**Sujet2 : Analyse intelligente des logs rattachés aux tickets JIRA avec interaction d'une base d'erreurs connues et génération de rapports d'aide à la décision**

**Contexte**

Les équipes de support et de maintenance accumulent une grande quantité de logs techniques associés aux tickets JIRA, mais l'analyse rapide et efficace de ces informations est souvent un défi. Une approche basée sur l’intelligence artificielle permettrait de non seulement analyser automatiquement les logs, mais aussi de s'appuyer sur une base d’erreurs connues, enrichie au fil des résolutions de tickets, pour améliorer les diagnostics et les recommandations. Cette base d'erreurs connues permettrait de recenser les problèmes récurrents, leurs solutions, et les bonnes pratiques, ce qui simplifierait l'analyse des incidents et la prise de décision.

**Objectifs du projet**

1. Extraire et analyser les logs associés aux tickets JIRA pour identifier les patterns d’erreurs et anomalies récurrentes.

2. Enrichir et interagir avec une base d’erreurs connues : Relier automatiquement les logs aux erreurs similaires dans la base d'erreurs pour identifier les causes probables et les solutions suggérées.

3. Classer et regrouper les logs en fonction de la nature des erreurs identifiées, pour faciliter la prise de décision et l'amélioration continue de la base d'erreurs.

4. Générer des rapports d'aide à la décision basés sur l'historique des tickets, les tendances des erreurs, et les recommandations automatisées.

5. Développer une interface utilisateur interactive permettant la visualisation des logs, l'accès aux solutions issues de la base d'erreurs connues et des rapports décisionnels.

**Méthodologie et Approches d’IA**

1. Collecte et Prétraitement des Logs et des Erreurs Connues

* Extraction des logs et des erreurs connues : Utiliser les APIs JIRA pour collecter les logs associés aux tickets. Connecter la base d'erreurs connues pour croiser ces informations avec les logs analysés.
* Nettoyage des données : Filtrer les logs pour extraire des informations essentielles (par ex., messages d'erreur, code de retour, composants impactés) et les formater pour l’analyse automatique.

2. Analyse des Logs par NLP et Modèles de Classification

* Traitement du Langage Naturel (NLP) : Appliquer des techniques de NLP pour extraire des informations clés des logs, comme les messages d’erreur et les descriptions d’événements, et pour identifier des termes correspondants dans la base d’erreurs connues.
* Classification et Similarité Sémantique : Utiliser des algorithmes de classification et de similarité sémantique (ex., BERT, Word2Vec) pour rapprocher les logs de tickets JIRA avec les entrées correspondantes dans la base d’erreurs connues, facilitant l’identification de solutions appropriées.
* Clustering pour la détection d’anomalies : Appliquer des techniques de clustering (K-means, DBSCAN) pour regrouper des logs similaires en clusters d'erreurs connues et en anomalies nouvelles, permettant de détecter de nouveaux types d’erreurs et de les intégrer dans la base.

3. Enrichissement et Exploitation de la Base d'Erreurs Connues

* Mise à jour dynamique de la base d'erreurs : Intégrer les nouvelles erreurs identifiées et leurs solutions dans la base de données, avec une catégorisation automatique pour un accès rapide lors des futures résolutions de tickets.
* Système de recommandation de solutions : En s’appuyant sur la base d'erreurs connues, recommander automatiquement des solutions adaptées à chaque ticket en fonction de la similarité des erreurs rencontrées.

4. Génération de Rapports d'Aide à la Décision

* Rapports de diagnostics et solutions : Pour chaque ticket, générer un rapport qui résume les logs et propose des solutions issues de la base d'erreurs connues, accompagné d'une analyse des causes probables et de suggestions pour la résolution.
* Rapports sur les tendances et indicateurs clés : Générer des rapports périodiques sur les tendances des erreurs (fréquence, types d’erreurs récurrentes), et sur l'efficacité des résolutions, permettant ainsi d’optimiser la base d’erreurs au fil du temps.

5. Développement d’une Interface Utilisateur Interactive

* Visualisation des logs et erreurs connues : Permettre une consultation des logs par type d’erreur, des solutions de la base d'erreurs connues, et une comparaison rapide avec les incidents similaires.
* Fonction de recherche intelligente : Permettre aux utilisateurs de rechercher des erreurs et des solutions par mots-clés ou par critères, en s’appuyant sur les technologies NLP.
* Alertes et notifications : Mettre en place un système d'alertes pour signaler les erreurs fréquentes, les tendances inquiétantes, ou les solutions à fort impact pour les équipes techniques.

**Technologies et outils recommandés**

* APIs de collecte de données : JIRA API pour les tickets et les logs, base de données SQL ou NoSQL pour la base d'erreurs connues (MongoDB, Elasticsearch pour la recherche en temps réel)
* NLP et Machine Learning : Python avec spaCy, NLTK pour l'analyse de texte, scikit-learn pour le clustering, BERT ou Sentence Transformers pour la similarité sémantique
* Stockage et recherche d'erreurs connues : Elasticsearch ou une base de données document pour indexer et interroger rapidement la base d'erreurs connues
* Interface et Visualisation : Framework web comme Flask ou Django pour l’interface, Tableau ou Plotly pour les visualisations de rapports et tendances
* Alertes et Notifications : Slack API, Email API ou d’autres services pour générer des notifications en temps réel sur les tendances ou erreurs fréquentes

**Résultats attendus**

* Réduction des temps de diagnostic et résolution : En croisant les logs avec une base d’erreurs connues, les équipes peuvent identifier plus rapidement les causes et solutions potentielles pour chaque ticket.
* Enrichissement continu de la base d’erreurs : Le système améliore en permanence la base d’erreurs connues, intégrant chaque nouveau problème et sa résolution, ce qui facilite la résolution des futurs incidents.
* Aide à la décision améliorée : Les rapports et analyses fournissent une vue d’ensemble des erreurs récurrentes et des performances de résolution, permettant aux équipes de mieux comprendre les problématiques et de prendre des décisions plus éclairées.

**Plan détaillé du projet sur 140 jours**

**Phase 1 : Compréhension des Données et Exploration (Jours 1-20)**

**Jour 1-5 : Analyse des tickets JIRA et des logs**

* Étudier la structure des tickets (champs disponibles, formats, relations)
* Explorer les logs associés aux tickets
* Identifier les types de données présentes (textes, timestamps, auteurs, etc.)
* Visualiser des statistiques descriptives des tickets et logs
* Documenter les observations

**Jour 6-10 : Étude des relations entre descriptions, logs et commentaires**

* Analyser la corrélation entre description, logs et commentaires
* Identifier des patterns récurrents (ex. mots-clés spécifiques aux erreurs)
* Étudier la distribution des commentaires par ticket
* Définir une stratégie de prétraitement des commentaires et descriptions
* Documenter les insights trouvés

**Jour 11-15 : Identification des patterns et types d’erreurs**

* Regrouper les tickets par module et feature
* Analyser les fréquences d’apparition des erreurs
* Étudier les tickets les plus récurrents et complexes
* Vérifier la cohérence des données et gérer les incohérences
* Rédiger une documentation technique sur les types d’erreurs

**Jour 16-20 : Recherche sur les approches existantes pour regrouper les tickets**

* Étudier les techniques de regroupement existantes (clustering, embeddings)
* Comparer les approches basées sur les logs et descriptions
* Identifier les modèles et frameworks les plus adaptés
* Synthétiser les résultats et proposer une méthodologie initiale
* Documenter les choix et justifications

**Phase 2 : Préparation des Données (Jours 21-50)**

**Jour 21-30 : Méthode 1 – Prétraitement NLP Classique**

* Nettoyage des textes (stopwords, stemming, lemmatisation)
* Extraction des entités nommées (NER) pour modules, composants, erreurs
* Vectorisation des données (TF-IDF, Word2Vec, Sentence-BERT)
* Indexation des tickets pour faciliter la recherche
* Comparaison des performances des différentes techniques

**Jour 31-40 : Méthode 2 – Utilisation des LLMs pour la structuration des données**

* Développer des prompts pour extraire les entités clés
* Expérimenter plusieurs modèles LLMs (ex. Llama-2, GPT)
* Structurer les tickets en groupes thématiques
* Comparer la structuration NLP vs. LLMs
* Documenter la méthodologie retenue

**Jour 41-50 : Validation et choix de la meilleure approche**

* Comparer les résultats des deux méthodes
* Sélectionner la technique la plus performante
* Définir un pipeline de traitement automatisé des tickets
* Tester sur un échantillon de tickets réels
* Rédiger la documentation finale sur la préparation des données

**Phase 3 : Regroupement des Tickets et Analyse des Logs (Jours 51-80)**

**Jour 51-60 : Recherche et expérimentation d’approches non supervisées**

* Tester K-Means, DBSCAN et clustering hiérarchique
* Appliquer des embeddings (BERT, SBERT, FastText)
* Évaluer la similarité cosine pour regrouper les tickets
* Comparer les performances et la qualité des clusters
* Documenter les résultats des expérimentations

**Jour 61-70 : Développement d’un modèle de regroupement optimal**

* Sélectionner l’approche la plus adaptée
* Optimiser les hyperparamètres des modèles choisis
* Automatiser le processus de regroupement
* Tester sur un jeu de données élargi
* Rédiger la documentation technique

**Jour 71-80 : Intégration et validation**

* Intégrer le modèle de regroupement dans le pipeline
* Valider les groupes de tickets obtenus
* Vérifier la cohérence des regroupements avec des experts
* Ajuster les paramètres si nécessaire
* Documenter les décisions finales

**Phase 4 : Construction et Optimisation de la KEDB (Jours 81-94)**

**Jour 81-87 : Création et mise en place de la KEDB**

* Définir la structure de la base de connaissances
* Mettre en place un pipeline d’insertion et mise à jour
* Intégrer les logs extraits et les groupes de tickets
* Tester les requêtes et l’indexation
* Documenter la mise en œuvre

**Jour 88-94 : Développement de l’API pour interroger la KEDB**

* Concevoir l’API backend pour la récupération des tickets
* Implémenter des filtres et requêtes optimisées
* Tester l’API avec plusieurs scénarios
* Documenter l’architecture de l’API

**Phase 5 : Recherche Sémantique et Extraction des Tickets Similaires (Jours 95-114)**

**Jour 95-104 : Mise en place d’une recherche vectorielle**

* Étudier les bases vectorielles FAISS et ChromaDB
* Expérimenter différentes métriques de similarité
* Implémenter la recherche vectorielle sur les tickets
* Tester et ajuster les résultats
* Documenter les choix techniques

**Jour 105-114 : Optimisation et validation de la recherche**

* Comparer les performances des moteurs de recherche
* Ajuster la pondération des critères de similarité
* Valider les résultats avec des experts
* Finaliser la documentation technique

**Phase 6 : Analyse des Tickets Similaires avec LLMs et Extraction des Caractéristiques (Jours 115-154)**

**Jour 115-130 : Expérimentation et comparaison des modèles LLMs**

* Sélection des modèles LLMs les plus adaptés
* Développement de prompts avancés
* Expérimentation sur extraction des causes et solutions
* Comparaison des performances et validation des résultats
* Documentation technique détaillée

**Jour 131-140 : Intégration et finalisation**

* Intégration des résultats dans la KEDB
* Validation globale du système
* Rédaction de la documentation finale
* Ajustements finaux avant le développement du chatbot

**Phase 7 : Développement du Chatbot Intelligent (Jours 141-160)**

**Jour 141-150 : Développement du backend et intégration**

* Définition des scénarios d’interaction
* Développement du backend et connexion à la KEDB
* Intégration des résultats d’analyse des tickets similaires
* Tests utilisateurs et ajustements
* Documentation technique

**Jour 151-160 : Optimisation et tests finaux**

* Affinage des réponses du chatbot
* Validation de l’expérience utilisateur
* Correction des derniers bugs
* Rédaction de la documentation finale
* Préparation du déploiement

## ****1. Compréhension des Données et Exploration (20 jours) 📊****

🔍 **Objectif : Explorer les données et comprendre les relations entre les descriptions, les logs et les commentaires.**

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **1-2** | Étude de la structure des tickets JIRA et logs (champs, formats, relations). |
| **3-4** | Analyse des relations entre descriptions, logs et commentaires (corrélations, liens). |
| **5-6** | Analyse des patterns récurrents et types d’erreurs les plus fréquents. |
| **7-8** | Identification des entités clés (modules, composants, erreurs). |
| **9-10** | Étude des similarités entre tickets et extraction des mots-clés. |
| **11-12** | Exploration des logs : structuration des informations contenues. |
| **13-14** | Recherche sur les méthodes existantes pour analyser et regrouper les tickets. |
| **15-16** | Définition des critères et objectifs du modèle. |
| **17-18** | Synthèse des insights et rédaction des conclusions. |
| **19-20** | Documentation technique. |

## ****2. Préparation des Données (30 jours) 🛠️****

🔍 **Objectif : Nettoyer, structurer et vectoriser les données avec deux approches (NLP Classique vs. LLMs).**

### ****Méthode 1 : Prétraitement NLP Classique****

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **21-22** | Nettoyage des descriptions et commentaires (stopwords, caractères spéciaux, normalisation). |
| **23-24** | Lemmatisation et stemming pour standardiser les termes. |
| **25-26** | Extraction des entités nommées (NER) : modules, composants, erreurs. |
| **27-28** | Vectorisation des données (TF-IDF, Word2Vec, Sentence-BERT). |

### ****Méthode 2 : Utilisation des LLMs****

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **29-30** | Prompt engineering pour structuration automatique des tickets. |
| **31-32** | Expérimentation avec différents modèles LLMs pour extraction des entités. |
| **33-34** | Comparaison entre NLP classique et LLMs pour structuration des tickets. |
| **35-36** | Validation et choix de la meilleure méthode. |
| **37-38** | Documentation technique. |

## ****3. Regroupement des Tickets et Analyse des Logs (30 jours) 🔎****

🔍 **Objectif : Trouver une méthode efficace pour regrouper les tickets similaires.**

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **39-40** | Définition des critères de regroupement (basés sur la sémantique et les logs). |
| **41-42** | Sélection des techniques de clustering (K-Means, DBSCAN, Hierarchical Clustering). |
| **43-44** | Expérimentation avec différents embeddings (Word2Vec, BERT, SBERT). |
| **45-46** | Analyse des logs pour améliorer le regroupement des tickets. |
| **47-48** | Tests des performances des techniques de clustering. |
| **49-50** | Comparaison des résultats et validation finale. |
| **51-52** | Documentation technique. |

## ****4. Construction et Optimisation de la KEDB (14 jours) 📂****

🔍 **Objectif : Construire une base de connaissances efficace.**

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **53-54** | Conception du schéma de la base de données. |
| **55-56** | Indexation et structuration des données pour interrogation rapide. |
| **57-58** | Développement du pipeline d’insertion et mise à jour automatique des tickets. |
| **59-60** | Tests de validation sur les données stockées. |
| **61-62** | Documentation technique. |

## ****5. Recherche Sémantique et Extraction des Tickets Similaires (20 jours) 🔍****

🔍 **Objectif : Mettre en place un système de recherche basé sur RAG et la similarité vectorielle.**

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **63-64** | Sélection des modèles de recherche sémantique (FAISS, ChromaDB). |
| **65-66** | Implémentation d’un index vectoriel pour rechercher les tickets similaires. |
| **67-68** | Expérimentation avec différentes métriques de similarité (cosine, euclidean). |
| **69-70** | Développement du moteur de recherche basé sur les embeddings. |
| **71-72** | Intégration avec la KEDB pour enrichir les résultats de recherche. |
| **73-74** | Documentation technique. |

## ****6. Analyse des Tickets Similaires avec LLMs et Extraction des Caractéristiques (40 jours) 🤖****

🔍 **Objectif : Analyser les tickets similaires en profondeur avec un modèle RAG.**

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **75-76** | Sélection des modèles LLMs pour extraction des insights. |
| **77-78** | Étude des stratégies de prompt engineering adaptées à RAG. |
| **79-80** | Expérimentation avec différents LLMs pour extraire les causes d’erreurs et solutions. |
| **81-82** | Mise en place de la récupération de contexte avec la base vectorielle. |
| **83-84** | Analyse des performances du système RAG. |
| **85-86** | Optimisation des prompts pour améliorer la précision des résultats. |
| **87-88** | Affinage des critères d’extraction des solutions et causes. |
| **89-90** | Intégration des résultats dans le pipeline global. |
| **91-92** | Finalisation et documentation technique détaillée. |

## ****7. Développement du Chatbot Intelligent (20 jours) 💬****

🔍 **Objectif : Développer un chatbot qui exploite la KEDB et le modèle RAG.**

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **93-94** | Définition des scénarios d’interaction et du parcours utilisateur. |
| **95-96** | Développement du backend et des APIs. |
| **97-98** | Connexion avec la KEDB et le moteur de recherche vectoriel. |
| **99-100** | Développement du module LLM pour analyser les requêtes. |
| **101-102** | Tests fonctionnels et ajustements. |
| **103-104** | Documentation technique. |

### ****⚡ (16 jours)****

| **Jour** | **Tâches détaillées** |
| --- | --- |
| **105-106** | Vérification de bout en bout (Pipeline complet). |
| **107-108** | Tests de performance et corrections finales. |
| **109-110** | Rédaction de la documentation finale. |
| **111-112** | Préparation des résultats et mise en production. |
| **113-114** | Validation avec tests utilisateurs. |
| **115-116** | Intégration finale et stabilisation. |
| **117-118** | Derniers ajustements et tests finaux. |