Un pipeline opérationnel fondé sur les LLM pour l'analyse automatique d'enquêtes à questions ouvertes

Esteban Ouhayoun[1], Claude Caligaris[1],Sean Manzambi[1],Margarita Roos[1]

[1] EDF, DTEO-CEDID, DATA & IA Factory

5 mai 2026

Résumé

Les enquêtes à questions ouvertes génèrent de grands volumes de verbatims courts, difficiles à exploiter automatiquement : les approches classiques de NLP y montrent rapidement leurs limites, et l'analyse manuelle reste coûteuse. Cet article présente un pipeline opérationnel entièrement fondé sur des LLM pour l'analyse de ces verbatims en contexte industriel, couvrant l'extraction contrôlée de thématiques, la classification multi-label des réponses et la génération automatique de recommandations. Nous décrivons en détail chaque étape, les choix de prompt et de formats de réponse retenus, ainsi que les compromis entre modèles distants et locaux. Les évaluations quantitatives montrent que ce pipeline surpasse les méthodes classiques (K-Means, UMAP+HDBSCAN) sur la cohérence thématique, tout en offrant une reproductibilité et une automatisation compatibles avec un déploiement en entreprise.

Mots-clés

LLM, analyse d'enquêtes, traitement automatique du langage naturel, classification multi-label, verbatims.

Abstract

Open-ended survey questions generate large volumes of short verbatim responses that are difficult to process automatically : classical NLP approaches quickly reach their limits, and manual analysis remains costly. This paper presents an operational pipeline fully based on LLM for analyzing such verbatims in an industrial context, covering controlled topic extraction, multi-label response classification, and automatic recommendation generation. We describe in detail each step, the prompt engineering and response format choices made, as well as the trade-offs between remote and local models. Quantitative evaluations show that this pipeline outperforms classical methods (K-Means, UMAP+HDBSCAN) in thematic coherence, while offering reproducibility and automation compatible with enterprise deployment.

Keywords

LLM, survey analysis, natural language processing, multi-label classification, verbatims.

1 Introduction

Les enquêtes comportant des questions ouvertes — c'est-à-dire invitant les répondants à s'exprimer librement en langage naturel sans contrainte de choix prédéfinis — constituent un outil central de collecte de feedback dans de nombreux contextes organisationnels : évaluation de la satisfaction des employés, retours d'expérience sur des services internes, baromètres sociaux. Elles permettent d'accéder à des informations riches et nuancées, difficilement capturables par des questionnaires fermés, et favorisent l'émergence d'opinions, d'arguments et de propositions originales, essentielles pour orienter la prise de décision stratégique ou l'amélioration des services.

Cependant, cette richesse informationnelle s'accompagne d'un défi opérationnel majeur : l'analyse systématique et reproductible de grands volumes de réponses textuelles, de longueur très variable, allant de quelques mots à quelques phrases. Historiquement, cette analyse reposait sur un codage manuel, coûteux en temps et sujet à une variabilité inter-annotateurs importante. Les approches automatiques fondées sur les méthodes classiques de traitement automatique du langage naturel (TALN) ont permis certains progrès, mais restent limitées face à la diversité lexicale des verbatims courts et nécessitent un ajustement fin des hyperparamètres pour chaque nouvelle enquête, ce qui rend leur déploiement industriel difficile.

L'émergence des grands modèles de langage (LLM) transforme profondément ce paysage. Leurs capacités avancées de compréhension sémantique, de regroupement conceptuel et de structuration de l'information offrent un potentiel inédit pour l'analyse automatisée de réponses ouvertes, sans requérir d'expertise en apprentissage automatique ni d'ajustement de paramètres spécifiques à chaque enquête.

Dans cet article, nous proposons un pipeline complet et reproductible d'analyse par LLM, conçu pour être déployé directement en environnement de production. Ce pipeline couvre l'intégralité du processus : préparation des données, extraction contrôlée de thématiques, classification multi-label des verbatims, et génération de comptes rendus. Nous décrivons précisément chaque composant — *prompts*, formats de réponse, choix de modèles — de sorte à en assurer la reproductibilité. Nous justifions également les choix techniques effectués au regard des contraintes applicatives, notamment en termes de confidentialité des données (mo-

dèles locaux vs. distants), de coût computationnel et de qualité des résultats.

1.1 Travaux antérieurs

L'analyse automatique des réponses ouvertes s'est d'abord appuyée sur des représentations lexicales (TF-IDF [7]), rapidement limitées par la brièveté des verbatims. Les représentations distribuées — *word embeddings* [5] puis architectures contextualisées initiées par BERT [1] — ont permis de dépasser ces limites. Des frameworks populaires tels que BERTopic ont ainsi démocratisé l'extraction de thématiques via le clustering d'embeddings (souvent UMAP associé à HDBSCAN [3]). Les évolutions récentes de ces modèles neuronaux thématiques font d'ailleurs l'objet de revues de littérature approfondies [9]. Toutefois, en contexte industriel, ces approches au clustering requièrent une forte sensibilité aux hyperparamètres et une interprétation humaine toujours nécessaire.

L'essor des LLM a ouvert de nouvelles perspectives : des frameworks tels que TopicGPT [6] proposent une génération semi-supervisée de thématiques, et des travaux récents [2] montrent leur intérêt pour assister le clustering. Cependant, la classification multi-label fine sur verbatims courts et la génération de recommandations structurées restent peu couvertes [8], et les aspects opérationnels — reproductibilité, confidentialité, coût — sont rarement abordés dans la littérature.

Dans ce contexte, nous proposons une approche entièrement fondée sur les LLM répondant aux exigences concrètes d'un déploiement en entreprise.

2 Méthode Proposée

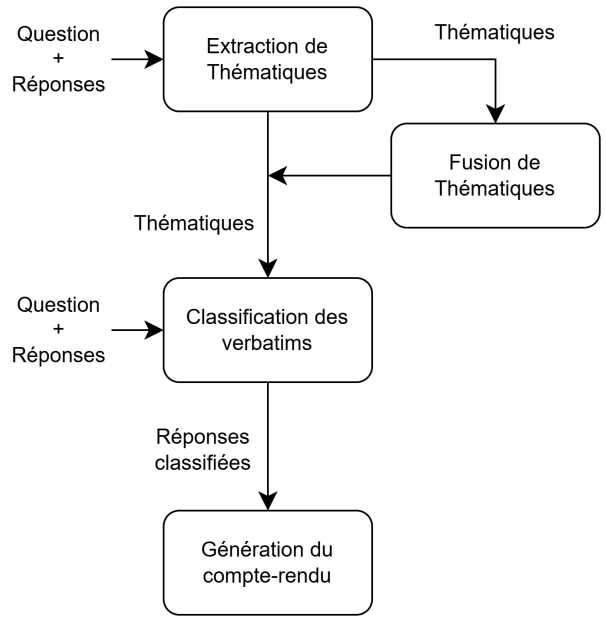

FIGURE 1 – Architecture du pipeline : de l'extraction des thématiques à la génération de compte-rendu.

La méthode proposée (figure 1) s'inspire du processus qu'un analyste humain suivrait naturellement pour traiter une enquête : lire un échantillon de réponses pour dégager les grandes thématiques, puis classer systématiquement l'ensemble des verbatims selon ces thématiques, et enfin synthétiser les résultats par thème. Chaque étape correspond à un appel LLM distinct, avec un *prompt* et un format de réponse dédiés. Cette séparation en étapes indépendantes permet une inspection humaine intermédiaire si nécessaire, et réduit fortement le risque d'hallucination par rapport à une approche en un seul *prompt*.

2.1 Extraction de thématiques

La première étape consiste à extraire les thématiques principales à partir d'un échantillon de réponses. Selon la taille de la fenêtre de contexte du LLM utilisé, tout ou partie des réponses sont fournies en entrée. À partir de ces réponses et, le cas échéant, de la question posée dans l'enquête, le LLM génère un ensemble de courtes phrases décrivant les thématiques identifiées (ex. : « Problème de Wi-Fi », « PC peu performants »).

Le *prompt* impose explicitement un nombre cible de thématiques à produire. Ce choix est motivé par un impératif opérationnel : trop de thématiques rendent le compte rendu final illisible et peu exploitable. Cette contrainte permet également de forcer le LLM à opérer des regroupements conceptuels plutôt que de produire une liste exhaustive de sous-thèmes. Cette étape est la plus critique du pipeline : des thématiques de mauvaise qualité ou trop fragmentées dégradent irrémédiablement la classification qui suit.

Le *prompt* utilisé est fourni en Annexe A.1.

2.2 Fusion de thématiques

Pour des questions très générales (ex. : « Quels problèmes rencontrez-vous avec l'informatique ? »), le LLM peut produire des thématiques partiellement redondantes (ex. : « problème avec le Wi-Fi » et « problème avec internet »). Une étape optionnelle de fusion permet de remédier à ce cas. Le LLM reçoit la liste complète des thématiques générées ainsi que la question d'enquête, et propose les regroupements pertinents sous la forme d'indices à fusionner.

Cette étape est particulièrement utile lorsque le modèle utilisé en étape 1 n'est pas un *reasoning model* ou manque de puissance pour respecter strictement la contrainte de quantité imposée dans le *prompt* d'extraction.

Le *prompt* utilisé est fourni en Annexe A.2.

2.3 Classification des verbatims

Une fois les thématiques établies, chaque verbatim est classifié. Le LLM reçoit la liste complète des thématiques (identifiées par un indice numérique et une courte description) ainsi qu'un lot de réponses à traiter, et associe à chaque réponse les indices des thématiques qui y sont effectivement mentionnées. Le LLM est explicitement autorisé à ne rien assigner si aucune thématique ne correspond.

Un point technique central est l'utilisation d'un *format de réponse* JSON strict (voir section 3.3.2), qui force le modèle à produire une sortie parseable et cohérente, et qui rappelle implicitement au modèle l'identité de chaque commentaire

à traiter. Les réponses sont traitées par lots de taille paramétrable, ce qui permet de trouver le meilleur compromis entre coût computationnel et qualité de classification.
Le *prompt* utilisé est fourni en Annexe A.3.

2.4 Génération du compte rendu

La dernière étape consiste à générer, pour chaque thématique, un résumé ainsi qu'un paragraphe de recommandations à partir des verbatims qui lui ont été assignés lors de la classification. En isolant les réponses par thème avant de les soumettre au LLM, on évite de noyer le modèle dans un corpus brut non structuré, ce qui améliore nettement la qualité et la précision des recommandations produites. Cette étape est conçue pour être facilement adaptable aux besoins spécifiques de l'organisation (format du rapport, ton, niveau de détail).
Le *prompt* utilisé est fourni en Annexe A.4.

3 Protocole Expérimental

3.1 Données

Les expérimentations ont été conduites sur un jeu de données constitué de 200 réponses à une question ouverte issue d'une enquête interne réelle. Ce corpus a fait l'objet d'une annotation manuelle servant de référence (*gold standard*), établie selon le protocole suivant : (1) extraction des thématiques par LLM, (2) vérification et légère correction manuelle des thématiques produites, (3) classification manuelle des 200 réponses par un annotateur unique.
Cette annotation d'une seule question par un seul annotateur constitue une limitation reconnue : l'idéal serait de disposer d'annotations portant sur plusieurs questions, réalisées par plusieurs annotateurs afin de mesurer la variabilité inter-annotateurs. Ce point est identifié comme une perspective d'extension du protocole.

3.2 Méthodes comparées

Plusieurs méthodes de NLP classiques ont été comparées à notre pipeline. Ces méthodes ne supportant pas l'assignation multi-label, la comparaison porte sur la cohérence des thématiques générées et sur la métrique ELM f1 score (voir section 3.4). Pour chaque méthode classique, de nombreuses combinaisons d'hyperparamètres ont été testées, ainsi que plusieurs modèles d'embeddings (Qwen3-embedding-8B, multilingual-e5-large-instruct, gemini-embedding).

3.2.1 Embeddings avec K-Means

K-Means génère un nombre de clusters fixé à l'avance. Son utilisation ici vise à évaluer dans quelle mesure les clusters produits à partir d'embeddings constituent une alternative à l'extraction de thématiques par LLM. Le nombre de clusters est fixé à 10 pour être comparable au nombre de thématiques cibles.

3.2.2 Embeddings avec UMAP [4] et HDBSCAN

Cette combinaison permet de déterminer automatiquement le nombre de clusters. Elle requiert cependant un ajustement fin de nombreux hyperparamètres (dimensions de réduction, taille minimale de cluster, seuils de densité), dont la sensibilité varie fortement d'une question à l'autre, ce qui compromet l'automatisation visée.

3.2.3 LLM en un seul *prompt*

Cette méthode soumet l'intégralité des réponses brutes au LLM en un unique appel, en lui demandant de produire directement le compte rendu. Elle constitue la baseline LLM naturelle : plus rapide (d'un facteur 10 à 100), elle permet d'évaluer l'apport de la décomposition en étapes distinctes de notre pipeline.

3.3 Études d'optimisation

3.3.1 *Prompts*

Les *prompts* ont été optimisés selon plusieurs axes : la langue (anglais vs. français), le nombre de verbatims traités par lot lors de la classification, le format des données en entrée (JSON structuré vs. texte brut), et la structure générale du *prompt*. Ces optimisations ont un impact déterminant sur les résultats.

3.3.2 Format de réponse

L'utilisation d'un *format de réponse* JSON strict est une composante essentielle du pipeline. Il garantit que les sorties du LLM sont systématiquement parseable, élimine les variabilités de formatage, et améliore la qualité des réponses en structurant implicitement la tâche pour le modèle. Par exemple, pour la classification, le *format de réponse* contraint les valeurs autorisées aux seuls indices de thématiques existants, rendant impossible l'hallucination d'un identifiant inexistant.

3.3.3 Choix du LLM

Le choix du LLM implique un arbitrage multi-critères : qualité des réponses, latence, coût, et confidentialité des données. Deux familles sont évaluées : les modèles distants accessibles via l'API interne de l'entreprise (Gemini, Mistral), qui offrent les meilleures performances mais impliquent l'envoi de requêtes réseau, et les modèles locaux exécutés sur un poste scientifique (HP Zbook, GPU RTX 500 Ada 4 Go VRAM via `llama.cpp`), garantissant la confidentialité pour les données sensibles. Les modèles locaux ont été sélectionnés pour offrir un temps d'inférence acceptable sur ce matériel, ciblant ainsi des modèles nécessitant moins de 4 Go de VRAM (à l'exception des poids des experts déchargés sur la RAM pour les architectures MoE). Le panel final regroupe les modèles les plus performants dans ces deux catégories sortis fin 2025.

3.4 Métriques d'évaluation

3.4.1 F1 score

Le F1 score est calculé entre la classification produite par le LLM à partir des thématiques de référence et les annotations manuelles. Il évalue exclusivement l'étape de classification et est utilisé pour comparer les LLM ainsi que les différentes configurations de *prompts*.

3.4.2 ELM f1 score (*ElementsLikeMe*)

Cette métrique permet de comparer des partitions en tenant compte de l'assignation multi-label, ce qui la rend adap-

tée à l'évaluation de l'ensemble du pipeline (de l'extraction à la classification). Elle est plus sévère que l'Information Mutuelle Normalisée (NMI - *Normalized Mutual Information*) classique : les méthodes produisant des thématiques incohérentes obtiennent des scores proches d'une assignation aléatoire, ce qui permet de discriminer efficacement les approches de bonne qualité.

4 Résultats

4.1 Analyse comparative des thématiques générées

Une analyse qualitative des thématiques a été privilégiée pour les méthodes classiques. En effet, ces dernières ne peuvent pas être évaluées quantitativement de la même manière que le pipeline proposé : leur incapacité à gérer l'assignation multi-label conduit systématiquement à des scores ELM f1 proches de l'aléatoire.

4.1.1 Embeddings avec K-Means

Avec un nombre de clusters fixé à 10, K-Means produit des résultats partiellement acceptables, mais plusieurs clusters regroupent des réponses relevant de thématiques distinctes, tandis que d'autres sur-spécialisent une même thématique. L'augmentation du nombre de clusters à 20 ou 30 suivie d'une étape de fusion n'élimine pas ces problèmes.

4.1.2 Embeddings avec UMAP et HDBSCAN

Cette méthode peut produire des résultats exploitables, mais uniquement avec une sélection précise des hyperparamètres. Or, les paramètres optimaux diffèrent selon les enquêtes, ce qui la rend impossible à automatiser sans intervention d'un expert pour chaque nouvelle enquête, ce qui est incompatible avec le contexte opérationnel visé.

4.1.3 LLM en un seul *prompt*

La méthode en un seul *prompt* produit des résumés et recommandations de qualité apparente satisfaisante, mais présente un problème opérationnel majeur : l'effet boîte noire. Le LLM peut halluciner des réponses, produire des résumés déconnectés des verbatims réels, ou fournir des proportions thématiques non fondées sur les données (lors des tests, le modèle a systématiquement produit des suites arithmétiques — ex. : 40 %, 36 %, 32 % — indépendamment des données). Il est donc impossible de valider les sorties sans relire l'intégralité des réponses, ce qui annule l'intérêt de l'automatisation.

4.2 Comparaison des LLM pour la classification

Les résultats sont séparés en deux tables correspondant aux deux scénarios de déploiement. Chaque évaluation a été lancée 5 fois. La table 1 concerne les modèles distants (Google, Mistral), plus performants mais impliquant l'envoi de données vers des serveurs tiers et un coût par requête. La table 2 concerne les modèles exécutés localement sur un poste scientifique (HP Zbook, RTX 500 4 Go VRAM, via llama.cpp), garantissant la confidentialité des données.
Le temps de traitement particulièrement élevé du modèle GPT-OSS-20B s'explique par son architecture orientée raisonnement (*reasoning model*). Pour une tâche de classification simple générant des réponses très courtes (50 à 100 tokens), sa longue phase de réflexion préalable (plusieurs milliers de tokens générés en interne) induit une surcharge computationnelle disproportionnée par rapport au gain de performance minime observé. Sans cette phase, sa vitesse serait supérieure à celle de Qwen3-30B-A3B-Instruct.
Pour illustrer l'apport des optimisations de *prompts*, deux modèles ont été testés avant et après optimisation : Gemini 2.5 Flash est passé d'un F1 de 0.72 à 0.754, et Qwen3-4B-Instruct de 0.58 à 0.678.

TABLE 1 – Comparaison de la classification des LLM en API

Modèle	f1 score	Temps (s)
Gemini 2.5 Flash	0.754	51.5
Mistral Medium	0.727	19.1
Gemini 2.5 Pro	0.712	130.5
Mistral Large 2	0.709	13.3
Mistral Small	0.693	16.3

TABLE 2 – Comparaison de la classification des LLM sur poste scientifique

Modèle	f1 score	Temps (s)
Qwen3-30B-A3B-Instruct (Q4_K_M)	0.698	142.7
Qwen3-4B-Instruct (Q6_K_M)	0.678	61.6
GPT-OSS-20B (Q8_0)	0.675	1452.2
Ministral-3-3B-Instruct (Q6_K_XL)	0.483	44.7
Qwen3-0.6B (Q8_0)	0.220	19.8

4.3 Comparaison des LLM de l'extraction à la classification

TABLE 3 – Comparaison de l'Extraction à la Classification

Modèle	ELM f1 score	Temps (s)
Gemini 2.5 Flash	0.287	90.28
GPT-OSS-20B & Qwen3-4B-Instruct	0.248	105.5
Qwen3-30B-A3B-Instruct (Q4_K_M)	0.204	262.41
Qwen3-4B-Instruct (Q6_K_M)	0.201	81.2
Aucune assignation	0.19	0
Random	0.16	0

Cette évaluation, plus coûteuse, a été lancée 10 fois par configuration en raison d'une variance élevée ; les scores 3 affichés sont des moyennes. La configuration « GPT-OSS-20B & Qwen3-4B-Instruct » utilise GPT-OSS-20B pour l'étape d'extraction et Qwen3-4B-Instruct pour la classification. Elle permet de démontrer un résultat important : bien que ces deux modèles aient des performances similaires en classification seule (table 2), l'utilisation d'un *reasoning model* à l'étape d'extraction améliore significative-

ment le score ELM global. Cela confirme que l'extraction de thématiques est le maillon critique du pipeline, et que le choix du modèle pour cette étape doit être prioritaire.

4.4 Évaluation de la génération de comptes rendus

Contrairement aux étapes de classification, l'étape finale de génération des recommandations n'a pas fait l'objet d'une évaluation par métrique automatique. La qualité d'un compte rendu est par essence subjective et fortement dépendante des attentes spécifiques (ton, formalisme) de l'entreprise et du commanditeur de l'enquête. En contexte opérationnel, cette étape a été validée qualitativement par les experts métiers. L'absence de « juge LLM » automatisé à cette étape répond également à une volonté de maintenir l'expert humain en validation finale du rapport.

5 Discussion

5.1 Adéquation aux objectifs et applicabilité industrielle

Contrairement aux méthodes classiques de clustering, notre pipeline automatise l'analyse sans requérir d'expertise en ML ni d'ajustement d'hyperparamètres, ce qui est essentiel pour un déploiement industriel. Il produit des thématiques sémantiquement cohérentes et gère la classification multi-label, reflétant fidèlement la réalité des verbatims où de multiples sujets peuvent être abordés. La décomposition en étapes garantit la contrôlabilité humaine et la reproductibilité via des *formats de réponse* stricts, évitant les erreurs de parsing en batch. Enfin, le pipeline est flexible : il permet l'usage de modèles locaux (ex. Qwen3-4B sur 4 Go VRAM) assurant la confidentialité des données tout en maintenant de bonnes performances (F1=0.678), ou de modèles distants pour une qualité maximale (F1=0.754). Indépendant de la langue, ce pipeline est conçu pour être mis en production à grande échelle.

5.2 Limites

L'évaluation quantitative repose sur un *gold standard* de 200 réponses annoté par un unique expert, avec une assistance initiale d'un LLM. Ce choix, bien qu'adapté aux contraintes industrielles, introduit un potentiel double biais (annotateur unique et effet d'ancrage). De futurs travaux nécessiteront des corpus de référence purement humains à annotation croisée pour garantir la robustesse absolue. Par ailleurs, la méthode entraîne un coût computationnel et financier non négligeable (fort volume de tokens envoyés en API), et reste fortement dépendante des capacités du LLM choisi, notamment l'importance cruciale d'un *reasoning model* pour l'étape d'extraction afin d'éviter l'étape de fusion.

6 Conclusion

Cet article présente un pipeline opérationnel complet d'analyse d'enquêtes à réponses ouvertes fondé exclusivement sur des LLM, conçu pour répondre aux contraintes concrètes d'un déploiement en entreprise : automatisation sans expertise en ML, reproductibilité, gestion de la confidentialité des données et production de résultats directement actionnables.

Les évaluations expérimentales montrent que les approches classiques (K-Means, UMAP + HDBSCAN) ne permettent pas de produire des thématiques cohérentes et exploitables de manière automatisée, en raison de leur sensibilité aux hyperparamètres et de leur incapacité à gérer l'assignation multi-label. À l'inverse, le pipeline proposé produit des regroupements thématiques de qualité supérieure, compatibles avec une classification multi-label, et directement mobilisables pour la génération de comptes rendus structurés.

Deux résultats quantitatifs méritent d'être soulignés. D'une part, l'optimisation des *prompts* et l'adoption de *formats de réponse* stricts apportent des gains significatifs de F1 score (jusqu'à +0.10 points), soulignant l'importance de l'ingénierie de *prompt* dans un contexte de production. D'autre part, l'utilisation d'un *reasoning model* à l'étape d'extraction est déterminante pour la qualité globale du pipeline, comme l'illustre l'écart observé sur la métrique ELM f1 score.

La décomposition en étapes distinctes — extraction, fusion optionnelle, classification, génération de rapport — constitue un compromis efficace entre automatisation et contrôlabilité, limitant les hallucinations observées dans les approches en un seul *prompt*. Elle contribue également à la reproductibilité du processus et facilite son adaptation à de nouveaux contextes applicatifs.

Parmi les perspectives d'extension, on peut citer : l'évaluation sur plusieurs questions annotées par différents annotateurs pour mesurer la robustesse inter-domaines ; l'exploration de stratégies hybrides combinant embeddings et LLM pour réduire le coût computationnel ; et l'adaptation du pipeline à d'autres types d'enquêtes qualitatives (entretiens, forums, avis clients).

Remerciements

Ce travail a été réalisé dans le cadre d'un stage au sein de la DATA & IA Factory d'EDF (DTEO-CEDID). L'auteur tient à exprimer sa gratitude à Claude Caligaris et Sean Manzambi pour leur encadrement technique et leur expertise tout au long du projet. Un remerciement particulier est adressé à Margarita Roos pour sa contribution précieuse et ses conseils lors de la rédaction de cet article. Enfin, nous remercions l'ensemble des membres de l'équipe pour leurs retours constructifs lors de nos points hebdomadaires.

Usage de l'IA générative

L'intelligence artificielle générative a été utilisée uniquement lors de la phase finale de rédaction pour la révision linguistique et l'amélioration du style rédactionnel.

Références

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Jianghan Liu, Ziyu Shang, Wenjun Ke, Peng Wang, Zhizhao Luo, Jiajun Liu, Guozheng Li, and Yining Li. LLM-guided semantic-aware clustering for topic modeling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 18420–18435, Vienna, Austria, July 2025. Association for Computational Linguistics.

[3] Leland McInnes, John Healy, and Steve Astels. hdbscan : Hierarchical density based clustering. *The Journal of Open Source Software*, 2 :205, 03 2017.

[4] Leland McInnes, John Healy, and James Melville. Umap : Uniform manifold approximation and projection for dimension reduction, 2020.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[6] Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. Topicgpt : A prompt-based topic modeling framework, 2024.

[7] Juan Enrique Ramos. Using tf-idf to determine word relevance in document queries. 2003.

[8] Han Wang, Nirmalendu Prakash, Nguyen Khoi Hoang, Ming Shan Hee, Usman Naseem, and Roy Ka-Wei Lee. Prompting large language models for topic modeling, 2023.

[9] Xiaobao Wu, Thong Nguyen, and Anh Tuan Luu. A survey on neural topic models : methods, applications, and challenges. *Artificial Intelligence Review*, 57(2), January 2024.

A *Prompts* utilisés

A.1 *Prompt* d'extraction de thématiques

Prompt : Extraction de thématiques

System :
Tu vas recevoir une liste de réponses à une question donnée. Ta mission est d'extraire les thématiques principales abordées dans ces commentaires.
Instructions :
1. **Analyse et Synthèse** : Identifie les sujets récurrents ou significatifs.
2. **Rédaction** : Pour chaque thématique, rédige une phrase courte et concise en français qui la résume.
3. **Contrainte de Quantité** : Si tu identifies plus de $n_topics thématiques, fusionne impérativement les sujets les plus proches pour respecter cette limite.
4. **Format de Sortie** : Réponds exclusivement sous la forme d'un objet JSON contenant deux clés :
- 'reasoning' : Une explication brève de ta logique de regroupement.
- 'sentences' : Un tableau de chaînes de caractères contenant les phrases finales.
Ne produis aucune explication en dehors du bloc JSON.
User :
Question : $question
Réponses utilisateurs :
$comments

A.2 *Prompt* de fusion de thématiques

Prompt : Fusion de thématiques

System :
Tu es un assistant expert en analyse sémantique de topics issus d'enquêtes utilisateurs. Ta tâche est d'identifier les topics qui sont très similaires ou suffisamment proches pour être fusionnés.
Tu dois :
- Proposer des regroupements uniquement si la similarité est forte (formulation ou intention proche).
- Ne pas forcer la fusion si les différences sont significatives.
- Si aucune fusion n'est pertinente, ne fais pas de fusion.
- Ton objectif est de faciliter la simplification du compte rendu tout en conservant la précision des thèmes abordés.
- Il doit rester uniquement une dizaine de topics après la fusion
User :
Voici la question posée dans l'enquête :
'$question'
Voici la liste des topics à analyser :
"'json
$descs "'
Donne en sortie uniquement une liste de listes d'indices représentant les groupes de descriptions à fusionner. Par exemple, si les descriptions aux indices 2, 5 et 7 doivent être fusionnées, retourne : [[2, 5, 7]]. Si plusieurs groupes doivent être fusionnés, retourne plusieurs sous-listes. Si aucune fusion n'est nécessaire, retourne une liste vide : [].

A.3 *Prompt* de Classification

Prompt : Classification

System : Tu es un assistant de classification. Tu recevras : - Une liste de thèmes (chacun avec un identifiant et une description courte en français). - Une liste de commentaires rédigés par des employés en réponse à la question : $question Ta tâche : Pour chaque commentaire, identifier tous les thèmes auxquels le commentaire fait référence. Il faut que tu sois totalement certain, sinon n'associe pas le thème. Output : Retourne un JSON où les clés sont 'Comment x', (x commence à 0) et la valeur est une liste contenant les ids de thèmes correspondant (peut être vide).
User : Classe les commentaires suivants selon les thèmes proposés.
Input : $json

A.4 *Prompt* de Génération de recommandation

Prompt : Génération de recommandation

System : Tu es un expert en communication d'entreprise, spécialisé dans l'analyse de retours employés et la formulation de recommandations concrètes.
User : Tu analyses une enquête interne portant sur le sujet suivant : $topic
La question posée à l'enquête est : $question
Voici des citations directes issues des réponses des employés : $comments
À partir de ces données, génère **un seul paragraphe court et concis** qui propose des pistes d'amélioration et des actions concrètes à mener.
- Le paragraphe doit être centré **exclusivement sur le sujet $topic**.
- Ne mentionne aucun autre sujet, aucune information hors contexte.
- Utilise un ton professionnel, clair et orienté action.
- Aucune mise en forme (pas de liste, pas de titres, pas d'astérisques).
- Aucun commentaire sur les méthodes ou l'analyse des données — seulement les recommandations.
Seulement le paragraphe. Rien d'autre.