

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
- ...

*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	24
4.2	Évaluation de l'hypothèse d'efficience	29
4.2.1	Étude d'optimisation des paramètres de convergence	29
4.3	Évaluation de l'hypothèse sur les coûts	39
4.3.1	Étude du temps d'annotation par un expert métier	39
4.3.2	Étude du temps de calcul des algorithmes	42
4.3.3	Étude du nombre de contraintes nécessaires	51
4.3.4	Estimation du temps total d'un projet d'annotation	51
4.4	Hypothèse de pertinence	53
4.4.1	Étude de la cohérence statistique de la base d'apprentissage en cours de construction	53
4.4.2	Étude de la pertinence sémantique de la base d'apprentissage en cours de construction	53
4.5	Hypothèse de rentabilité : « <i>quel gain à chaque itération ?</i> »	55
4.5.1	Étude d'estimation des cas d'arrêts de la méthode	55
4.6	Hypothèse de robustesse : « <i>quelle influence d'une erreur ?</i> »	56
4.6.1	Étude de simulation d'erreurs d'annotations	56
4.6.2	Étude d'annotation avec des paradigmes différents	56
4.7	Autres études à réaliser	58
4.7.1	Choix du nombre de clusters ==> problème de recherche complexe .	58

4.7.2	Impact d'un modèle de langage ==> nécessite de nombreuses données spécifiques au domaine	58
4.7.3	Paradigme d'annotation (intention vs dialogue) ==> problème d'UX + objectif métier	58
4.7.4	(et plein d'autres que j'ajouterai au fur et à mesure de ma rédaction)	58

5

Conclusion

5.1	Rappel de la problématique ??	59
5.2	Avantage et limites de la méthodes ??	59
5.3	Ouverture ??	59

Annexes

A

Annexe théorique

A.1	Les algorithmes de clustering	61
A.1.1	Kmeans	61
A.1.2	Hierarchique	61
A.1.3	Spectral	61
A.1.4	DBScan	61
A.1.5	Affinity Propagation	61
A.2	Evaluation d'une clustering	62
A.2.1	Homogénéité – Complétude – Vmeasure	62
A.2.2	FMC	62

B

Annexe technique

B.1	package pypi interactive-clustering	63
B.2	package pypi interactive-clustering-gui	63
B.3	package pypi features-maximization-metrics	63
B.4	experimentations jupyter notebook	63

C

Annexe des jeux de données

C.1	french bank cards	65
C.2	DNA press title	65

LISTE DE CODES

Bibliographie	67
Liste des TODOs	69
Liste des figures	73
Liste des tableaux	77
Liste des algorithmes	79
Liste de codes	81
Glossaire	83
Index	85

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 détecté 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 ;
- 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,

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)

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.

cf. partie étude

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.

citation

Sur la base de cet échantillon, l'expert peut entamer son étape d'**annotation de contraintes** (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

description technique plus tard ? ref subsection :3.3.4

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 comparaison entre deux modélisations thématiques d'un même jeu de données ;
3. `cognitivefactory-interactive-clustering-gui`, encapsulant les algorithmes précédents et intégrant la logique de la méthode dans une interface graphique.

Pour les sections suivantes, nous suivrons l'exemple suivant (cf. Code 3.1) pour présenter nos implémentations.

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

1. thématique : *crédit* vs. *assurance*; *sport* vs. *culture*, ...
2. action : *souscrire* vs. *résilier*; *activer* vs. *désactiver*; *s'informer* vs. *réaliser*, ...
3. besoin : *souscrire un crédit* vs. *souscrire une assurance*; *s'informer en sport* vs. *s'informer en culture*, ...

```

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 (*stopwords*),
citation
- la conversion des mots vers leur forme racine (*lemmatisation*)
citation
- et la suppression des mots en fonction de leur profondeur dans l'arbre de dépendance syntaxique(*dependency parsing*).
citation

Ces traitements sont réalisés en bénéficiant des fonctionnalités mises à disposition d'un modèle de langue de type SpaCy, avec par défaut l'utilisation du modèle `fr-core-news-md`.
citation

Pour nos études, nous définissons quatre niveaux de prétraitements facilement identifiables :
citation

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`.

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

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 :

- citation 1. **TF-IDF**, utilisant la fréquence d'occurrence des mots pour représenter une phrase, et noté `vect.tfidf` pour nos études ;
- citation 2. **SpaCy**, 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 """
25
26 # Vectorisation des données.
27 dict_of_vectors = vectorize(
28     dict_of_texts=dict_of_preprocess_texts,
29     vectorizer_type="tfidf",
30 )
```

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

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 :

- 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. Fig. 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 .

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

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

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.

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 # Création du gestionnaire de contraintes.

```

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

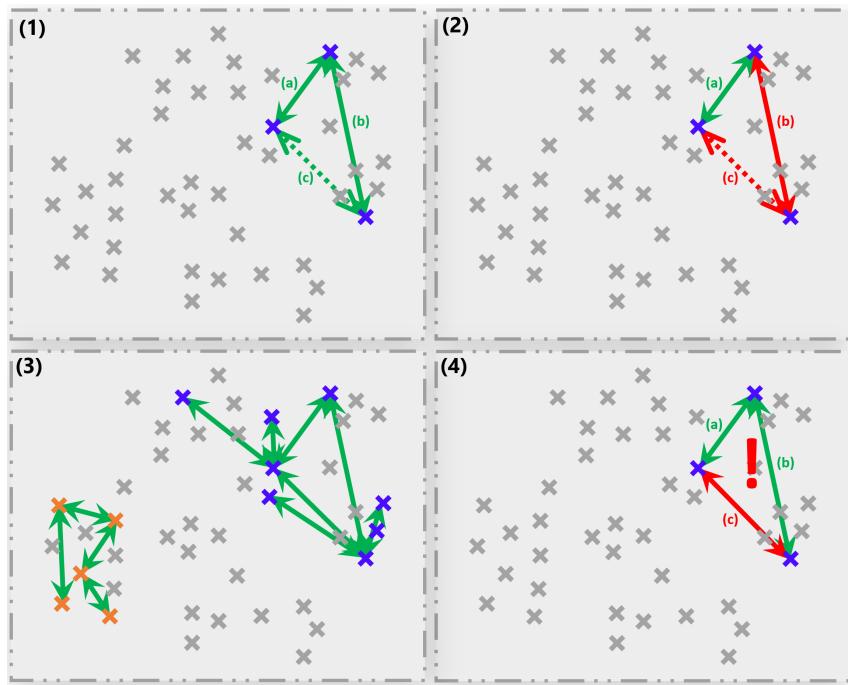


FIGURE 3.1 – Exemples des propriétés de transitivité des contraintes MUST-LINK (flèches vertes) et CANNOT-LINK (flèches rouges). (1) et (2) représente les possibilités de déduction d'une contrainte ((c)) en fonction des deux autres ((a) et (b)). (3) représente deux composants connexes définis par la 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)).

```

5 constraints_manager = managing_factory(
6     manager="binary",
7     list_of_data_IDs = list(dict_of_texts.keys()), # ["0", "1", "2",
8     "3", "4", "5", ... , "N"]
9 )
10 # Ajout de contraintes.
11 constraints_manager.add_constraint(
12     data_ID1="0", # "Comment signaler un vol de carte bancaire ?"
13     data_ID2="1", # "J'ai égaré ma carte bancaire, que faire ?"
14     constraint_type="MUST_LINK",
15 )
16 constraints_manager.add_constraint(
17     data_ID1="3", # "Le distributeur a avalé ma carte !"
18     data_ID2="4", # "En retirant de l'argent, le GAB a gardé ma
19     # carte ..."
20     constraint_type="MUST_LINK",
21 )
22 constraints_manager.add_constraint(
23     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.
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*, noté `clust.kmeans.cop`, et sa version *MPC*, noté `clust.kmeans.mpc`;
2. **DBscan**, dans sa version *C-DBScan*, noté `clust.cdbscan`;
3. **Hiérarchique**, dans sa version *xxx*, 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*, noté `clust.spec`;
5. **Propagation par affinité**, dans sa version *xxx*, 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*. Pour plus de détails, les descriptions en pseudo-code de ces algorithmes sont disponibles en annexe. Enfin, un exemple d'utilisation ce module est consultable dans Code 3.4.

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",

```

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

```

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

```

1 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.

citation
+ foot-note

travaux
TPS en
annexe ?

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 :

- 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.

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

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

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

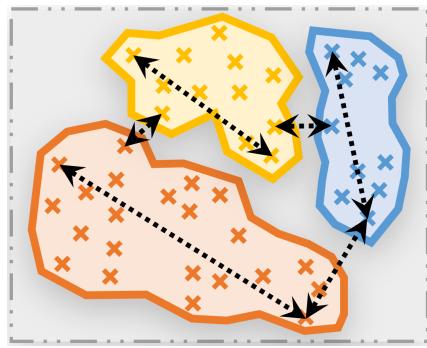


FIGURE 3.2 – Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (`samp.farhest.same`), et de données issues de clusters différents et étant les plus proches les unes des autres (`samp.closest.diff`).

Une classe abstraite définit les prérequis des algorithmes implémentés (avoir une méthode `sample`) et une *factory* est disponible pour instancier rapidement un objet d'échantillonnage. Un exemple d'utilisation ce module est consultable dans Code 3.5.

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
3     sampling_factory
4
5 # Initialiser un objet d'échantillonnage.
6 sampler = sampling_factory(
7     algorithm="random",
8     random_seed=42,
9 )
10
11 # Run sampling.
12 selection = sampler.sample(
13     constraints_manager=constraints_manager,
14     nb_to_select=2,
15     clustering_result=clustering_result, # optionnel pour "random"

```

```
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 """
25 """
```

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 raffinage 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.

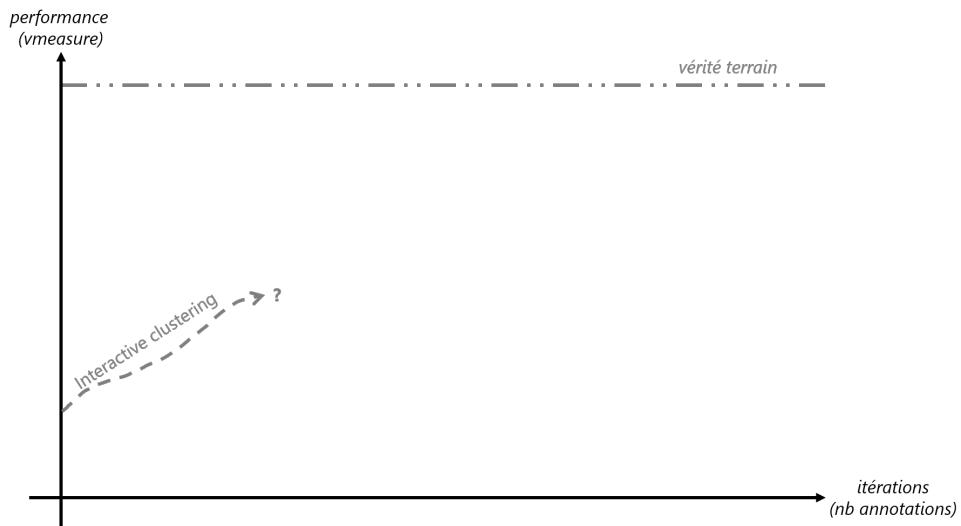


FIGURE 4.1 – Illustration des études réalisées sur le *clustering* interactif (*étape 0/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre 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 et/ou R, sont disponibles dans SCHILD, 2021. 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	24
4.2	Évaluation de l'hypothèse d'efficience	29
4.2.1	Étude d'optimisation des paramètres de convergence	29
4.3	Évaluation de l'hypothèse sur les coûts	39

4.3.1	Étude du temps d'annotation par un expert métier	39
4.3.2	Étude du temps de calcul des algorithmes	42
4.3.3	Étude du nombre de contraintes nécessaires	51
4.3.4	Estimation du temps total d'un projet d'annotation	51
4.4	Hypothèse de pertinence	53
4.4.1	Étude de la cohérence statistique de la base d'apprentissage en cours de construction	53
4.4.2	Étude de la pertinence sémantique de la base d'apprentissage en cours de construction	53
4.5	Hypothèse de rentabilité : « <i>quel gain à chaque itération ?</i> » . .	55
4.5.1	Étude d'estimation des cas d'arrêts de la méthode	55
4.6	Hypothèse de robustesse : « <i>quelle influence d'une erreur ?</i> » . .	56
4.6.1	Étude de simulation d'erreurs d'annotations	56
4.6.2	Étude d'annotation avec des paradigmes différents	56
4.7	Autres études à réaliser	58
4.7.1	Choix du nombre de clusters ==> problème de recherche complexe .	58
4.7.2	Impact d'un modèle de langage ==> nécessite de nombreuses données spécifiques au domaine	58
4.7.3	Paradigme d'annotation (intention vs dialogue) ==> problème d'UX + objectif métier	58
4.7.4	(et plein d'autres que j'ajouterai au fur et à mesure de ma rédaction)	58

4.1 Évaluation de l'hypothèse d'efficacité

Nous aimerais 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. »

Afin de vérifier cette hypothèse, nous mettrons 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 critiquerons l'évolution de la nouvelle base d'apprentissage obtenue et sa similitude avec la base d'apprentissage initiale.

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

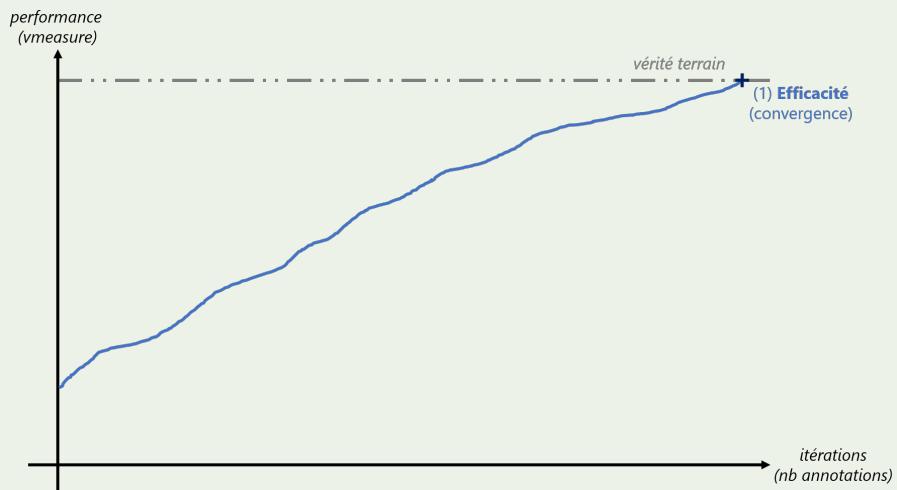


FIGURE 4.2 – Illustration des études réalisées sur le *clustering* interactif (étape 1/6) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre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

4.1.1 Étude de convergence vers une vérité terrain pré-établie

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 : simuler l'annotation d'une base d'apprentissage

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.

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. 4.1.

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

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

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

Pour cette étude, nous essayons une tentative pour chaque combinaison de 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 : **absent** (noté `prep.no`), **simple** (noté `prep.simple`), **avec lemmatisation** (noté `prep.lemma`) et **avec filtres** (noté `prep.filter`);

référence,
lien vers
AN-
NEXE,
+ des-
cription
condi-
tions de
création
du JDD

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.farhtest.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 fixé à 50.

Il y a donc 192 combinaisons testées, 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 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 intéresserons à l'évolution de la **v-measure** 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.

Pour information : Les scripts de l'expérience (*notebooks Python*) sont disponibles dans un dossier dédié de SCHILD, 2021.

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.

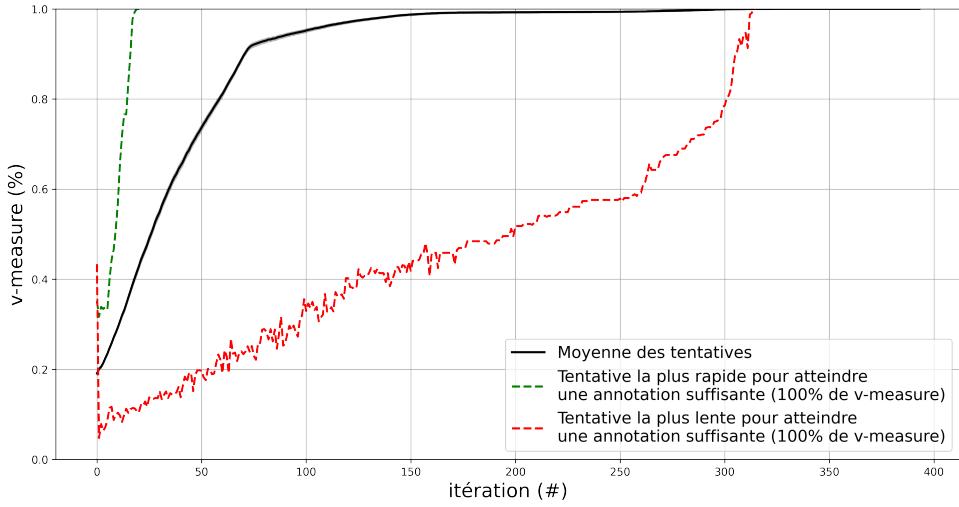


FIGURE 4.3 – Évolution de la moyenne de la **v-measure** entre un résultat obtenu et la vérité terrain en fonction du nombre d’itération de la méthode de *clustering* interactif, moyenne réalisée itération par itération sur l’ensemble des tentatives. Représentation des tentatives ayant été les plus rapides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

Annotations		Performances (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

La première et principale conclusion de cette étude concerne la preuve que la méthode est efficace. En effet, les différentes simulations ont bien convergé vers la vérité terrain (atteinte de l’asymptote à 100% de **v-measure**), montrant qu’il est possible pour un expert métier de créer une base d’apprentissage à l’aide d’une méthodologie d’annotation basée sur le *clustering* inter-

actif.

Cette découverte permet de confirmer plusieurs espoirs portés sur la méthode.

Tout d'abord, la vérité terrain a été retrouvée sans formaliser concrètement la structure de données. Là où une annotation par label aurait requis au préalable une définition des catégories possibles pour les données à étiqueter, 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 ?*

De plus, 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).

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 à un annotateur, un tel volume représente tout de même une grande quantité de travail.

Commentaire de Gautier
22/05/2023 (A DÉ-TAILLER)
Oui, complètement d'accord ici, mais en fait ça va plus loin que ça non ?
Déjà, on a une quantité de ressources allouées à la tâche en effet plus fabile (car choisir entre "simi-

cela peut décourager les experts métiers en début de projet, surtout pour des projets ayant des jeux de données de plus grandes tailles. Toutefois, les résultats obtenus montrent une forte dispersion du nombre d'itérations nécessaire, et certaines tentatives ont été bien plus efficientes dans l'utilisation de leurs contraintes. La tentative la plus rapide a convergé à l'itération 19, soit 950 contraintes, ce qui est un volume d'annotation bien plus abordable ! On peut donc espérer trouver un paramétrage optimal de la méthode permettant de diminuer significativement le nombre moyen de contraintes nécessaires afin d'obtenir une base d'apprentissage exploitable avec un volume d'annotations acceptable. Cet aspect fait l'objet de l'étude décrite dans la section 4.2 (hypothèse d'efficience).

D'autre part, le choix d'annoter toutes les contraintes possibles sur les données (**annotation exhaustive**) n'est pas forcément judicieux. En effet, si nous nous référons à 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 90% de **v-measure** (quitte à affiner le résultat manuellement pour combler la "perte" moyenne de 10% de **v-measure**). Cet aspect sera ajouté à l'objectif de l'étude décrite dans la section 4.2 (hypothèse d'efficience).

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

4.2 Évaluation de l'hypothèse d'efficience

Suite à la validation de l'hypothèse d'efficacité (convergence de la méthode, cf. section 4.1), nous aimerions 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 du prétraitement des données, de la vectorisation des données, de l'échantillonnage des contraintes à annoter et du *clustering* sous contraintes. »

Afin de vérifier cette hypothèse, nous mettrons 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 réaliserons l'analyse statistique de la taille d'effet de différents paramètres sur la vitesse de convergence du *clustering* itératif.

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.

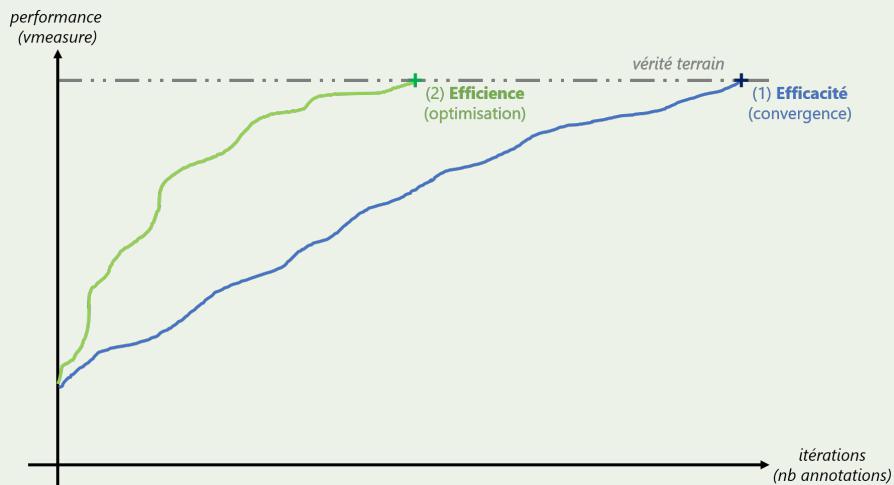


FIGURE 4.4 – Illustration des études réalisées sur le *clustering* interactif (*étape 2/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre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

4.2.1 Étude d'optimisation des paramètres de convergence

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).¹⁰

Protocole expérimental : analyser la taille d'effet des paramètres d'implémentation 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.

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.

Pour résumer le protocole expérimental que nous décrivons c-dessous, vous pouvez vous référer au pseudo-code décrit dans Alg. 4.2.

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

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

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

Sortie(s): meilleures 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 avons toujours 192 combinaisons testées, 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 intéresserons à l'évolution de la **v-measure** entre la vérité terrain et notre segmentation des données obtenue, et nous affinerons 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** entre le résultat obtenu et la vérité terrain,

- c'est-à-dire un état de semi-parcours vers une convergence totale¹¹ ;
2. le cas d'une **annotation suffisante**, correspondant au nombre d'itérations nécessaires à la méthode pour avoir 100% de **v-measure** entre le résultat obtenu et la vérité terrain, 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 métier.

Enfin, nous utiliserons une ANOVA à mesures répétées afin de déterminer l'effet des paramètres de notre implémentation sur le nombre d'annotations requis pour converger vers la vérité terrain.

i Pour information : Ces analyses sont réalisées à l'aide du logiciel R, et le test de Tukey (HSD) est utilisé pour les comparaisons post-hoc. Les scripts de l'expérience (*notebooks Python*) sont disponibles dans un dossier dédié de SCHILD, 2021.

citation

Résultats obtenus

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

- Les tentatives les plus rapides furent celles avec un prétraitement des données `prep.no` ou `prep.simple` ou `prep.lemma`, une vectorisation des données `vect.tfidf`, un clustering sous contraintes `clust.hier.sing`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 11 itérations, soit 550 annotations, dont 299 (respectivement 304 et 281) contraintes MUST-LINK.
- Les tentatives les plus lentes furent celles avec un prétraitement des données `prep.no`, une vectorisation des données `vect.tfidf`, un clustering sous contraintes `clust.spec`, et un échantillonnage de contraintes `samp.farthest.same`. Ces tentatives ont requis 315 itérations, soit 15 750 annotations, dont 1 032 contraintes MUST-LINK.

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

Pour obtenir une **annotation suffisante** (*atteindre une v-measure de 100%*), la moyenne des itérations est de 76.29 (min : 19, max : 328, écart-type : 46.44), soit une moyenne de 3 801.19

11. Le seuil de 90% a été choisi au cours de l'étude de convergence (cf. hypothèse d'efficacité, section 4.1).

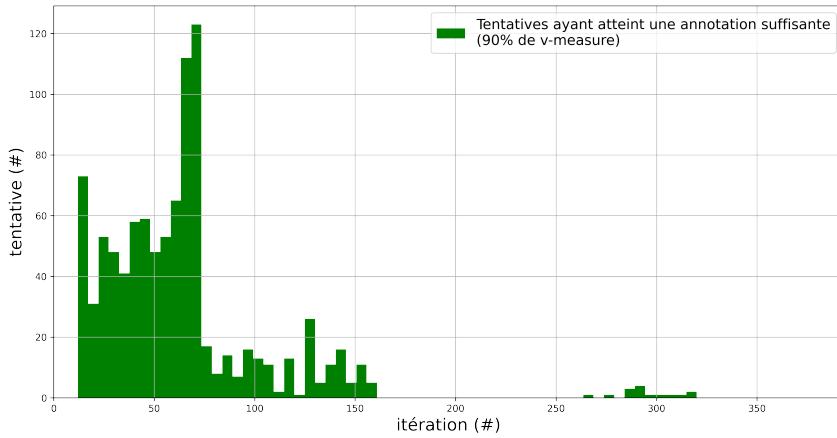


FIGURE 4.5 – Répartition des tentatives en fonction de l’itération de la méthode à laquelle elles atteignent le seuil d’une annotation partielle, c'est-à-dire l’itération à laquelle elles parviennent à 90% de **v-measure** entre un résultat obtenu et la vérité terrain. L’histogramme est réduit à 60 pics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

Description des facteurs analysés		Description statistique			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.farhtest.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.

annotations (min : 950, max : 16 400, écart-type : 2 314.91). La figure 4.6 représente la répartition de ces itérations au cours des différentes tentatives. On peut noter les deux cas intéressants suivants :

- Les tentatives les plus rapides furent celles avec un prétraitement des données `prep.simple`, une vectorisation des données `vect.tfidf`, un clustering sous contraintes `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis 19 itérations, soit 950 annotations, dont 638 (respectivement 641) contraintes MUST-LINK.
- Les tentatives les plus lentes furent celles avec un prétraitement des données `prep.no`, une vectorisation des données `vect.tfidf`, un clustering sous contraintes `clust.spec`, et un échantillonnage de contraintes `samp.farthest.same`. Ces tentatives ont requis 394 itérations, soit 16 400 annotations, dont 1 309 contraintes MUST-LINK.

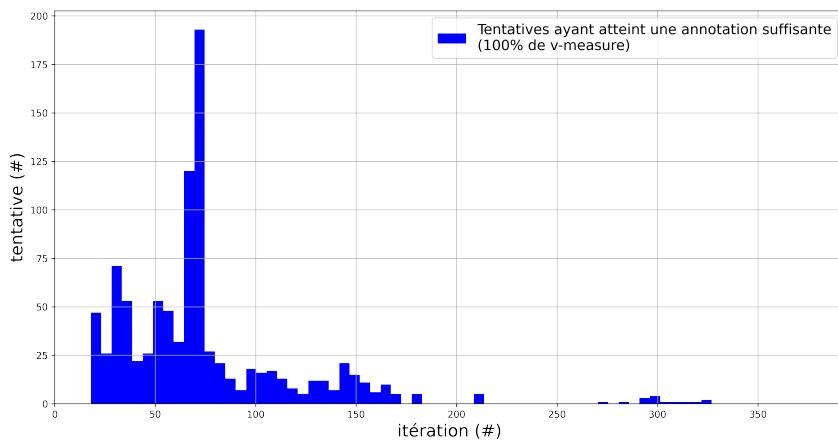


FIGURE 4.6 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation suffisante, c'est-à-dire l'itération à laquelle elles parviennent à 100% de **v-measure** entre un résultat obtenu et la vérité terrain. L'histogramme est réduit à 60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

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 contraintes `clust.hier.comp` ou `clust.hier.wards`, 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.

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

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

- Les tentatives les plus lentes furent celles avec un prétraitement des données `prep.simple`, une vectorisation des données `vect.frcorenewsmd`, un clustering sous contraintes `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.

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).

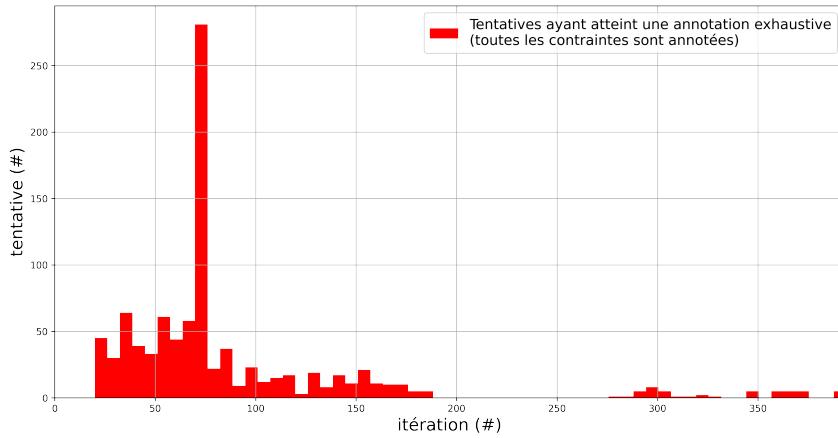


FIGURE 4.7 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation exhaustive, c'est-à-dire l'itération à laquelle toutes les contraintes possibles entre les données ont été annotées. L'histogramme est réduit à 60 pics pour simplifier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

Description des facteurs analysés		Description statistique			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'annotations.

Chapitre 4. Étude de la méthode

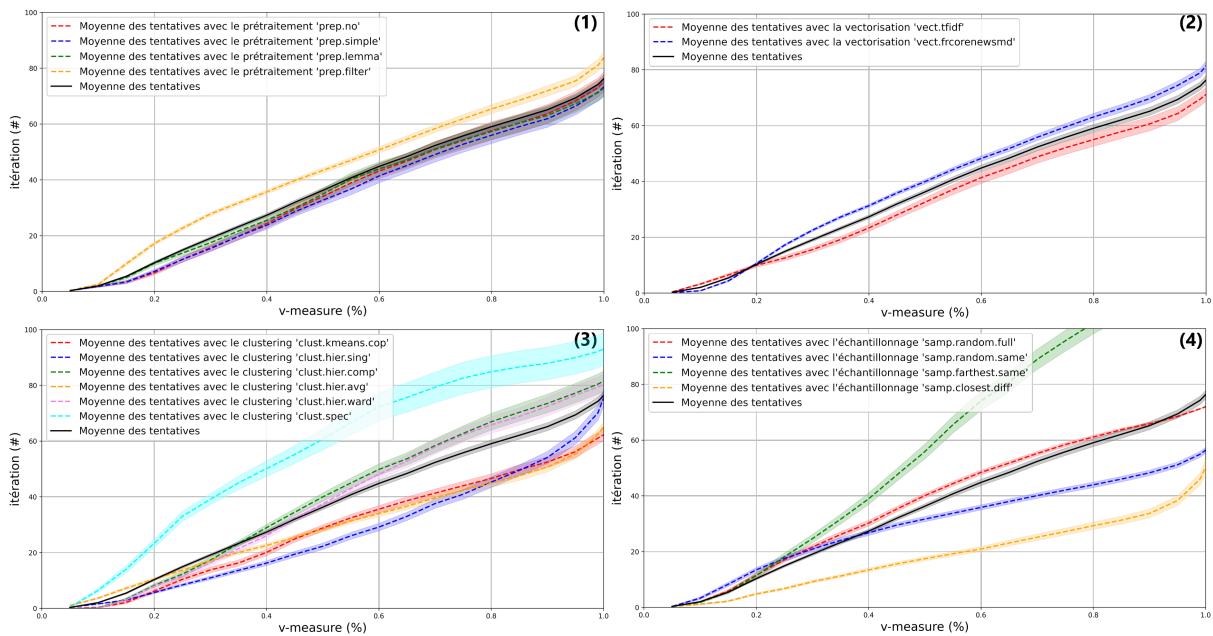


FIGURE 4.8 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de *clustering* interactif pour obtenir un seuil défini de **v-measure** entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes valeurs que peuvent prendre les facteurs analysés et affichées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

tion. Pour trouver si une telle implémentation existe et quels en sont les paramètres optimaux, nous avons analysé l’impact de différentes paramétrages sur les tâches principales de la méthode (**prétraitement**, **vectorisation**, **clustering sous contraintes**, **échantillonnage**) en nous basant sur des simulations d’annotation d’un jeu de données.

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

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

1. pour une **annotation partielle** (90% de **v-measure**), le meilleur paramétrage moyen est constitué du prétraitement simple (**prep.simple**), de la vectorisation TF-IDF (**vect.tfidf**), du clustering hiérarchique à lien moyen (**clust.hier.avg**) et de l’échantillonnage des données les plus proches dans des clusters différents (**samp.closest.diff**). Avec ce paramé-

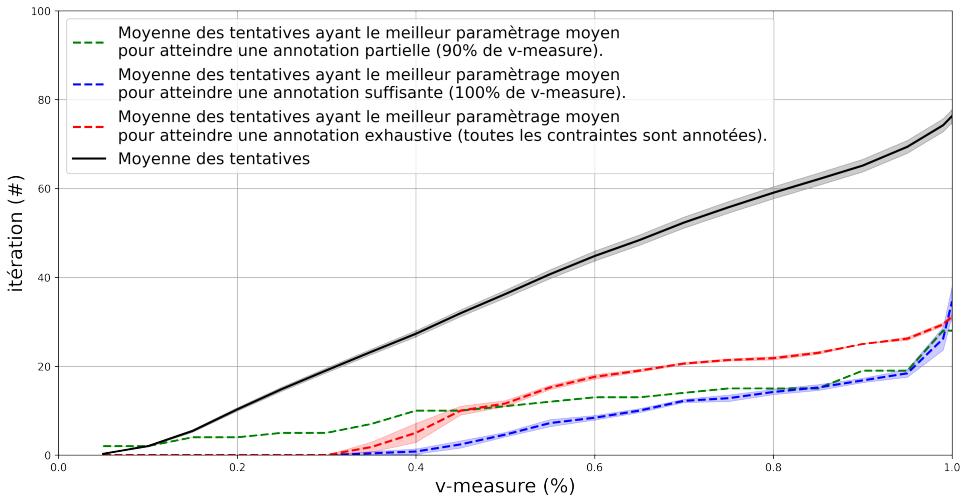


FIGURE 4.9 – Évolution des moyennes du nombre d’itérations nécessaire de la méthode de *clustering* interactif pour obtenir un seuil défini de **v-measure** entre un résultat obtenu et la vérité terrain, moyennes réalisées sur les différentes seuils d’annotations étudiés : l’annotation partielle (*atteindre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

trage, 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 certaines limites discutées dans la section 4.1 (hypothèse d’efficacité).

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.

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 ?"), nous pouvons conclure que la charge totale nécessaire à l’annotation avec une méthodologie basée sur le *clustering* interactif est comparable à celles des méthodes traditionnelles. De plus, cette méthode ne demande pas de formalisation concrète de la structure de données à annoter pour faire émerger une base d’apprentissage au cours des

itérations, donc le *clustering* interactif devient une méthode d'annotation adaptée à l'activité des annotateurs.

Néanmoins, quelques pistes sont encore à explorer pour compléter cette analyse d'efficience.

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. 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 un expert métier pour qui un score de **v-measure** n'est pas révélateur de la pertinence métier de la segmentation proposée des données. Il manque donc une stratégie d'évaluation de pertinence de la base d'apprentissage en cours de construction et de la suffisance des annotations réalisées pour faire refléter la vision de l'annotateur dans le résultat. 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

Pour compléter l'étude réalisée sur l'hypothèse d'efficience (optimisation des paramètres de convergence, cf. section 4.2), nous aimerais vérifier l'hypothèse suivante :

à compléter

❖ 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. »

Afin de vérifier cette hypothèse, nous organiserons plusieurs expériences pour simuler ou déterminer ces durées : une étude du temps d'annotation par un expert métier (cf. section 4.3.1), une étude du temps de calcul des algorithmes (cf. section 4.3.2) et une étude du nombre de contraintes nécessaires (cf. section 4.3.3). Nous conclurons l'estimation du temps total d'un projet d'annotation en section 4.3.4.

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 nombre d'itérations de la méthode (*nombre d'annotations par un expert métier*).

4.3.1 Étude du temps d'annotation par un expert métier

Protocole expérimental

Nous voulons estimer le temps nécessaire à un opérateur pour annoter un lot de contraintes. Pour cela, nous allons chronométrer plusieurs expert métiers en train d'annoter un même échan-

Chapitre 4. Étude de la méthode

tillon et modéliser le nombre de contraintes par minute ainsi que son évolution au cours de plusieurs sessions d'annotation.

⚠️ 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 devront supprimer en amont et d'un commun accord les données trop spécifiques.

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

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: pour tout annotateur faire
4:   tant que la base de contraintes n'a pas été entièrement annotée faire
5:     chronomètre : START
6:     annotation : annoter une partie des contraintes
7:     revue : revue des contraintes en conflits d'annotation
8:     chronomètre : STOP
9:     mesure : estimer la différence de chronomètre pour cette session
10:    fin tant que
11:   fin pour
12:   modélisation : entraîner un modèle linéaire généralisé du temps d'annotation
13:   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

Pour cette étude, nous procéderons 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 une collecte des titres d'articles de journaux classés par catégorie de publication (*économie, sport,...*). 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 14 catégories les plus communes. L'échantillon résultant est décrit en annexe .

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 la vitesse de convergence vers la vérité terrain), nous utilisons l'échantillonnage purement aléatoire (`samp.rand.full`).

Ensuite ...

A REDIGER : consignes annotateurs

TODO :
AN-
NEXE
JEU DE
DON-
NEES

A REDIGER : description interface

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

i Pour information : Ces analyses sont réalisées en Python à l'aide des librairies `datetime` et `statsmodels` (SEABOLD et PERKTOLD, 2010). Le projet à importer dans l'outil d'annotation ainsi que les scripts de l'expérience (*notebooks* Python) sont disponibles dans un dossier dédié de SCHILD, 2021.

citation

Résultats obtenus

A REDIGER : Description statistiques des résultats

A REDIGER : Modélisation du temps

A REDIGER : équation du temps

La figure 4.11 représente cette modélisation du temps d'annotation en comparaison avec les mesures réalisées lors de l'expérience.

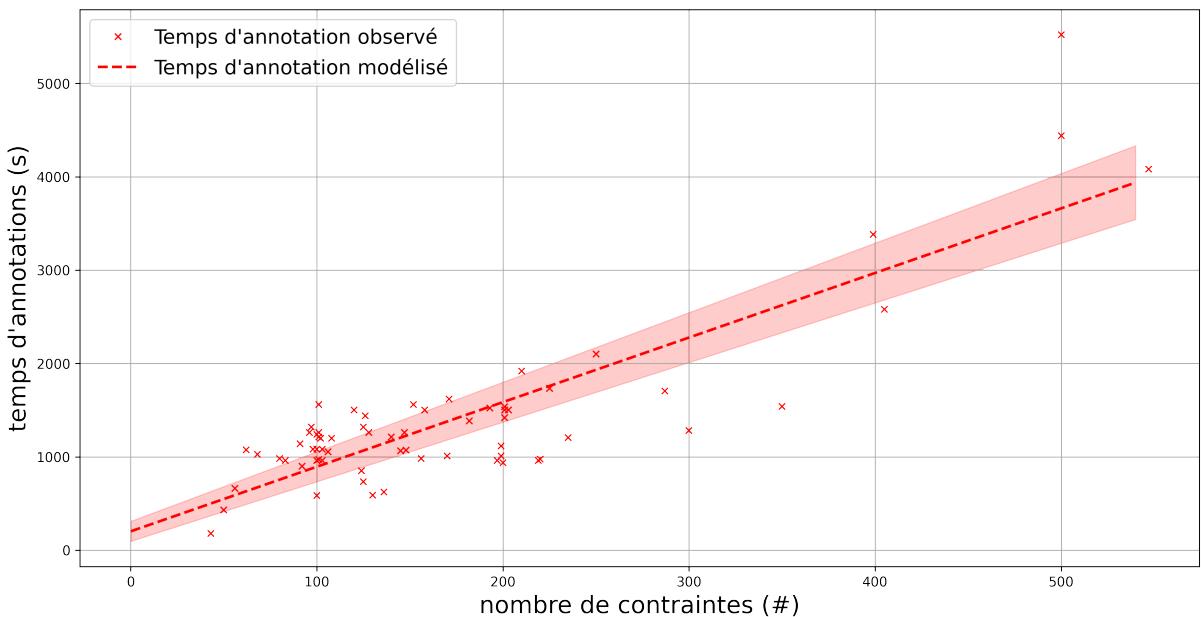


FIGURE 4.11 – Estimation du temps nécessaire (en secondes) pour annoter un lot de contraintes.

La figure 4.12 représente l'évolution de la vitesse d'annotation de quatre opérateurs (les deux plus rapides et les deux plus lents). Ces données sont l'objet d'une étude de cas dans la discussion ci-dessous.

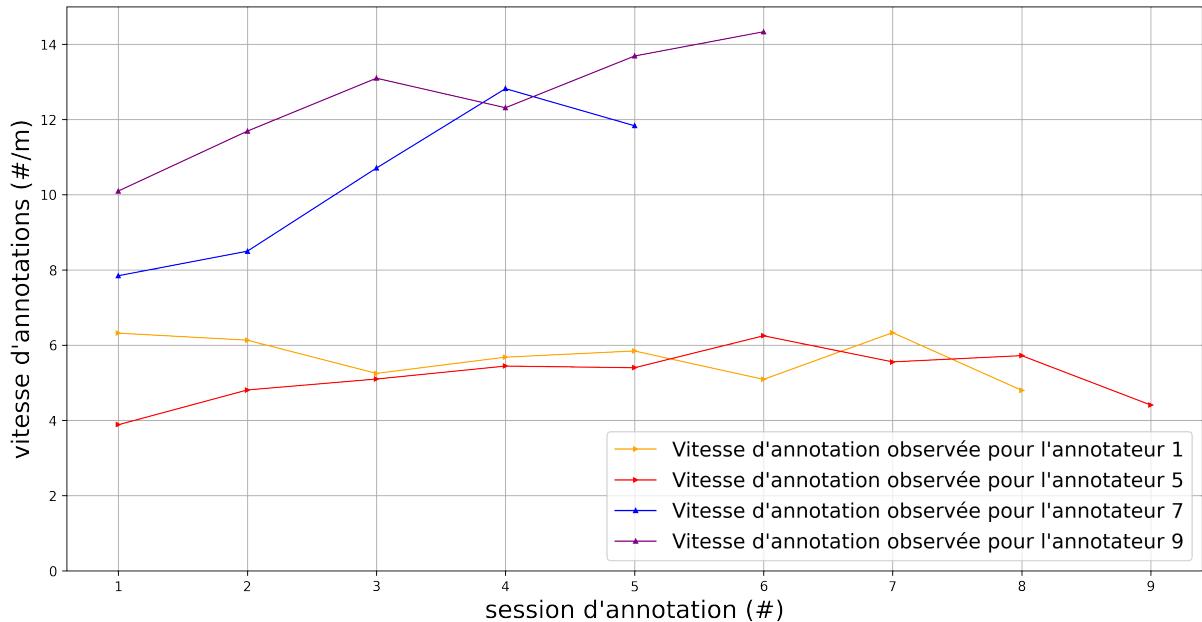


FIGURE 4.12 – Etude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations

Discussion

A REDIGER : Discussion temps moyen

A REDIGER : Discussion vitesse d'annotation

4.3.2 Étude du temps de calcul des algorithmes

Protocole expérimental : estimer le temps de calcul des algorithmes du *clustering* interactif

Pour confirmer le choix du paramétrage pour une convergence optimale (cf. hypothèse d'efficience en section 4.2), nous voulons étudier le temps de calcul de chaque algorithme intervenant dans notre implémentation du *clustering* interactif. Pour cela, nous allons chronométrer plusieurs exécutions de ces algorithmes avec différents arguments d'entrée (la taille du jeu de données, le nombre de clusters et le nombre de contraintes annotées, ...) et modéliser le temps de calcul résultant en fonction de ces paramètres.

⚠️ Attention : Pour utiliser des jeux de données de tailles différentes tout en maîtrisant leur contenu, nous avons dupliqués aléatoirement des données en générant des fautes de frappes. Pour cette étude, nous faisons l'hypothèse que cela n'a pas d'impact majeur sur le temps d'exécution des différents algorithmes.

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

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

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

```

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

```

Pour cette étude, 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`) ;
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 annotées** (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 annotées** (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'entrée¹²),

12. Les arguments d'entrée peuvent influencer le contexte d'utilisation à partir du du : nombre de données, nombre de contraintes annotées, nombre de contraintes à sélectionner, nombre de *clusters* existant, nombre de *clusters* à trouver

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* est utilisée) et nous nous limitons aux variables responsables d'un maximum de la variance des mesures (la méthode d'*Elbow* est utilisée pour choisir les facteurs pertinents). Sur cette base, nous entraînons un modèle linéaire généralisé (*GLM*) pour représenter le temps d'exécution moyen de l'algorithme : ce modèle sera caractérisé par le coefficient de détermination généralisé R^2 de *Cox et Snel*, la log-vraisemblance llf et la log-vraisemblance llf_null du modèle *null*. Pour finir, nous discuterons des valeurs des coefficients obtenus sur l'impact du temps d'exécution.

i Pour information : Ces analyses sont réalisées en Python à l'aide des librairies `datetime` et `statsmodels` (SEABOLD et PERKTOLD, 2010). Les scripts de l'expérience (*notebooks* Python) sont disponibles dans un dossier dédié de SCHILD, 2021.

citation

Résultats obtenus

En ce qui concerne la tâche de **prétraitement**, une première analyse montre que les modélisations des trois implémentations sont similaires ($p\text{-valeur} > 0.980$). Nous ferons 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 ($R^2 : > 0.999$, $llf : -379.35$, $llf_null : -1\ 353.98$) nous permet de déduire l'équation suivante¹³ :

$$time(prep) \simeq 8.39 \cdot 10^{-1} + 6.31 \cdot 10^{-3} \cdot dataset_size \quad (4.1)$$

ref annexe

La figure 4.13 représente cette modélisation du temps de calcul des algorithmes de prétraitements en comparaison avec les mesures réalisées lors de l'expérience.

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

Pour les algorithmes de vectorisation `vect.tfidf`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size` ($r : 0.977$). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, $llf : 262.35$, $llf_null : 70.04$) nous permet de déduire l'équation suivante¹⁴ :

$$time(vect.tfidf) \simeq -1.40 \cdot 10^{-2} + 9.54 \cdot 10^{-5} \cdot dataset_size \quad (4.2)$$

ref annexe

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 ($R^2 : > 0.999$, $llf : -186.80$, $llf_null : -399.39$) nous permet de déduire l'équation suivante¹⁵ :

ref annexe

13. Cette déduction est en accord avec la complexité théorique de l'algorithme qui est en $\mathcal{O}(dataset_size)$.

14. Cette déduction est en accord avec la complexité théorique de l'algorithme qui est en $\mathcal{O}(dataset_size)$.

15. Cette déduction est en accord avec la complexité théorique de l'algorithme qui est en $\mathcal{O}(dataset_size)$.

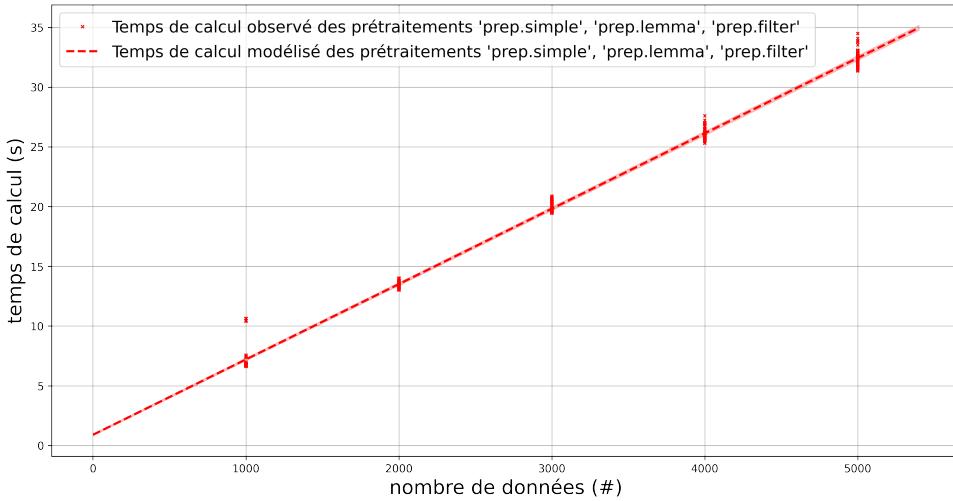


FIGURE 4.13 – Estimation du temps nécessaire (en secondes) pour effectuer une tâche de **prétraitement** en fonction du nombre de données à traiter. Les paramétrages `prep.simple`, `prep.lemma` et `prep.filter` ayant des temps de calculs similaires, leurs modélisations n’ont pas été séparées.

$$time(vect.frcorenewsmd) \simeq 1.89 + 4.11 \cdot 10^{-3} \cdot \text{dataset_size} \quad (4.3)$$

La figure 4.14 représente ces modélisations de temps de calcul des algorithmes de vectorisation en comparaison avec les mesures réalisées lors de l’expérience.

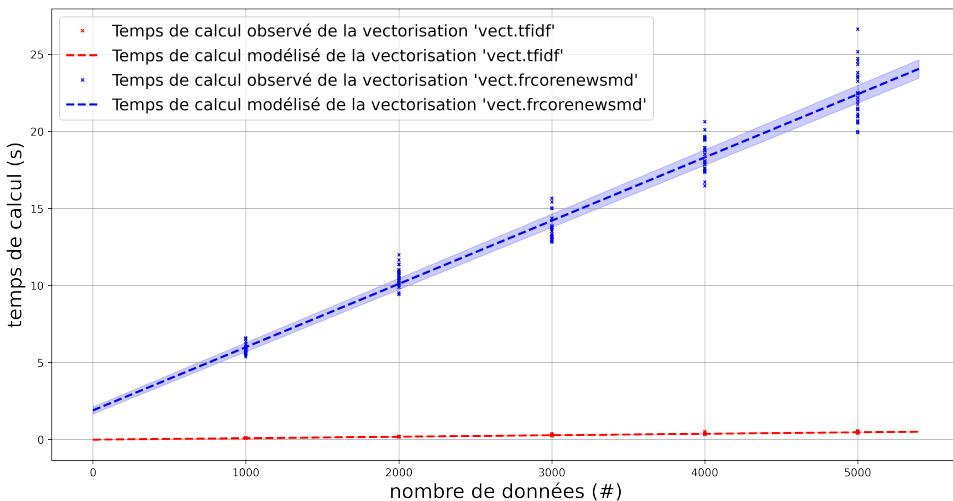


FIGURE 4.14 – Estimation du temps nécessaire (en secondes) pour effectuer une tâche de **vectorisation** en fonction du nombre de données à traiter.

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

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

4.3. Évaluation de l'hypothèse sur les coûts

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 ($R^2 : 0.904$, `llf` : $-9.20 \cdot 10^4$, `llf_null` : $-1.00 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$time(clust.kmeans.cop) \simeq -2.40 \cdot 10^2 + 2.11 \cdot 10^{-1} \cdot dataset_size \quad (4.4)$$

ref annexe

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 ($R^2 : > 0.999$, `llf` : $-5.38 \cdot 10^4$, `llf_null` : $-6.10 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$time(clust.hier.sing) \simeq -8.87 \cdot 10^2 + 6.37 \cdot 10^{-4} \cdot dataset_size^2 \quad (4.5)$$

ref annexe

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 ($R^2 : > 0.999$, `llf` : $-5.40 \cdot 10^4$, `llf_null` : $-6.11 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$time(clust.hier.comp) \simeq -9.25 \cdot 10^2 + 6.42 \cdot 10^{-4} \cdot dataset_size^2 \quad (4.6)$$

ref annexe

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 ($R^2 : 0.942$, `llf` : $-5.82 \cdot 10^4$, `llf_null` : $-6.45 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$time(clust.hier.avg) \simeq -1.10 \cdot 10^3 + 1.02 \cdot 10^{-3} \cdot dataset_size^2 \quad (4.7)$$

ref annexe

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 ($R^2 : > 0.999$, `llf` : $-5.39 \cdot 10^4$, `llf_null` : $-6.14 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$time(clust.hier.ward) \simeq -9.79 \cdot 10^2 + 6.82 \cdot 10^{-4} \cdot dataset_size^2 \quad (4.8)$$

ref annexe

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 ($R^2 : 0.534$, `llf` : $-7.88 \cdot 10^5$, `llf_null` : $-8.27 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$time(clust.spec) \simeq 11.00 + 7.56 \cdot 10^{-6} \cdot dataset_size^2 \quad (4.9)$$

ref annexe

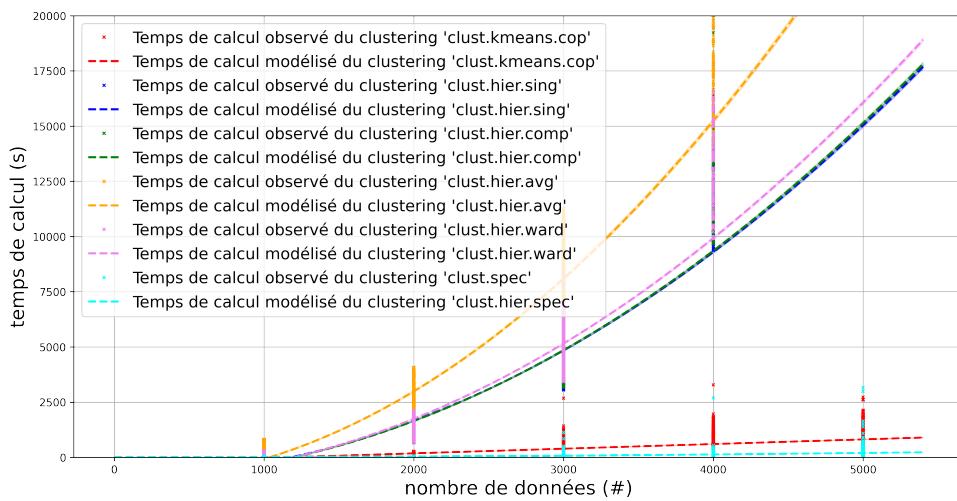


FIGURE 4.15 – Estimation du temps nécessaire (en secondes) pour effectuer une tâche de **clustering** en fonction du nombre de données à traiter.

La figure 4.15 représente ces modélisations de temps de calcul des algorithmes de *clustering* en comparaison avec les mesures réalisées lors de l’expérience.

En ce qui concerne la tâche d'**échantillonnage de contraintes**, une première analyse montre que les modélisations des quatre implémentations sont différentiables ($p\text{-valeur} : < 10^{-3}$). Nous ferons 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` : $-4.33 \cdot 10^4$, `llf_null` : $-1.17 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$time(samp.rand.full) \simeq -4.78 \cdot 10^{-1} + 8.47 \cdot 10^{-7} \cdot dataset_size^2 \quad (4.10)$$

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` : $-3.17 \cdot 10^4$, `llf_null` : $-6.84 \cdot 10^4$) nous permet de déduire l'équation suivante :

$$time(samp.rand.same) \simeq -1.36 \cdot 10^{-1} + 1.92 \cdot 10^{-7} \cdot dataset_size^2 \quad (4.11)$$

Pour les algorithmes de l'échantillonnage de contraintes `samp.farhtest.same`, l'analyse de la corrélation des facteurs avec les mesures de temps d'exécution indique qu'une modélisation minimale et suffisante peut être réalisée à partir du facteur `dataset_size2` ($r : 0.981$). Le second facteur le plus corrélé (mais non retenu) est l'interaction `dataset_size2 · previous_nb_clusters` ($r : 0.700$). Le modèle linéaire généralisé retenu ($R^2 : > 0.999$, `llf` : $-4.52 \cdot 10^4$, `llf_null` : $-1.02 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$time(samp.farhtest.same) \simeq -2.25 \cdot 10^{-1} + 5.32 \cdot 10^{-7} \cdot dataset_size^2 \quad (4.12)$$

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_size`² ($r : 0.995$). 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` : $-5.80 \cdot 10^4$, `llf_null` : $-1.36 \cdot 10^5$) nous permet de déduire l'équation suivante :

$$time(samp.closest.diff) \simeq -6.58 \cdot 10^{-1} + 1.46 \cdot 10^{-6} \cdot dataset_size^2 \quad (4.13)$$

ref annexe

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

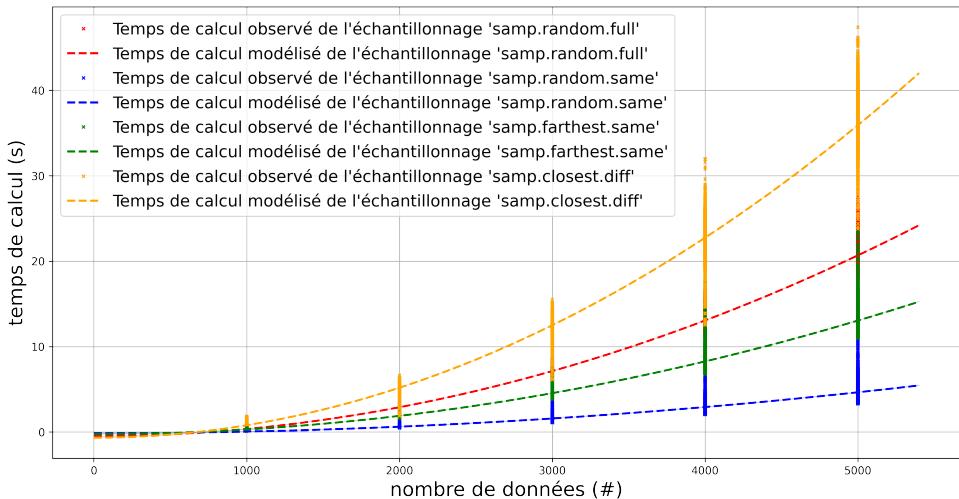


FIGURE 4.16 – Estimation du temps nécessaire (en secondes) pour effectuer une tâche d'**échantillonnage de contraintes** en fonction du nombre de données à traiter.

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 pour 5 000 données). Cette différence, visible en figure 4.15, 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).

💡 **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 4.13), de vectorisation (figure 4.14), et d'échantillonnage de contraintes (cf. figure 4.16) ont des complexités presque négligeables au regard des temps d'exécution du *clustering* (pour 5 000 données : environ 60 secondes contre près de 800 secondes pour `clust.kmeans.cop` et près de 15 000 secondes pour `clust.hier.sing`). Nous maintenons donc les paramétrages obtenus pour ces tâches en section 4.2 sans analyses complémentaires et bornons l'ensemble des ces temps de calcul par un temps constant de 60 secondes.

Pour conclure, dans l'optique d'atteindre de manière efficiente 90% de v-measure (annotation partielle) avec un coût global minimal, nous retenons l'usage du **paramétrage favori** constitué du prétraitements 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 :

$$time(parametrage.favori) \simeq -1.80 \cdot 10^2 + 2.11 \cdot 10^{-1} \cdot dataset_size \quad (4.14)$$

Pour poursuivre cette étude et estimer le coût total de la méthode, nous devons maintenant estimer le nombre moyen d'itérations nécessaires de la méthode en fonction de la taille du jeu de données à annoter (cf. section 4.3.3).

4.3.3 Étude du nombre de contraintes nécessaires

Protocole expérimental

A REDIGER :

i Pour information : Les scripts de l'expérience (*notebooks Python*) sont disponibles dans un dossier dédié de SCHILD, 2021.

Résultats obtenus

Discussion

4.3.4 Estimation du temps total d'un projet d'annotation

Protocole expérimental

A REDIGER :

16. Pour 1 000 données : environ 30 secondes ; Pour 5 000 données : environ 15 minutes.

i Pour information : Les scripts de l'expérience (*notebooks Python*) sont disponibles dans un dossier dédié de SCHILD, 2021.

Résultats obtenus

Discussion

4.4 Hypothèse de pertinence

Nous aimerais vérifier l'hypothèse suivante :

à compléter

❖ Hypothèse de pertinence ❖

« La vitesse de convergence du *clustering* interactif peut être optimisée en réglant différents paramètres. Nous étudierons l'influence du prétraitement des données, de la vectorisation des données, de l'échantillonnage des contraintes à annoter et du *clustering* sous contraintes (cf. figure 4.17. »



FIGURE 4.17 – Illustration des études réalisées sur le *clustering* interactif (étape 4/6) 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*).

4.4.1 Étude de la cohérence statistique de la base d'apprentissage en cours de construction

Protocole expérimental

Description succincte du protocole expérimental dans l'encadré d'hypothèse ?

Résultats obtenus

Discussion

4.4.2 Étude de la pertinence sémantique de la base d'apprentissage en cours de construction

Protocole expérimental

Description succincte du protocole expérimental dans l'encadré d'hypothèse ?

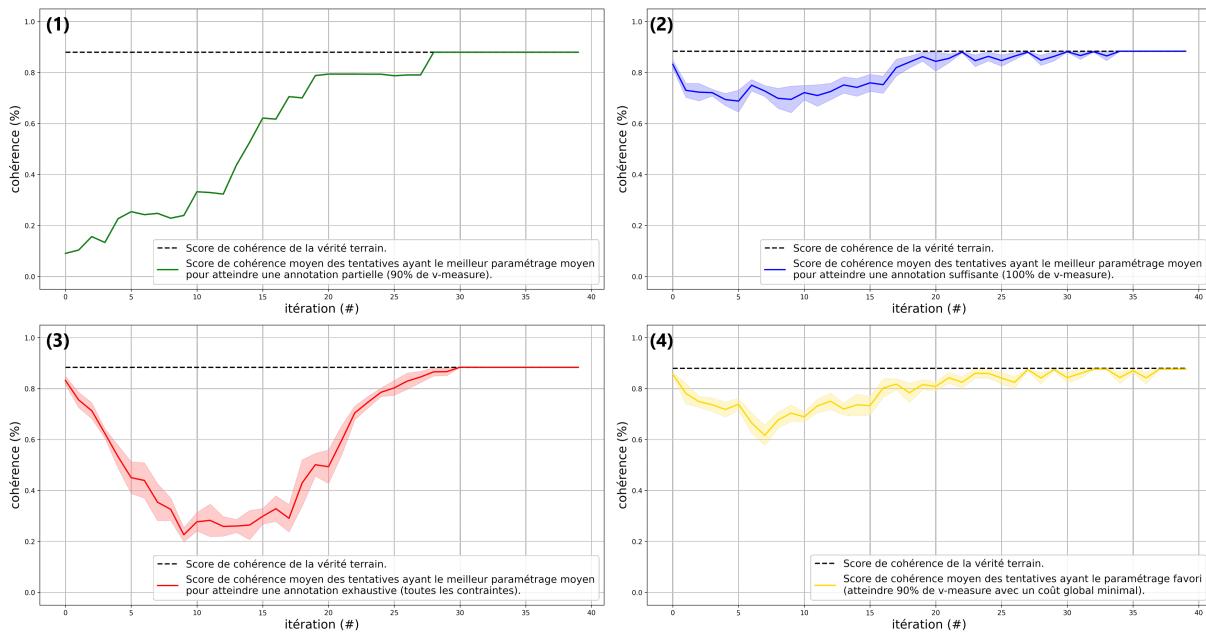


FIGURE 4.18 – Évolution du score de cohérence moyen des tentatives en fonction de leur paramétrage : (1) meilleur paramétrage moyen une annotation partielle (90% de v-measure), (2) meilleur paramétrage moyen une annotation suffisante (100% de v-measure), (3) meilleur paramétrage moyen une annotation exhaustive (annoter toutes les contraintes possibles), et (4) paramétrage favori (90% de v-measure avec un coût minimal).

Note : Le score de cohérence de la vérité terrain peut varier en fonction des méthodes de prétrainements et de vectorisation utilisées.

Résultats obtenus

Discussion

4.5 Hypothèse de rentabilité : « quel gain à chaque itération ? »

Nous aimerais vérifier l'hypothèse suivante :

à reformuler

❶ Hypothèse de rentabilité ❶

« Il est possible d'estimer quand méthodologie d'annotation basée sur le *clustering* interactif **a convergé** vers un résultat satisfaisant (cf. figure 4.19). »

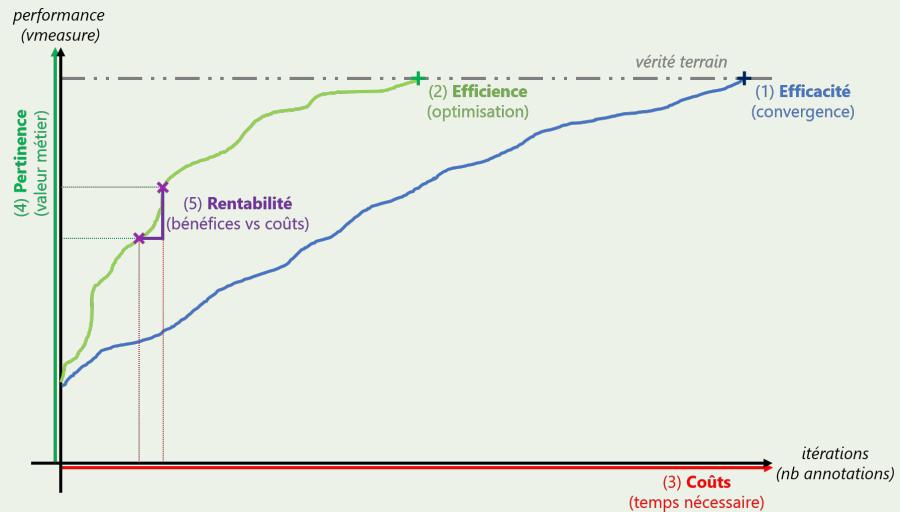


FIGURE 4.19 – Illustration des études réalisées sur le *clustering* interactif (étape 5/6) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre d'annotations par un expert métier*).

4.5.1 Étude d'estimation des cas d'arrêts de la méthode

Protocole expérimental

Description succincte du protocole expérimental dans l'encadré d'hypothèse ?

Résultats obtenus

Discussion

4.6 Hypothèse de robustesse : « quelle influence d'une erreur ? »

à reformuler

Nous aimerais vérifier l'hypothèse suivante :

❶ Hypothèse de robustesse ❶

« Il est possible d'estimer l'influence d'une différence d'annotation lors d'une méthodologie d'annotation basée sur le *clustering* interactif (cf. figure 4.20. »

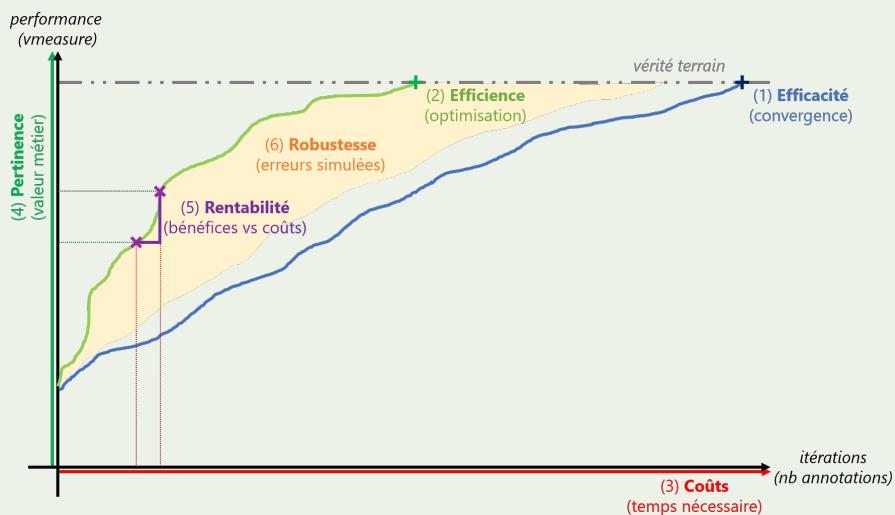


FIGURE 4.20 – Illustration des études réalisées sur le *clustering* interactif (étape 6/6) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre d'annotations par un expert métier*).

4.6.1 Étude de simulation d'erreurs d'annotations

Protocole expérimental

Description succincte du protocole expérimental dans l'encadré d'hypothèse ?

Résultats obtenus

Discussion

4.6.2 Étude d'annotation avec des paradigmes différents

Protocole expérimental

Description succincte du protocole expérimental dans l'encadré d'hypothèse ?

Résultats obtenus

Discussion

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	61
A.1.1	Kmeans	61
A.1.2	Hierarchique	61
A.1.3	Spectral	61
A.1.4	DBScan	61
A.1.5	Affinity Propagation	61
A.2	Evaluation d'une clustering	62
A.2.1	Homogénéité – Complétude – Vmeasure	62
A.2.2	FMC	62

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	63
B.2	package pypi interactive-clustering-gui	63
B.3	package pypi features-maximization-metrics	63
B.4	experimentations jupyter notebook	63

- 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	french bank cards	65
C.2	DNA press title	65

C.1 french bank cards

C.2 DNA press title

Bibliographie

- LAMPERT, T., DAO, T.-B.-H., LAFABREGUE, B., SERRETTE, N., FORESTIER, G., CREMILLEUX, B., VRAIN, C., & GANCARSKI, P. (2018). Constrained distance based clustering for time-series : a comparative and experimental study. *Data Mining and Knowledge Discovery*, 32(6), 1663-1707. <https://doi.org/10/gfbpj8>
- SCHILD, E. (2021, novembre 5). *cognitivefactory/interactive-clustering-comparative-study*. Zenodo. <https://doi.org/10.5281/ZENODO.5648255>
- 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*, 18(2), 1-19. <https://doi.org/10.4018/IJDWM.298007>
- SEABOLD, S., & PERKTOLD, J. (2010). statsmodels : Econometric and statistical modeling with python. *9th Python in Science Conference*.

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 ?	10
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
citation	13
citation	14
citation	14
ref :section2 :clustering	17
citation	17
ref	17
citation + footnote	18
travaux TPS en annexe ?	18
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
Ajouter lien vers la documentation des notebooks en footnote	22
référence, lien vers ANNEXE, + description conditions de création du JDD	25
Commentaire Gautier 22/05/2023 : (A DÉTAILLER AILLEURS ?) Oui, complètement d'accord ici, mais en fait ça va plus loin que ça non ? Déjà, on a une quantité de ressources allouées à la tâche en effet plus fabile (car choisir entre "similaire" et "non similaire" est clairement plus simple que d'assigner un label parmi N). Mais on a aussi une diminution des ressources allouées au maintien d'une stratégie d'annotation : en effet, pas besoin de définir à l'avance de type system ou autre, tout est construit à la volée. Ce deuxième point est particulièrement intéressant à discuter je pense, car on sait normalement que le maintien d'objectifs en mémoire de travail peut aider à maintenir un niveau d'engagement sur une tâche cognitive. Du coup, ça pose d'autant plus la question de l'expérience utilisateur : annoter avec un CI sera-t-il moins engageant qu'annoter avec une méthode classique ?	28
référence, lien vers ANNEXE	30
citation	31
remarque sur la valeur de eta2	36
remarque sur la valeur de eta2	36
à compléter	39

Liste des TODOs

TODO : ANNEXE JEU DE DONNEES	40
A REDIGER : consignes annotateurs	40
A REDIGER : description interface	40
citation	41
A REDIGER : Description statistiques des résultats	41
A REDIGER : Modélisation du temps	41
A REDIGER : équation du temps	41
A REDIGER : Discussion temps moyen	42
A REDIGER : Discussion vitesse d'annotation	42
citation	45
ref annexe	45
ref annexe	45
ref annexe	45
ref annexe	47
ref annexe	47
ref annexe	47
ref annexe	47
ref annexe	47
ref annexe	47
ref annexe	47
ref annexe	47
ref annexe	48
ref annexe	48
ref annexe	48
ref annexe	49
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
A REDIGER :	51
à compléter	53
Description succincte du protocole expérimental dans l'encadré d'hypothèse ?	53
Description succincte du protocole expérimental dans l'encadré d'hypothèse ?	53
à reformuler	55
Description succincte du protocole expérimental dans l'encadré d'hypothèse ?	55
à reformuler	56
Description succincte du protocole expérimental dans l'encadré d'hypothèse ?	56
Description succincte du protocole expérimental dans l'encadré d'hypothèse ?	56
SECTION À RÉDIGER	58
Style d'écriture : "je" ou "nous" ou "on" ?	72
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	27
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>).	29
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.	32

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.	33
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.	35
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) clustering 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>	36
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>).	37
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 nombre d'itérations de la méthode (<i>nombre d'annotations par un expert métier</i>).	39
4.11 Estimation du temps nécessaire (en secondes) pour annoter un lot de contraintes.	41
4.12 Etude de cas d'évolution de la vitesse d'annotation de contraintes (en contraintes par minutes) en fonction des différentes sessions d'annotations	42
4.13 Estimation du temps nécessaire (en secondes) 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.	46
4.14 Estimation du temps nécessaire (en secondes) pour effectuer une tâche de vectorisation en fonction du nombre de données à traiter.	46
4.15 Estimation du temps nécessaire (en secondes) pour effectuer une tâche de clustering en fonction du nombre de données à traiter.	48
4.16 Estimation du temps nécessaire (en secondes) pour effectuer une tâche d' échantillonnage de contraintes en fonction du nombre de données à traiter.	49
4.17 Illustration des études réalisées sur le <i>clustering</i> interactif (<i>étape 4/6</i>) en schématisant l'évolution de la performance (<i>accord avec la vérité terrain calculé en v-measure</i>) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (<i>nombre d'annotations par un expert métier</i>).	53

4.18	Évolution du score de cohérence moyen des tentatives en fonction de leur paramétrage : (1) meilleur paramétrage moyen une annotation partielle (90% de v-measure), (2) meilleur paramétrage moyen une annotation suffisante (100% de v-measure), (3) meilleur paramétrage moyen une annotation exhaustive (annoter toutes les contraintes possibles), et (4) paramétrage favori (90% de v-measure avec un coût minimal).	54
Note :	<i>Le score de cohérence de la vérité terrain peut varier en fonction des méthodes de prétraitements et de vectorisation utilisées.</i>	
4.19	Illustration des études réalisées sur le clustering interactif (<i>étape 5/6</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>).	55
4.20	Illustration des études réalisées sur le clustering interactif (<i>étape 6/6</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>).	56

Liste des figures

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.	32
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.	34
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.	35

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.	30
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.	40
4.4	Description en pseudo-code du protocole expérimental de l'étude du temps d'exécution des algorithmes du <i>clustering</i> interactif	43

LISTE DES ALGORITHMES

Liste de codes

3.1	Jeu exemple pour présenter notre implémentation du clustering interactif.	12
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.	15
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, 61
 dbSCAN, 61
 hierarchique, 61
 kmeans, 61
 spectral, 61

vmeasure, 62

Index
