Report on AI ChatBot for Legal Services Automation Project

1. Introduction

Project Overview:

The goal of this project is to develop an AI chatbot that can process legal documents (PDFs of corporate acts) submitted by users and identify predefined services from a company service menu. The chatbot will interact with users to gather missing information, answer questions, and finally provide a quote with the services and prices. If the user does not upload the legal document, they can still converse with the chatbot to explain their needs.

Dataset:

Three datasets were developed for this project:

- Dataset 1: Document content, decisions, and services (100 rows)
- Dataset 2: Title, description, category, and relationships between decisions and services (21 rows)
- Dataset 3: Title, scope, city, state, and value of each service (509,013 rows)

Tools and Technologies:

1. VScode for development environment
2. Python as development language
3. LangChain as frameworkt for chatbot
4. Pinecone for vector database
5. HugginFace for embeddings
6. OpenAPI for LLM
7. Streamlit for UI
8. Docker for build and deploy
9. Railway as application server

2. Data Exploration and Preprocessing

Datasets 2 and 3 were developed specifically for the project, so no adjustments were necessary.

However, Dataset 1 was constructed from PDFs, some of which were not created with text but rather with scanned documents. It was necessary to create a function to extract the content using Optical Character Recognition (OCR). This function was designed to be generic and was later used as a tool for the LangChain agent.

```python
def extract_text_from_pdf(pdf_path):
    """
    Extract text from a PDF file using PyMuPDF (fitz).
    Use OCR for scanned pages.

    Args:
    - pdf_path (str): Path to the PDF file.

    Returns:
    - str: Extracted text.
    """
    text = ""
    document = fitz.open(pdf_path)
    for page_num in range(len(document)):
        page = document.load_page(page_num)
        text += page.get_text()
        if not text.strip():  # If no text is extracted, use OCR
            pix = page.get_pixmap()
            img = Image.open(io.BytesIO(pix.tobytes()))
            text += pytesseract.image_to_string(img)
    cleaned_text = clean_text(text)
    return cleaned_text
```

A function was also created to remove characters that are not alphanumeric, basic punctuation, line breaks, tabs, and unnecessary spaces.

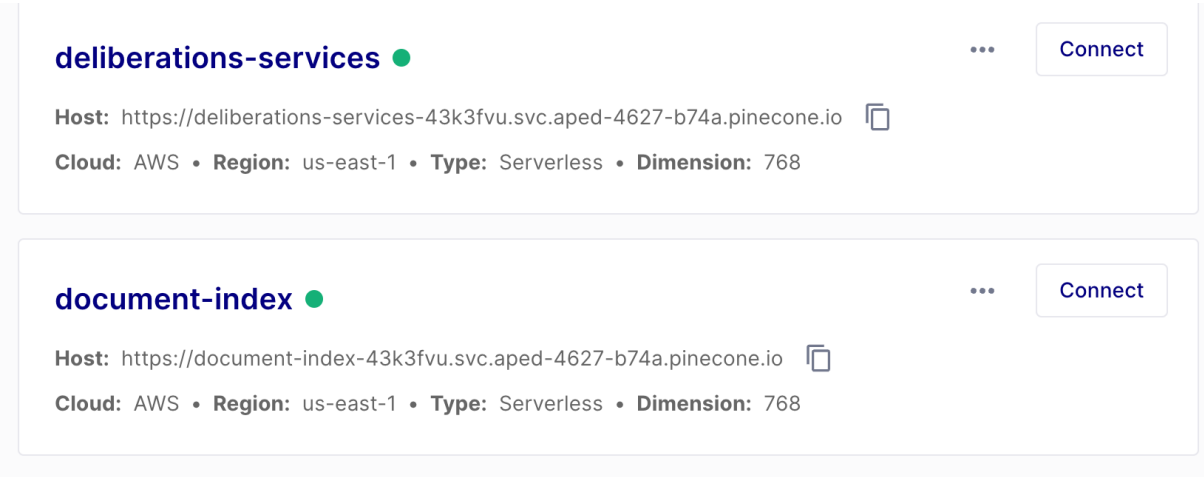
```python
def clean_text(text):
    """
    Clean the extracted text by removing unnecessary characters and formatting.

    Args:
    - text (str): The extracted text.

    Returns:
    - str: The cleaned text.
    """
    # Remove multiple spaces and replace with a single space
    text = re.sub(r'\s+', ' ', text)
    # Remove line breaks and tabs
    text = text.replace('\n', ' ').replace('\t', ' ')
    # Remove non-alphanumeric characters and basic punctuation
    text = re.sub(r'[^a-zA-Z0-9áàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ.,;!?()\-\'" ]', '', text)
    return text
```

3. Vector Database

For the vector database, we used Pinecone and created indexes for Dataset 1 (document-index) and Dataset 2 (deliberations-services). For the embeddings, the *neuralmind/bert-base-portuguese-cased* model from HuggingFace was used.



However, the document-index was not used in the project because, despite being potentially useful for supporting the classification of deliberations and services, it could expose sensitive information present in the documents. Additionally, since it contains only 100 samples, it is too small a dataset to effectively train a model.

No index was created for Dataset 3 because it contains information that can change regularly, such as prices, which may vary according to the client. This dataset will be explored with another type of LangChain agent tool.

4. LangChain Agent and Tools

A ZERO_SHOT_REACT_DESCRIPTION type LangChain agent was created using OpenAI (gpt-3.5-turbo) as the LLM. Prompt engineering was performed by passing a prefix to the agent.

```
agent_kwargs = {'prefix': f'''
                Você é um especialista em atos societários e trabalha para a empresa A2 Soluções inteligentes.
                Qualquer recomendação que você fizer deverá ser para profissionais da sua empresa que é a A2 Soluções inteligentes.
                Não mande consultar qualquer outro profissional que não seja da A2 Soluções inteligentes.
                Sempre responsa em Português do Brasil.
                '''}


agent = initialize_agent(      You, 2 hours ago • project to deploy
    tools,
    llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True,
    agent_kwargs=agent_kwargs,
    handle_parsing_errors=True,
    max_iterations=3,
    early_stopping_method='generate',
)
```

Three tools were created for the agent:

```
tools = [
    Tool(
        name = "Deliberations",
        func=deliberations.run,
        description="""
        Use para qualquer todas as questões que envolverem assuntos relacionados a empresas e atos societários.
        """
    ),
    Tool(
        name="PDF Extraction",
        func=pdf_tool.run,
        description="""
        Use esta ferramenta para extrair texto de arquivos PDF.
        Só execute essa ferramenta se você tiver um arquivo PDF para extrair texto.
        Nunca use mais de uma vez.
        """
    ),
    Tool(
        name = "Services prices",
        func=python.run,
        description = f"""
        Essa ferramenta só dever ser utilizada depois que a ferramenta 'Deliberations' for utilizada.
        Use essa ferramenta para responder perguntas sobre os preços dos serviços que estão armazenados em um dataframe pandas 'df'.
        Execute python pandas operations em 'df' para ajudá-lo a obter a resposta certa.
        Nunca passe o input diretamente para a função 'run', sempre faça a manipulação necessária antes de passar para a função 'run'.
        'df' tem as seguintes colunas: {df.columns}
        Os serviços podem ser consultados por seus nomes na coluna 'Serviços'.
        Você pode pesquisar os serviços utilizando df[df['Serviços'].str.contains('exemplo de serviços')]
        A coluna 'Valor' contém o valor dos serviços. Caso esteja em branco, considere o valor como 100.
        A coluna 'Atributo' contém o nome da cidade onde o serviço será prestado.
        Sempre apresente os valores no formato R$ 0,00.
        """      You, 2 hours ago • project to deploy
    ),
]
```

Deliberations: Responsible for querying information about decisions and services in the vector database. Prompt engineering was performed to ensure the best results.

```
vectorstore = LangChainPinecone(
    index=index,
    embedding=embeddings,
    text_key="description"
)

llm = ChatOpenAI(
    openai_api_key=os.environ.get("OPENAI_API_KEY"),
    model_name='gpt-3.5-turbo',
    temperature=0.0
)

prompt = PromptTemplate.from_template(
    """
    Use as seguintes informações de contexto para responder quais são as deliberações e serviços necessários para atender ao ato societário.
    Devolva uma lista de deliberações e serviços que devem ser realizados para atender ao ato societário.
    Devolva a cidade onde os serão prestados.
    Se você não souber a resposta, apenas diga que não sabe, não tente inventar uma resposta.

    {context}

    Questão: {question}      You, 2 hours ago • project to deploy
    Resposta útil:
    """
)

deliberations = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=vectorstore.as_retriever(),
    chain_type_kwargs={"prompt": prompt}
)
```

PDF Extraction: Responsible for extracting the content from the PDF, whether it is text-based or image-based, and removing unnecessary content. Utilizes the functions developed in the data exploration and preprocessing phase.

```
class PDFExtractionTool:
    def __init__(self, extraction_function):
        self.extraction_function = extraction_function

    def run(self, pdf_path):
        return self.extraction_function(pdf_path)

pdf_tool = PDFExtractionTool(extract_text_from_pdf)
```

Services prices: Responsible for querying the service prices within a pandas dataframe. This is a very interesting feature as it allows access to anything at the pure Python level.

```
df = pd.read_csv("data/prices.csv")
python = PythonAstREPLTool(locals={"df": df}, verbose=True)
```

5. Deploy

To provide a graphical interface for the chatbot, the Streamlit package was used. Docker was utilized to create an image of the application, ensuring that all project dependencies are correctly installed regardless of the environment in which it is executed. Railway was chosen to run the container with the Docker image.

The app is available at https://chatbot-production-bd08.up.railway.app/

6. Future Work

- Implement a memory mechanism for the agent.
- Modify the Services prices tool so that the data is consumed from a relational database or a Rest API.
- Create automated methods for evaluating the model's responses.
- Improve the deliberations-services index to enhance the agent's responses.
- Perform more prompt engineering to improve interaction between the agent's tools.
- Test with open-source LLM models and compare the results.
- Create a dataset to train a specific model for the classification of deliberations and services.