UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

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

THÈSE

présentée et soutenue publiquement le 01 mai 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

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

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 souhaites remercier :

☐ ma femme

☐ ma tortue

☐ ma famille

☐ mes amis

☐ mes collègies

☐ mes encadrants

☐ chatGPT

☐ ...

Je dédie cette thèse
à quelqu'un de bien.

Table des matières

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 succinte ;
  - 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 paramétrer, réponses contrôlées, o (-) demande de l'entrainement, 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 "*Estabilishing a Niche*"

SECTION À RÉDIGER

- ☐ Cadre industriel
  - Algorithme fixe

- Données spécifiques

☐ GAP : Besoins de données

- Collecte de données spécifiqus 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'nanotation. 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.

- 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

Rappel des contrainte indus-trielles

Sommaire

2.1 Rappel sur le fonctionnement d'un chatbot

SECTION À RÉDIGER

titre : Ap-proche statis-tique vs symbo-lique

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 $(intention, enites)$ o

citation

• Description du cas d'un chatbot non-supervisé / à base d'un modèle de langage o

citation

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 organisation principales

a distinguer suivant l'approche statistiques 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 touts les aspects.

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

• Data scientistes : o Experts en IA o Peu de connaissance métier, i.e. peu de regarde 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 lorque 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 datascientists, 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 trie 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 dela 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 Entraînement et tests

- Le coeur "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 maitrisé par les utilisateurs o problème d'ergo 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 etre 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 tux de bonnes réponse ? Ou c'est plu 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 structure 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

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

2.3.2 Création assistée par des regroupements non-supervisés

• Constat : o 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 : o Un clustering pourrait simplifier cette tâche ! o Rappel : grandes lignes du fonctionnement d'un algorithme de clustering ? o NB : une section ou une annexe détaillera les algorithmes de les plus utilisés o KMeans : Classique, Incontournable, Rapide, Efficace o Hiérarchique : Lent mais facile à implémenter o Spectral : Permet des topologies complexes o DBScan : Classique, Incontournable, Rapide, Efficace, Peu d'hyperparamètre o Affinity propagation : o Metric learning : Lent mais plus adapté au corpus o ...

clustering topic modeling, ...

• Avantages : o Regroupement automatique o Découverte de la structure

• Inconvénients : o Les résultats sont souvent peu pertinents o Similarité par entités, et pas par intentions o Nuances métiers non comprises o Plusieurs soucis si le jeu de données est déséquilibré ou spécifique o Absence d'un modèle de langue spécifique au contexte... o parfois besoin d'hyperparamètres complexes à déterminer

2.3.3 Conception assistée par des regroupements semi-supervisés

• Solution : o On peut envisager ainsi de corriger le clustering en y insérant des contraintes métiers Lampert et al., 2018 o Méthodes semi-supervisée o NB : une section ou une annexe détaillera les algorithmes de clustering sous contraintes o KMeans : Classique, Incontournable, Rapide, Efficace o Hiérarchique : Lent mais facile à implémenter o Spectral : Permet des topologies complexes o DBScan : Classique, Incontournable, Rapide, Efficace, Peu d'hyperparamètre o Affinity propagation : o Metric learning : Lent mais plus adapté au corpus o ...

• Interactions possibles avec le clustering (sur la base de proposition de l'humain) o Sur les données / sur le résultat : ajouts de contraintes sur les données, suppressions ou modifications manuelles de données, réorganisation manuelles des clusters, ... o Sur les paramètres : modifier les hyper-paramètres, modifier le nombre de clusters, modifier les embeddings, utiliser d'autres algorithmes, ... o Besoin de visualisation : vue des contraintes, de la représentation vectorielle, ...

• Avantage : o On a réglé les problèmes de pertinence en ajoutant des contraintes

• Inconvénients : o Choisir comment modéliser ces contraintes peut être complexe o Surtout énorme en ajoutant des contraintes o Choisir les contraintes pertinentes est une tâche difficile

2.3.4 Conception basée sur des méthodes d'apprentissage actif

• Solution : o On peut demander à la machine de définir les contraintes dont elle a besoin pour s'améliorer / confirmer son comportement o 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 o Sous-entendu : Préférer la collaboration à la supériorité (que ce soit celle de la machine ou celle de l'expert) o 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) o Sur les données / sur le résultat : proposition de suppression de données aberrantes, proposition d'ajout de contraintes à des endroits stratégiques, ... o Sur les paramètres : réévaluation des paramètres, combiner plusieurs algorithmes et synthétiser le résultat, ...

• Avantage : o On a réglé les problèmes de pertinence et de coûts en ajoutant des contraintes

• Inconvénients / problème à résoudre : o Accepter de collaborer avec la machine (problème UX, ergo, accompagnement au changement) o 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'entrainement 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

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.

citation

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.

citation

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.

citation

à reformuler plus tard.

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.

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)

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.

utiliser l'appellation clustering ou segmentation ?

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

Pour l'**initialisation** de la méthode (cf. Alg. 3.1, *lignes 1 à 3*), nous définissons une liste vide de contraintes : tout au long du processus, cette liste contiendra l'ensemble de la connaissance que l'expert transmettra au système sous la forme de contraintes simple 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 coeur 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 l'expert, la formulation de l'énoncer d'annotation doit être judicieusement définie : par exemple, les contraintes peuvent représenter une similitude sur la thématique concernée [1], sur l'action désirée [2], ou encore sur le besoin de l'utilisateur [3]. On notera que des incohérences peuvent s'introduire, ayant pour conclusions de devoir à la fois considérer comme similaire et différentes deux données.

descriptio technique plus tard ? ref subsection :3.3.4

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

figure, ref subsection :3.3.2

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*, ...

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.

cf. partie étude

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.

description technique plus tard ? ref subsection :3.3.3

description technique plus tard ? ref subsection :3.3.X

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.

```
# Définir les données.
dict_of_texts = {
    "0": "Comment signaler un vol de carte bancaire ?",
    "1": "J'ai égaré ma carte bancaire, que faire ?",
    "2": "J'ai perdu ma carte de paiement",
    "3": "Le distributeur a avalé ma carte !",
    "4": "En retirant de l'argent, le GAB a gardé ma carte...",
    "5": "Le distributeur ne m'a pas rendu ma carte bleue.",
    # ...
    "N": "Pourquoi le sans contact ne fonctionne pas ?",
}
```

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 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 filtre**, correspondant au traitement de base avec l'élagage de l'arbre de dépendance syntaxique de la phrase, noté `prep.filter`.

D'autre part, la partie `utils.vectorization`[5] permet de transformer les données en une représentation exploitable pour la machine. Deux modes de vectorisation sont mis à disposition :

1. **TF-IDF**, 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`. citation

Vous avez un exemple d'utilisation des modules de prétraitements et de vectorisation dans Code 3.2.

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

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

```
# Import des dépendances.
from cognitivefactory.interactive_clustering.utils.preprocessing
    import preprocess
from cognitivefactory.interactive_clustering.utils.vectorization
    import vectorize

# Prétraitement des données.
dict_of_preprocess_texts = preprocess(
    dict_of_texts=dict_of_texts,
    apply_stopwords_deletion=False,
    apply_parsing_filter=False,
    apply_lemmatization=False,
    spacy_language_model="fr_core_news_md",
)
"""
    {"0": "comment signaler un vol de carte bancaire",
     "1": "j ai egare ma carte bancaire, que faire",
     "2": "j ai perdu ma carte de paiement",
     "3": "le distributeur a avale ma carte",
     "4": "en retirant de l argent le gab a garde ma carte",
     "5": "le distributeur ne m a pas rendu ma carte bleue",
     # ...
     "N": "pourquoi le sans contact ne fonctionne pas"}
"""

# Vectorisation des données.
dict_of_vectors = vectorize(
    dict_of_texts=dict_of_preprocess_texts,
    vectorizer_type="tfidf",
)
```

3.3.2 Gestion des contraintes

En ce qui concerne la **manipulation de contraintes**, nous utilisons le module `contraints`[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) :

$$(\forall D_1, D_2, D_3)\texttt{MUSTLINK}(D_1, D_2) \wedge \texttt{MUSTLINK}(D_2, D_3) \Rightarrow \texttt{MUSTLINK}(D_1, D_3) \tag{3.1}$$

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

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

Pour respecter ces propriétés, le gestionnaire de contraintes doit calculer les transitivités à chaque ajout ou suppression de contraintes. On distinguera donc une contrainte ajoutée (`added`) d'une contrainte déduite par transitivité (`inferred`).

Il se peut que la contrainte en cours d'ajout contredise les contraintes déduites : nous parlons alors d'incohérence ou de conflit (cf. Fig. 3.1). Dans ce cas, l'ajout de la dernière contrainte n'est pas prise en compte et le gestionnaire renvoie une erreur permettant d'identifier ce conflit. Ce conflit peut venir simplement venir d'une erreur d'inattention, mais peut aussi venir d'une déduction basée sur des ajouts antérieurs erronés .

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

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

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 noyau par similarité pour former un plus *cluster* plus conséquent, ou être distingué d'autres noyaux pour former plusieurs *clusters*.

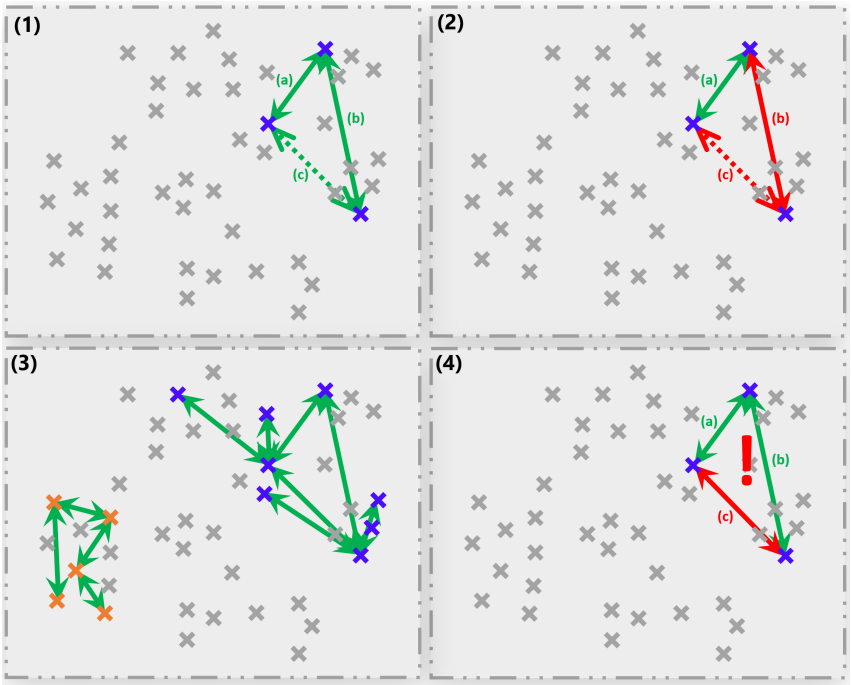

FIGURE 3.1 – Exemples des propriétés de transitivité des contraintes `MUST-LINK` (flèches vertes) et `CANNOT-LINK` (flèches rouges). **(1)** et **(2)** représente les possibilités de déduction d'une contrainte (`(c)`) en fonction des deux autres (`(a)` et `(b)`). **(3)** représente deux composants connexes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

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.

```
# Import des dépendances.
from cognitivefactory.interactive_clustering.constraints.factory import managing_factory

# Création du gestionnaire de contraintes.
constraints_manager = managing_factory(
    manager="binary",
    list_of_data_IDs = list(dict_of_texts.keys()),  # ["0", "1", "2", "3", "4", "5", ..., "N"]
)

# Ajout de contraintes.
constraints_manager.add_constraint(
    data_ID1="0",  # "Comment signaler un vol de carte bancaire ?"
    data_ID2="1",  # "J'ai égaré ma carte bancaire, que faire ?"
    constraint_type="MUST_LINK",
)
```

```
constraints_manager.add_constraint(
    data_ID1="3",  # "Le distributeur a avalé ma carte !"
    data_ID2="4",  # "En retirant de l'argent, le GAB a gardé ma
        carte..."
    constraint_type="MUST_LINK",
)
constraints_manager.add_constraint(
    data_ID1="0",  # "Comment signaler un vol de carte bancaire ?"
    data_ID2="N",  # "Pourquoi le sans contact ne fonctionne pas ?"
    constraint_type="CANNOT_LINK",
)
  # NB: ajouter une contrainte "MUST_LINK" entre "1" et "N" lèverait
      une erreur.

# Récupération des composants connexes.
# Récupérartion des composants connexes.
connected_components = constraints_manager.get_connected_components()
"""
  [['0', '1'],
   ['2'],
   ['3', '4'],
   ['5'],
   ['N']]
"""
```

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) :

ref :section2 :clustering

1. **KMeans**, dans sa version *COP* et sa version *MPC*, notés `clust.kmeans.cop` et `clust.kmeans.mpc` pour nos études ;
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`.

citation
citation
citation
citation
citation
citation

Une classe abstraite définit les prérequis des algorithmes implémentés (avoir une méthode `cluster`) et une *factory* est disponible pour instancier rapidement un objet de *clustering*. 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.

ref

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

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

```
# Import des dépendances.
from cognitivefactory.interactive_clustering.clustering.factory
    import clustering_factory

# Initialiser un objet de clustering.
clustering_model = clustering_factory(
    algorithm="kmeans",
   model="COP",
    random_seed=42,
)

# Lancer le clustering.
clustering_result = clustering_model.cluster(
    constraints_manager=constraints_manager,  # contient les
       contraintes
    nb_clusters=2,
    vectors=dict_of_vectors,
)
"""
    {"0": 0,  # "Comment signaler un vol de carte bancaire ?"
     "1": 0,  # "J'ai égaré ma carte bancaire, que faire ?"
     "2": 0,  # "J'ai perdu ma carte de paiement"
     "3": 1,  # "Le distributeur a avalé ma carte !"
     "4": 1,  # "En retirant de l'argent, le GAB a gardé ma carte..."
     "5": 1,  # "Le distributeur ne m'a pas rendu ma carte bleue."
     # ...
     "N": 1}  # "Pourquoi le sans contact ne fonctionne pas ?"
"""
```

Les algorithmes KMeans (COP), hiérarchique et spectral (SPEC) ont été implémentés au début de ce doctorat. Dans le cadre d'un projet industriel 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.

citation + footnote

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 ;

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

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

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

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

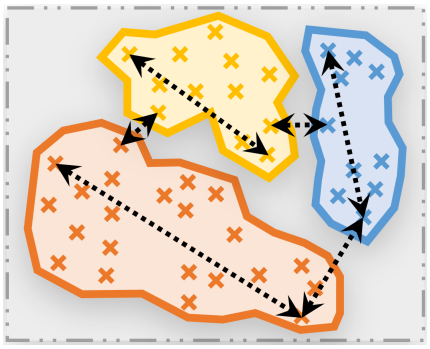

FIGURE 3.2 – Exemples d'échantillonnages, sur la base de trois clusters, de données issues de mêmes clusters et étant les plus éloignées les unes des autres (`samp.farhtest.same`), et de données issues de clusters différents et étant les plus proches les unes des autres (`samp.closest.diff`).

Une classe abstraite définit les prérequis des algorithmes 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
      sampling_factory
3
4 # Initialiser un objet d'échantillonnage.
5 sampler = sampling_factory(
6   algorithm="random",
7     random_seed=42,
8 )
9
```

```
# Run sampling.
selection = sampler.sample(
  constraints_manager=constraints_manager,
  nb_to_select=2,
  clustering_result=clustering_result,  # optionnel pour "random"
  vectors=dict_of_vectors,  # optionnel pour "random"
)
"""
  [("0", '5'),  # "Comment signaler un vol de carte bancaire ?" vs "Le distributeur ne m'a pas rendu ma carte bleue."
   ("0", '2'),  # "Comment signaler un vol de carte bancaire ?" vs "J'ai perdu ma carte de paiement"
   ("2", 'N')]  # "J'ai perdu ma carte de paiement" vs "Pourquoi le sans contact ne fonctionne pas ?"
"""
```

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'entrainement 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.

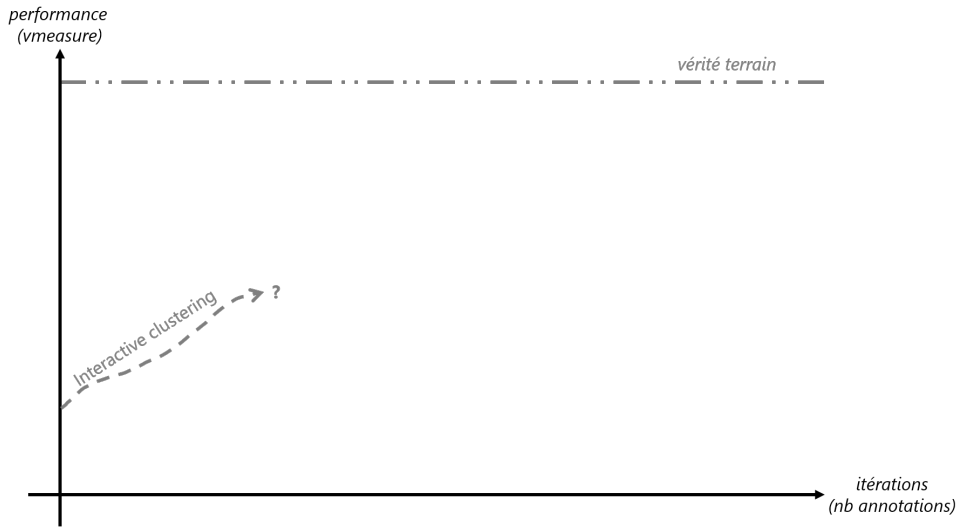

FIGURE 4.1 – Illustration des études réalisées sur le *clustering* interactif (*étape 0/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre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

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.

footnote documentation

Sommaire

4.1 Hypothèse d'efficacité : « *est-ce que la méthode fonctionne ?* »

Nous aimerions 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 l'hypothèse et l'espoir de convergence d'une base d'apprentissage en cours de construction vers sa vérité terrain.

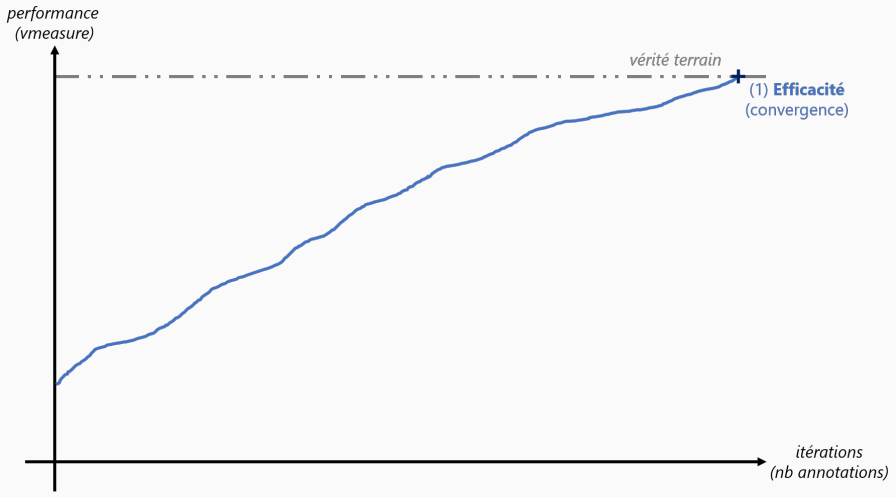

FIGURE 4.2 – Illustration des études réalisées sur le *clustering* interactif (*étape 1/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre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

4.1.1 Étude de convergence vers une vérité terrain pré-établie

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).

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 conversationnel. 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. 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.

référence, lien vers ANNEXE

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 itération 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 de critiquer la ressemblance entre deux données de son domaine d'expertise. Une tentative de l'application de notre méthode s'arrête lorsque toutes les contraintes possibles entre les données ont été annotés par l'expert.

Ajouter format spécial pour cette note ?

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 quatre niveaux suivants : `prep.no`, `prep.simple`, `prep.lemma` et `prep.filter` ;
2. la **vectorisation** des données, avec les deux niveaux suivants : `vect.tfidf` et `SpaCy (vect.frcorenewsmd)` ;
3. le **clustering sous contraintes** des données, avec les six niveaux suivants : `clust.kmeans.cop`, `clust.hier.sing`, `clust.hier.comp`, `clust.hier.avg`, `clust.hier.ward` et `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 quatre niveaux suivants : `samp.random.full`, `samp.random.same`, `samp.farhtest.same` et `samp.closest.diff`. La choix de la taille d'échantillon n'est pas étudié ici, et cette taille est arbitrairement fixé à `50`.

Tester C-DBScan ?

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.

Afin d'affiner notre évaluation, nous porterons aussi attention aux seuils d'annotations suivants :

1. le ***clustering* initial**, correspondant à la segmentation des données sans contraintes, ne bénéficiant d'aucune connaissance métier ;

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.

Pour résumer ce protocole expérimental, vous pouvez vous référez 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)

pour tout arrangement d'algorithmes et de paramètres à tester **faire**
    initialisation : récupérer les données de la vérité terrain sans leur label, créer une liste vide de contraintes
    prétraitement : supprimer le bruit dans les données
    vectorisation : transformer les données en vecteurs
    clustering initial : regrouper les données par similarité
    évaluation : estimer l'équivalence entre le clustering obtenu et la vérité terrain
    répéter
        échantillonnage : sélectionner de nouvelles contraintes à annoter
        simulation d'annotation : ajouter des contraintes grâce à la comparaison des labels de la vérité terrain
        clustering : regrouper les données par similarité avec les contraintes
        évaluation : estimer l'équivalence entre le clustering obtenu et la vérité terrain
    jusqu'à annotation de toutes les contraintes possibles
    évaluation finale : espérer avoir un score d'équivalence de 100% entre le clustering obtenu et la vérité terrain
fin pour

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

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

Résultats obtenus

Toutes les 960 tentatives (192 combinaisons de paramètres, 5 tentatives pour chaque combinaison) ont convergé vers la vérité terrain (i.e. on atteint une `v-measure` de 100%). La figure 4.3 représente 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 y sont aussi représentées.

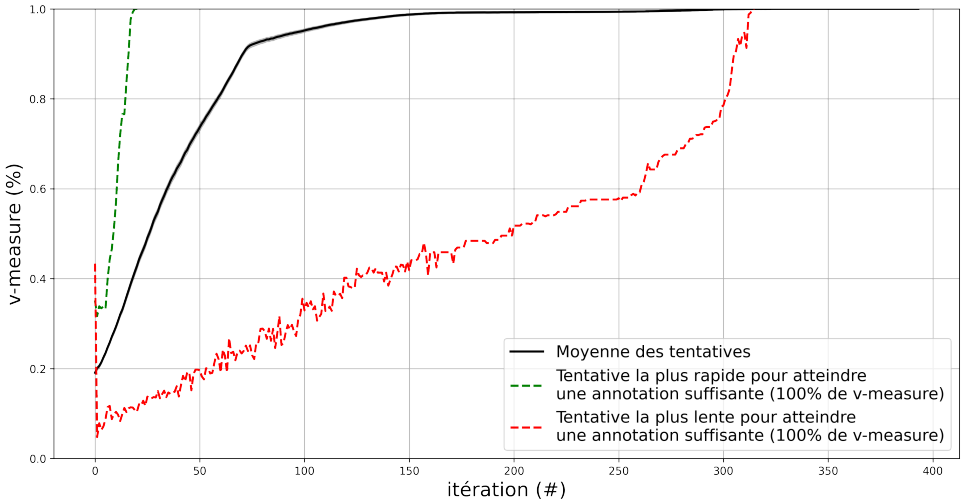

FIGURE 4.3 – Évolution de la moyenne de la `v-measure` entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de *clustering* interactif, moyenne réalisée itération par itération sur l'ensemble des tentatives. Représentation des tentatives ayant été les plus rapides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

Lors du ***clustering* initial**, on remarque que la `v-measure` entre le clustering (sans contraintes) et la vérité terrain atteint une moyenne de `19.05`% (min : `03.42`%, max : `47.75`%, écart-type : `13.38`%) ;

Pour obtenir une **annotation suffisante** (*atteindre une* `v-measure` *de 100%*), la moyenne des itérations est de `76.29` (min : `19`, max : `328`, écart-type : `46.44`), soit une moyenne de `3 801.19` annotations (min : `950`, max : `16 400`, écart-type : `2 314.91`). La figure 4.4 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 avec un prétraitement `prep.simple`, une vectorisation `vect.tfidf`, un clustering `clust.hier.comp` ou `clust.hier.ward`, et un échantillonnage `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 avec prétraitement `prep.no`, une vectorisation `vect.tfidf`, un clustering `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`.

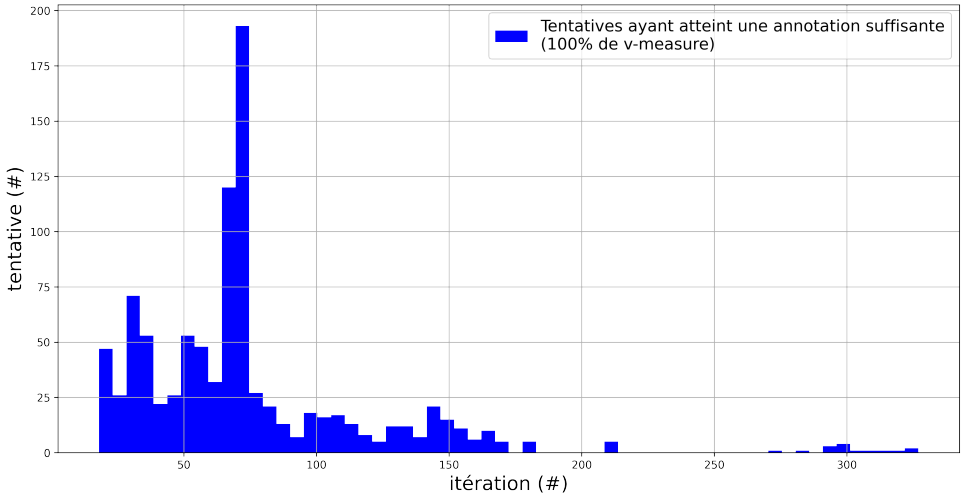

FIGURE 4.4 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation suffisante, c'est-à-dire l'itération à laquelle elles parviennent à 100% de `v-measure` entre un résultat obtenu et la vérité terrain. La moyenne est située à `76.29` itérations, et l'histogramme est réduit à 60 pics pour simplifier l'affichage.

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 4.5 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 avec un prétraitement `prep.no` ou `prep.lemma`, une vectorisation `vect.tfidf`, un clustering `clust.hier.comp` ou `clust.hier.wars`, et un échantillonnage de contraintes `samp.closest.diff`. Ces tentatives ont requis `20` itérations, soit `1 000` annotations, dont `653` (respectivement `668`) contraintes `MUST-LINK`.

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

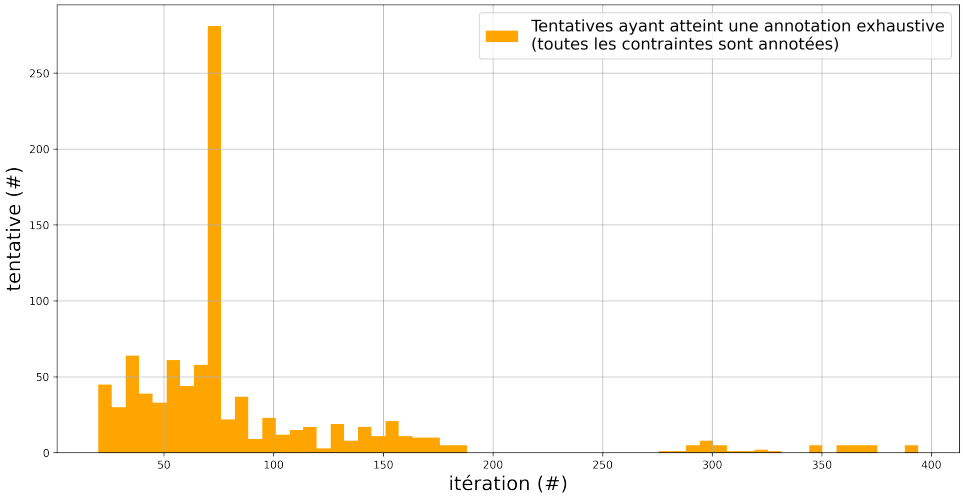

FIGURE 4.5 – Répartition des tentatives en fonction de l'itération de la méthode à laquelle elles atteignent le seuil d'une annotation exhaustive, c'est-à-dire l'itération à laquelle toutes les contraintes possibles entre les données ont été annotées. La moyenne est située à `88.98` itérations, et l'histogramme est réduit à 60 pics pour simplifier l'affichage.

Discussion

La première et principale conclusion de cette étude concerne la preuve que la méthode est fonctionnelle. En effet, les différentes simulations ont bien convergé vers la vérité terrain, 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* interactif.

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 pleinement opérationnel.

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, il faudrait une moyenne de `88.98` itérations pour être exhaustif, soit `4 431.34` annotations, ce qui correspond à près de `9` 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. 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 (écart-type de `68.21`). On peut donc espérer trouver un paramétrage optimal de la méthode permettant de diminuer significativement le nombre de contraintes requis et obtenir ainsi 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).

footnote ?

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 80% de `v-measure` est déjà atteinte autour de l'itération 60, alors que l'asymptote à 100% n'est atteinte que vers l'itération 150.

WARNING : moyenne selon itération vs moyenne selon vmeasure

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 Hypothèse d'efficience : « *est-ce que l'implémentation est optimale ?* »

Suite à la validation de l'hypothèse d'efficacité (convergence de la méthode, cf. section 4.1), nous aimerions vérifier l'hypothèse suivante :

à compléter

Hypothèse d'efficience

« **La vitesse de convergence du *clustering* interactif peut être optimisée en ajustant différents paramètres. 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 4.6 illustre l'hypothèse et l'espoir d'une convergence "optimale" d'une base d'apprentissage en cours de construction vers sa vérité terrain.

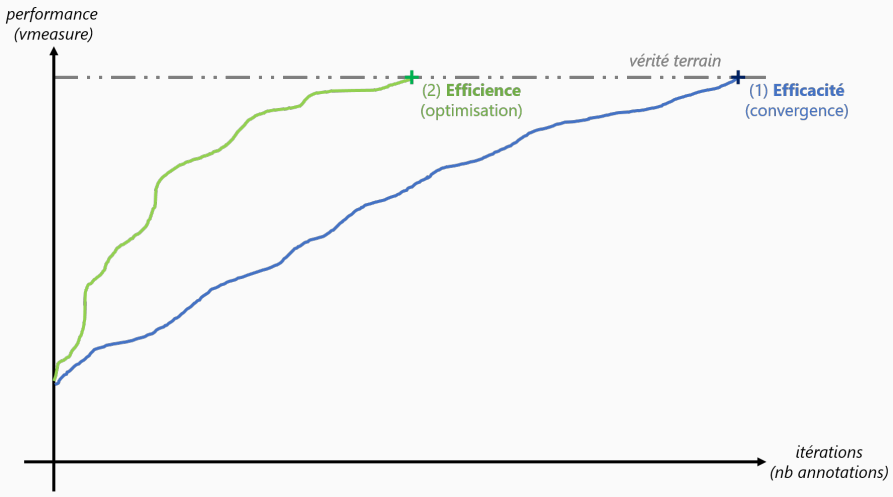

FIGURE 4.6 – Illustration des études réalisées sur le *clustering* interactif (*étape 2/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre d'annotations par un expert métier*).

4.2.1 Étude d'optimisation des paramètres de convergence

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).

Protocole expérimental : analyser la taille d'effet paramètres 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. Pour cette étude, nous allons reprendre le protocole expérimental de l'étude de convergence en section 4.1.1 visant à simuler la création d'une base d'apprentissage.

référence, lien vers ANNEXE

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. (***nouveau***) le cas d'une **annotation partielle**, correspondant au nombre d'itérations nécessaires à la méthode pour avoir 80% de `v-measure` entre le résultat obtenu et la vérité terrain, c'est-à-dire un état de semi-parcours vers une convergence totale [9] ;
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.

pourquoi un nouveau seuil à 80%

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. Ces mesure seront réalisées à l'aide du logiciel R, et les `Tukey HSD` est utilisé pour les comparaisons post-hoc.

citation

Pour résumer ce protocole expérimental, vous pouvez vous référez au pseudo-code décrit dans Alg. 4.2.

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

Résultats obtenus

Le tableau 4.1 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaires pour atteindre une **annotation partielle** (*atteindre une* `v-measure` *de 80%*). Les analyses de variance mettent en relief l'effet significatif sur cette convergence du prétraitement (eta-carré : `0.992`, p-valeur : $< 10^{-3}$), de la vectorisation (eta-carré : `0.998`, p-valeur : $< 10^{-3}$), du clustering (eta-carré : `0.999`, p-valeur : $< 10^{-3}$) et de l'échantillonnage (eta-carré : `0.999`, 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` ou `clust.hier.sing`, et l'échantillonnage `samp.closest.diff`. La moyenne du nombre d'itération requis pour ce paramétrage est de `13.00` (écart-type : `2.11`), soit 650 annotations (écart-type : `105.50`).

9. Le seuil de 80% a été arbitrairement choisi pour cette étude.

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)

pour tout arrangement d'algorithmes et de paramètres à tester **faire**
    initialisation : récupérer les données de la vérité terrain sans leur label, créer une liste vide de contraintes
    prétraitement : supprimer le bruit dans les données
    vectorisation : transformer les données en vecteurs
    clustering initial : regrouper les données par similarité
    évaluation : estimer l'équivalence entre le clustering obtenu et la vérité terrain
    répéter
        échantillonnage : sélectionner de nouvelles contraintes à annoter
        simulation d'annotation : ajouter des contraintes grâce à la comparaison des labels de la vérité terrain
        clustering : regrouper les données par similarité avec les contraintes
        évaluation : estimer l'équivalence entre le clustering obtenu et la vérité terrain
    jusqu'à annotation de toutes les contraintes possibles
fin pour
analyse : déterminer les tailles d'effets des algorithmes et paramètres

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

Description des facteurs analysés		Description statistique			Description des tailles d'effets	
Facteur	Niveau	Moyenne	Rang	SE	η^2	`p-valeur`
prétraitement	`prep.simple`	55.95	(1)	0.33	0.992	$7.72e^{-13}$ $(***)$
	`prep.lemma`	57.25	(2)			
	`prep.no`	57.59	(2)			
	`prep.filter`	65.36	(4)			
vectorisation	`vect.tfidf`	55.00	(1)	0.30	0.998	$1.56e^{-06}$ $(***)$
	`vect.frcorenewsmd`	63.08	(2)			
clustering	`clust.hier.avg`	44.89	(1)	0.32	0.999	$< 2e^{-16}$ $(***)$
	`clust.hier.sing`	45.27	(1)			
	`clust.kmeans.cop`	46.55	(3)			
	`clust.hier.ward`	65.79	(4)			
	`clust.hier.comp`	66.90	(5)			
	`clust.spec`	84.83	(6)			
échantillonnage	`samp.closest.diff`	29.27	(1)	0.33	0.999	$< 2e^{-16}$ $(***)$
	`samp.random.same`	44.93	(2)			
	`samp.random.full`	61.07	(3)			
	`samp.farhtest.same`	101.88	(4)			

TABLE 4.1 – ANOVA du nombre d'itérations nécessaires pour l'obtention de 80% 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.

Le tableau 4.2 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaires pour atteindre une **annotation suffisante** (*atteindre une `v-measure` de 100%*). 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`, le 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`).

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	0.987	$1.17e^{-13}$ $(***)$
	`prep.simple`	73.30	(2)			
	`prep.no`	75.24	(2)			
	`prep.filter`	83.77	(4)			
vectorisation	`vect.tfidf`	71.16	(1)	0.36	0.991	$9.30e^{-07}$ $(***)$
	`vect.frcorenewsmd`	81.43	(2)			
clustering	`clust.kmeans.cop`	62.23	(1)	0.42	0.997	$< 2e^{-16}$ $(***)$
	`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	0.998	$< 2e^{-16}$ $(***)$
	`samp.random.same`	56.38	(2)			
	`samp.random.full`	71.95	(3)			
	`samp.farhtest.same`	126.55	(4)			

TABLE 4.2 – 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.

Le tableau 4.3 retranscrit l'influence de chacun des paramètres sur le nombre d'itérations nécessaires pour atteindre une **annotation exhaustive** (*annoter toutes les contraintes possibles*). 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`, le 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`).

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	0.909	$1.10e^{-08}$ $(***)$
	`prep.filter`	89.55	(2)			
	`prep.simple`	89.64	(2)			
	`prep.no`	90.81	(4)			
vectorisation	`vect.tfidf`	85.50	(1)	0.39	0.985	$2.53e^{-06}$ $(***)$
	`vect.frcorenewsmd`	92.46	(2)			
clustering	`clust.kmeans.cop`	64.99	(1)	0.39	0.999	$< 2e^{-16}$ $(***)$
	`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	0.997	$< 2e^{-16}$ $(***)$
	`samp.random.full`	72.80	(2)			
	`samp.closest.diff`	98.38	(3)			
	`samp.farhtest.same`	132.75	(4)			

TABLE 4.3 – 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.

La figure 4.7 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 4.8 représente cette même évolution pour les meilleurs paramétrages moyens destinés à atteindre les trois seuils d'annotation définis (partiel, suffisant, exhaustif).

figure trop petite ?

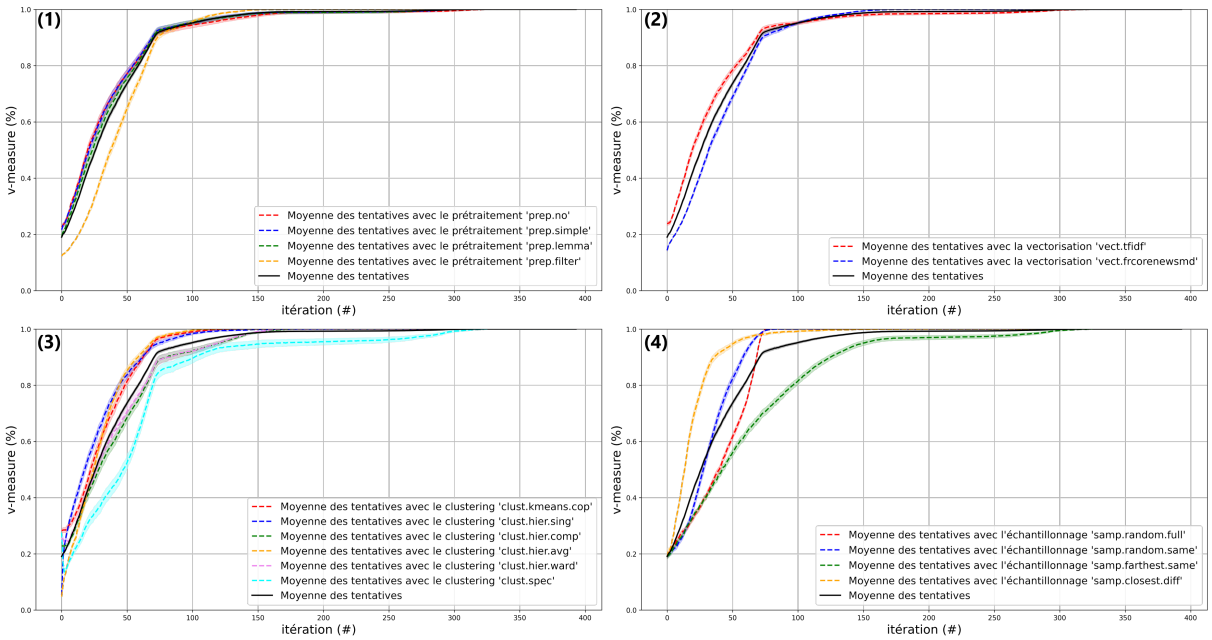

FIGURE 4.7 – Évolution des moyennes de la `v-measure` entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de *clustering* interactif, moyennes réalisées itération par itération sur les différentes valeurs que peuvent prendre les facteurs analysés et affichées par facteur : **(1)** prétraitement, **(2)** vectorisation, **(3)** clustering et **(4)** échantillonnage.

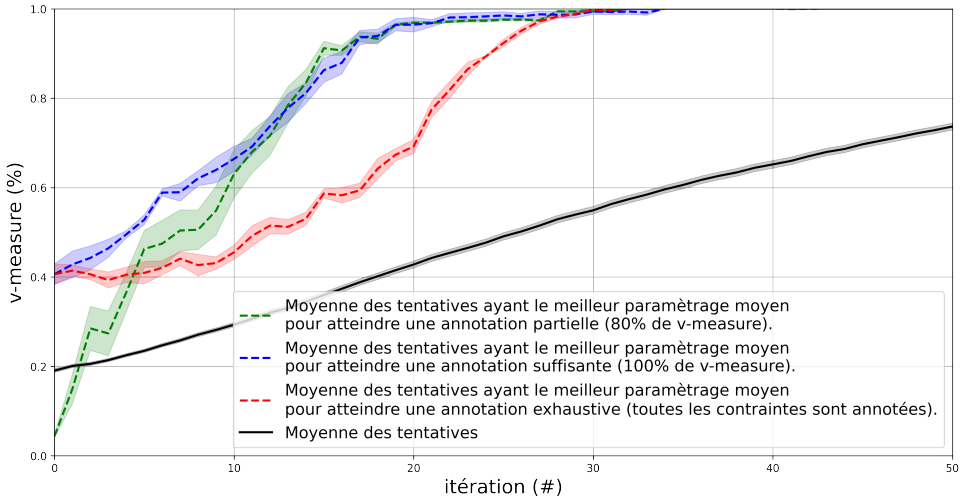

FIGURE 4.8 – Évolution des moyennes de la `v-measure` entre un résultat obtenu et la vérité terrain en fonction du nombre d'itération de la méthode de *clustering* interactif, moyennes réalisées itération par itération sur les différentes seuils d'annotations étudiés : l'annotation partielle (*atteindre une* `v-measure` *de 80%*), l'annotation suffisante (*atteindre une* `v-measure` *de 100%*) et l'annotation exhaustive (*annoter toutes les contraintes possibles*).

Discussion

meilleur paramétrage

4.3 Hypothèse sur les coûts : « *combien dois-je investir ?* »

à compléter

Nous aimerions vérifier l'hypothèse suivante :

Hypothèse sur les coûts

« Il est possible de **mesurer le temps nécessaire** à une méthodologie d'annotation basée sur le *clustering* interactif pour obtenir un résultat exploitable (cf. figure 4.9. »

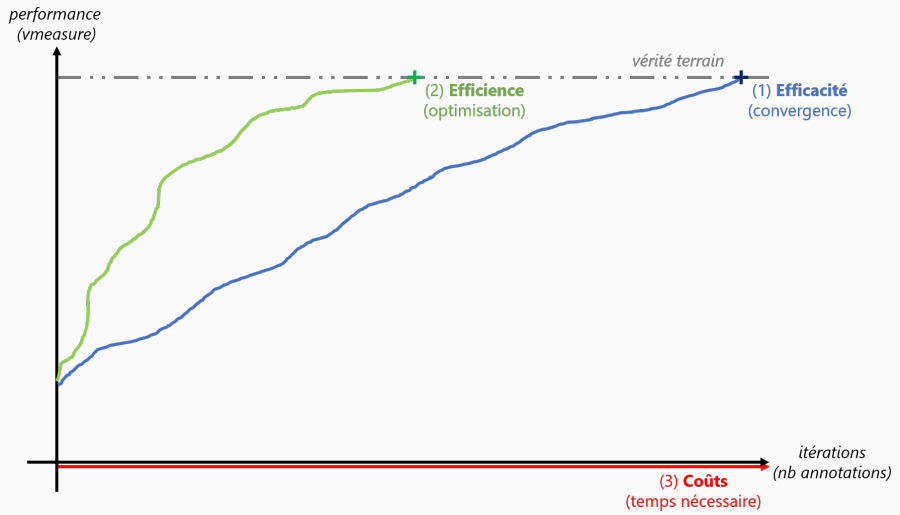

FIGURE 4.9 – Illustration des études réalisées sur le *clustering* interactif (*étape 3/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre d'annotations par un expert métier*).

4.3.1 Étude d'estimation du temps d'annotation par un expert métier

Protocole expérimental

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

Résultats obtenus

Discussion

4.3.2 Étude d'estimation du temps de calcul des algorithmes

Protocole expérimental

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

Résultats obtenus

Discussion

4.3.3 Étude d'estimation du temps total d'un projet d'annotation

Protocole expérimental

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

Résultats obtenus

Discussion

4.4 Hypothèse de pertinence : « *est-ce le résultat est exploitable ?* »

Nous aimerions 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 4.10. »

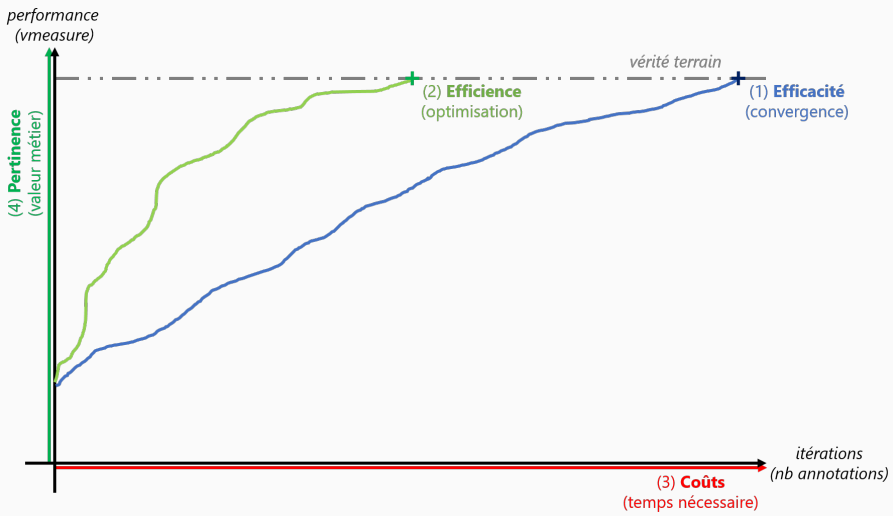

FIGURE 4.10 – Illustration des études réalisées sur le *clustering* interactif (*étape 4/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre 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 ?

Résultats obtenus

Discussion

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

à reformuler

Nous aimerions vérifier l'hypothèse suivante :

Hypothèse de rentabilité

« Il est possible d'estimer quand méthodologie d'annotation basée sur le *clustering* interactif **a convergé** vers un résultat satisfaisant (cf. figure 4.11. »

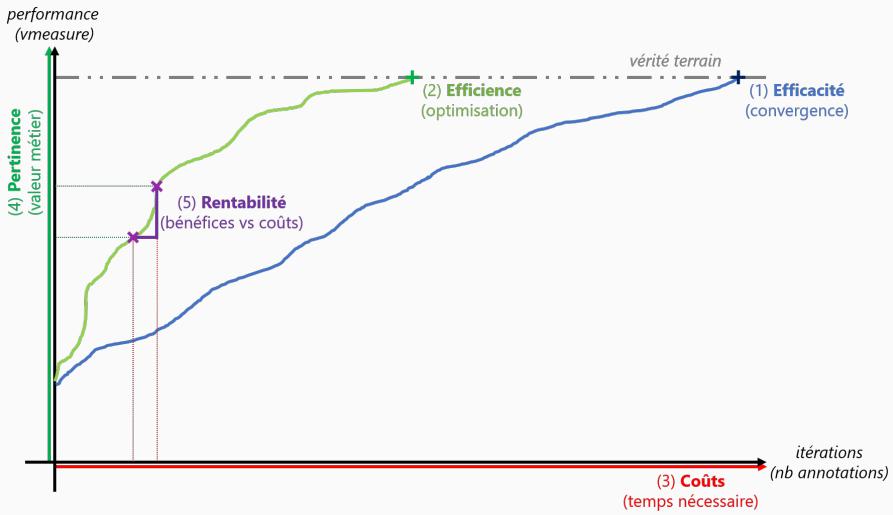

FIGURE 4.11 – Illustration des études réalisées sur le *clustering* interactif (*étape 5/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre 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 ?* »

Nous aimerions vérifier l'hypothèse suivante : à reformuler

Hypothèse de robustesse

« Il est possible d'**estimer l'influence d'une différence d'annotation** lors d'une méthodologie d'annotation basée sur le *clustering* interactif (cf. figure 4.12. »

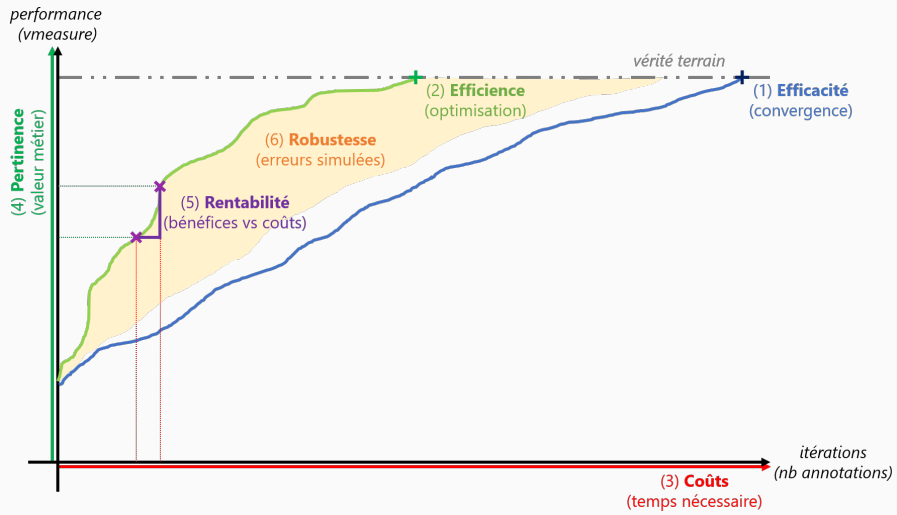

FIGURE 4.12 – Illustration des études réalisées sur le *clustering* interactif (*étape 6/6*) en schématisant l'évolution de la performance (*accord avec la vérité terrain calculé en v-measure*) d'une base d'apprentissage en cours de construction en fonction du nombre d'itérations de la méthode (*nombre 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

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

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

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

Liste des TODOs

Style d'écriture : "je" ou "nous" ou "on" ?

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste de codes

Glossaire

clustering !!TODO!!. 7

Index