A Weekly Update on

DiaCare: An Intelligent Diabetes Management Application with LLM-Augmented Chatbot and ML-Based Early Risk Prediction

27 February, 2025

Group Information

Group-01 CSE299 (Section-17)

1. Saif Mohammed - 2121913042
2. Nazibul Islam Nabil - 2222456642
3. Humayra Rahman Nipa - 2121128042
4. Umme Suraia Haque Setu - 2031278642

Weekly Update Brief

- **Machine Learning:** Completed Preprocessing, including data cleaning, handling categorical data, feature engineering, and train-test split.
- **LLM Chatbot:** A comparative analysise of
  - different Large Language Models
  - different embedding models
  - different chunking strategies

Machine Learning (Preprocessing)

Data Loading

In [53]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import sklearn as scikit_learn
```

In [54]:

```python
csv_path = "/Users/saifmohammed/Desktop/CSE299-Junior-Design-Project/ML/Test/Copy Dataset/Diabetes_Final_Data_V21.csv"
df = pd.read_csv(csv_path)
df
```

Out[54]:

	age	gender	pulse_rate	systolic_bp	diastolic_bp	glucose	height	weight	bmi	family_diabetes	hypertensive	family_hypertens
0	42	Female	66	110	73	5.88	1.65	70.2	25.75	0	0	
1	35	Female	60	125	68	5.71	1.47	42.5	19.58	0	0	
2	62	Female	57	127	74	6.85	1.52	47.0	20.24	0	0	
3	73	Male	55	193	112	6.28	1.63	57.4	21.72	0	0	
4	68	Female	71	150	81	5.71	1.42	36.0	17.79	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
5432	74	Male	83	164	89	6.47	1.60	64.0	24.99	0	1	
5433	75	Male	67	141	104	8.31	1.65	62.0	22.75	0	0	
5434	40	Female	67	134	114	7.61	1.50	69.0	30.72	0	1	
5435	36	Female	62	139	80	4.90	1.52	41.5	17.87	0	0	
5436	26	Female	80	134	93	5.15	1.47	67.1	30.92	0	0	

Figure: Data loading

Analyzing Different Chunking Models

```
[96]: cluster_chunker = ClusterSemanticChunker(
          embedding_function=embedding_function,
          max_chunk_size=100,
          length_function=token_count
      )

      cluster_chunker_chunks = cluster_chunker.split_text(all_text)

      analyze_chunks(cluster_chunker_chunks, use_tokens=True)
```

```
Number of Chunks: 451

 ================================================== 200th Chunk ==================================================
 Anion gap >10 >12 >12 Variable
Mental status Alert Alert/ drowsy Stupor/coma Stupor/coma

 ================================================== 201st Chunk ==================================================
 NB: Effective serum osmolality: 2 [measured Nat (mEq/L) + Glucose (mmol/L); Anion Gap: (Na*)
-[Cl + HCO3 (mEq/L)]

No token overlap found
```

LLM Semantic Chunker

```
[101]: from sentence_transformers import SentenceTransformer
       import numpy as np

       # Load a pre-trained model for semantic text similarity
       model = SentenceTransformer("sentence-transformers/paraphrase-MiniLM-L6-v2")

       # Function to split the document into sentences or parts
       def split_document_into_sentences(all_text):
```

Figure: Different Chunking Strategies

Analyzing Cosine Similarity for Different Text Embedding Models

Cosine Similarity for intfloat/e5-small-v2

[51]:
```python
import numpy as np

def cosine_similarity(vec1, vec2):
    dot_product = np.dot(vec2, vec1)  # (79, 384) @ (384,) → (79,)
    norm_vec1 = np.linalg.norm(vec1)
    norm_vec2 = np.linalg.norm(vec2, axis=1)  # Compute norms for each document
    return dot_product / (norm_vec1 * norm_vec2)

# Convert document embeddings to NumPy array
document_embeddings = np.array(document_result)  # Shape: (79, 384)

# Compute cosine similarity for each document
similarities = cosine_similarity(query_result, document_embeddings)

# Get the top 5 most similar documents
top_indices = np.argsort(similarities)[::-1][:5]  # Sort in descending order and take top 5

# Print results
print("Top 5 Similar Documents:")
for i, idx in enumerate(top_indices):
    print(f"{i+1}. Document {idx} - Similarity: {similarities[idx]:.4f}")
```

```
Top 5 Similar Documents:
1. Document 9 - Similarity: 0.8518
2. Document 11 - Similarity: 0.8379
3. Document 15 - Similarity: 0.8342
4. Document 46 - Similarity: 0.8193
5. Document 59 - Similarity: 0.8179
```

Figure: Cosine Similarity for Different Models

Achievements

- Completed preprocessing for machine learning model development.
- Conducted an analysis of various LLM models.
- Evaluated multiple embedding models.
- Assessed different chunking models
- Performed a comparative analysis of these models

Technology Stack

- **Programming Language:** Python
- **Framework:** LangChain
- **Libraries:** Pandas, Matplotlib, Seaborn, langchain-experimental, mistralai, spacy, scikit-learn, langchain-ollama, sentencepiece, torch
- **Embedding Model:** OllamaEmbeddings
- **Vector Database:** chromadb
- **LLM:** HuggingFace LLM
- **Document Processing:** PyMuPDF, PyPDF

Work Distribution (This Week)

- **Saif Mohammed - 2121913042**
  - LLM Chatbot Text Embedding
- **Nazibul Islam Nabil - 2222456642**
  - LLM Chatbot Chunking Methods
- **Humayra Rahman Nipa - 2121128042**
  - ML Preproessing
- **Umme Suraia Haque Setu - 2031278642**
  - ML Preproessing

References

- A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2023.
- LangChain Engineer, "LLM Part: Learn RAG From Scratch – Python AI Tutorial," YouTube. [Online]. Available: *https://www.youtube.com/*. [Accessed: Feb. 19, 2025].
- P. Lewis, E. Denoyer, and S. Riedel, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv preprint arXiv:2005.11401*, May 2020. [Online]. Available: *https://arxiv.org/abs/2005.11401*.
- GPT-4 Tutorial, "RAG Prototyping: GPT-4 Tutorial: How to Chat With Multiple PDF Files (1000 pages of Tesla's 10-K Annual Reports)," YouTube. [Online]. Available: *https://www.youtube.com/*. [Accessed: Feb. 19, 2025].