SPARK : Exploiter les échanges techniques passés pour améliorer le support client

Steve Bellart[1] Arnaud Deleruyelle[1]
(1) Talan, 14 rue Pergolèse, 75116 Paris, France
steve.bellart@talan.com, arnaud.deleruyelle@talan.com

RÉSUMÉ

S.P.A.R.K. (SAP Process Augmented Response Knowledge) est un projet qui vise à concevoir une architecture de génération augmentée par récupération (RAG) adaptée aux dialogues techniques issus de requêtes clients. L'objectif est d'améliorer l'efficacité des consultants SAP dans la résolution des demandes clients en exploitant les connaissances contenues dans des échanges antérieurs similaires. Chaque ticket résolu contient un dialogue entre un client décrivant un problème et un consultant proposant une solution technique détaillée. L'accès rapide à ces solutions déjà éprouvées constitue un atout majeur. Cet article aborde les défis spécifiques liés à l'exploitation des données conversationnelles techniques, présente les solutions proposées pour optimiser la récupération et la génération de réponses pertinentes, et traite des perspectives futures.

ABSTRACT

SPARK : Leveraging Past Technical Dialogues to Enhance Customer Support

S.P.A.R.K. (SAP Process Augmented Response Knowledge) is a project aimed at designing a Retrieval-Augmented Generation (RAG) architecture tailored to technical dialogues derived from customer support requests. Its goal is to enhance the efficiency of SAP consultants in resolving client requests by leveraging knowledge contained in similar past interactions. Each previously resolved ticket includes a dialogue where the client describes an issue, and the consultant provides a detailed technical solution. Quick access to these proven solutions represents a significant advantage. This paper addresses specific challenges related to utilizing conversational technical data, introduces proposed solutions to optimize retrieval and relevant response generation, and discusses future directions for the project.

MOTS-CLÉS : RAG, SAP, Embeddings, Protection des données.

KEYWORDS: RAG, SAP, Embeddings, Privacy.

1 Introduction

Les centres de services (CDS) des entreprises disposent généralement d'archives regroupant l'ensemble des tickets résolus par le passé. Chaque ticket comprend à la fois un besoin (dysfonctionnement, demande d'évolution, de développement ou d'expertise spécifique) exprimé par un client, ainsi qu'une solution détaillée fournie par le ou les consultant(s) et/ou expert(s) chargé(s) de sa résolution. Généralement sous forme conversationnelle, ces échanges entre clients et CDS contiennent une richesse d'informations issues d'expertises accumulées sur des centaines, voire des milliers de cas déjà traités.

Grâce à une architecture de génération augmentée par récupération (RAG), l'intérêt d'exploiter ces données historiques pour assister les consultants face à de nouveaux tickets similaires ou identiques est devenu possible. Une telle approche permettrait aux consultants des CDS de ne plus systématiquement repartir de zéro, réduisant ainsi la dépendance exclusive à l'expertise individuelle des consultants.

Cependant, contrairement à l'utilisation traditionnelle de RAG sur des documents rédigés, le format conversationnel des données sources, leur caractère technique spécifique (ici lié aux systèmes SAP) et la sensibilité potentielle des informations qu'elles contiennent posent plusieurs défis majeurs. En plus de la sélection des modèles d'IA les plus adaptés, ces aspects doivent être pris en compte pour garantir la performance et la sécurité de l'outil développé.

Cet article revient d'abord sur des travaux antérieurs pertinents, puis détaille les défis spécifiques à notre contexte, puis expose les solutions actuellement déployées ainsi que les perspectives futures du projet, celui-ci étant toujours en cours de développement.

2 Travaux antérieurs

Les modèles de RAG ont suscité un intérêt croissant dans le domaine du traitement automatique du langage naturel, en raison de leur capacité à produire des réponses informatives, contextualisées et vérifiables. Cette architecture hybride combine un modèle génératif de type LLM avec un composant de récupération documentaire, permettant de dépasser les limites des modèles purement paramétriques dont la connaissance est figée dans leurs poids. Introduit par Lewis et al. (Lewis *et al.*, 2020), le modèle RAG s'est notamment illustré dans des tâches de question-réponse ouverte, en réduisant significativement les phénomènes d'hallucination par l'appui sur des contenus externes vérifiables.

Diverses variantes ont été proposées pour raffiner cette architecture, telles que RAG-Sequence et RAG-Token (Lewis *et al.*, 2020), Fusion-in-Decoder (Izacard & Grave, 2021), REALM (Guu *et al.*, 2020) ou encore RETRO (Borgeaud *et al.*, 2022), explorant des stratégies alternatives d'intégration de l'information récupérée. Plus récemment, le modèle Atlas (Izacard *et al.*, 2022) a démontré l'intérêt de combiner le paradigme RAG avec des approches *few-shot*, en injectant directement dans le prompt du modèle de langage des exemples de tâches similaires afin de guider plus finement la génération.

L'application de ces approches aux données conversationnelles a principalement été étudiée dans le cadre de chatbots. Des travaux comme MultiDoc2Dial (Feng *et al.*, 2021) ont proposé des mécanismes de segmentation documentaire permettant de récupérer, à partir de plusieurs sources hétérogènes, les informations pertinentes nécessaires à la génération de réponses cohérentes dans des dialogues complexes. D'autres approches, telles que BlenderBot 3 (Shuster *et al.*, 2022b) ou les architectures modulaires présentées dans (Shuster *et al.*, 2022a), ont amélioré la pertinence d'un RAG comme moteur d'un chatbot. Plus récemment, RAGate (Wang *et al.*, 2024) a introduit un mécanisme de filtrage dynamique du contexte, optimisé pour les interactions en dialogue ouvert.

En revanche, l'utilisation des échanges conversationnels techniques, tels que les tickets issus de centres de service, comme source d'entrée structurée dans une architecture RAG demeure peu explorée. Bien que ces dialogues constituent une ressource précieuse pour capturer des cas d'usage réels, leur exploitation nécessite des traitements spécifiques, notamment pour structurer l'information de manière exploitable par les systèmes génératifs. Dans ce contexte, plusieurs travaux ont proposé des méthodes de création automatique de résumés à partir de conversations. Des systèmes tels que ceux décrit dans (Zhang *et al.*, 2021) ont montré de bonnes performances sur des corpus comme SummScreen (Chen

et al., 2021) ou MediaSum (Zhu *et al.*, 2021), confirmant la faisabilité d'une synthèse efficace des conversations informelles. Enrichir ces résumés par des représentations structurées, telles que des graphes (Chen & Yang, 2021; Gao *et al.*, 2023) s'est également révélé bénéfique pour la cohérence et la fidélité des contenus générés.

Par ailleurs, des approches non supervisées ou faiblement supervisées, s'appuyant sur des graphes lexicaux (Park & Lee, 2022) ou des paradigmes auto-apprenants, offrent des perspectives intéressantes pour des contextes à faibles ressources ou en l'absence d'annotations manuelles. Enfin, l'émergence des grands modèles de langage a marqué une nouvelle étape : ceux-ci sont capables, via un simple prompt, de produire des résumés conversationnels de bonne qualité, y compris dans des domaines aussi exigeants que le médical (Giorgi *et al.*, 2023).

Malgré ces avancées, l'application des modèles RAG à des données conversationnelles spécialisées, comme les tickets issus de centres de service dans des environnements techniques, présente encore plusieurs verrous. Parmi ceux-ci figurent la prise en compte du vocabulaire métier spécifique, la dilution des informations essentielles au sein de longues séquences dialoguées, ainsi que les contraintes liées à la sensibilité ou à la confidentialité des données. La mise en œuvre d'architectures robustes, respectueuses de ces contraintes, constitue donc un défi central pour l'adaptation de RAG dans ce contexte applicatif.

3 Les défis de l'exploitation de tickets techniques

L'application de l'architecture RAG sur les tickets issus d'un centre de services pose un ensemble de défis spécifiques liés à la nature, la structure et le contenu des données traitées. Contrairement aux documents classiques, un ticket n'est pas un texte figé : il s'agit d'un échange conversationnel dont seule la situation initiale est connue au moment de sa réception. La recherche de similarité doit donc être envisagée de manière asymétrique, en comparant un problème partiel avec des cas complets (problème + résolution). Cette section détaille les principales difficultés identifiées dans le cadre de notre étude.

La protection des données Les données issues d'un centre de services contiennent fréquemment des éléments sensibles : noms de clients, de sociétés, de consultants, ou encore références internes à des projets ou infrastructures techniques. Leur anonymisation peut par ailleurs constituer un impératif contractuel ou réglementaire, mais aussi méthodologique.

D'une part, du point de vue contractuel, un consultant traitant une demande ne doit pas être exposé à des informations permettant d'identifier les clients ayant déjà rencontré un problème similaire. D'autre part, la présence de ces éléments nominaux biaise les mécanismes d'attention des modèles d'embedding, ceux-ci ayant tendance à surpondérer les entités nommées (noms propres, acronymes, etc.) dans le calcul de similarité. Cela peut entraîner un appariement artificiel entre deux tickets mentionnant un même client, indépendamment de la nature réelle du problème. La figure 1 illustre ce phénomène : des tickets sans lien thématique se retrouvent artificiellement proches dans l'espace des embeddings en raison de la répétition d'un même nom d'entreprise ((Zeng *et al.*, 2024) détaille l'enjeu de l'anonymisation dans les RAG).

Statistique	Valeur
Nb moyen de tokens par ticket	1620
Nb médian de tokens par ticket	1124
Écart-type sur le nb de tokens	1572
Nb moyen de messages par ticket	10.62
Nb médian de messages par ticket	7
Écart-type sur le nb de messages	10.49
Nb total de tickets analysés	170

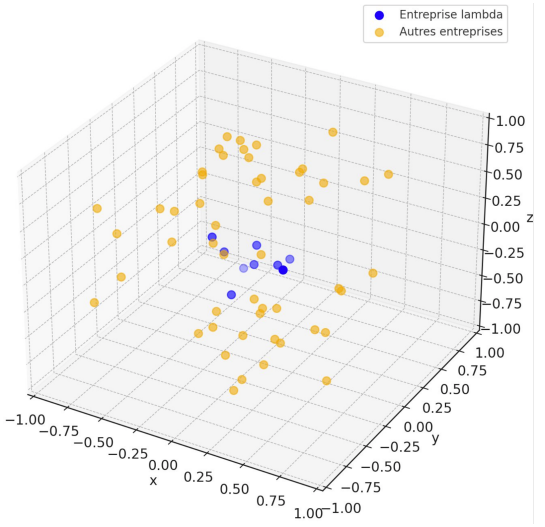

FIGURE 1 – Exemple de rapprochement entre tickets du même client malgré des sujets de conversations différents. La représentation résulte d'une projection via une analyse en composante principale (ACP) des vecteurs d'embedding. Le tableau présente les statistiques descriptives des tickets analysés. Les estimations du nombre de tokens ont été effectuées avec le tokenizer cl100k_base (utilisé notamment par les modèles GPT-4 et GPT-3.5-turbo).

L'aspect conversationnel Les échanges entre clients et consultants suivent généralement une dynamique asynchrone, entrecoupée de clarifications, d'allers-retours, voire de digressions. L'information clé (problème métier, diagnostic, solution) est ainsi diluée dans le flux conversationnel. Par exemple, dans un ticket concernant un problème de calcul du coût de revient "CCR" sur SAP, la cause réelle (configuration manquante dans une transaction tierce) peut n'apparaître qu'après plusieurs échanges. À l'inverse, certains échanges sont très courts et contiennent peu ou pas d'informations pertinentes, résultant en une grande divergence sur la qualité et la quantité des informations d'un ticket (voir tableau de la figure 1).

Exemple de ticket – Problème de calcul variante B8 transaction CK40N

[CLIENT] : Bonjour, nous rencontrons un problème sur le calcul de la variante B8 via CK40N. Les variantes ZT00, ZT01 et ZT04 fonctionnent correctement, seule ZT08 bloque.

[SUPPORT] : Bonjour, votre demande a été transmise à l'équipe de consultants.

[CONSULTANT1] : Bonjour, nous analysons la situation et revenons vers vous dès que possible.

[CONSULTANT2] : Bonjour, j'ai repris l'analyse. Rien de bloquant a priori. Les dates présélectionnées ne semblent pas en cause. Les erreurs sur l'ID Z1 apparaissent aussi sur ZT00/01/04 sans empêcher le calcul.
Message bloquant observé sur ZT08 : "Témoin de suppression apposé au sous-ens. [ARTICLE] périm. valor. 1000" (CK054).
Diagnostic : segment de valorisation marqué comme supprimé.
Procédure : retirer le témoin si calcul souhaité.

[CLIENT] : Merci. L'article [ARTICLE] est bien exclu dès la première étape de CK40N, comme pour les autres variantes.

[CLIENT] : Nous pensons que ZT08 a été configurée pour bloquer à chaque anomalie, contrairement aux autres. Pouvez-vous vérifier et l'aligner avec ZT00 ou ZT04 ?

[CONSULTANT2] : Merci pour la précision. Cela confirme la piste. On creusera ce point.

[CONSULTANT1] : La configuration de ZT08 a été alignée avec les autres. Les modifications sont en QAL, OT inclus dans la requête.

[CLIENT] : Bien reçu.

[CONSULTANT1] : Test réalisé sur ZT08 : plus d'erreur bloquante. Transport prévu le 10/01.

[CLIENT] : Transport effectué le 11/01.

[CLIENT] : Pouvez-vous clôturer le ticket ?

[CONSULTANT1] : C'est fait.

FIGURE 2 – Dialogue extrait et anonymisé de notre jeu de données. Les éléments anonymisés sont les éléments entre crochets. Ce ticket illustre un problème de calcul de coût de revient dans SAP. CK40N désigne la transaction SAP permettant de calculer les coûts ; les codes ZT** correspondent à différentes variantes de calcul configurables ; CK054 est un message d'erreur système ; QAL représente l'environnement de qualification (tests avant production) ; OT désigne un ordre de transport pour déployer les modifications. Les mentions [ARTICLE] ont été anonymisées.

Ce phénomène de dispersion s'oppose à l'efficacité des représentations vectorielles classiques, qui capturent mal les relations longues ou implicites entre les répliques. Il devient alors difficile d'indexer ou de comparer efficacement les tickets, d'autant plus si le problème n'est jamais clairement explicité dans la conversation.

La mise en place de métriques L'évaluation d'un système RAG repose généralement sur la mesure de la qualité des documents récupérés ou sur la qualité de la génération finale. Dans notre cas, nous voulons dans un premier temps mesurer la qualité des documents récupérés (ici des tickets archivés), mais nous n'avons pas d'annotation sur la similarité entre les tickets a priori.

Le vocabulaire technique et spécifique Les modèles d'embedding couramment utilisés ont été entraînés sur des données génériques (Wikipedia, StackExchange, etc.) et présentent des limites dans la représentation de termes spécialisés. Or, les tickets d'un CDS font généralement appel à un vocabulaire métier très spécifique, comprenant des abréviations internes, des noms de modules, des codes transactionnels ou encore des syntaxes propres à des environnements particuliers.

Ces termes rares ou absents des corpus d'entraînement entraînent une perte d'information lors du passage en vecteurs, rendant difficile la comparaison de tickets traitant de problématiques techniques similaires mais exprimées dans un langage métier.

4 Pré-traitement pour une similarité efficace

Face aux défis décrits ci-dessus, le projet SPARK vise à concevoir une architecture progressive et robuste pour exploiter les archives de tickets d'un CDS. Le projet est encore en cours de développement, mais plusieurs modules ont d'ores et déjà été implémentés. L'objectif initial a été de fiabiliser la détection de similarité entre tickets, avant d'exploiter plus largement les cas similaires pour assister la résolution automatique ou semi-automatique de nouveaux tickets.

Ce projet s'inscrit dans le contexte d'une intégration avec un système de gestion des incidents et services (ISTM - Incident and Service Management) déployé sur SAP Solution Manager (SOLMAN). SAP (Systems, Applications & Products in Data Processing) est un éditeur allemand de solutions logicielles d'entreprise, notamment spécialisé dans les systèmes de planification des ressources d'entreprise (ERP). Le module ISTM de SAP Solution Manager centralise la gestion des tickets de support et des demandes de service, générant un flux de données via les services CDS (Core Data Services). Pour exploiter ce flux de données, le projet est développé sur la plateforme Business Technology Platform (BTP) de SAP (SAP, 2024). En particulier, nous avons conçu le pipeline décrit par le schéma 3, qui vise à résoudre les problèmes mentionnés dans la section précédente, notamment

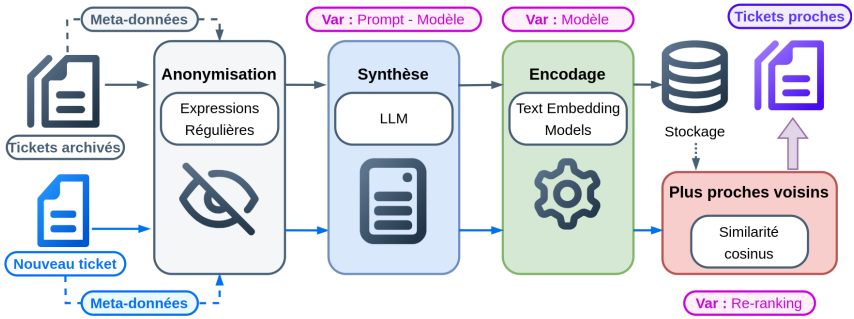

FIGURE 3 – Pipeline de traitement des tickets, pour la récupération automatique des demandes similaires. Les tickets archivés suivent ce flux jusqu'à un stockage des synthèses et vecteurs générés. Un nouveau ticket suit le même flux (avec un prompt adapté lors de la **Synthèse** prenant en compte le fait que nous n'avons à disposition que l'en-tête et le premier message du client à disposition). Une fois le vecteur calculé, celui-ci est comparé a ceux dans le stockage afin de détecter les tickets proches de ce nouveau ticket. Les cellules **Var** donnent les composants à paramétrer pour chacune des parties de la pipeline.

par un pré-traitement des données d'origine et par la mise en place d'un protocole de test pour aiguiller les différents choix de conception.

Anonymisation des données En premier lieu, un nettoyage systématique des tickets a été effectué afin de retirer toute information nominative. Grâce à la connaissance préalable des participants à chaque ticket (client, consultant, entreprise) dans leur métadonnées, un ensemble de règles basées sur des expressions régulières a été mis en place pour anonymiser les noms, prénoms, noms d'entreprise et autres informations sensibles de façon efficace. Ce prétraitement permet non seulement de se conformer aux exigences légales de confidentialité, mais aussi d'éviter des biais dans la mesure de similarité entre tickets.

Synthèse des échanges Pour pallier la dilution d'information dans les dialogues, nous avons expérimenté l'utilisation de modèles LLM pour générer une synthèse structurée de chaque ticket. À partir d'un échange brut, le modèle a un prompt lui demandant d'extraire les éléments clés : description du problème initial "PROBLÈME", causes identifiées "PISTES", solution mise en œuvre "SOLUTION". Par exemple, à partir du ticket 2, nous obtenons la figure 4.

Exemple de ticket – Problème de calcul variante B8 transaction CK40N

PROBLÈME : Le [CLIENT] rencontre un dysfonctionnement lors du calcul de la variante B8 dans la transaction CK40N. Contrairement aux variantes ZT00, ZT01 et ZT04, qui s'exécutent correctement, la variante ZT08 échoue systématiquement sans permettre l'affichage complet des erreurs de traitement. Ce blocage empêche l'analyse complète du calcul du coût de revient pour les articles concernés. Le [CLIENT] précise que l'article [ARTICLE] est bien exclu du calcul dès la première étape.
PISTES : Après la prise en charge, plusieurs vérifications sont menées. Une première analyse montre que les autres variantes (ZT00, ZT01, ZT04) génèrent elles aussi des messages d'erreur (liés notamment à un ID de gamme incorrect), mais sans empêcher l'exécution complète. Pour ZT08, un message spécifique de type `CK054` est identifié, signalant un "témoin de suppression" sur un segment de valorisation. Toutefois, cette erreur n'est pas considérée comme la cause principale du blocage. L'hypothèse privilégiée est que ZT08 a été configurée de façon plus stricte : toute anomalie interrompt immédiatement le processus, ce qui empêche de voir l'ensemble des erreurs. Une discussion avec le [CLIENT] confirme cette hypothèse.
SOLUTION : Réalisation d'une modification de la configuration de la variante ZT08 afin qu'elle adopte le même comportement que ZT00 et ZT04, c'est-à-dire tolérer les erreurs techniques sans interrompre le calcul global. Cette modification est réalisée dans l'environnement de qualité (QAL) et validée par des tests [CLIENT], qui confirment l'absence d'erreur bloquante. Le transport de la correction en production est effectué le 11/01, clôturant ainsi le ticket.

FIGURE 4 – Résumé de la figure 2

Ce résumé synthétique permet ensuite une représentation vectorielle plus compacte, réduisant le bruit conversationnel et facilitant les comparaisons avec d'autres tickets, en particulier les blocs "PROBLÈME" et "SOLUTION" résumant la situation initiale et la solution finalement apportée.

5 Mise en place de métrique et benchmark

Pour évaluer la qualité des rapprochements entre tickets basés sur la similarité cosinus de leurs vecteurs, nous avons constitué un jeu d'évaluation comprenant 24 tickets manuellement annotés comme proches (ou non) d'une part, et d'autre part d'un ensemble de 146 autres tickets sur des sujets différents, soit un corpus de 170 tickets au total sur les plus de 120 000 tickets que nous possédons. Étant donné l'absence d'annotations initiales, nous avons décidé de nous concentrer sur un champ réduit de tickets et d'alterner de façon itérative, annotation et évaluation de la solution pour la suite du projet. Les annotations des 24 tickets sélectionnés ainsi que les résumés associés à ces tickets ont été validés par trois experts métier. Cette base nous permet de faire une première évaluation de la capacité des différents prétraitements et choix de conceptions, dans la détection des tickets réellement pertinents.

Nous souhaitons à terme annoter de façon fine un jeu de référence de 500 tickets clients, en utilisant comme pré-sélectionneur les différentes itérations du projets.

5.1 Métrique

Afin d'évaluer la pertinence des similarités identifiées entre tickets, nous introduisons une métrique simple mais informative, fondée sur ce jeu de 24 tickets répartis en trois groupes thématiques de 8 tickets chacun. Les tickets d'un même groupe sont considérés comme similaires entre eux.

Pour chaque ticket t_i de ce jeu, nous calculons les 7 plus proches voisins $\{v_1, v_2, \ldots, v_7\}$ dans l'espace des embeddings vectoriels contenant un les vecteurs des 170 tickets. Chaque voisin v_k (avec $k \in \{1, \ldots, 7\}$) rapporte un score pondéré si celui-ci appartient au même groupe que t_i :

$$S(t_i) = \sum_{k=1}^{7} \delta_{i,k} \cdot \frac{1}{k}$$

où $\delta_{i,k} = 1$ si v_k appartient au même groupe que t_i, et 0 sinon. Cette pondération donne plus d'importance aux voisins les plus proches dans le classement.

Le score global est obtenu en normalisant la somme des scores de tous les tickets par le score maximal théorique (obtenu si tous les voisins sont corrects pour chaque ticket) :

$$S_{tot} = \frac{1}{N \cdot S} \sum_{i=1}^{N} S(t_i)$$

où $N = 24$ est le nombre total de tickets utilisés dans ce score, et $S = \sum_{k=1}^{7} \frac{1}{k}$ est le score maximal qu'un ticket peut atteindre. Cette métrique retourne donc une valeur entre 0 et 1, où 1 indique que tous les tickets ont retrouvé uniquement des voisins similaires dans leur groupe, en ordre parfait.

5.2 Stratégies

Le pipeline présenté en figure 3 a été exploré sous différentes configurations afin d'évaluer l'impact des choix de conception sur la qualité des similarités détectées, mesurée via la métrique décrite précédemment. Chaque composante annotée *Var* dans le schéma correspond à un paramètre variable du système testé comparativement. Deux variantes de prompt ont été évaluées pour générer les résumés de tickets : un prompt manuel basé sur les principes du *prompt engineering* (RESUME), et une version optimisée automatiquement avec PromptPerfect (PERFECT PROMPT RESUME (Jina AI, 2024)). Plusieurs modèles d'encodage ont été testés pour la génération des embeddings textuels : TEXT EMBEDDING 3 (LARGE et SMALL) et ADA 002 (OpenAI), TITAN EMBED TEXT (Amazon), ainsi que SAP NEB (SAP). En parallèle, les résumés ont été générés à l'aide de LLMs tels que GPT-4o, Gemini 1.5 Pro et Claude 3.5 Sonnet. Enfin, une stratégie de re-ranking a été appliquée pour affiner le tri final des tickets similaires : après génération du résumé du ticket cible, un prompt extrait une liste de 10 mots-clés représentatifs. La présence de ces mots dans les résumés des 10 tickets les plus proches (selon la similarité cosinus initiale) attribue un score, et les résultats sont réordonnés selon ce score, avec la similarité cosinus comme critère de départage.

Les prompts pour la génération des résumés que nous avons utilisés sont fournis dans la section Annexes.

5.3 Résultats et analyse

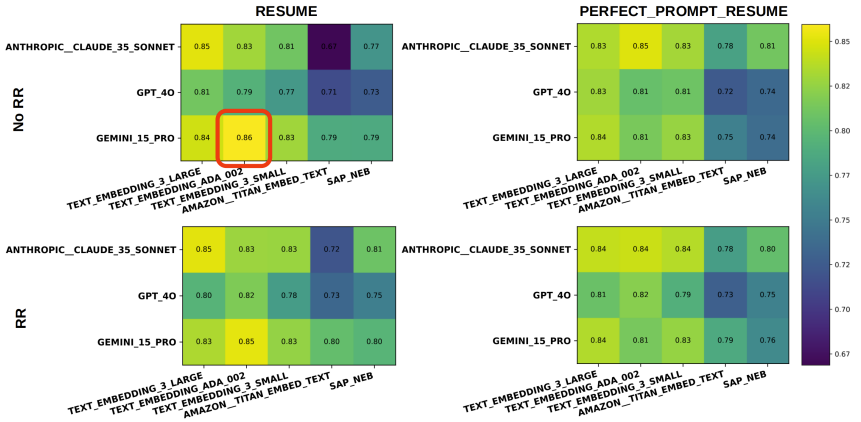

FIGURE 5 – Score S_{tot} obtenu en fonction des différentes stratégies testées. Le meilleur score est atteint avec une combinaison utilisant *Gemini 1.5 Pro* pour la génération de résumés, *ADA 2* pour le calcul des embeddings, le prompt d'origine (manuel), et sans re-ranking basé sur mots-clés.

La figure 5 présente les résultats obtenus pour l'ensemble des configurations testées, mesurés via le score S_{tot}. Cette visualisation permet de comparer l'impact relatif de chaque composant du pipeline : type de prompt, choix du LLM, modèle d'embedding, et application (ou non) d'un re-ranking lexical.

Re-ranking lexical L'ajout de la stratégie de re-ranking basée sur la détection de mots-clés permet d'améliorer certains cas peu performants en rehaussant la pertinence locale. Toutefois, elle tend également à dégrader légèrement les configurations initialement les plus efficaces. Globalement, nous observons une amélioration marginale : la moyenne des scores passe de $S_{tot} = 0.79$ sans re-ranking à $S_{tot} = 0.80$ avec re-ranking. Ce gain, bien que réel, reste peu significatif, d'autant plus que la meilleure configuration se voit légèrement pénalisée par cette étape additionnelle.

Formulation des prompts Les performances obtenues avec les deux types de prompts (manuel vs. optimisé via PromptPerfect) sont globalement proches. De manière quelque peu contre-intuitive, le prompt manuel s'avère légèrement plus efficace. Nous attribuons cette différence au style des résumés produits : les résumés générés avec le prompt optimisé sont souvent plus formels, tandis que ceux issus du prompt manuel sont plus concis et directs. Or, dans notre contexte asymétrique, où seul le problème est connu pour le nouveau ticket, un résumé plus simple semble favoriser une meilleure mise en correspondance. Des tests antérieurs avaient d'ailleurs montré des résultats inversés lorsque cette asymétrie n'était pas prise en compte.

Choix des modèles L'analyse globale révèle que le couple *LLM + modèle d'embedding* constitue le facteur ayant le plus d'impact sur la performance finale. Pour la stratégie de prompt et de re-ranking variante, on observe une stabilité des performances selon la combinaison de modèles utilisée. Les modèles d'embedding d'OpenAI se distinguent nettement, indépendamment du LLM avec lequel ils sont associés à l'inverse de GPT-4o qui est en retrait. Les meilleures combinaisons sont obtenues en associant un LLM non-OpenAI (comme *Gemini 1.5 Pro* ou *Claude 3.5 Sonnet*) avec un modèle d'embedding OpenAI.

Meilleure configuration La meilleure configuration atteint un score de $S_{tot} = 0.86$, ce qui témoigne d'un fort pouvoir discriminant, mais laisse encore apparaître des cas d'erreur. Une analyse qualitative a mis en évidence un exemple caractéristique : un ticket issu des 146 tickets hors-sujets a été identifié comme le 3^{e} voisin le plus proche d'un ticket du jeu de référence. Fait marquant, les deux résumés commencent exactement par la même phrase : *"Le client rencontre un problème de ..."*, malgré des contenus ensuite divergents et un vocabulaire technique très différent. Ce type d'erreur souligne une limite des modèles actuels d'embedding, qui semblent encore accorder une importance excessive à la structure phrastique plutôt qu'au contenu technique réel.

Cette observation renforce notre volonté de mettre en œuvre, dans les travaux à venir, un *fine-tuning* supervisé sur notre propre corpus, afin d'adapter les représentations vectorielles au niveau de granularité et au vocabulaire spécifique des tickets métiers.

6 Conclusion et perspectives

L'exploitation des archives de tickets d'un centre de service via des modèles RAG constitue une opportunité prometteuse pour accélérer la résolution des incidents techniques. Toutefois, la nature conversationnelle, technique et sensible de ces données rend leur traitement plus complexe que celui de documents classiques.

Le projet SPARK a permis d'initier une première architecture adaptée à ce contexte, en s'appuyant sur l'anonymisation des échanges, la synthèse automatique des dialogues et une évaluation rigoureuse de la détection de similarité. Les premiers résultats confirment la faisabilité de l'approche, mais soulignent également la nécessité d'affiner les représentations vectorielles.

Parmi les perspectives en cours, nous envisageons :

— l'exploitation des pièces jointes associées aux tickets (captures d'écran, logs, documents techniques), afin d'enrichir le contexte de recherche ;

— le lancement d'une campagne d'annotation plus large auprès des experts du CDS pour raffiner les métriques et alimenter un fine-tuning des modèles d'embedding ;

— l'évaluation de la chaîne complète RAG, incluant la génération d'une suggestion de solution à partir des tickets similaires détectés.

Ces travaux ouvrent la voie vers un assistant intelligent spécialisé pour les centres de services, capable de tirer parti des connaissances accumulées au fil du temps tout en respectant les contraintes opérationnelles et réglementaires du domaine.

Références

BENAMARA F., HATOUT N., MULLER P. & OZDOWSKA S., Éds. (2007). *Actes de TALN 2007 (Traitement automatique des langues naturelles)*, Toulouse. ATALA, IRIT.

BORGEAUD S., MENSCH A., HOFFMANN J., CAI T., RUTHERFORD E., MILLICAN K., VAN DEN DRIESSCHE G., LESPIAU J.-B., DAMOC B., CLARK A., DE LAS CASAS D., GUY A., MENICK J., RING R., HENNIGAN T., HUANG S., MAGGIORE L., JONES C., CASSIRER A., BROCK A., PAGANINI M., IRVING G., VINYALS O., OSINDERO S., SIMONYAN K., RAE J. W., ELSEN E. & SIFRE L. (2022). Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, p. 2206–2240.

CHEN J. & YANG D. (2021). Structure-aware abstractive conversation summarization via discourse and action graphs. In *Proceedings of the 2021 Conference of the North American Chapter of the ACL : Human Language Technologies (NAACL-HLT)*, p. 1380–1391 : Association for Computational Linguistics.

CHEN M., CHU Z., WISEMAN S. & GIMPEL K. (2021). Summscreen : A dataset for abstractive screenplay summarization. *arXiv preprint arXiv :2104.07091*.

DIAS G., Éd. (2015). *Actes de TALN 2015 (Traitement automatique des langues naturelles)*, Caen. ATALA, HULTECH.

FENG S., PATEL S. S., WAN H. & JOSHI S. (2021). MultiDoc2Dial : Modeling dialogues grounded in multiple documents. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 6162–6176.

GAO S., CHENG X., LI M., CHEN X., LI J., ZHAO D. & YAN R. (2023). Dialogue summarization with static-dynamic structure fusion graph. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 13858–13873 : Association for Computational Linguistics.

GIORGI J., TOMA A., XIE R., CHEN S. S., AN K. R., ZHENG G. X. & WANG B. (2023). Wanglab at mediqa-chat 2023 : Clinical note generation from doctor-patient conversations using large language models. *arXiv preprint arXiv :2305.02220*.

GUU K., LEE K., TUNG Z., PASUPAT P. & CHANG M.-W. (2020). REALM : Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, p. 3929–3938.

IZACARD G. & GRAVE E. (2021). Leveraging passage retrieval with generative models for open-domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, p. 874–880.

IZACARD G., LEWIS P., LOMELI M., HOSSEINI L., PETRONI F., SCHICK T., DWIVEDI-YU J., JOULIN A., RIEDEL S. & GRAVE E. (2022). Atlas : Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv :2208.03299*.

JINA AI (2024). Promptperfect. https://promptperfect.jina.ai/. Consulté en avril 2025.

LAIGNELET M. & RIOULT F. (2009). Repérer automatiquement les segments obsolescents à l'aide d'indices sémantiques et discursifs. In A. NAZARENKO & T. POIBEAU, Éds., *Actes de TALN 2009 (Traitement automatique des langues naturelles)*, Senlis : ATALA LIPN.

LANGLAIS P. & PATRY A. (2007). Enrichissement d'un lexique bilingue par analogie. In (Benamara *et al.*, 2007), p. 101–110.

LEWIS P., PEREZ E., PIKTUS A., PETRONI F., KARPUKHIN V., GOYAL N., KÜTTLER H., LEWIS M., YIH W.-T., ROCKTÄSCHEL T., RIEDEL S. & KIELA D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, p. 9459–9474.

PARK S. & LEE J. (2022). Unsupervised abstractive dialogue summarization with word graphs and pov conversion. In *Proceedings of the 2nd Workshop on Designing Meaning Representations (DMR)*, p. 1–9.

SAP (2024). Sap erp — enterprise resource planning software. https://www.sap.com/products/erp.html. Consulté en avril 2025.

SERETAN V. & WEHRLI E. (2007). Collocation translation based on sentence alignment and parsing. In (Benamara *et al.*, 2007), p. 401–410.

SHUSTER K., KOMEILI M., ADOLPHS L., ROLLER S., SZLAM A. & WESTON J. (2022a). Language models that seek for knowledge : Modular search & generation for dialogue and prompt completion. *arXiv preprint arXiv :2203.13224*.

SHUSTER K., XU J., KOMEILI M., JU D., SMITH E. M., ROLLER S., UNG M., CHEN M., ARORA K., LANE J., DIAB M., SCHULZ H., KIELA D. & WESTON J. (2022b). Blenderbot 3 : a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv :2208.03188*.

WANG X., SEN P., LI R. & YILMAZ E. (2024). Adaptive retrieval-augmented generation for conversational systems. *arXiv preprint arXiv :2407.21712*.

ZENG S., ZHANG J., HE P., XING Y., LIU Y., XU H., REN J., WANG S., YIN D., CHANG Y. *et al.* (2024). The good and the bad : Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv :2402.16893*.

ZHANG Y., NI A., YU T., ZHANG R., ZHU C., DEB B., CELIKYILMAZ A., AWADALLAH A. H. & RADEV D. (2021). An exploratory study on long dialogue summarization : What works and what's next. *arXiv preprint arXiv :2109.04609*.

ZHU C., LIU Y., MEI J. & ZENG M. (2021). MediaSum : A large-scale media interview dataset for dialogue summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the ACL : Human Language Technologies (NAACL-HLT)*, p. 5927–5934 : Association for Computational Linguistics.

7 Annexes

Prompt pour la génération des résumés

Je vais te donner un échange à propos d'un besoin ou d'un souci technique autour des ERP. Ce message commence par un en-tête indiquant la priorité et d'autres informations diverses, puis tu as les échanges de messages. C'est toujours le client (qui doit rester anonyme) qui a émis le besoin et qui échange en premier, suivi d'un message du support disant que le ticket est pris en charge. Je veux que tu me résumes ces conversations en détaillant le problème initial (en gardant le client anonyme), puis que tu discusses des pistes envisagées pour répondre à ce problème et enfin que tu expliques en détail la solution finale retenue et sa mise en place. Je veux voir clairement et à l'identique les noms de parties 'PROBLÈME', 'PISTES' et 'SOLUTION' dans le résumé. Pour anonymiser le contenu, je veux que tu fasses référence au 'client' comme étant la ou les personnes ayant émis le besoin d'aide et que tu fasses référence à 'Talan' comme l'entreprise prenant en charge le besoin et qui propose les solutions. Ne donne pas de noms de personnes. Voici l'échange à traiter :

Prompt pour la génération des résumés issu de PromptPerfect

Rôle
Tu es un consultant informatique expert SAP capable de comprendre finement les problématiques SAP.
Tâche
Je souhaite que tu résumes le contenu partagé de manière synthétique. Note la proposition sur une échelle de 1 à 5 et recommence jusqu'à ce que le résumé dispose d'une note de 5, n'affiche pas la notation.
Format de réponse attendu
Un résumé en trois parties :

- **PROBLÈME** : détaille le problème initial.
- **PISTES** : expose les solutions envisagées.
- **SOLUTION** : détaille la solution retenue et sa mise en place.

Directives de confidentialité
- Le client doit toujours rester anonyme ; fais référence au 'client' comme étant la ou les personnes ayant émis le besoin d'aide.
- Fais référence à 'TALAN' comme l'entreprise prenant en charge le besoin et qui propose les solutions.
- Ne donne JAMAIS les noms, prénoms des personnes ni des organisations.

Consignes
- Ne commence jamais ta réponse par un en-tête.
- Rédige une réponse bien formatée optimisée pour la lisibilité.
- Sois concis dans ta réponse. Évite tout préambule et fournis la réponse directement.
- Ne termine pas par une formule d'ouverture ou de demande d'information.
- Évite le langage de moralisation ou d'hésitation.
- Utilise les listes, puces et autres dispositifs d'énumération avec parcimonie, préférant d'autres méthodes de formatage comme les en-têtes. N'utilise des listes que lorsqu'une énumération claire est nécessaire.
- Utilise des listes numérotées uniquement lorsque des éléments doivent être classés. Sinon, utilise des puces.
- Si une partie de ta réponse inclut des informations externes aux sources fournies, indique-le clairement et encourage une vérification indépendante.
- N'oublie pas d'être précis, complet, et de respecter toutes les directives fournies ci-dessus.

Contexte
Le contenu partagé correspond à un ticket de résolution. Ce contenu commence par un en-tête, la priorité et d'autres informations, puis est suivi d'échanges correspondant à des messages.

Voici l'échange à traiter :

Prompt pour la détection de mots-clefs

Je vais te donner un échange à propos d'un besoin ou d'un souci technique autour des ERP. Ce message commence par un en-tête indiquant la priorité et d'autres informations diverses, puis tu as les échanges de messages. C'est toujours le client (qui doit rester anonyme) qui a émis le besoin et qui échange en premier, suivi d'un message du support disant que le ticket est pris en charge. Je veux que tu me donnes une liste de 10 mots-clés les plus techniques possibles caractérisant cet échange. Donne directement les mots-clés séparés par des points-virgules ';'. Aucun des mots ne doit caractériser Talan ou une autre entité mais bien caractériser l'aspect technique. Voici l'échange à traiter :