**Text Analytics Final Project: Knowledge Database Design Document**

**Team Members:**

**Ganti Uday**

**Mounica Pothureddy**

**Prasuna Challa**

**Veera Mukesh Aripaka**

**Anantha Sai Ram Padala**

**PHASE – I**

1. **Data Sources Identification**

Here's a formatted summary of the identified data sources for augmenting LLM (Language Model) prompting for the MSBAIS Chatbot:

* **Jira Ticketing System (CSV File):** Extracts current issues, resolutions, and updates from the Jira ticketing system.
* **Canned Responses Document (Potential Queries):** Contains predefined responses for common queries or topics often asked by applicants or reapplicants, enabling quick and accurate responses.
* **MSBAIS-FAQ Document:** Includes a compilation of frequently asked questions (FAQs) about the MSBAIS program, serving as a reference for informative responses.
* **MSBAIS Current Students Canvas Course Modules:** Extracts information from course modules within the MSBAIS Canvas platform, covering topics such as Course Advising and registration, Graduate Teaching/Research Assistantships, CPT Internships, etc. To extract this Information we need to scrape the webpage.
* **Course Availability (Staff Schedule**): Provides information on course availability for upcoming semesters based on staff schedules, aiding students in planning their course selections.
* **Summer Internships (Companies and Roles**): Lists companies and corresponding roles that have been hired for summer internships, offering insights into potential internship opportunities for students.
* **Other Academic and Administrative Topics**: Covers various academic and administrative aspects, including Reduced Course Load (RCL) for international students, Practice Center projects, independent study, Academic integrity, Transfer credits, Academic probation, Professional development certificates, and guidelines on how to apply for graduation.

These diverse data sources collectively enrich the knowledge base of the MSBAIS Chatbot, enabling it to respond accurately and comprehensively to a wide array of inquiries, thereby enhancing the overall user experience for both prospective and current MSBAIS students. Regular updates and maintenance of these data sources are essential to ensure the chatbot remains up-to-date and reliable.

1. **Data Schema Development**

Pinecone is a vector database that can be used to store and search for embeddings of data from a variety of sources. Embeddings are a type of data that represent the semantic information. Vector databases offer optimized storage and querying capabilities for embeddings.

To use Pinecone, we first need to create an index. An index is a collection of records, each of which has a unique ID, an array of floats representing a vector embedding, metadata (key-value pairs), and sparse vectors.

Below is the figure of how a record looks in a vector database:

A screenshot of a phone

Description automatically generated

Once you have created an index, you can start adding data to it. For example, to add data from a Jira Ticketing system which is a json data file with questions and answers, we would first need to create chunks of data initially. A chunk is a collection of related records. Once after creating chunks of data, we can generate embeddings for each chunk and then upsert the embeddings into the Pinecone index created.

The following schema is used to store and search for embedding in a Pinecone index for all of our data sources:

ID: unique ID for each record

Values: embedding vector values

Metadata: text (e.g., the question or response from the Jira ticket)

A close-up of a white background

Description automatically generated

This schema allows to search for related information about msbais course, registration, or about university details etc. For example, you could use a similarity search to find documents that are similar to a given query document. You could also use the metadata fields to filter your search results.

1. **Vectorization Techniques**

Vectorization techniques are essential for converting textual data into numerical vectors, allowing machine learning models to process and analyze the text. The choice of vectorization technique depends on the specific task, dataset, and desired outcomes. Here are some widely used vectorization techniques suitable for converting textual data into numerical representations:

1. **TF-IDF (Term Frequency-Inverse Document Frequency):**

* Represents word importance in a document relative to a corpus, emphasizing less common words within a document.
* Ideal for text classification, clustering, and information retrieval tasks.

1. **Word2Vec:** Maps words to dense vectors, preserving semantic relationships and enabling applications like sentiment analysis and language modeling.
2. **Doc2Vec (Paragraph Vectors):** Extends Word2Vec to learn fixed-length feature representations for variable-length pieces of text, useful for document-level analyses.
3. **BERT (Bidirectional Encoder Representations from Transformers):**
   * BERT is a transformer-based model that generates contextualized word embeddings.
   * Contextual embeddings capture word meanings based on surrounding context, improving representation accuracy.
   * Ideal for a range of NLP tasks including text classification, question-answering, and named entity recognition.
4. **GloVe (Global Vectors for Word Representation**): Learns word vectors by analyzing global word co-occurrence statistics, offering a global perspective of word meanings and relationships.
5. **Tf-Idf weighted Word Embeddings:** Enhances word embeddings using TF-IDF weighting, giving more weight to important words, improving word representations.
6. OpenAI Ada:

* OpenAI Ada is a highly advanced language model based on the GPT-3.5 architecture, known for its ability to generate human-like text and comprehend context effectively.
* Ada can be used for vectorization by encoding text into numerical representations, making it suitable for various NLP tasks, including text summarization, language translation, chatbot development, and more.

The Document Embedding API, from open API, is the best approach relative to TF-IDF, one Hot encoding & word2Vec approach that we have seen because of the below:

1. Contextual Embeddings:

* OpenAI Ada: Ada generates contextual embeddings, considering the entire context of a sentence or paragraph. It understands and represents words based on their surroundings, capturing nuances and context effectively.
* TF-IDF, One-Hot Encoding, Word2Vec: These techniques generate fixed embeddings without considering context, potentially losing nuances and subtle meanings.

1. Semantic Understanding:

* OpenAI Ada: The model demonstrates a deep understanding of semantics, allowing it to generate embeddings that encapsulate the underlying meaning of the text.
* TF-IDF, One-Hot Encoding, Word2Vec: While effective, these approaches may fall short of representing semantic relationships accurately.

1. Adaptability to Various NLP Tasks:

* OpenAI Ada: The Ada model is versatile and can be fine-tuned for specific NLP tasks, making it adaptable and suitable for a wide array of applications.
* TF-IDF, One-Hot Encoding, Word2Vec: While these techniques are versatile, they may not be as easily adaptable to different NLP tasks without further modifications.