterings + vote collaboratif ? algorithmes sans le nombre de clusters en hyper-paramètres

4.7.2 Impact d'un modèle de langage ==> nécessite de nombreuses données spécifiques au domaine

- o Piste de résolution : script d'étude comparative déjà prêt, mais il manque les données opensources...

4.7.3 Paradigme d'annotation (intention vs dialogue) ==> problème d'UX + objectif métier

- o Etude Ergo, sort de mon domaine d'expertise

4.7.4 (et plein d'autres que j'ajouterai au fur et à mesure de ma rédaction)

- o

Chapitre 5

Conclusion

5.1 Rappel de la problématique ??

TODO

5.2 Avantage et limites de la méthodes ??

TODO

5.3 Ouverture ??

TODO

Annexe A

Annexe théorique

Sommaire

A.1	Les algorithmes de clustering	91
A.1.1	Kmeans	91
A.1.2	Hierarchique	91
A.1.3	Spectral	91
A.1.4	DBScan	91
A.1.5	Affinity Propagation	91
A.2	Evaluation d'une clustering	92
A.2.1	Homogénéité – Complétude – Vmeasure	92
A.2.2	FMC	92

A.1 Les algorithmes de clustering

A.1.1 Kmeans

kmeans

A.1.2 Hierarchique

hierarchique

A.1.3 Spectral

spectral

A.1.4 DBScan

dbscan

A.1.5 Affinity Propagation

affinity propagation

A.2 Evaluation d'une clustering

A.2.1 Homogénéité – Complétude – Vmeasure

la VMeasure est la moyenne harmonique entre l'homogénéité et la complétude.

A.2.2 FMC

Annexe B

Annexe technique

Sommaire

B.1	package pypi interactive-clustering	93
B.2	package pypi interactive-clustering-gui	93
B.3	package pypi features-maximization-metrics	93
B.4	experimentations jupyter notebook	93

- B.1 package pypi interactive-clustering
- B.2 package pypi interactive-clustering-gui
- B.3 package pypi features-maximization-metrics
- B.4 experimentations jupyter notebook

Annexe C

Annexe des jeux de données

Sommaire

C.1	BANK CARDS : french bank cards	95
C.2	MLSUM : press titles	95

C.1 BANK CARDS : french bank cards

[FR] 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 assistant conversationnel destiné à traiter ces demandes courantes.

Contenu : Les questions sont formulées en français. L’ensemble de données est divisé en 10 intentions de 50 questions chacune, pour un total de 500 questions.

Périmètre des intentions : Les intentions sont construites de telle manière que toutes les questions issues d’une même intention ont la même réponse ou action. Le périmètre couvert concerne : la perte ou le vol de cartes ; la carte avalée ; la commande des cartes ; la consultation du solde bancaire ; l’assurance fournie par une carte ; le déverrouillage de la carte ; la gestion de cartes virtuelles ; la gestion du découvert bancaire ; la gestion des plafonds de paiement ; la gestion du mode sans contact.

Origine : Le périmètre des intentions est inspiré par un chatbot actuellement en production, et la formulation des questions est inspirée de demandes courantes de clients.

C.2 MLSUM : press titles

We present MLSUM, the first large-scale MultiLingual SUMmarization dataset. Obtained from online newspapers, it contains 1.5M+ article/summary pairs in five different languages – namely, French, German, Spanish, Russian, Turkish. Together with English newspapers from the popular CNN/Daily mail dataset, the collected data form a large scale multilingual dataset which can enable new research directions for the text summarization community. We report cross-lingual comparative analyses based on state-of-the-art systems. These highlight existing biases which motivate the use of a multi-lingual dataset.

Annexe C. Annexe des jeux de données

For constraints annotation experiment based on data similarity, this dataset have been subsetted (randomly pick 75 articles in the following 14 most used topics : 'economie', 'politique', 'sport', 'planete' (renamed in 'ecologie'), 'sciences', 'police-justice', 'disparitions', 'emploi', 'sante', 'musiques', 'arts', 'educations', 'climat' (renamed in 'meteo'), 'immobilier') and filtered (keep articles that have an obvious topics regarding their titles, without their bodies). Two reviewers have working on this task in order to limit the subjectivity of the filtering. This subsetted dataset is used (1) to estimate needed time to annotate titles similarity with constraints (MUST-LINK, CANNOT-LINK) and (2) to test interactive clustering methodology (constraints annotation and constrained clustering).

Bibliographie

- 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>
- BALEDENT, A. (2023, décembre 1). *De la complexité de l'annotation manuelle : méthodologie, biais et recommandations.* <https://theses.hal.science/tel-04011353>
- BRYSBAERT, 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>
- 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
- DAVIDSON, I., & RAVI, S. S. (2005). Agglomerative Hierarchical Clustering with Constraints : Theoretical and Empirical Results (A. M. JORGE, L. TORGÓ, P. BRAZDIL, R. CAMACHO & J. GAMA, Éd.). *Knowledge Discovery in Databases : PKDD 2005*, 3721, 59-70. Récupérée octobre 22, 2020, à partir de http://link.springer.com/10.1007/11564126_11
- 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>
- 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>
- 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.
- 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.
- 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.

BIBLIOGRAPHIE

- KHAN, M. A., TAMIM, I., AHMED, E., & AWAL, M. A. (2012). Multiple Parameter Based Clustering (MPC) : Prospective Analysis for Effective Clustering in Wireless Sensor Network (WSN) Using K-Means Algorithm. *WSN*, 04(01), 18-24. <https://doi.org/10.4236/wsn.2012.41003>
- KIRCH, W. (Éd.). (2008). Pearson's Correlation Coefficient. In *Encyclopedia of Public Health* (p. 1090-1091). Springer Netherlands. Récupérée juillet 6, 2023, à partir de https://link.springer.com/10.1007/978-1-4020-5614-7_2569
- LAMIREL, J.-C., CUXAC, P., & HAJLAOUI, K. (2017). A Novel Approach to Feature Selection Based on Quality Estimation Metrics. In F. GUILLET, B. PINAUD & G. VENTURINI (Éd.), *Advances in Knowledge Discovery and Management* (p. 121-140, T. 665). Springer International Publishing. Récupérée novembre 23, 2018, à partir de http://link.springer.com/10.1007/978-3-319-45763-5_7
- LAMPERT, T., DAO, T.-B.-H., LAFABREGUE, B., SERRETTE, N., FORESTIER, G., CRÉMILLEUX, B., VRAIN, C., & GANÇARSKI, P. (2018). Constrained distance based clustering for time-series : a comparative and experimental study. *Data Min Knowl Disc*, 32(6), 1663-1707. <https://doi.org/10/gfbpj8>
- MANNING, C. D., & SCHÜTZE, H. (2000). *Foundations of statistical natural language processing* (2e éd. avec des corrections). MIT Press.
- 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>
- 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>
- 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.
- 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>
- SCHILD, E. (2022, 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., 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>
- SEABOLD, S., & PERKTOLD, J. (2010). Statsmodels : Econometric and Statistical Modeling with Python, 92-96. <https://doi.org/10.25080/Majora-92bf1922-011>
- 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>
- TEAM, R. C. (2017). *R : A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. <https://www.R-project.org/>
- THORNDIKE, R. L. (1953). Who belongs in the family ? *Psychometrika*, 18(4), 267-276. <https://doi.org/10.1007/BF02289263>
- TUKEY, J. W. (1949). Comparing Individual Means in the Analysis of Variance. *Biometrics*, 5(2), 99. <https://doi.org/10.2307/3001913>
- VAN ROSSUM, G., & DRAKE, F. L. (2009). *Python 3 Reference Manual* (CreateSpace).
- 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.

BIBLIOGRAPHIE

Liste des TODOs

TITRE A REVOIR : Faciliter => Accélérer ? Améliorer l'accessibilité ? Limiter les biais et erreurs ?	x
CHAPITRE À REFORMULER FAÇON SWALES	1
SECTION À RÉDIGER	1
SECTION À RÉDIGER	1
Remarque Gautier 20/02/2023 : utilité du travail Un aspect à réfléchir ici : on a besoin de données, en effet, et par conséquent on génère une industrie de l'annotation. Tout se passe un peu comme si on déportait tout le travail nécessaire pour accompagner les clients qui utilisent le chatbot sur les phases d'annotation. Ca pose une question importante de l'utilité du travail : travaille-t-on pour l'humain ou pour la machine ? (ca permet d'aborder la question des débats anti-IA aussi) Pour éviter la déshumanisation du travail, c'est donc très important de réduire l'adhérence aux données et le besoin d'annotation.	2
SECTION À RÉDIGER	2
TRANSITION À COMPLÉTER	3
Rappel des contraintes industrielles	3
titre : Approche statistique vs symbolique	3
SECTION À RÉDIGER	3
Remarque Gautier 20/02/2023 : Le "usuel" est clairement à discuter ici. Il y a deux approches à la connaissance, qui sont ici à discuter, je pense : - une approche statistique, qui cherche DIRECTEMENT à générer la connaissance à partir de la masse de données ingérée (on y retrouve les approches génératives, par exemple) - une approche symbolique, dans laquelle on décide de passer par des représentations symboliques intermédiaires (les intentions et entités) comme médiateur de la réponse qu'on apporte au client Il n'y a pas d'approche qui soit "usuelle", à mon sens, mais uniquement deux approches de la connaissance différentes, chacune à ses avantages, et en l'occurrence on peut apprécier le pragmatisme de l'approche symbolique, puisque ça a un côté très efficace et ça permet de garder le contrôle sur le vocabulaire (les symboles) qu'on souhaite couvrir. Quelle que soit ta position sur le sujet, je ne pense pas que tu puisses directement parler de fonctionnement usuel sans passer d'abord en revue les différentes approches qu'on peut choisir pour concevoir un chatbot	3
citation	4
citation	4
SECTION À RÉDIGER	4
a distinguer suivant l'approche statistiques et l'approche symbolique	4

Liste des *TODOs*

Remarque Gautier 20/02/2023 : Vu le chaos du monde du travail concernant la définition du data scientist, et en quoi il est différent d'un data engineer, analyst, etc..., ce sera important que tu livres ta définition et ton point de vue sur ce qu'est un DS. En fait on pourrait imaginer trouver des experts métiers et des chefs de projets qui connaissent l'IA. On peut même les y former (c'est une des approches qu'on suit souvent). Mais c'est juste pas pratique à faire. Je me demande, à la lecture de cette section, si le problème n'est pas plutôt un problème de division des compétences ici, plutôt que de acteurs. On divise les compétences (connaissance des algorithmes, des données, du métier, de l'organisation d'un projet), et c'est de cette division que naissent les différents acteurs d'un projet. Ca serait intéressant de trouver un exemple d'un chatbot conçu par une seule personne qui prend en charge tous les aspects. . . reformuler cette section par "compétences nécessaires" et montrer qu'elles sont en générales réparties entre plusieurs acteurs	4
Remarque Gautier 20/02/2023 : La aussi, ça mérite presque une digression (et ton point de vue perso) sur les méthodes de travail et l'agilité en particulier. Le cahier des charges et la spécification ont l'avantage de contractualiser le travail à faire, et lorsque le travail est très divisé c'est important. Mais dans la pratique, aujourd'hui tout le monde dit qu'il est Agile. hors, dans l'agilité, on n'est pas sensé avoir de contractualisation. Pourquoi en faire une ici ?	5
Remarque Gautier 20/02/2023 : Au delà de ce que tu écris (avec lequel je suis d'accord), on a aussi un problème plus large. En choisissant une approche symbolique (cf mon commentaire plus haut), ça implique que la création et l'utilisation des chatbots fait se rencontrer deux mondes symboliques : - le monde symbolique des experts travaillant dans le métier (i.e. les banquiers) - le monde symbolique des utilisateurs (i.e les clients) Il serait intéressant de discuter les raisons pour lesquels ces mondes symboliques peuvent converger (objectifs identiques et partagés, caractère humain...) et diverger (compétences et connaissances très inégales). Ca permet d'avoir un regard critique sur l'organisation du travail, et justement de prôner l'idée que l'on doit retirer le plus possible les facteurs de divergence durant la symbolisation de la connaissance.	5
Remarque Gautier 20/02/2023 : oui, cf mon commentaire plus haut sur la rencontre des mondes symboliques. C'est pour moi un désavantage de cette approche, et ça explique peut être en partie le succès des approches non supervisées style ChatGPT	6
Remarque Gautier 20/02/2023 : Quels sont les objectifs de l'AC ? C'est seulement d'améliorer le tux de bonnes réponse ? Ou c'est plus large que ça ? (corriger les erreurs d'interprétation, faire converger les conceptions symboliques, éduquer les équipes, etc...)	6
SECTION À RÉDIGER	6
clustering, topic modeling,	7
à reformuler plus tard.	9
citation	10
Remarque Gautier 20/02/2023 : erreur de routine, erreur par manque de connaissance, ... Il faudra discuter les causes de ces erreurs	10
citation	10
citation	10
à reformuler plus tard.	10
AJOUTER SCHEMA : Diagramme d'état ? du point de vue de l'utilisateur ?	10
utiliser l'appellation clustering ou segmentation ?	11
cf. partie étude	11

citation	11
description technique plus tard ? ref subsection :3.3.4	11
figure, ref subsection :3.3.2	12
cf. partie étude	12
description technique plus tard ? ref subsection :3.3.3	12
description technique plus tard ? ref subsection :3.3.X	12
ref :section2 :clustering	17
SECTION À RÉDIGER : FMC	20
SECTION À RÉDIGER : IC-GUI page d'annotation	20
SECTION À RÉDIGER : IC-GUI gestion d'état de l'application	20
SECTION À RÉDIGER : IC-GUI page d'analyse (en cours)	20
SECTION À RÉDIGER	20
SECTION À RÉDIGER	20
divers à compléter (technique ? méthode ? ...).	21
remarque sur la valeur de eta2	37
remarque sur la valeur de eta2	37
description à faire	44
ref complexité théorique algo en annexe	54
ref complexité théorique algo en annexe	55
ref complexité théorique algo en annexe	55
ref complexité théorique algo en annexe	56
ref complexité théorique algo en annexe	56
ref complexité théorique algo en annexe	56
ref complexité théorique algo en annexe	56
ref complexité théorique algo en annexe	56
ref complexité théorique algo en annexe	56
ref complexité théorique algo en annexe	57
ref complexité théorique algo en annexe	57
ref complexité théorique algo en annexe	58
ref complexité théorique algo en annexe	58
ref : chapitre 3	66
A REDIGER	77
A REDIGER : C'est pas adapté pour un expert métier, mais ça peut être utilisé pour de l'affichage	77
A REDIGER	77
footenote avec détails sur OpenAI ?	78
A REDIGER : On peut voir 3 phases : INEXPLOITABLE / DE PLUS EN PLUS EXPLOITABLE / EXPLOITABLE ; les partiellement exploitables sont pas très présents.	79
A REDIGER : tableau avec exemples ?	79
A REDIGER	80
A REDIGER : C'est super pratique, super accessible	80
A REDIGER : C'est parfois un peu ambigu	80
A REDIGER : objectif de l'expérience	82
A REDIGER	82
A REDIGER	82
A REDIGER	82
A REDIGER	83
A REDIGER : C'est assez instable, cf. échantillonnage optimisé	83

Liste des TODOs

A REDIGER : objectif de l'expérience	83
A REDIGER	83
A REDIGER	83
A REDIGER	83
A REDIGER	84
A REDIGER : Ca montre quand plus rien bouge, fixer un seuil	84
A REDIGER : objectif de l'expérience	86
A REDIGER	86
A REDIGER	86
A REDIGER : Avec et sans corrections, avec sans sans closest.	86
A REDIGER	86
A REDIGER : Super important de corriger	86
SECTION À RÉDIGER	87
Style d'écriture : "je" ou "nous" ou "on" ?	104
Style d'écriture : "je" ou "nous" ou "on" ?	

Liste des figures

3.1	Exemples des propriétés de transitivité des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges). (1) et (2) représente les possibilités de déduction d'une contrainte ((c)) en fonction des deux autres ((a) et (b)). (3) représente deux composants connexes définis par la transitivité des contraintes MUST-LINK . Enfin, (4) représente un cas de conflit où une contrainte ((c)) ne correspond pas à sa déduction faite à partir des autres contraintes ((a) et (b)).	16
3.2	Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (<code>samp.farthest.same</code>), et de données issues de clusters différents et étant les plus proches les unes des autres (<code>samp.closest.diff</code>).	19
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>).	22
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>).	24
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	28
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>).	30
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.	33

Liste des figures

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.	35
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.	36
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>	38
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>).	39
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>).	41
4.11 Capture d'écran de l'application web permettant utiliser 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 chosier 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.	44

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).	45
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 contrainte a été sélectionnée et annotée, si elle est à revoir ou si une incohérence la concernant est détectée.	
4.13 Estimation du temps nécessaire (en minutes) pour annoter un lot de contraintes.	46
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	47
4.15 Estimation du temps nécessaire (en minutes) pour effectuer une tâche de prétraitement en fonction du nombre de données à traiter. Les paramétrages <code>prep.simple</code> , <code>prep.lemma</code> et <code>prep.filter</code> ayant des temps de calculs similaires, leurs modélisations n'ont pas été séparées.	54
4.16 Estimation du temps nécessaire (en minutes) pour effectuer une tâche de vectorisation en fonction du nombre de données à traiter.	55
4.17 Estimation du temps nécessaire (en minutes) pour effectuer une tâche de clustering en fonction du nombre de données à traiter.	57
4.18 Estimation du temps nécessaire (en minutes) pour effectuer une tâche d' échantillonnage de contraintes en fonction du nombre de données à traiter.	58
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.	63
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)).	63
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 annotations 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).	67
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>).	68
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>).	70

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.	73
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 modèle de langue et est retranscrite en trois niveaux : exploitable en vert, partiellement exploitable en orange, et non exploitable en rouge.	80
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>).	81
4.27 Évolution de l'accord entre annotation de contraintes d'un expert et le résultat de <i>clustering</i> sur lequel est basé l'échantillonnage de contraintes. Les évolutions moyenne 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.	82
4.28 Évolution de la différence de résultats entre deux itérations de <i>clustering</i> . Les évolutions moyenne 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.	83
4.29 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.	85
4.30 Évolutions 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 prenant en compte différents taux d'erreurs d'annotations entre 0 et 50%.	86

Liste des tableaux

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.	27
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.	34
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.	35
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.	37
4.5	extrait de l'évolution de l'analyse linguistique du <i>cluster</i> ...	77
4.6	Détails ...	78

Liste des tableaux

Liste des algorithmes

3.1	Description en pseudo-code de la méthode d'annotation proposée employant le clustering interactif	11
4.1	Description en pseudo-code du protocole expérimental de l'étude de convergence du <i>clustering</i> interactif vers une vérité terrain pré-établie.	25
4.2	Description en pseudo-code du protocole expérimental de l'étude d'optimisation de la convergence du <i>clustering</i> interactif vers une vérité terrain pré-établie. . . .	31
4.3	Description en pseudo-code du protocole expérimental de l'étude du temps d'annotation d'un lot de contraintes par un expert métier.	43
4.4	Description en pseudo-code du protocole expérimental de l'étude du temps d'exécution des algorithmes du <i>clustering</i> interactif	52
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 <i>clustering</i> interactif.	61
4.6	Description en pseudo-code du protocole expérimental de l'étude de vérification manuelle de la valeur métier d'une base d'apprentissage.	71
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.	75
4.8	Description en pseudo-code du protocole expérimental de l'étude d'un résumé automatique des <i>clusters</i> à l'aide d'un modèle de langue pour vérifier la valeur métier d'une base d'apprentissage.	79

LISTE DES ALGORITHMES

Liste de codes

3.1	Jeu exemple pour présenter notre implémentation du clustering interactif.	13
3.2	Démonstration de notre implémentation du prétraitement et de la vectorisation sur le jeu d'exemple.	14
3.3	Démonstration de notre implémentation de gestion des contraintes sur le jeu d'exemple.	16
3.4	Démonstration de notre implémentation du clustering sous contraintes sur le jeu d'exemple.	17
3.5	Démonstration de notre implémentation de l'échantillonnage sur le jeu d'exemple.	19

LISTE DE CODES

Glossaire

clustering !!TODO!!.

Glossaire

Index

chatbot, 4
classification, 4
ner, 4
clustering, 7
 affinity propagation, 91
 dbSCAN, 91
 hierarchique, 91
 kmeans, 91
 spectral, 91

vmeasure, 92

