RAG FRAMEWORK – WRENCHBOT

Current Outputs:

```
(env_haystack) C:\Users\lenovo\env_haystack\Scripts>python app2.py
C:\Users\lenovo\env_haystack\lib\site-packages\torchvision\datapoints\__init__.py:12: UserWarning: The torchvision.datap
oints and torchvision.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some APIs
may still change according to user feedback. Please submit any feedback you may have in this issue: https://github.com/p
ytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/vision/issues/7319 to learn more about
the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_beta_
transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
C:\Users\lenovo\env_haystack\lib\site-packages\torchvision\transforms\v2\__init__.py:54: UserWarning: The torchvision.da
tapoints and torchvision.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some AP
Is may still change according to user feedback. Please submit any feedback you may have in this issue: https://github.co
m/pytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/vision/issues/7319 to learn more abo
ut the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_be
ta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
faiss_document_store.db removed successfully.
Failed to remove faiss_index after 3 retries.
FAISS Document Store initialized successfully.
```

```
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\lenovo\env_haystack\lib\site-packages\torchvision\datapoints\__init__.py:12: UserWarning: The torchvision.datap
oints and torchvision.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some APIs
may still change according to user feedback. Please submit any feedback you may have in this issue: https://github.com/p
ytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/vision/issues/7319 to learn more about
the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_beta_
transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
C:\Users\lenovo\env_haystack\lib\site-packages\torchvision\transforms\v2\__init__.py:54: UserWarning: The torchvision.da
tapoints and torchvision.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some AP
Is may still change according to user feedback. Please submit any feedback you may have in this issue: https://github.co
m/pytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/vision/issues/7319 to learn more abo
ut the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_be
ta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'faiss_do
cument_store.db'. Retrying in 1 seconds...
PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'faiss_do
cument_store.db'. Retrying in 1 seconds...
PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'faiss_do
cument_store.db'. Retrying in 1 seconds...
Failed to remove faiss_document_store.db after 3 retries.
Failed to remove faiss_index after 3 retries.
FAISS Document Store initialized successfully.
Device set to use cpu
```

```
Device set to use cpu
 * Debugger is active!
 * Debugger PIN: 103-829-097
127.0.0.1 - - [07/Jan/2025 14:51:58] "GET / HTTP/1.1" 200 -
Writing Documents: 10000it [00:21, 463.65it/s]
Batches: 100%|| 1/1 [00:00<00:00,  1.63it/s]
Batches: 100%|| 156/156 [00:35<00:00,  4.39it/s]
Documents Processed: 10000 docs [00:37, 269.01 docs/s]
Batches:  63%|| 111/177 [00:34<00:12,  5.42it/s]
```

```
-7.99922273e-02  9.51894745e-02  2.87396330e-02 -9.31561962e-02
 7.81395286e-02  1.51837226e-02 -5.32930307e-02  3.73006128e-02
-9.28860717e-03  2.55008438e-03  1.39783412e-01 -2.36369576e-02
 3.16954330e-02  6.42700046e-02 -3.68429795e-02  5.94265349e-02
-1.22152433e-01  8.72031972e-02  4.21575978e-02  8.06708410e-02
-7.81026259e-02 -3.75187546e-02 -2.86708195e-02 -6.59003258e-02
 2.09816098e-02 -3.38083766e-02 -3.14570926e-02 -8.87605101e-02
-2.52922028e-02 -9.50792059e-03 -6.75161257e-02  8.69221911e-02
 1.07280679e-01 -1.81774683e-02 -4.47932035e-02 -2.53155660e-02
 2.06599338e-03 -9.35200676e-02  4.61754709e-04  2.03861427e-02
 4.66542244e-02  6.24116583e-05 -1.74690410e-02  4.70426567e-02
 2.41117738e-02 -4.25333995e-03  8.14483911e-02 -1.30149465e-08
-1.28905512e-02 -4.07029390e-02  1.94521081e-02 -6.13717735e-02
-3.06136701e-02 -7.61105074e-03 -4.14322428e-02  3.81088024e-03
-5.18862866e-02  3.96385863e-02  3.73026133e-02 -1.03896789e-01
-1.31597277e-02 -2.03386229e-02 -3.71955819e-02 -5.35645038e-02
-3.26641053e-02 -3.65897454e-02  7.67884254e-02  1.84700545e-02
 8.56179371e-03  3.15458863e-03  9.59504172e-02  1.67213026e-02
-1.94673128e-02  8.66204947e-02  1.59361977e-02  6.12183399e-02
 7.59519869e-03  8.39055981e-03 -9.90616065e-03  3.95239145e-02
 3.10579943e-03 -6.12430014e-02 -1.11253403e-01 -2.44702604e-02
 3.27657759e-02  2.34473720e-02 -9.74710379e-03 -7.57475523e-03
-3.65024991e-02  7.69933546e-03  5.16887009e-02 -7.08034844e-04
-3.59880738e-02  2.77542230e-02 -4.11142409e-02  5.01930937e-02
 2.67442353e-02 -4.74321730e-02 -2.80732829e-02 -1.95737667e-02
 1.52277332e-02  7.85472840e-02 -4.42028530e-02 -2.42939107e-02
-9.44590475e-03 -1.85781289e-02 -6.22452945e-02  6.01181462e-02
-3.30157718e-03  8.09450969e-02 -2.94133257e-02 -7.80349225e-02]
127.0.0.1 - - [07/Jan/2025 14:54:21] "POST /initialize HTTP/1.1" 200 -
```

WEB INTERFACE:

came from.

explain power BI?

Microsoft Power BI is an embedded BI tool for Microsoft Office 365. It lets you use Microsoft Office products with Microsoft Power BI. The tool can be used to create and manage BI reports and dashboards. It can also be used as a tool to help you with other Microsoft products.

reports in wrench?

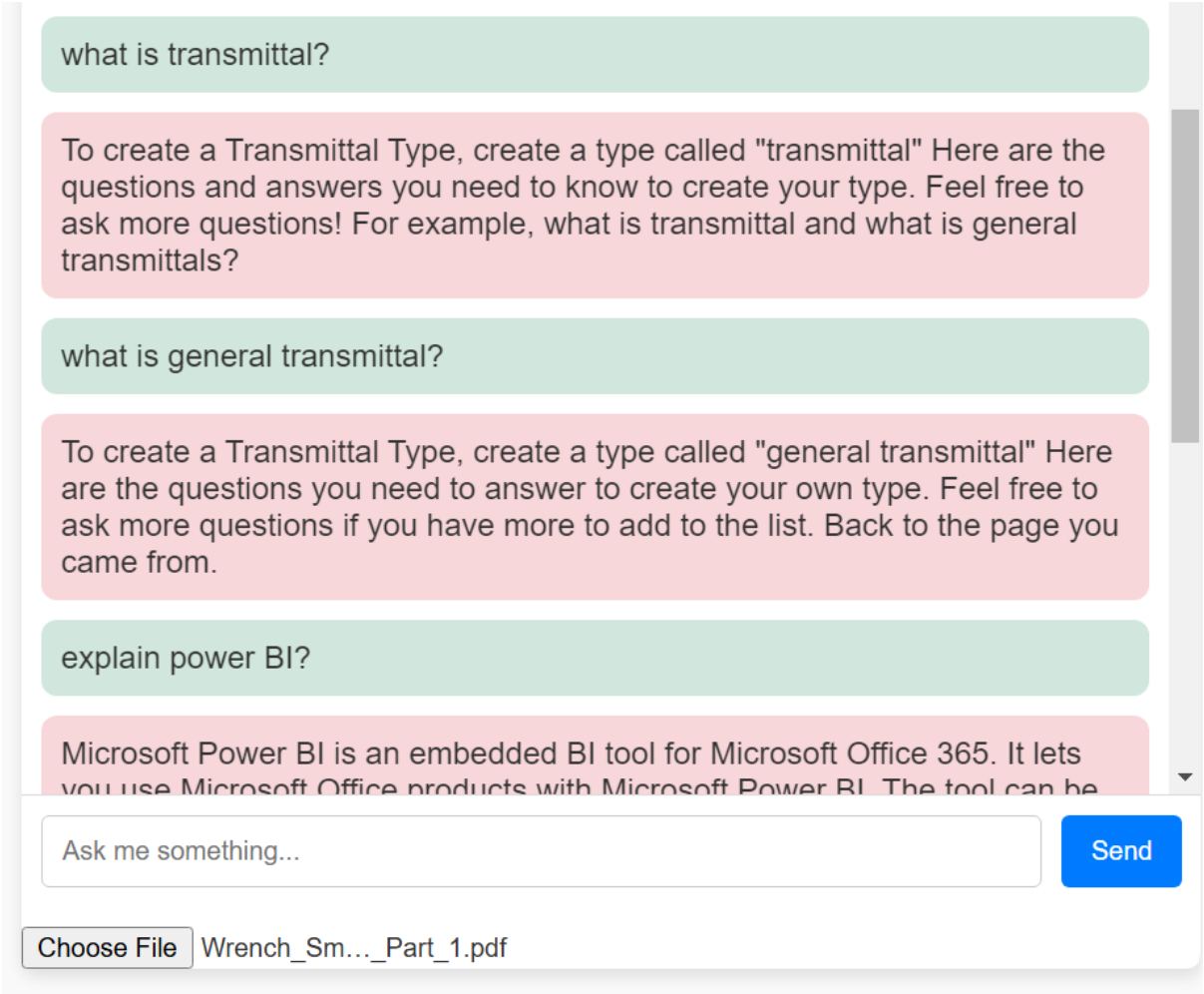
Wrench SmartProject has reports that are critical to project management and data management can be derived. Reports are configured in SmartProject Desktop version. While configuring Wrench Reports, in the Report Generation section, click About Wrench. Here you go! For more information on Wrench, visit the Wrench website.

tell me about wrench?

Ask me something...

Send

Choose File Wrench_Sm..._Part_1.pdf



---

1. **<u>Vector Embeddings:</u>**

- I use sentence-transformers/all-MiniLM-L6-v2 to create vector embeddings of document content. These embeddings allow the system to efficiently store and search for relevant information using the FAISS Document Store.

2. **<u>Large Language Model (LLM):</u>**

- Hugging Face's deepset/roberta-large-squad2 is used for question answering. It processes user queries and the relevant context retrieved from documents, then provides a precise answer based on the content.

3. **<u>BART for Paraphrasing:</u>**

- The generated answer is passed through BART (facebook/bart-large-cnn), a powerful transformer model, for paraphrasing. This helps produce more natural and varied responses for the user.

Code:

```
from flask import Flask, request, jsonify, render_template
from flask_cors import CORS
from haystack.document_stores import FAISSDocumentStore
from haystack.nodes import EmbeddingRetriever
from transformers import pipeline as hf_pipeline
import fitz  # PyMuPDF
import os
import logging
import time

# Initialize Flask App
app = Flask(__name__)
CORS(app)

# Directory to save uploaded files
UPLOAD_FOLDER = "data/uploads/"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def safe_remove(file_path, retries=3, delay=1):
    """Attempt to remove a file with retries."""
    for _ in range(retries):
        try:
            if os.path.exists(file_path):
                os.remove(file_path)
```

```
            print(f"{file_path} removed successfully.")
            return
        except PermissionError as e:
            print(f"PermissionError: {e}. Retrying in {delay} seconds...")
            time.sleep(delay)
    print(f"Failed to remove {file_path} after {retries} retries.")

# Ensure file cleanup
safe_remove("faiss_document_store.db")
safe_remove("faiss_index")

# Initialize FAISS Document Store
try:
    document_store = FAISSDocumentStore(
        sql_url="sqlite:///faiss_document_store.db",
        faiss_index_factory_str="Flat",
        embedding_dim=384  # Match the embedding model dimension
    )
    print("FAISS Document Store initialized successfully.")
except Exception as e:
    logging.error(f"Error initializing FAISS Document Store: {str(e)}")
    raise

# Initialize Embedding Retriever
retriever = EmbeddingRetriever(
    document_store=document_store,
```

```
    embedding_model="sentence-transformers/all-MiniLM-L6-v2",
    use_gpu=True
)

# Update embeddings if documents are present
if document_store.get_document_count() > 0:
    document_store.update_embeddings(retriever)

# Initialize Hugging Face QA pipeline
qa_pipeline = hf_pipeline(
    "question-answering",
    model="deepset/roberta-large-squad2",
    tokenizer="deepset/roberta-large-squad2",
    framework="pt"
)

# Initialize BART paraphrasing pipeline
paraphrase_pipeline = hf_pipeline("text2text-generation",
model="facebook/bart-large-cnn")

def paraphrase_answer(answer):
    """Paraphrase the generated answer using BART model."""
    paraphrased = paraphrase_pipeline(answer, max_length=512,
num_return_sequences=1)
    return paraphrased[0]['generated_text']

def extract_text_from_pdf(pdf_path):
```

```
    """Extract text from a PDF and organize it into sections."""
    doc = fitz.open(pdf_path)
    sections = []
    for page_num, page in enumerate(doc, start=1):
        content = page.get_text("blocks")  # Extract content as blocks
        for block in content:
            text = block[4].strip()
            if len(text) > 20:  # Ignore very short lines
                sections.append({
                    "content": text,
                    "meta": {"page_number": page_num, "source": pdf_path}
                })
    return sections

def format_answer(query, raw_answer, context):
    """Format the response into structured output with additional details."""
    return (
        f"Here you go!\n\n"
        f"**{query.capitalize()}**\n\n"
        f"**Answer:** {raw_answer}\n\n"
        f"**Details:**\n{context}\n\n"
        f"Feel free to ask more questions!"
    )

def generate_structured_response(query, answer, context_list):
    """Generate a structured response dynamically."""
```

```
    if answer == "No relevant information found.":
        return (
            f"I'm sorry, I couldn't find any relevant information about '{query}'. "
            "Could you try rephrasing your question or provide more details?"
        )

    # Create a structured response
    context_summary = "\n".join(context_list)
    return (
        f"**{query.capitalize()}**\n\n"
        f"**Answer:** {answer}\n\n"
        f"**Additional Details:**\n{context_summary}\n\n"
        "Let me know if you'd like further clarification or have more questions!"
    )

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/initialize', methods=['POST'])
def initialize():
    global document_initialized
    try:
        if 'file' not in request.files:
            return jsonify({"error": "No file uploaded."}), 400
```

```
    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No selected file."}), 400

    if not file.filename.endswith('.pdf'):
        return jsonify({"error": "Only PDF files are allowed."}), 400

    file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
    file.save(file_path)

    # Extract text and index it
    sections = extract_text_from_pdf(file_path)
    documents = [
        {"content": section["content"], "meta": section["meta"]}
        for section in sections
    ]
    document_store.write_documents(documents)

    # Update embeddings in FAISS
    document_store.update_embeddings(retriever)
    document_initialized = True

    # Print embeddings for debugging
    embeddings = retriever.embed_queries([doc['content'] for doc in
documents])
    for idx, embedding in enumerate(embeddings):
        print(f"Document {idx + 1} embedding: {embedding}")
```

```
        return jsonify({"message": "Document uploaded and processed successfully!"}), 200
    except Exception as e:
        logging.error(f"Error during initialization: {str(e)}")
        return jsonify({"error": str(e)}), 500

@app.route('/ask', methods=['GET'])
def ask():
    if not document_initialized:
        return jsonify({"error": "No document has been uploaded or processed yet."}), 400

    query = request.args.get('query')
    if not query:
        return jsonify({"error": "No query provided."}), 400

    # Handle basic greetings or acknowledgments
    lower_query = query.lower()
    if lower_query in ["hi", "hello"]:
        return jsonify({"response": "Hello! How can I assist you today?"}), 200
    elif lower_query in ["thank you", "thanks"]:
        return jsonify({"response": "You're welcome! Feel free to ask more questions."}), 200

    try:
        # Retrieve relevant documents
```

```
        retrieved_docs = retriever.retrieve(query=query, top_k=10)  # Adjust top_k as needed
        contexts = [doc.content for doc in retrieved_docs]

        # Process combined context with QA pipeline
        combined_context = " ".join(contexts[:3])  # Combine top 3 contexts
        result = qa_pipeline(question=query, context=combined_context)

        # Format the answer
        raw_answer = result.get("answer", "No relevant information found.")
        context_snippet = "\n".join(contexts[:3])  # Include additional context for clarity
        formatted_answer = format_answer(query, raw_answer, context_snippet)

        # Paraphrase the answer before sending it back
        paraphrased_answer = paraphrase_answer(formatted_answer)

        return jsonify({"query": query, "answer": paraphrased_answer}), 200
    except Exception as e:
        logging.error(f"Error during question answering: {str(e)}")
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
    app.run(debug=True)
```

INSTALLATIONS:

- pip install Flask
- pip install flask-cors
- pip install farm-haystack[colab,inference]
- pip install transformers
- pip install pymupdf
- pip install faiss-cpu # If using CPU version(I used CPU version)
- # OR
- pip install faiss-gpu # If using GPU version
- pip install torch

VERSIONS:

- **farm-haystack**: 1.26.4
- **transformers**: 4.47.1
- **torch**: 2.0.1+cu118 (for CUDA support, if applicable)
- **sentence-transformers**: 3.3.1
- **faiss-cpu**: 1.7.2
- **rank-bm25**: 0.2.2
- **chromadb**: 0.5.23
- **pydantic**: 1.10.19
- **Flask**: 3.1.0
- **Flask-Cors**: 5.0.0

Llama 2 and 3

- **Pros**:
    - High-performance, large-scale models suitable for a range of NLP tasks.
    - Provides strong language understanding and generation capabilities.
    - Can be fine-tuned for various specific applications.
- **Cons**:

    - Permission restrictions limiting its usage in some contexts.
    - Potential computationally expensive for specific environments or use cases.

MiniLM (Currently Used)

- **Pros**:
    - **Lightweight** and optimized for fast inference, ideal for resource-constrained environments.
    - **Open-source**, providing flexibility for custom applications and integrations.
    - Suitable for a wide range of NLP tasks, with good balance between performance and efficiency.
    - **Low computational requirements**, making it easier to deploy and use in a variety of systems.
- **Cons**:
    - May not perform as well as larger models for complex or long-form tasks.
    - Limited capacity for fine-tuning on specialized, highly specific domains.

GPT-Neo 125M

Pros

- Open-source and freely available for research and development.
- Lightweight model, fast for inference and experimentation.
- Suitable for smaller tasks or prototyping.
- Fine-tuning possible for specific use cases.

Cons

- Limited performance due to the small number of parameters (125M).

- Lacks the ability to handle complex queries or tasks compared to larger models.
- Lower accuracy and coherence for long-form or detailed text generation.

GPT-2

Pros

- Freely available and widely used in research.
- Proven effectiveness for various NLP tasks like summarization, text generation, and more.
- Scalable with multiple parameter sizes (small, medium, large).
- Good documentation and community support.

Cons

- Outdated compared to more modern models like GPT-3 or GPT-4.
- Lacks fine-tuned versions for instruction-based tasks or chat-specific purposes.
- Requires significant computational resources for larger parameter sizes.

DialoGPT

Pros

- Optimized for conversational tasks; designed specifically for dialogue generation.
- Open-source and available for custom fine-tuning.
- Pre-trained on conversational datasets, making it more effective for chatbots and dialogue applications.

Cons

- Limited general-purpose capabilities outside conversational tasks.
- Performance varies significantly depending on the dialogue complexity.

- Outperformed by newer models like ChatGPT and other conversational LLMs.

Milvus

Pros

- An open-source vector database for similarity search and embedding storage.
- Optimized for managing and querying large-scale embeddings, such as those from language models.
- Can be integrated with many LLMs like GPT-Neo, GPT-J, or GPT-2 for efficient retrieval-based applications.
- Excellent scalability and performance for retrieval-based tasks.

Cons

- Not an LLM; requires external models for generating or managing embeddings.
- Complexity in setup and integration for non-expert users.
- Requires significant computational resources for large datasets.

GPT-J

Pros

- Open-source and comparable to GPT-3 in terms of architecture.
- Larger model (6B parameters) than GPT-Neo 125M, providing better accuracy and text coherence.
- Suitable for many NLP tasks, including summarization, question-answering, and more.
- Supports fine-tuning for custom applications.

Cons

- High computational requirements for training and inference.

- Limited support and pre-trained datasets compared to proprietary models like GPT-3 or GPT-4.
- Lacks conversational fine-tuning by default.

GPT-Neo (Varied Sizes)

Pros

- Open-source with multiple sizes (e.g., 125M, 1.3B, 2.7B).
- Good balance of performance and accessibility for medium and large parameter models.
- Widely used for text generation, summarization, and other NLP tasks.
- Community-driven development and documentation.

Cons

- Performance depends on the model size (smaller models struggle with complex tasks).
- Requires more computational resources than smaller models like GPT-2 or GPT-Neo 125M.
- No native instruction-following fine-tuning like proprietary GPT-3 models.

T5 (Text-to-Text Transfer Transformer)

- **Pros**:
    - Unified framework for diverse NLP tasks.
    - Strong performance on various benchmarks.
    - Versatile and can be fine-tuned for many tasks.
- **Cons**:
    - Large and computationally expensive.
    - Requires significant resources for fine-tuning.
    - Overkill for simple tasks.

BART (Bidirectional and Auto-Regressive Transformers)

- **Pros**:
    - Excellent for both understanding and generation tasks.
    - Performs well in text generation and summarization.
    - Flexible for multiple NLP tasks.
- **Cons**:
    - Computationally expensive.
    - Fine-tuning requires significant resources.
    - Struggles with long sequences.

Flan-T5

- **Pros**:
    - Fine-tuned for instruction-following tasks.
    - Improves task-specific accuracy.
    - Great for applications that require following user instructions.
- **Cons**:
    - Requires substantial computational resources.
    - Performance may vary on non-instructional tasks.
    - May be resource-heavy for smaller applications.

Report by: Nandana Manoj