MAY 14, 2025

Kowalski - Your Helpful AI Assistant

Christian Goll <cgoll@suse.com>

SUSE

Outline

Introduction

Introducing Kowalski

Kowalski Workflow

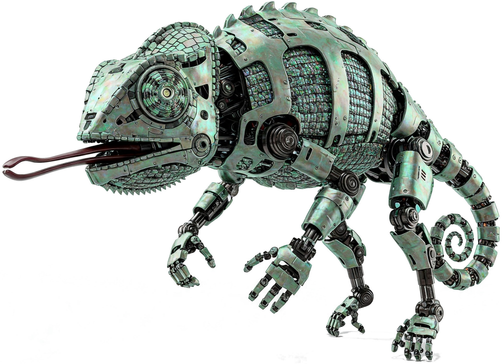
User Interaction

Benefits

The Challenge: LLMs and System Configuration

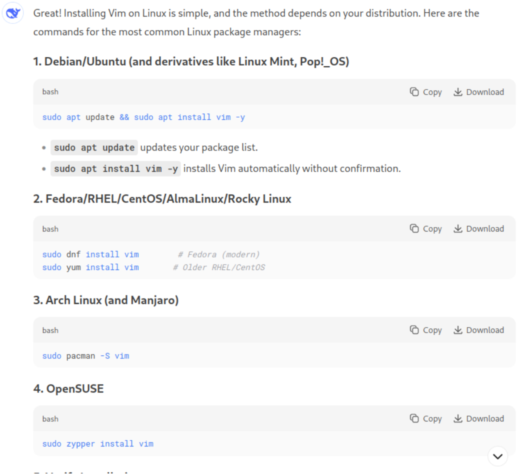
Current state

- Large Language Models (LLMs) are powerful for general tasks.
- LLMs are allready used for system configuration tasks
- Halucinations and wrong tools likely to be presented



Example

- Answers with standard prompt are extensive
- User has to select the right one
- Sensitive information may be shared!



Great! Installing Vim on Linux is simple, and the method depends on your distribution. Here are the commands for the most common Linux package managers:

1. Debian/Ubuntu (and derivatives like Linux Mint, Pop!_OS)

```bash
sudo apt update && sudo apt install vim -y
```

- `sudo apt update` updates your package list.
- `sudo apt install vim -y` installs Vim automatically without confirmation.

2. Fedora/RHEL/CentOS/AlmaLinux/Rocky Linux

```bash
sudo dnf install vim        # Fedora (modern)
sudo yum install vim       # Older RHEL/CentOS
```

3. Arch Linux (and Manjaro)

```bash
sudo pacman -S vim
```

4. OpenSUSE

```bash
sudo zypper install vim
```

Problem

- LLMs are created by training text completion on scraped sources
- LLMs don’t have any context

You know nothing, John Snow!

Solution

- Context is needed for accurate solutions
- Should be easy to get context of actual system.

I know things!

Introducing Kowalski: A Specialized AI Assistant

- Kowalski is designed to bridge the gap between LLMs and system configuration.
- It provides LLMs with highly relevant and context-aware information.
- Focus on extracting and utilizing knowledge from technical documentation, specifically SUSE/SLE docs.

Excurse: Vector Embeddings

Definition

Numerical representations of data (words, texts, etc.) in a vector space

Key Properties

- Capture semantic relationships (e.g., ”king” – ”man” + ”woman” ≈ ”queen”).
- Fixed-length (e.g., 300 dimensions for Word2Vec).
- Similar items cluster together in the vector space.

Example

```
"cat"   [0.2, -0.5, 0.7], "dog"   [0.3, -0.4, 0.6]
```

How are Embeddings Created?

Common Methods

GloVe:

Uses global word co-occurrence statistics from a corpus.

Word2Vec (Skip-gram/CBOW):

Predicts surrounding words (or target word) using shallow neural networks.

Vector space search

- Searching nearest neighbors in high dimesional vector space isn’t trivial
- Facebook created `faiss` library which does this job

Applications of Vector Embeddings

Use Cases:

- **NLP:** Search engines, chatbots, translation.
- **Computer Vision:** Image similarity (e.g., CLIP).
- **Recommendation Systems:** User/item embeddings (e.g., Netflix).

Tools:

- `gensim` (Word2Vec), `sentence-transformers`, `OpenAI Embeddings API`.

Kowalski’s Core Idea: Contextualized Information

- Enhance the LLM’s ability to perform configuration tasks.
- Achieve this by providing accurate information extracted directly from trusted sources (documentation).
- Supplement documentation knowledge with live system information.

Phase 1: Documentation Ingestion

The Starting Point: SUSE/SLE Documentation

- Kowalski begins by processing official documentation.
- This documentation is a rich source of configuration knowledge.
- The goal is to make this information machine-readable and easily retrievable.

Extraction Process: Identifying Key Information

What is Extracted?

- Mentioned files (e.g., configuration files, scripts).
- Commands to be executed (e.g., installation, configuration commands).
- Key parameters and values within documentation examples.

How is it Extracted?

- Utilizing parsing techniques.
- Applying Natural Language Processing (NLP) to identify relevant entities.
- Defining patterns and rules to capture structured information.

Building the Knowledge Base: Internal Database

- Extracted files, commands, and related documentation snippets are stored.
- An internal database is created to organize this structured information.
- This database serves as a central repository of documentation-derived knowledge.

Semantic Understanding: Creating Embeddings

- Embeddings are created from the documentation content.
- These embeddings capture the semantic meaning of the text.
- This allows for searching based on concept and relevance, not just keywords.

The User Interaction: Asking Kowalski

User Query

- A user poses a question to Kowalski regarding a system configuration task.
- The query is the starting point for retrieving relevant information.

Query Embedding: Understanding the User’s Need

- The user’s query is also converted into an embedding.
- This puts the query into the same semantic space as the documentation embeddings.
- Enables comparing the user’s request to the stored knowledge.

Finding Relevant Information: Vector Search with FAISS

- FAISS (Facebook AI Similarity Search) is used for efficient vector similarity search.
- The embedding of the user query is compared against the documentation embeddings.
- The nearest (most semantically similar) help documents are identified.

Enriching the Context: Adding System Specifics

Going Beyond Documentation

- The extracted relevant documentation snippets are a primary source of context.
- Kowalski then checks if files mentioned in the documentation exist on the *actual* system.
- The content of these existing system files is extracted.

The Enriched Context for the LLM

- The LLM receives a context that includes:
- Relevant information from the documentation (extracted text).
- Information about specific files found on the user’s system that were mentioned in the documentation.
- This provides the LLM with a much more accurate picture of the user’s situation.

How it Works Together: The Kowalski Workflow

1. Documentation Ingestion & Extraction
2. Database Creation & Embedding Generation
3. User Query
4. Query Embedding
5. Vector Search (FAISS)
6. Retrieve Relevant Docs
7. Check for Mentioned System Files
8. Extract System File Content
9. Provide Enriched Context to LLM
10. LLM Generates Configuration Advice

Benefits of Using Kowalski

- **Accuracy:** LLM responses are grounded in trusted documentation.
- **Relevance:** Context is tailored to the specific documentation and the user’s system.
- **Efficiency:** Users get direct answers for configuration tasks.
- **Reduced Hallucination:** Providing specific context minimizes the risk of incorrect information.
- **Improved LLM Performance:** LLMs can provide more helpful and actionable configuration steps.