RAPPORT DE PROJET D'EXPERTISE

Intégration des Large Language Models (LLMs) pour l'Analyse des Modes de Défaillances, de leurs Effets et de leur Criticité (AMDEC)

NOUHAN KOUROUMA

GÉNIE INDUSTRIELLE
INTELLIGENCE ARTIFICIELLE ET DATA SCIENCE

Encadré par : PR. TAWFIK MASROUR

Résumé

Ce projet propose une nouvelle approche d'amélioration de l'Analyse des Modes de Défaillances, de leurs Effets et de leur Criticité (AMDEC) en intégrant les Large Language Models (LLMs). Notre méthode automatise le processus d'AMDEC en combinant la collecte de données, le prétraitement, l'utilisation d'algorithmes d'intelligence artificielle et l'évaluation des risques. Une étude de cas dans l'industrie automobile a validé notre approche, en utilisant les avis clients négatifs pour identifier les caractéristiques importantes. De plus, nous avons développé un tableau de bord pour suivre les résultats de l'analyse des avis clients et les améliorations apportées par l'entreprise, offrant ainsi une vision complète de l'évolution des problèmes.

Table des matières

Table des figures

1 Introduction Générale

Dans le contexte actuel du développement de produits et de l'amélioration des services clients, deux domaines d'innovation se démarquent : l'Analyse des Modes de Défaillances, de leurs Effets et de leur Criticité (AMDEC) et l'application de techniques d'Intelligence Artificielle (IA), notamment les Large Language Models (LLMs) et le Traitement Automatique du Langage Naturel (NLP). Ces approches révolutionnent la manière dont les entreprises abordent les défis liés à la qualité des produits, à la satisfaction client, et à la gestion des risques.

D'une part, l'AMDEC traditionnelle a longtemps été un pilier du processus de développement de produits, fournissant une méthodologie systématique pour identifier et atténuer les modes de défaillance potentiels. Cependant, les méthodes manuelles présentent des défis, notamment en termes de processus intensifs en main-d'œuvre et de susceptibilité aux erreurs humaines. Dans ce contexte, l'intégration des LLMs offre une opportunité de moderniser et d'automatiser le processus, en exploitant la puissance des modèles de langage pour extraire des informations pertinentes à partir de diverses sources.

D'autre part, l'amélioration des services clients a également connu une transformation significative grâce à l'application de l'IA, en particulier à travers les approches de NLP. Les méthodes classiques de collecte d'informations auprès des clients, souvent statiques et limitées, ont laissé place à des approches plus dynamiques. Dans ce contexte, notre contribution se situe dans le développement d'une méthodologie novatrice basée sur le NLP, permettant d'extraire en profondeur des informations à partir des avis libres des clients sans les restreindre à des questionnaires préétablis.

Notre projet propose donc une convergence de ces deux tendances en intégrant les LLMs dans le processus d'AMDEC, spécifiquement dans l'analyse des retours clients pour l'amélioration des produits. Cette approche dynamique offre la possibilité d'automatiser l'identification des modes de défaillance, tout en fournissant une compréhension approfondie des besoins et des préoccupations des clients.

La structure de notre travail comprendra une revue de la littérature sur l'état de l'art dans ces deux domaines, mettant en évidence les avantages et les limitations des LLMs dans le contexte de l'AMDEC et du NLP. Nous proposerons ensuite un cadre intégratif, illustré par une étude de cas démontrant la mise en œuvre pratique de notre méthodologie. Enfin, nous discuterons des enseignements tirés et des perspectives futures pour le développement de cette approche synergique.

2 Revue de Littérature

Dès 1996, Wirth et al. ont soutenu qu'une approche basée sur la connaissance de l'AMDEC pourrait améliorer la méthode conventionnelle de réalisation d'une AMDEC en utilisant diverses bases de connaissances pour soutenir des descriptions précises des produits à l'aide d'un vocabulaire contrôlé, et pour faciliter la réutilisation ultérieure des connaissances collectées lors d'une AMDEC [Wirth et al., 1996].

Les développements récents en matière d'IA et d'apprentissage automatique ont ouvert de nouvelles possibilités pour l'amélioration de l'AMDEC. Liu et al. ont souligné comment les méthodes de prise de décision multicritère peuvent soutenir l'évaluation des risques dans l'AMDEC [Liu et al., 2019], tandis que Soltanali et al. ont proposé une plateforme intelligente d'AMDEC avec des modèles hybrides, combinant quantification de l'incertitude, techniques d'apprentissage automatique et prise de décision multicritère [Soltanali and Ramezani, 2023]. Na'amnh et al. ont introduit des modèles améliorés d'évaluation des risques utilisant l'inférence floue et les réseaux neuronaux, surpassant les méthodes classiques, le modèle flou se révélant supérieur pour la prise de décision [Na'amnh et al., 2021].

De plus, les chercheurs ont exploré des approches basées sur les données en utilisant l'apprentissage automatique pour mettre à jour et prédire en continu les numéros de priorité des risques (RPN) pour de nouveaux modes de défaillance [Peddi et al., 2023]. Hassan et al. ont utilisé avec succès des données historiques et des réseaux neuronaux convolutionnels pour automatiser la priorisation des exigences contractuelles [Hassan et al., 2023]. Yucesan et al. ont utilisé des méthodes floues de meilleur-pire et de réseau bayésien flou pour évaluer les paramètres de risque dans l'AMDEC [Yucesan et al., 2021]. La conception basée sur les données a également retenu l'attention, comme le démontre Filz et al., qui ont montré les avantages de combiner les données d'événements de maintenance passés avec l'expérience des employés pour soutenir la planification de la maintenance [Filz et al., 2021]. De plus, Hodkiewicz et al. ont présenté l'application d'approches ontologiques pour améliorer la représentation explicite des concepts liés à l'AMDEC [Hodkiewicz et al., 2021].

L'intégration de LLMs, en particulier ChatGPT, dans le processus d'AMDEC suscite de l'intérêt. ChatGPT utilise des algorithmes d'apprentissage profond pour générer des réponses textuelles de type humain [Zhao et al., 2023]. La capacité de ChatGPT à comprendre le contexte et à apprendre de nouvelles données offre des avantages potentiels dans les tâches d'AMDEC [Thomas, 2023]. La mise en œuvre de ChatGPT dans l'AMDEC implique l'utilisation de sa fonctionnalité de base et son renforcement avec des connaissances spécifiques à l'entreprise [Diemert and Weber, 2023]. La combinaison d'outils d'IA tels que ChatGPT et de l'expertise humaine renforcerait les points forts des deux dans le processus d'AMDEC. Les travaux com-

binant l'AMDEC avec l'utilisation de techniques LLM sont encore rares. Un exemple est l'étude de Spreafico et Sutrisno, qui présentent une méthode utilisant un chatbot pour l'analyse automatique des défaillances sociales dans la durabilité des produits [Spreafico and Sutrisno, 2023]. Trois études de cas ont confirmé son potentiel tout en soulignant les limites du chatbot.

La littérature examinée met en évidence l'importance des outils logiciels dans le soutien du processus d'AMDEC, améliorant la collaboration, l'évaluation des risques et l'efficacité globale. De plus, l'intégration de LLM offre des perspectives pour améliorer davantage l'AMDEC en exploitant les capacités avancées de traitement du langage naturel. La combinaison d'outils d'IA avec l'expertise humaine est considérée comme un moyen d'obtenir des résultats supérieurs dans l'AMDEC. Cependant, malgré les progrès réalisés dans les approches basées sur l'IA pour l'AMDEC, des lacunes subsistent. Tout d'abord, la littérature actuelle

3 Méthodologie

Notre méthodologie se base sur deux approches différentes : la première approche est le fine-tuning et la seconde approche est le prompt ingéniering.

Comme le montre la Figure 1, notre projet vise les trois niveaux, à savoir : détection des noms de pièces, des problèmes liés à ces pièces et aussi les informations cruciales qui nous permettrons de comprendre le contexte général du problème. Mais pour le moment, nous allons nous concentrer sur la création d'un modèle capable de détecter les noms de pièces dans les avis clients et par la suite, nous pouvons chercher une approche pour réaliser les autres tâches. Pour cela, nous avons décidé d'utiliser une approche basée sur le Fine-tuning des modèles Transformers, notamment le modèle GPT-3 L'avantage de ce genre d'approche est que les modèles de base sont déjà si performants qu'ils peuvent vite donner de bons résultats sur peu de données mais l'inconvénient est qu'ils sont coûteux.

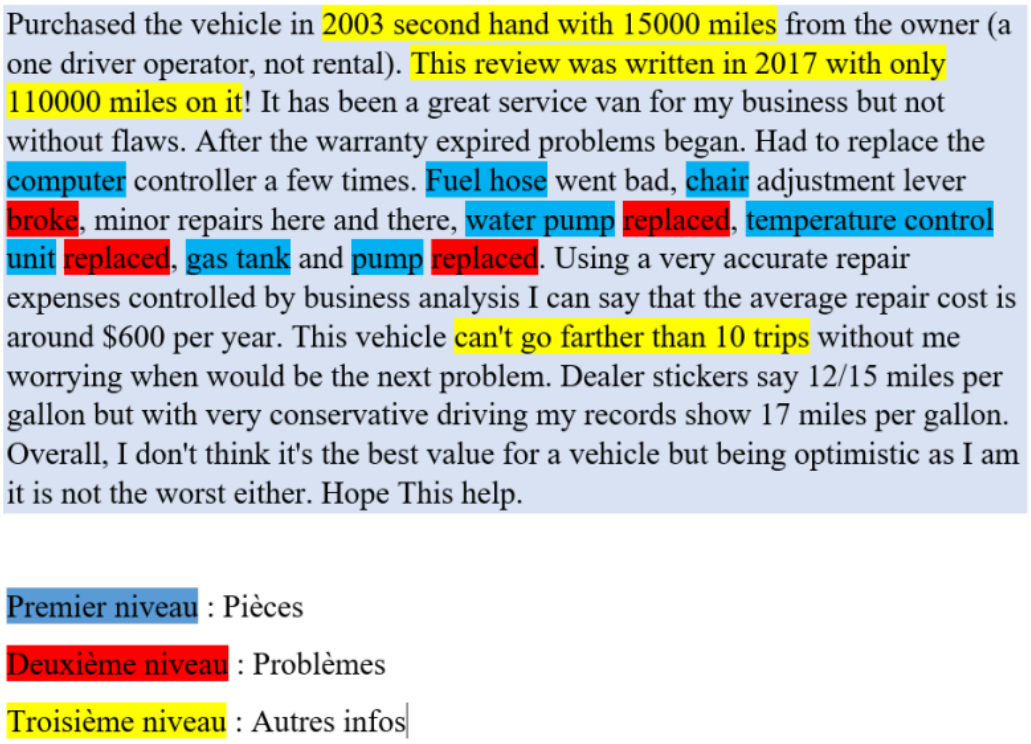


FIGURE 1 – Illustration du processus d'extraction

3.1 Approche Fine-tuning

3.1.1 Collecte des données

Pour recueillir les données destinées à l'entraînement de nos modèles, nous avons exploité la plateforme Kaggle Kaggle [2024] qui constitue une base de données en ligne dédiée aux projets de Data Science. Cette plateforme nous a donné accès à cinquante ensembles de données distincts portant sur les avis de clients concernant cinquante marques de véhicules différentes. Tous ces ensembles de données présentent une structure uniforme avec les mêmes noms de colonnes, comprenant notamment :

- **Review_Date** : La date à laquelle l'auteur a publié son avis.
- **Author_Name** : Le nom de l'auteur ayant rédigé l'avis sur le véhicule.
- **Vehicle_Title** : Le titre ou le nom du véhicule objet de l'avis.
- **Review_Title** : Le titre donné par l'auteur à son avis, offrant un aperçu du contenu.
- **Review** : Le texte détaillé de l'avis rédigé par l'auteur sur le véhicule.
- **Rating** : La notation attribuée par l'auteur pour évaluer le véhicule, généralement sur une échelle (par exemple, de 1 à 5 étoiles).

Ces colonnes renferment des informations cruciales sur les avis relatifs aux véhicules, telles que la date de publication, l'auteur, le véhicule concerné, le titre de l'avis, le contenu de celui-ci, ainsi que la note attribuée.

3.1.2 Traitement des données

3.1.2.1 Étiquetage Cette phase du projet revêt une importance cruciale, car elle implique la création de notre colonne cible (*"Target"*) qui stockera les listes de pièces de véhicules. Pour ce faire, nous avons effectué du web scraping sur deux sites web : *wikipedia.com* [wik] et *listexplained.com* [lis]. Ces sites nous ont fourni un accès à une liste non exhaustive de pièces de véhicules, totalisant environ sept cents références. Nous sommes conscients que notre liste ne couvre pas toutes les pièces de véhicules et ne prend pas en compte toutes les appellations possibles pour une même pièce. Cependant, pour le moment, nous nous en tiendrons à cela et nous améliorerons notre liste au fil de notre progression.

Pour conclure cette phase d'étiquetage, nous avons parcouru l'ensemble de notre ensemble de données, créant ainsi deux nouvelles colonnes :

- **"List_part"** : qui contient une liste des pièces de véhicules trouvées dans chaque avis.
- **"Count_part"** : qui indique le nombre de pièces présentes dans chaque avis.

À ce stade, nous pouvons affirmer que notre ensemble de données est en bonne voie et que nous sommes prêts à passer à l'étape du fine-tuning.

3.1.2.2 Préparation du dataset pour le fine-tuning

Une fois que nous avons déterminé que le fine-tuning est la solution appropriée, indiquant que notre prompt a été optimisé au mieux de ses capacités et que des problèmes persistants avec le modèle ont été identifiés, il est impératif de préparer les données pour l'entraînement du modèle. Dans cette étape cruciale, nous devons constituer un ensemble varié de conversations de démonstration, similaires aux conversations auxquelles nous demanderons au modèle de répondre lors de l'inférence en production.

Chaque exemple dans notre jeu de données doit être une conversation structurée selon le même format que notre API de complétion de chat, comprenant une liste de messages où chaque message a un rôle, un contenu et éventuellement un nom. Il est essentiel d'inclure dans ces exemples d'entraînement des situations où le modèle sollicité ne se comporte pas comme souhaité. Les messages fournis en tant qu'assistant dans les données doivent représenter les réponses idéales que nous souhaitons que le modèle fournisse.

Cette approche garantit que le modèle est formé sur des exemples diversifiés et spécifiques, incluant des cas où des améliorations sont nécessaires. Cela permet de guider le modèle vers les réponses idéales que nous attendons dans des situations variées lors de l'utilisation du modèle en production.

La dernière étape consiste à formater notre dataset, pour cela nous avons ajouté une séquence de fin à tous les prompts : il termine par la chaine de caractère "-> " qui indique au modèle qu'il peut commencer à faire la complétion c'est-à-dire générer la liste des pièces ; et une séquence de fin, toutes les complétions se termine par le mot " END" pour indiquer au modèle qu'il doit s'arrêter, on fait cela pour éviter que le modèle ne continue de générer des mots non souhaités. Et maintenant nous convertissons notre dataset en fichier Jsonl qui sera le format utilisé pour le fine-tuning.

3.1.3 Entrainement des Modèles

Après avoir minutieusement préparé et formaté nos données, nous entamons la phase d'entraînement, cruciale pour la création du modèle. Pour ce faire, nous utilisons l'outil en ligne de commande (CLI) d'OpenAI, exécutant ainsi une série de commandes. Nos données sont ensuite téléchargées, accompagnées des hyperparamètres que nous avons définis, et le processus d'entraînement est enclenché.

Dans notre approche, nous avons choisi un processus d'entraînement itératif qui permet un ajustement continu du modèle, affinant ses performances de manière constante. Cette méthodologie de fine-tuning itératif implique plusieurs cycles d'amélioration du modèle, contrastant avec une seule itération de fine-tuning. Nous adaptons progressivement notre modèle, perfectionnant ses performances au fil des itérations.

L'objectif principal de ce projet est de former un modèle polyvalent capable d'accomplir diverses tâches de détection, notamment la reconnaissance des noms de pièces, l'identification des problèmes liés aux pièces ou les défauts, ainsi que la compréhension du contexte. Initialement, nos efforts se concentrent exclusivement sur la détection des noms de pièces, puis nous intégrerons progressivement les autres objectifs au fil du temps.

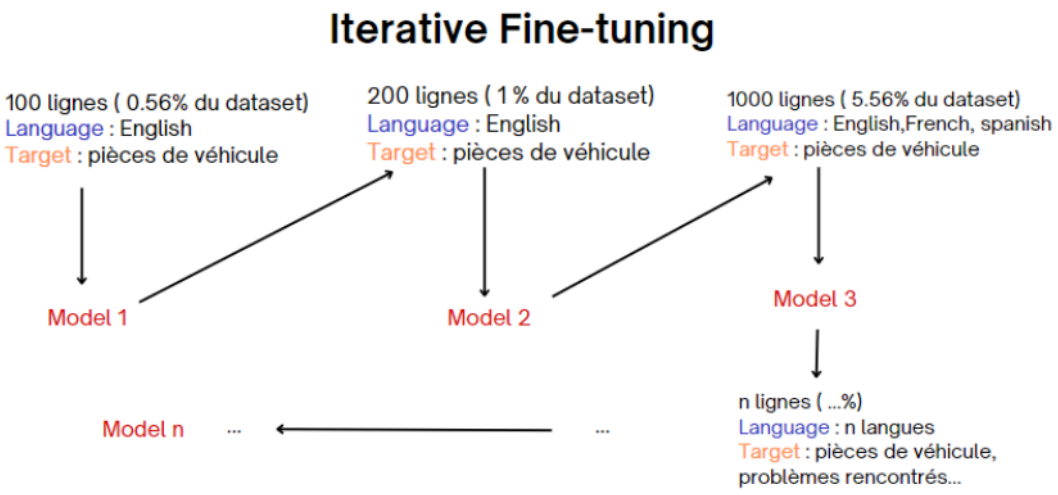


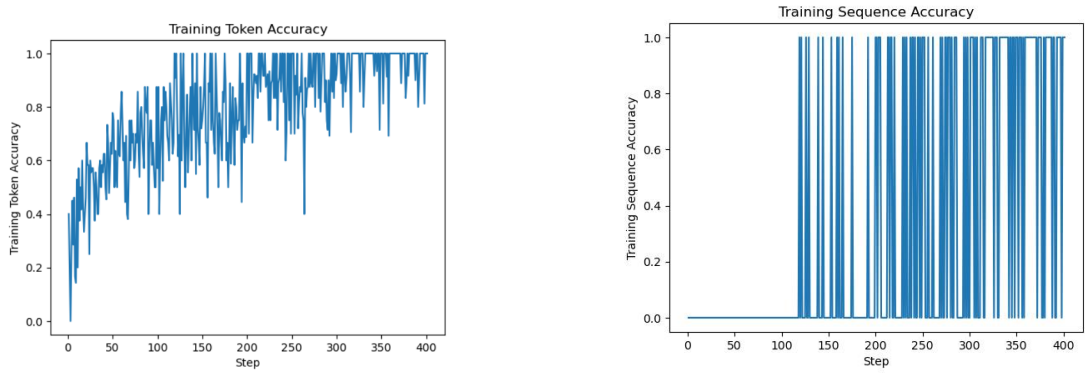
FIGURE 2 – Iterative Fine-tuning

3.1.4 Évaluation des modèles

Notre premier modèle, `partfinder_t_001`, a été entraîné sur 1000 lignes de notre ensemble de données, représentant ainsi 5,56% du total. À la suite de l'entraînement, les performances du modèle sont accessibles dans un fichier `results.csv`. Ce fichier contient une ligne pour chaque étape d'entraînement, où une étape fait référence à une passe avant et arrière sur un lot de données.

- `elapsed_tokens` : le nombre de jetons que le modèle a traités jusqu'à présent (y compris les répétitions)
- `elapsed_examples` : le nombre d'exemples que le modèle a traités jusqu'à présent (y compris les répétitions), un exemple correspondant à un élément du lot (batch). Par exemple, avec `batch_size = 4`, chaque étape augmentera `elapsed_examples` de 4.
- `training_loss` : la perte (`loss`) sur le lot d'entraînement
- `training_sequence_accuracy` : le pourcentage de complétions dans le lot d'entraînement pour lesquelles les jetons prédits par le modèle correspondent exactement aux jetons de complétion réels.
- `training_token_accuracy` : le pourcentage de jetons dans le lot d'entraînement qui ont été prédits correctement par le modèle.

L'évolution des performances de notre modèle à chaque étape de son entraînement est présentée sur les figures ci-dessous.



(a) Training Token Accuracy (b) Training Sequence Accuracy

FIGURE 3 – Métriques d'entrainement

3.2 Approche Prompt Engineering

Notre premier dataset, nous permettait de pouvoir extraire certaines informations cruciales, telles ques les pannes les plus couremment détectées. Cette fois-ci, nous allons nous concentrer sur la génération des graphes DFMEA en utilisant les technique de prompt-engineering.

3.2.1 Définition du Prompt Engineering

Le concept de "Prompt Engineering", traduit littéralement par "Ingénierie de prompt", englobe des techniques et des méthodes spécifiquement élaborées pour optimiser les formulations d'instructions dans le contexte du traitement du langage naturel (NLP) et des modèles linguistiques de grande envergure basés sur l'apprentissage automatique, tels que GPT-3 ou GPT-4. L'objectif fondamental est d'obtenir des réponses de qualité supérieure, plus précises et plus ciblées. En effet, la manière dont une question ou une instruction est formulée exerce une influence substantielle sur la qualité et la pertinence des réponses générées dans ce domaine particulier.

3.2.2 Techniques de prompts Engineering

3.2.2.1 Zero-Shot Prompting Le "zero-shot prompting" implique la génération d'une réponse sans fournir d'exemples ou de contexte préalable aux LLMs. Cette technique est idéale lorsque des réponses rapides sont nécessaires pour des questions de base ou des sujets généraux.

3.2.2.2 One-Shot Prompting Cela consiste à donner un exemple de prompt et un exemple de réponse que l'on aimerait que le modèle génère.

3.2.2.3 Few-Shot Prompting Contraiment au One-Shot, le Few-shot Prompting consiste à donner aux modèles deux ou plusieurs exemples de prompts et complexion. Cela permet au modèle de savoir comment générer les réponses selon l'output que l'on désire.

3.2.2.4 Information Retrieval nformation retrieval prompting est le fait d'utiliser les LLMs sur les sources externes. Cela permet au modèle d'aller chercher les informations sur lesquelles il n'a pas été entrainé au préalable tel qu'à partir des documents, des sites web, des bases de données, etc.

3.2.3 Prompt Engeering dans la génération des table DFMEA

Dans le cadre de notre projet, nous devrons trouver une technique de prompte engineering éfficace pour pouvoir générer notre table en se basant sur l'input de l'utilisateur.

Pour cela, nous avons opté pour le one-shot, car notre modèle étant déjà très performant, n'avait pas besoin de beaucoup d'exemple pour comprendre la tâche à réaliser. Voici la structure de notre prompt

Système : Il s'agit des instructions fournies au modèle concernant son comportement et la manière dont il devrait se percevoir.

Exemple : C'est une illustration de prompt complétion servant de référence pour le modèle lors de la génération de ses réponses. Il est composé de deux parties :

- **UserContent :** Exemple de l'entrée utilisateur.
- **ModelResponse :** Un exemple de réponse idéale que le modèle est censé produire.

Complétion : C'est la réponse du modèle à la requête de l'utilisateur, prenant en considération non seulement son prompt initial, mais également l'exemple de prompt fourni dans le contexte, ainsi que les instructions spécifiées au niveau du système. Le processus est resumé dans la figure 4

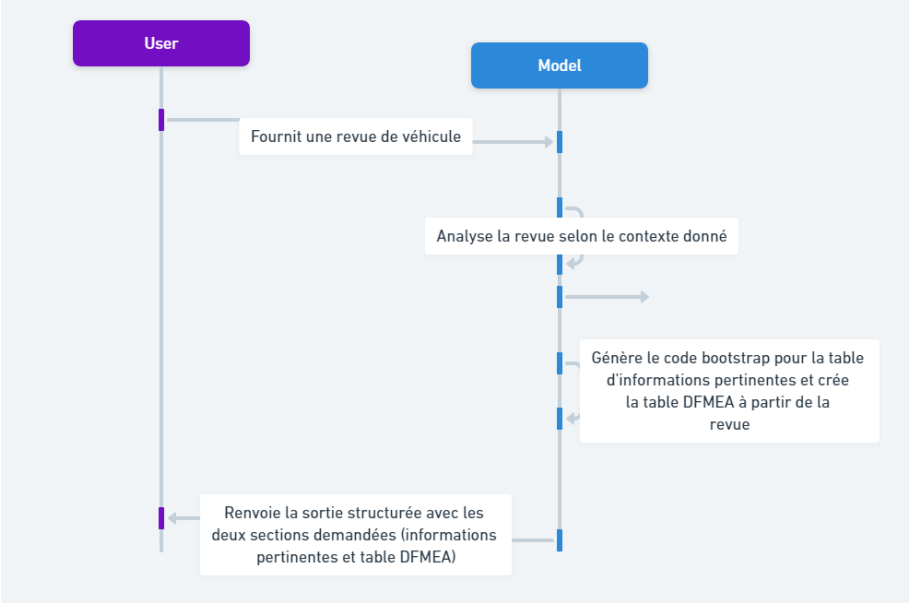


FIGURE 4 – Prompt-completion architecture

3.2.4 Retrieval Augmented Generation

Le RAG (Retrieval Augmented Generation est une technique qui permet de donner la possibilité aux LLMs de rechercher dans des sources externes telles que les documents, les bases de données, l'internet...

RAG prend une entrée et récupère un ensemble de documents pertinents ou de soutien à partir d'une source donnée (par exemple, Wikipedia). Les documents sont concaténés en tant que contexte avec la requête d'entrée d'origine, puis alimentés dans le générateur de texte qui produit la sortie finale.

Le RAG repose sur l'utilisation d'embeddings, en particulier sur la recherche de similarité entre ces embeddings. Les embeddings sont des représentations vectorielles de données, dans ce cas, de documents textuels. Le processus de recherche de similarité vise à mesurer la proximité sémantique entre différentes unités de texte.

Ce processus est illustré sur la figure ci-dessous.

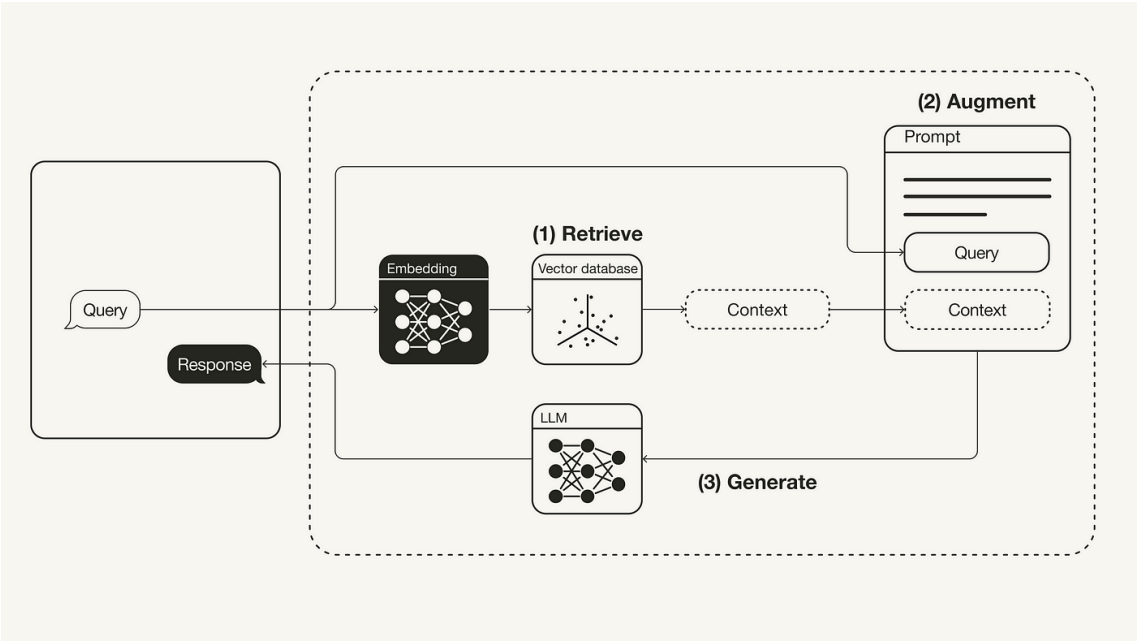


FIGURE 5 – Retrieval Augmented Generation Architecture

4 Plateforme

Notre objectif final étant de développer une plateforme intuitive, opérationnelle et sécurisée, dans cette section, nous allons détailler le processus de la création de la plateforme et son fonctionnement.

4.1 Les Outils Utilisés

4.1.1 Le Framework Django

Dans le cadre du développement de la plateforme, plusieurs outils ont été employés pour assurer une implémentation efficace et robuste. Parmi ces outils, Django, un framework web en Python, a été choisi comme base fondamentale. Django offre une structure organisée, une gestion simplifiée des bases de données, des fonctionnalités de sécurité avancées, et facilite le déploiement rapide d'applications web.

4.1.2 La base de données Postgresql

La base de données, basée sur PostgreSQL, est spécifiquement conçue pour stocker les informations sur les pièces de véhicules , ainsi que les statistiques liées à leur évolution. PostgreSQL a été choisi en raison de sa flexibilité, de sa robustesse, et de sa prise en charge de fonctionnalités avancées, ce qui en fait un choix idéal pour le stockage et la gestion des données prédictives.

4.1.3 OpenAI Api

L'API OpenAI est une interface de programmation d'application (API) fournie par OpenAI, une entreprise spécialisée dans l'intelligence artificielle. L'API OpenAI donne aux développeurs la possibilité d'intégrer les modèles de langage d'OpenAI, tels que GPT (Generative Pre-trained Transformer), dans leurs propres applications, produits ou services.

4.1.4 LangChain

LangChain est un framework conçu pour développer des applications alimentées par des modèles de langage, en facilitant leur intégration dans diverses applications. Il se compose principalement de schémas et de modèles.

4.2 Architecture de la plateforme

Notre plateforme intègre une interface de chat, un tableau de bord et une base de données. Les utilisateurs soumettent des avis sur les véhicules via le chat. Le modèle, utilisant le prompt engineering ou le RAG, génère de complètes réponses contextuelles (Table DFMEA, informations sur les éléments dans la base de données).
Ces réponses sont affichées dans le chatbot et enregistrées dans la base de données. Le modèle peut également utiliser les données de la base pour des réponses personnalisées. Les informations de la base sont visualisées sur le dashboard, aidant l'entreprise à prendre des décisions éclairées basées sur les tendances observées.

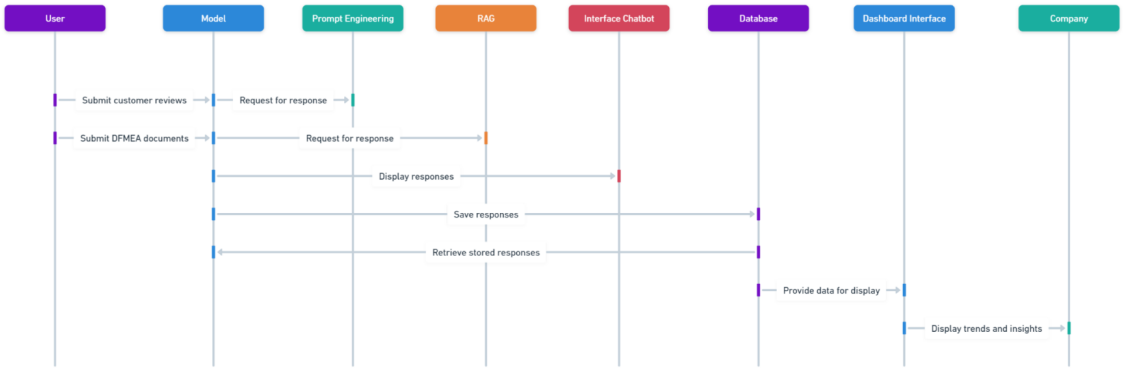


FIGURE 6 – Resumé de l'architecture de la plateforme

4.2.1 L'interface Chat

L'interface de chat offre la fonctionnalité permettant aux utilisateurs de générer des rapports DFMEA (Design Failure Mode and Effect Analysis) de l'entreprise via le modèle RAG. En interagissant avec cette interface, les utilisateurs peuvent poser des questions ou soumettre des requêtes spécifiques concernant les aspects de la conception des véhicules.

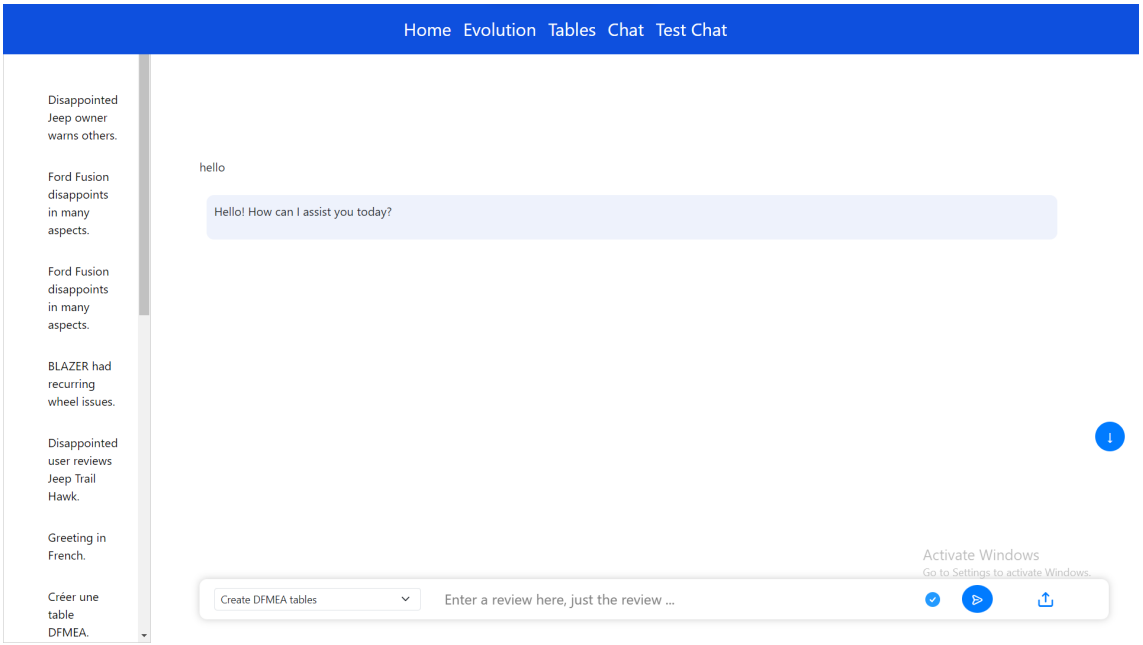


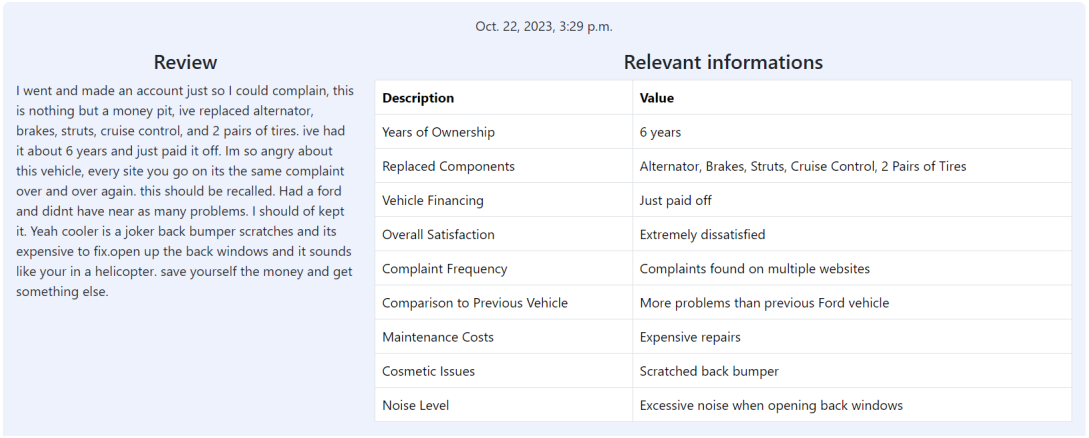
FIGURE 7 – Chat Interface

4.2.1.1 Génération des tables DFMEA :

Lorsqu'un utilisateur soumet un avis contenant des plaintes sur un véhicule, l'interface de chat entre en action en utilisant le modèle RAG. Dans le contexte donné, le modèle est instruit de générer du code HTML pour une table d'informations pertinentes ainsi qu'une table DFMEA (Design Failure Mode and Effect Analysis).

Le modèle, grâce au prompt engineering, produit le code HTML nécessaire, excluant explicitement les balises et éléments non autorisés. Il crée ainsi deux tables distinctes : une table d'informations pertinentes, comprenant des détails tels que l'année d'achat, le kilométrage actuel, les réparations majeures, etc., et une table DFMEA détaillée, catégorisant les composants du véhicule, les modes potentiels de défaillance, les effets associés, les causes probables, les contrôles actuels, ainsi que des mesures de gravité, d'occurrence, de détection et de nombre de priorités de risque (RPN).

L'interface de chat, étant un interpréteur HTML, affiche ensuite ces tables générées directement dans la conversation. Le résultat est une présentation claire et concise des informations pertinentes et de l'analyse DFMEA, prête à être interprétée et utilisée par l'utilisateur. Ce processus garantit une réponse précise et immédiate en réponse aux exigences spécifiques définies dans le contexte.



Oct. 22, 2023, 3:29 p.m.

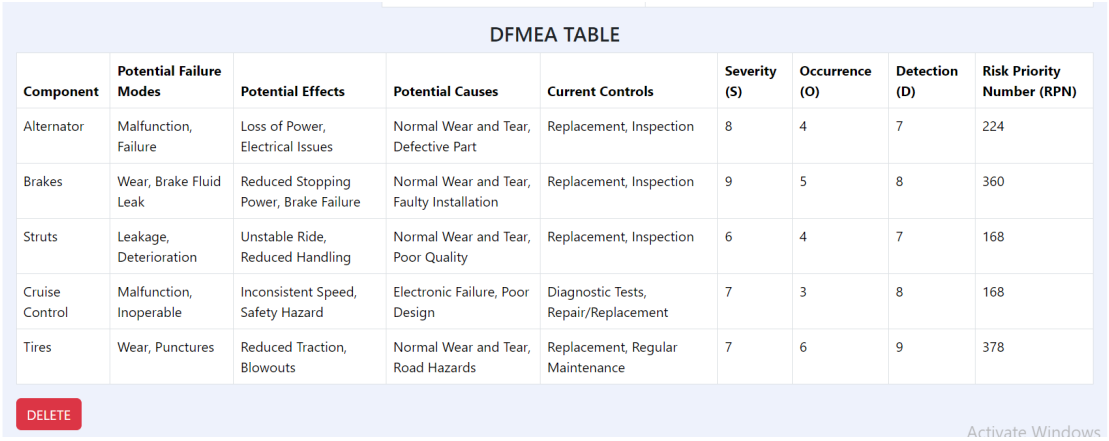
Review

I went and made an account just so I could complain, this is nothing but a money pit, ive replaced alternator, brakes, struts, cruise control, and 2 pairs of tires. ive had it about 6 years and just paid it off. Im so angry about this vehicle, every site you go on its the same complaint over and over again. this should be recalled. Had a ford and didnt have near as many problems. I should of kept it. Yeah cooler is a joker back bumper scratches and its expensive to fix.open up the back windows and it sounds like your in a helicopter. save yourself the money and get something else.

Relevant informations

Description	Value
Years of Ownership	6 years
Replaced Components	Alternator, Brakes, Struts, Cruise Control, 2 Pairs of Tires
Vehicle Financing	Just paid off
Overall Satisfaction	Extremely dissatisfied
Complaint Frequency	Complaints found on multiple websites
Comparison to Previous Vehicle	More problems than previous Ford vehicle
Maintenance Costs	Expensive repairs
Cosmetic Issues	Scratched back bumper
Noise Level	Excessive noise when opening back windows

FIGURE 8 – Exemple d'output du modèle : Informations importantes



DFMEA TABLE

Component	Potential Failure Modes	Potential Effects	Potential Causes	Current Controls	Severity (S)	Occurrence (O)	Detection (D)	Risk Priority Number (RPN)
Alternator	Malfunction, Failure	Loss of Power, Electrical Issues	Normal Wear and Tear, Defective Part	Replacement, Inspection	8	4	7	224
Brakes	Wear, Brake Fluid Leak	Reduced Stopping Power, Brake Failure	Normal Wear and Tear, Faulty Installation	Replacement, Inspection	9	5	8	360
Struts	Leakage, Deterioration	Unstable Ride, Reduced Handling	Normal Wear and Tear, Poor Quality	Replacement, Inspection	6	4	7	168
Cruise Control	Malfunction, Inoperable	Inconsistent Speed, Safety Hazard	Electronic Failure, Poor Design	Diagnostic Tests, Repair/Replacement	7	3	8	168
Tires	Wear, Punctures	Reduced Traction, Blowouts	Normal Wear and Tear, Road Hazards	Replacement, Regular Maintenance	7	6	9	378

DELETE

Activate Windows

FIGURE 9 – Exemple d'output du modèle : table DFMEA

4.2.2 Dashboard

Le Dashboard nous permet de visualiser les tendances des informations clients dans notre base de données et de prendre des décisions en conséquence. C'est un résumé de toutes les interactions des utilisateurs avec le chatbot.
Les fonctionnalités clés de ce tableau de bord incluent une analyse en temps réel, l'évaluation continue de l'impact des améliorations, une exploitation approfondie des avis et la mise à jour automatique des indicateur.

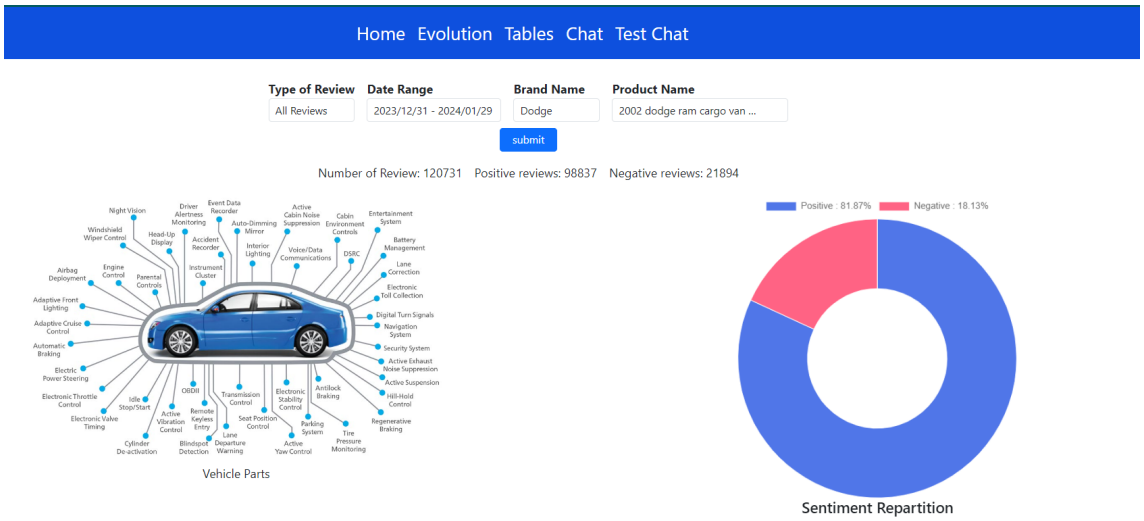


FIGURE 10 – Une Vue du dashboard

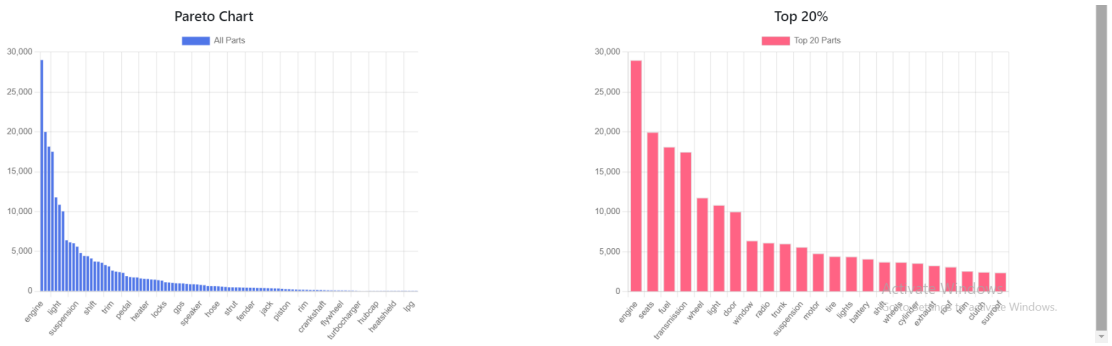


FIGURE 11 – Diagramme Pareto des pièces les plus citées

Utilisant le diagramme de Pareto, également appelé la règle des 80/20, nous simplifions l'analyse en identifiant et hiérarchisant les problèmes majeurs ou causes les plus prédominantes contribuant aux défauts, erreurs ou inefficacités.

Pour approfondir davantage, nous pouvons également visualiser l'évolution temporelle de chaque pièce. Ce processus est actualisé en temps réel à chaque nouvel avis client émis. Ci-dessous, un aperçu des statistiques spécifiques à chaque pièce dans notre tableau de bord.

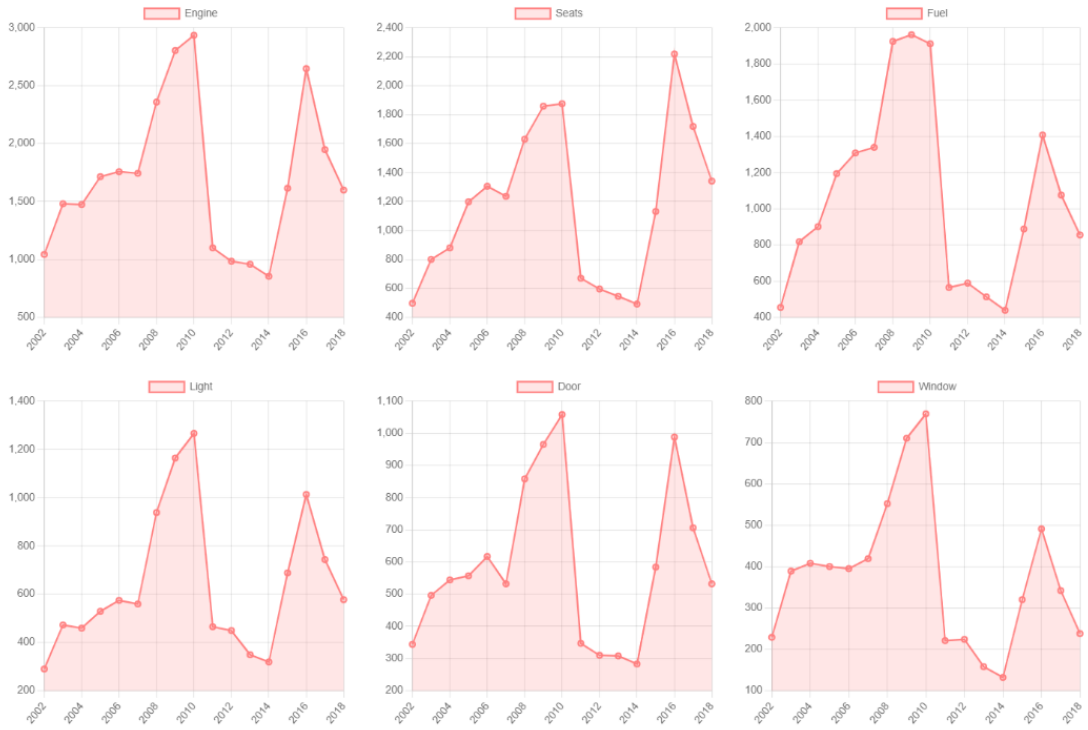


FIGURE 12 – Évolution des pièces dans les avis par période

Comme illustré sur la Figure 12, chaque pièce individuelle est représentée en fonction du nombre d'occurrences dans les avis clients. Ce processus est dynamique et peut être adapté pour analyser spécifiquement les avis négatifs, positifs, par mois, par produit, etc.

5 Perspectives d'Amélioration

5.1 Intégration des LLMs Open Source

L'utilisation de LLMs propriétaires, tels que ceux d'OpenAI, peut s'avérer coûteuse. Par exemple, l'utilisation du modèle GPT-3.5 d'OpenAI coûte environ 0,003$ pour 1 000 tokens en entrée et 0,006$ pour 1 000 tokens en sortie.

En revanche, l'intégration de LLMs open source tels que Llama, Mistral ... offre plusieurs avantages. Premièrement, ils sont généralement gratuits à utiliser, ce qui peut entraîner des économies de coûts significatives. Deuxièmement, ils offrent une plus grande transparence et flexibilité, car les utilisateurs peuvent modifier et adapter les modèles en fonction de leurs besoins spécifiques.

5.2 Fine-tuning sur les données DFMEA réelles

Une autre perspective d'amélioration que nous envisageons est le fine-tuning des LLMs sur des données DFMEA réelles. Le DFMEA (Design Failure Mode and Effects Analysis) est une méthode structurée pour identifier les modes de défaillance potentiels d'un produit ou d'un processus, leurs causes et leurs effets sur le fonctionnement du système.

En formant les LLMs sur des données DFMEA réelles, nous pouvons adapter les modèles pour mieux comprendre et analyser les modes de défaillance spécifiques à différents contextes industriels. Cela pourrait conduire à une analyse plus précise et plus pertinente des modes de défaillance, de leurs effets et de leur criticité.

De plus, le fine-tuning peut permettre aux modèles de mieux comprendre le jargon et les terminologies spécifiques à l'industrie, ce qui peut améliorer la précision et l'utilité de l'analyse AMDEC.

5.3 Personnalisation des Modèles pour Diverses Applications Industrielles

Nous envisageons également la possibilité de personnaliser les LLMs pour une variété d'applications industrielles, pas seulement l'AMDEC. Chaque industrie a ses propres défis et exigences uniques, et il est essentiel que nos outils d'analyse soient capables de répondre à ces besoins spécifiques.

6 Conclusion

Au cours de ce projet, nous avons mis en œuvre une approche de fine-tuning pour entraîner des modèles GPT-3 à reconnaître et détecter des problèmes dans les avis clients relatifs aux pannes de véhicules.

Nous avons également créé une plateforme qui intègre un chatbot avec le RAG. Ce chatbot est capable de générer des graphes DFMEA et d'extraire des informations importantes à partir des documents DFMEA d'une entreprise. Ces fonctionnalités offrent une interface conviviale et intuitive pour l'analyse des modes de défaillance, facilitant ainsi la prise de décision basée sur les données.

De plus, notre plateforme comprend un tableau de bord qui présente des statistiques sur les pièces de véhicules et leur évolution dans le temps en fonction des avis clients. Ces informations peuvent aider les entreprises à identifier les tendances et les problèmes potentiels, et à prendre des décisions éclairées pour améliorer la qualité et la fiabilité de leurs produits.

En somme, notre travail a démontré le potentiel des LLMs pour l'analyse des modes de défaillance et la prise de décision basée sur les données. À l'avenir, nous envisageons d'explorer davantage la personnalisation des modèles pour diverses applications industrielles, ainsi que l'intégration de LLMs open source et le fine-tuning sur des données DFMEA réelles. Nous sommes impatients de poursuivre ce travail passionnant et prometteur.

Références

List explained. URL `https://www.listexplained.com`.

Wikipedia. URL `https://www.wikipedia.com`.

S. Diemert and J.H. Weber. Can large language models assist in hazard analysis ? 2023. doi : 10.48550/arXiv.2303.15473.

M.-A. Filz, J.E.B. Langner, C. Herrmann, and S. Thiede. Data-driven failure mode and effect analysis (fmea) to enhance maintenance planning. *Computers in Industry*, 129 :103451, 2021. doi : 10.1016/j.compind.2021.103451.

F. Hassan, T. Nguyen, T. Le, and C. Le. Automated prioritization of construction project requirements using machine learning and fuzzy failure mode and effects analysis (fmea). *Automation in Construction*, 154 :105013, 2023. doi : 10.1016/j.autcon.2023.105013.

M. Hodkiewicz, J.W. Klüwer, C. Woods, T. Smoker, and E. Low. An ontology for reasoning over engineering textual data stored in fmea spreadsheet tables. *Computers in Industry*, 131 :103496, 2021. doi : 10.1016/j.compind.2021.103496.

Kaggle. Kaggle - your machine learning and data science community, 2024. URL `https://www.kaggle.com/`.

H.-C. Liu, X.-Q. Chen, C.-H. Duan, and Y.-M. Wang. Failure mode and effect analysis using multi-criteria decision making methods : a systematic literature review. *Computers & Industrial Engineering*, 135 :881–897, 2019. doi : 10.1016/j.cie.2019.06.055.

S. Na'amnh, M.B. Salim, I. Husti, and M. Daróczi. Using artificial neural network and fuzzy inference system based prediction to improve failure mode and effects analysis : A case study of the busbars production. *Processes*, 9(8) :1444, 2021. doi : 10.3390/pr9081444.

S. Peddi, K. Lanka, and P.R.C. Gopal. Modified fmea using machine learning for food supply chain. *Materials Today : Proceedings*, 2023. doi : 10.1016/j.matpr.2023.04.353. In Press.

H. Soltanali and S. Ramezani. Smart failure mode and effects analysis (fmea) for safety–critical systems in the context of industry 4.0. pages 151–176, 2023. doi : 10.1007/978-981-19-9909-3_7.

C. Spreafico and A. Sutrisno. Artificial intelligence assisted social failure mode and effect analysis (fmea) for sustainable product design. *Sustainability*, 15(11) :8678, 2023. doi : 10.3390/su15118678.

D. Thomas. Revolutionizing failure modes and effects analysis with chatgpt : Unleashing the power of ai language models. *Journal of Failure Analysis and Prevention*, 23(3) :911–913, 2023. doi : 10.1007/s11668-023-01659-y.

R. Wirth, B. Berthold, A. Krämer, and G. Peter. Knowledge-based support of system analysis for the analysis of failure modes and effects. *Engineering Applications of Artificial Intelligence*, 9(3) :219–229, 1996. doi : 10.1016/0952-1976(96)00014-0.

M. Yucesan, M. Gul, and E. Celik. A holistic fmea approach by fuzzy-based bayesian network and best-worst method. *Complex & Intelligent Systems*, 7(3) :1547–1564, 2021. doi : 10.1007/s40747-021-00279-z.

W. X. Zhao, K. Zhou, J. Li, T. Tang, and X. et al. Wang. A survey of large language models. 2023. doi : 10.48550/arXiv.2303.18223.