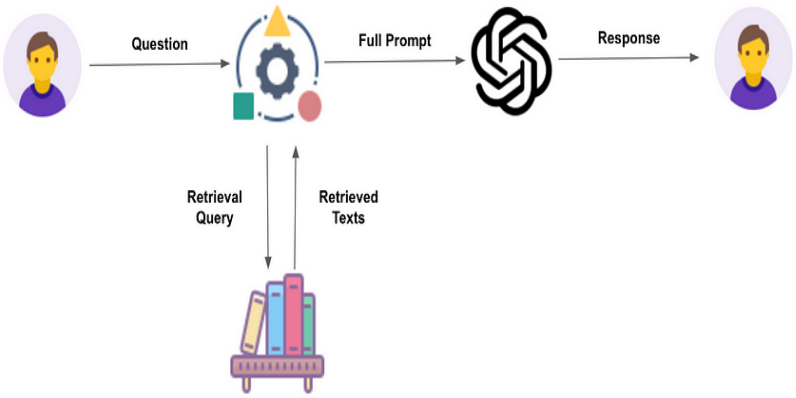
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

Estudio de arquitecturas basadas en Retrieval Augmented Generation para la mejora de generación de consultas Jira Query Language



Estudiante: Aróstegui García, Alberto

Director/Directora: Egaña Aranguren, Mikel; López Novoa, Unai

Curso: 2023-2024

Fecha: 30 de abril de 2024

Resumen:
Este trabajo se enfoca en explorar los límites de los ensayos a tracción para probetas de materiales compuestos.

Palabras Clave: *Materiales Compuestos*

Abstract:
English

Key Words:

Laburpena:
Euskera

Gako-hitzak:

Índice

Índice de figuras

Índice de tablas

Abreviaturas

API Application Programming Interface

JQL Jira Query Language

LLM Large Language Model

RAG Retrieval Augmented Generation

1. Contexto

Este trabajo de fin de grado responde a las necesidades de la empresa LKS Next-GobTech, una empresa de desarrollo de software con enfoque en la innovación. De cara a comprender los motivos por los que esta empresa requiere de lo estudiado en este trabajo, se han de poner en contexto las herramientas y metodologías que utilizan para mejorar la calidad del producto que se desarrolla. Partiendo del TFG de un compañero de escuela, Joel García Escribano, se ha estudiado la posibilidad de añadir una arquitectura de RAG (Retieval Augmented Generation) para aumentar la precisión de las respuestas que ofrece.

1.1. Gestión de proyectos

La gestión de proyectos es el conjunto de metodologías utilizadas para coordinar la organización, la motivación y el control de recursos con el fin de alcanzar un objetivo. En el caso de una empresa que se dedica a desarrollar software existen varias necesidades que tienen que ser suplidas, como gestionar varios proyectos a la vez o la disponibilidad de toda la información de manera centralizada para poder ser accedida por cualquier desarrollador, supervisor o jefe de proyecto.

Dentro de LKS Next-GobTech se coordinan varios proyectos a la vez en todo momento, por lo que hace falta un programa de software capaz de ofrecer herramientas que ayuden a la gestión de estos.

1.2. Herramientas

1.2.1. JIRA

JIRA es una herramienta de software propietario desarrollada por Atlassian para coordinar proyectos basados en tareas, llamadas incidencias dentro de la jerga de la aplicación. Esta herramienta sirve tanto para uso interno, como para que acceda el cliente, pudiendo encontrar un punto centralizado donde compartir información sobre el progreso y el estado del proyecto.

Las incidencias son la división atómica de paquetes de trabajo, que representan una tarea cuantificable asignable a un desarrollador y que ayudan a medir el desarrollo llevado a cabo. Al disponer de estados para las incidencias, se puede consultar de manera sencilla cómo progresa el proyecto.

Dentro de estas se pueden registrar distintos datos, como el tiempo que se prevé que va a tomar la tarea y el tiempo real que toma, mediante registros de trabajo, medidos en horas. Asimismo, se puede incluir información de interés para quien

vaya a ser asignado el desarrollo de la incidencia, como una descripción, un resumen o enlaces externos a documentación relevante.

En un proyecto JIRA gestionado en LKS Next-GobTech se gestiona un flujo para las incidencias detallado a continuación: el desarrollador que la realice marcará la incidencia como hecha, a lo que un desarrollador senior validará el trabajo realizado y decidirá si es correcto o si ha de se mejorado. Una vez confirmado, se marcará como validada y podrá pasar a la vista del cliente, que podrá comprobar el trabajo realizado.

1.2.2. Git - Gitlab

Al igual que se necesita controlar el estado de trabajos en el proyecto, también es necesario llevar un control de versiones para un óptimo desarrollo de software. En el caso de LKS Next-GobTech se utiliza git [1] como herramienta y Gitlab como punto centralizado donde guardar los repositorios.

Gitlab es una plataforma que permite gestionar las versiones del software y la colaboración entre desarrolladores. De esta manera, se crea un repositorio para cada proyecto que tiene la empresa y para cada uno de estos repositorios se otorgan permisos de modificación a los desarrolladores que vayan a trabajar en ese proyecto.

Además, se utiliza la integración de JIRA con Gitlab para relacionar las incidencias con cambios realizados en el repositorio asignado al proyecto, de manera que tanto la confirmación del trabajo realizado como del tiempo invertido pueden ser contrastados.

1.3. JiraGPT Next

Partiendo del trabajo realizado por Joel García, se dispone de JiraGPT Next como una herramienta que ayuda a recuperar incidencias filtradas utilizando lenguaje natural. De esta manera, una persona que no posea conocimiento técnico en la generación de consultas JQL podrá filtrar incidencias facilmente.

Tras esta herramienta se encuentra una llamada de API a un LLM que, utilizando una plantilla para guiar al modelo, pedirá que se traduzca la pregunta en lenguaje natural a una consulta JQL que responda a lo que se pide.

La idea detrás de este nuevo trabajo es realizar un estudio de la mejora de precisión obtenida utilizando arquitecturas RAG. Para ello, se propone modificar la estructura que se sigue para la generación de consultas JQL utilizando LangChain y bases de conocimiento de las que recuperar información relevante para la generación de la consulta.

Con este estudio se pretende observar si las distintas arquitecturas propuestas suponen un cambio significativo en la precisión de las consultas generadas.

2. Tecnologías

A continuación se detallarán las distintas tecnologías que serán estudiadas durante este TFG. Cabe recalcar que varias de estas distintas tecnologías propuestas, como los grafos de conocimiento o las ontologías, han requerido de un estudio previo para poder ser implementadas en el proyecto.

Independientemente de los resultados que se obtengan con cada una de ellas, es necesario tener en cuenta el proceso de familiarización con las mismas, así como el tiempo invertido en su estudio y posterior implementación para un desempeño óptimo.

2.1. Large Language Models

Los modelos de lenguaje grandes, conocidos en inglés como Large Language Model (LLM), son modelos computacionales de lenguaje que se componen de redes neuronales de muchos parámetros entrenados en grandes cuerpos de texto. Estos modelos principalmente son entrenados para predecir la siguiente palabra en una oración, pudiendo así generar respuestas basadas en el contexto proporcionado.

2.2. Retrieval Augmented Generation

Se conoce como Retrieval Augmented Generation (RAG) a la arquitectura que combina la recuperación de información con la generación de texto. Esta arquitectura se compone de dos partes principales: un modelo de recuperación y un modelo de generación. El modelo de recuperación se encarga de recuperar información relevante de una base de conocimiento, mientras que el modelo de generación se encarga de generar texto basado en la información recuperada.

Esta arquitectura es especialmente útil cuando se trabaja con modelos de lenguaje grandes, ya que mejora el problema de las alucinaciones. En lugar de generar respuestas en base al conocimiento del que disponen durante el entrenamiento, que puede dar resultados erróneos, el modelo puede acceder a bases de conocimiento factual con las que puede generar respuestas más precisas y acordes al contexto.

2.2.1. Funcionamiento

El funcionamiento típico de esta arquitectura consta de un flujo dividido en dos partes principales, la recuperación de contexto y la generación de respuestas, que se explicarán brevemente a continuación:

Durante la recuperación de contexto se consulta en una base de conocimiento, que podría ser una base de datos vectorial, un grafo de conocimiento o una ontología,

entre otros. Para hacer una consulta, se ha de contrastar la pregunta que un usuario haga con la información contenida, para obtener la información más relevante posible. Esta tarea requiere gran atención ya que es crucial de cara al desempeño que vaya a lograr el sistema.

Una vez se ha recuperado la información, se pasa a la generación de respuestas. En esta etapa, se utiliza la información recuperada junto con la pregunta inicial para guiar al modelo de lenguaje en la generación de respuestas. De esta manera, el modelo puede generar respuestas más precisas y acordes al contexto proporcionado.

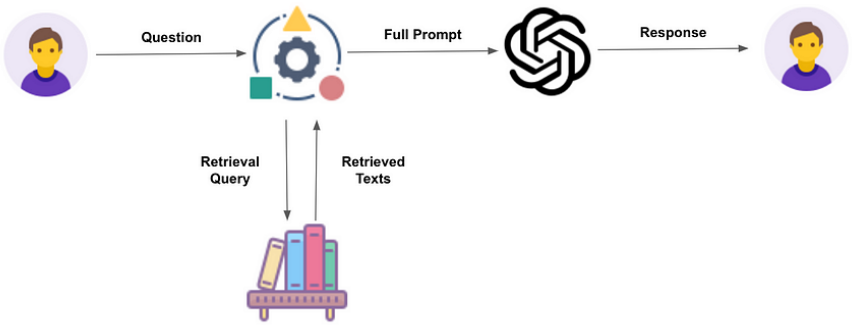


Figura 1: Esquema de funcionamiento de una arquitectura RAG.

2.3. Ontologías

Como posible base de conocimiento relevante para la generación de consultas JQL se propone crear una ontología que represente las reglas que existen en las consultas JQL. La información que se pretende representar en la ontología se ha extraído directamente de la documentación oficial de Jira, brindada por Atlassian, donde se detallan las reglas que se deben seguir para la creación de consultas JQL [2].

Como posible base de conocimiento relevante para la generación de consultas JQL se propone crear una ontología que represente las reglas que existen en las consultas JQL. La información que se pretende representar en la ontología se ha extraído directamente de la documentación oficial de JIRA, brindada por Atlassian, donde se detallan las reglas que se deben seguir para la creación de consultas JQL [2].

Referencias

[1] Scott Chacon y Ben Straub. *Pro git*. Apress, 2014.
[2] Atlassian. *Use advanced search with Jira Query Language (JQL)*. Accessed: 2024-03-30.