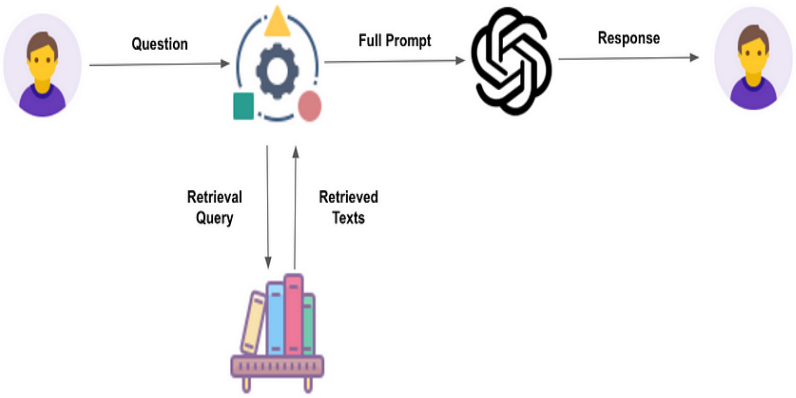
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

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: 2 de mayo 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

PLN Procesamiento del Lenguaje Natural

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 desarrollan. Partiendo del TFG de un compañero de escuela, Joel García Escribano, que consiste en un asistente conversacional que genera consultas JQL a partir de preguntas hechas con lenguaje natural, se ha estudiado la posibilidad de añadir una arquitectura de RAG (Retieval Augmented Generation) para aumentar la precisión de las respuestas que ofrece.

A continuación, se detallan las herramientas y metodologías que utiliza LKS Next-GobTech para la gestión de proyectos y se introduce la herramienta JiraGPTNext, que es el objeto de estudio de este trabajo.

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. Planificación

En esta sección se detallará la planificación del trabajo de fin de grado, en la que se incluirán los objetivos, la metodología y el cronograma de trabajo. Además, se incluyen los recursos necesarios para llevar a cabo el proyecto.

2.1. Tareas

2.2. Presupuesto

A lo largo de esta subsección se detalla el presupuesto necesario para el estudio transcurrido en este trabajo de fin de grado. Se incluyen los costes de los recursos humanos, los costes de los recursos materiales y los costes de los recursos software.

2.2.1. Costes de software

El estudio de este trabajo se ha realizado con herramientas de software de código abierto en la medida de lo posible, si bien también se ha hecho uso de llamadas de API de OpenAI.

Como editores de código y ontologías, se han utilizado Visual Studio Code y Protégé respectivamente, ambos de código abierto y gratuitos. Para la gestión de versiones se ha utilizado Git, también de código abierto y gratuito.

2.2.2. Costes de mano de obra

3. 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.

3.1. Modelos grandes de lenguaje

Dentro del campo de la inteligencia artificial y el Procesamiento del Lenguaje Natural (PLN), los modelos grandes de lenguaje, conocidos en inglés como Large Language Model (LLM) han sido una de las tecnologías más revolucionarias de los últimos años. Estos modelos son capaces de aprender de grandes cantidades de texto y generar texto de manera coherente y con sentido, pudiendo así responder a preguntas basándose en el contexto proporcionado.

Los LLMs se basan en arquitecturas de redes neuronales profundas, como los transformers [2], que permiten procesar secuencias de texto de manera más eficiente. Gracias a su mecanismo de atención, el cual permite al modelo enfocarse en las partes más relevantes de la secuencia de texto, los transformers han sido la base de muchos de los modelos de lenguaje más grandes y potentes de la actualidad.

A diferencia de modelos lingüísticos anteriores, los LLMs son capaces de aprender de manera no supervisada, lo que les permite obtener información de grandes cantidades de texto sin necesidad de etiquetas. Esto ha permitido el desarrollo de modelos masivos, como GPT-3 [3], que han demostrado ser capaces de realizar tareas de generación de texto, traducción, resumen, entre otras, con resultados sorprendentes.

3.2. 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.

3.2.1. 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.

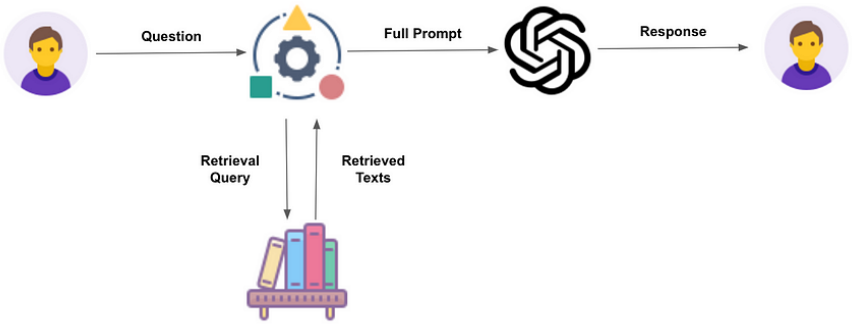


Figura 1: Esquema de funcionamiento de una arquitectura RAG.

3.3. Ontologías

Una ontología en el campo de la informática es una especificación formal y explícita de una conceptualización. En otros términos, una ontología es una representación de un conjunto de conceptos dentro de un dominio y las relaciones entre esos conceptos.

El lenguaje usado para definir ontologías es el *Web Ontology Language* (OWL), que es un lenguaje de marcado semántico para publicar y compartir ontologías en la web. OWL es desarrollado por el *World Wide Web Consortium* (W3C) y es una extensión de *Resource Description Framework* (RDF), que es un modelo de datos basado en grafos para describir recursos web.

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 [4]. Esta ontología serviría para interpretar las reglas que hay que seguir al generar consultas JQL, además, consta de ejemplos en cada una de las clases definidas, que ayuda a comprender mejor el funcionamiento de las reglas.

Esta ontología ha sido desarrollada utilizando el software *Protégé* [5], una herramienta de código abierto para la creación de ontologías desarrollada por la Universidad de Stanford.

Referencias

[1] Scott Chacon y Ben Straub. *Pro git*. Apress, 2014.

[2] Ashish Vaswani et al. «Attention is All you Need». En: *Advances in Neural Information Processing Systems*. Ed. por I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[3] Tom Brown et al. «Language Models are Few-Shot Learners». En: *Advances in Neural Information Processing Systems*. Ed. por H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, págs. 1877-1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[4] Atlassian. *Use advanced search with Jira Query Language (JQL)*. Accessed: 2024-03-30. URL: https://support.atlassian.com/jira-service-management-cloud/docs/use-advanced-search-with-jira-query-language-jql/.

[5] Stanford University. *Protégé*. Accessed: 2024-03-20. URL: https://protege.stanford.edu/.