Evaluating Performance Metrics in Bias Mitigation for Generated Patient Vignettes Using Large Language Models

Shaurya Kumar, Chinmay Agrawal

Abstract

Recent advancements in Large Language Models (LLMs) have introduced new possibilities for automating medical vignette generation, a process that could significantly aid in patient diagnosis. However, concerns about bias in the outputs of these models necessitate a comprehensive evaluation of their performance and fairness. Our study leverages a sophisticated LLM pipeline to generate medical vignettes from a dataset curated from clinical guidelines, biomedical literature, and clinical trials. The LLM pipeline involves multiple stages: a context retriever that loads and embeds clinical guidelines, biomedical literature, and clinical trials; a prompt generator that creates specific prompts based on user queries; a strong LLM (e.g., GPT-4) that generates vignettes; and a post-processing stage that refines the outputs for benchmarking. This workflow ensures that the generated vignettes are contextually relevant and tailored to the specified medical conditions.To explain the motivations for using these specific metrics, we delve into the statistical and model-based scoring methods. ROUGE and BLEU scores measure n-gram overlaps, while GPTScore and Semantic Entropy assess semantic similarity and fluency. Our dataset comprises clinical guidelines and biomedical literature, and we detail the sources and examples used in our analysis. By performing rigorous tests, we present the numerical outcomes of these metrics, demonstrating their significance in evaluating the quality of generated patient vignettes. In conclusion, our research emphasizes the importance of selecting appropriate evaluation metrics to ensure the accuracy and reliability of patient vignettes generated by LLMs. These findings contribute to improving the effectiveness of LLMs in clinical settings, ultimately aiding in unbiased and accurate patient diagnosis.

Introduction

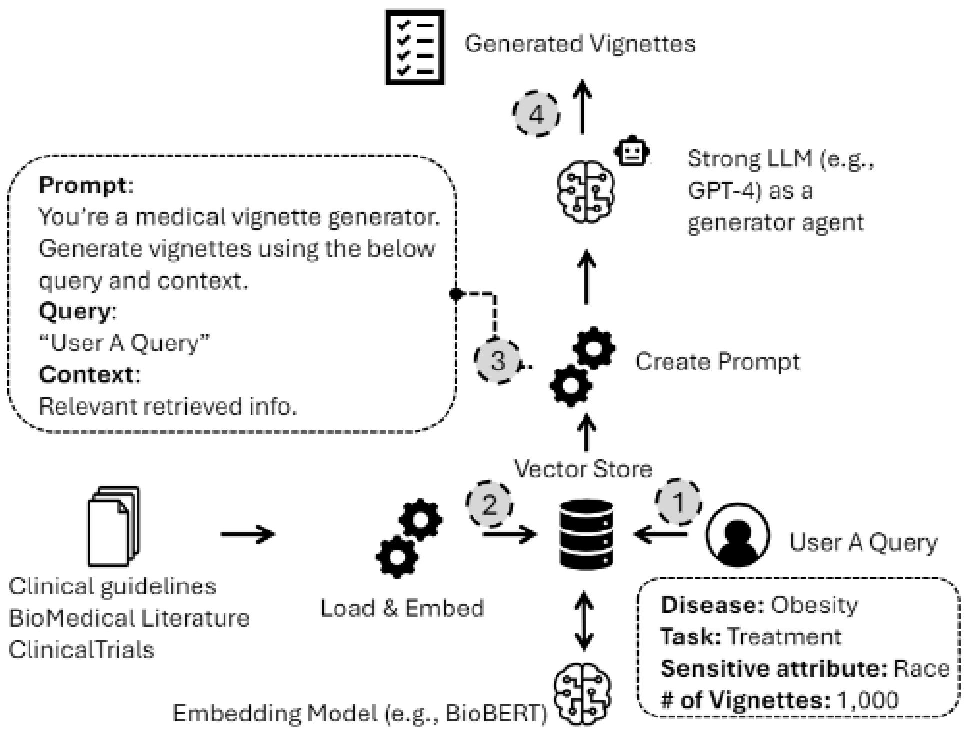
New developments in Large Language Models (LLMs) have introduced transformative possibilities for automating the generation of medical vignettes, which can significantly enhance the accuracy and efficiency of patient diagnosis. The project aims to develop a robust tool that integrates these advanced models with comprehensive datasets, ensuring that the vignettes generated are both contextually accurate and clinically relevant.

Our tool utilizes a multi-stage LLM pipeline designed to process and generate medical vignettes. The pipeline begins with a context retriever that loads and embeds data from clinical guidelines, biomedical literature, and clinical trials. This embedded information is then used by a prompt generator to create specific queries tailored to the user's needs. The generated prompts are processed by a powerful LLM, such as GPT-4, which produces detailed medical vignettes. Finally, a post-processing stage refines these vignettes to ensure they meet high standards of relevance and accuracy.

A critical aspect of our project is the evaluation of the summarization quality of the generated vignettes. To assess this, we employ a variety of evaluation metrics, including ROUGE and BLEU scores for measuring n-gram overlaps, and GPTScore and Semantic Entropy for evaluating semantic similarity and fluency. These metrics provide a comprehensive view of the performance and reliability of the vignettes, highlighting areas for improvement and ensuring that the outputs are unbiased and clinically useful.

By rigorously testing and presenting the outcomes of these metrics, we demonstrate their importance in the evaluation process. Our findings underscore the necessity of selecting appropriate evaluation metrics to guarantee the accuracy and dependability of LLM-generated patient vignettes. This research contributes to the ongoing efforts to refine LLM applications in clinical settings, ultimately aiding in the development of unbiased and precise diagnostic tools.

Figure 1.1


Generated Vignettes
4
Strong LLM (e.g., GPT-4) as a generator agent
Prompt: You're a medical vignette generator. Generate vignettes using the below query and context.
Query: "User A Query"
Context: Relevant retrieved info.
3
Create Prompt
Vector Store
2
1
User A Query
Clinical guidelines
BioMedical Literature
ClinicalTrials
Load & Embed
Disease: Obesity
Task: Treatment
Sensitive attribute: Race
# of Vignettes: 1,000
Embedding Model (e.g., BioBERT)

As seen in Figure 1.2 , the high-level workflow diagram of an AI-driven system details the interactions between components to process user inputs and generate refined outputs. The steps below detail the process:

1. **User Inputs**: The process begins with the user providing input data, which could be a query, a request for information, or any other type of data needing processing.
2. **Context Retriever**: This component is responsible for accessing and retrieving relevant information from the knowledge base. It ensures that the system has all the necessary context to understand and process the user input effectively.
3. **Knowledge Base**: A structured repository where all relevant information and data are stored. This database is essential for providing contextual information to the system.
4. **Prompt Generator**: Using the retrieved context, this component generates prompts or initial responses. These prompts serve as the foundation for further processing by the system.
5. **LLM (Large Language Model)**: The core processing unit of the system, the LLM takes the generated prompts and performs sophisticated language modeling tasks. It generates an initial response based on the given prompts and context.
6. **Refinement**: This step involves refining the initial response generated by the LLM. The refinement process may include additional processing, adjustments, or improvements to ensure the output is accurate and meets the user's needs.
7. **Post Processing**: After refinement, the output undergoes post-processing to enhance readability, format the response appropriately, or apply any final adjustments. This ensures the output is polished and ready for delivery.
8. **Benchmark**: The final output is compared against benchmark criteria to evaluate its quality, accuracy, and relevance. This step ensures the system's performance is consistently meeting the desired standards.

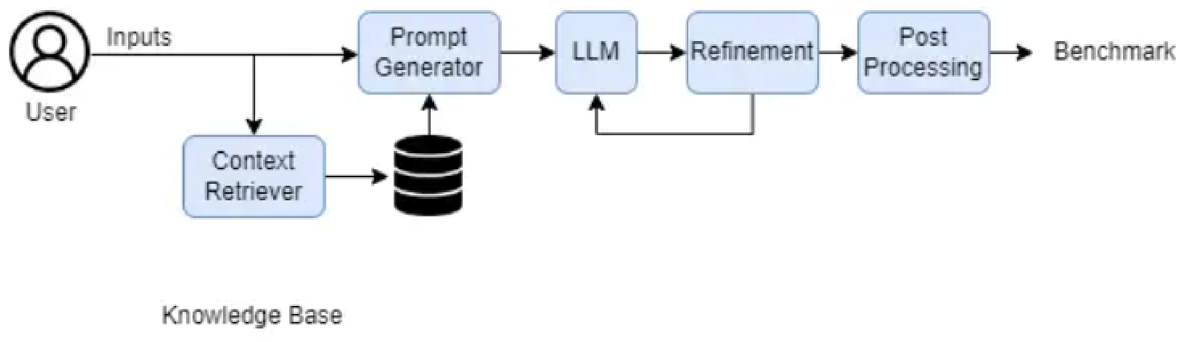


Figure 1.2

Methods

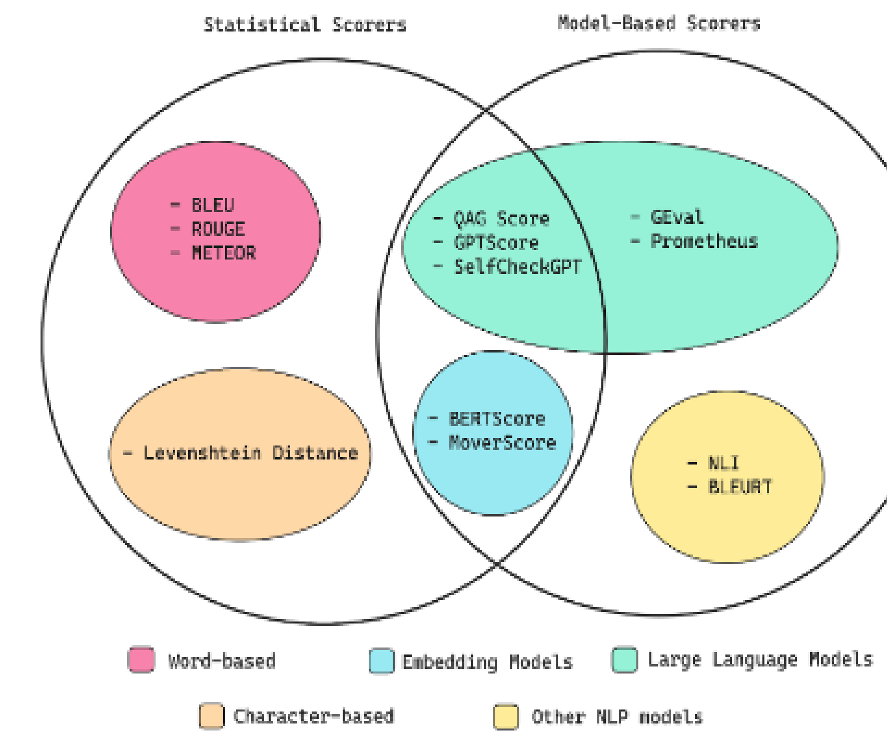


Figure 2.1

As seen in figure 2.1 , various Statistical and Model - Based scores are utilized as benchmark performance measures for Large Language Models. The use of every one would be redundant hence this project aims to explore the most efficient ones. Below outlines the 3 scores used in our Project along with the motivations behind their choosing and implementation and benefits:

1. **BLEU (Bilingual Evaluation Understudy)**
   - *Motivation*: BLEU is primarily used to evaluate the quality of machine-translated text against a set of high-quality reference translations. The core motivation behind BLEU is to measure the correspondence of the candidate translation to the references at the word or phrase level, focusing on precision—the proportion of words in the machine-generated output that appear in the reference translations.
   - *Utility*: BLEU is computationally inexpensive and correlates moderately well with human judgment, making it a standard baseline in machine translation evaluation. It's particularly useful for assessing consistency in vocabulary usage and basic grammatical structures in translated texts.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Key:
BP - brevity penalty, which penalizes short generated texts.
W_n - weight assigned to n-gram precision
P_n - precision of n-grams

2. **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**
   - *Motivation*: ROUGE is designed to evaluate text summarization and other tasks where recall is as important as precision. It measures the overlap of n-grams, word sequences, and sometimes longer units like sentences between the generated text and reference texts.
   - *Utility*: Unlike BLEU, which emphasizes precision, ROUGE captures how much of the reference content appears in the generated text, making it valuable for tasks where covering all relevant information is crucial, such as summarization.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \quad (1)$$

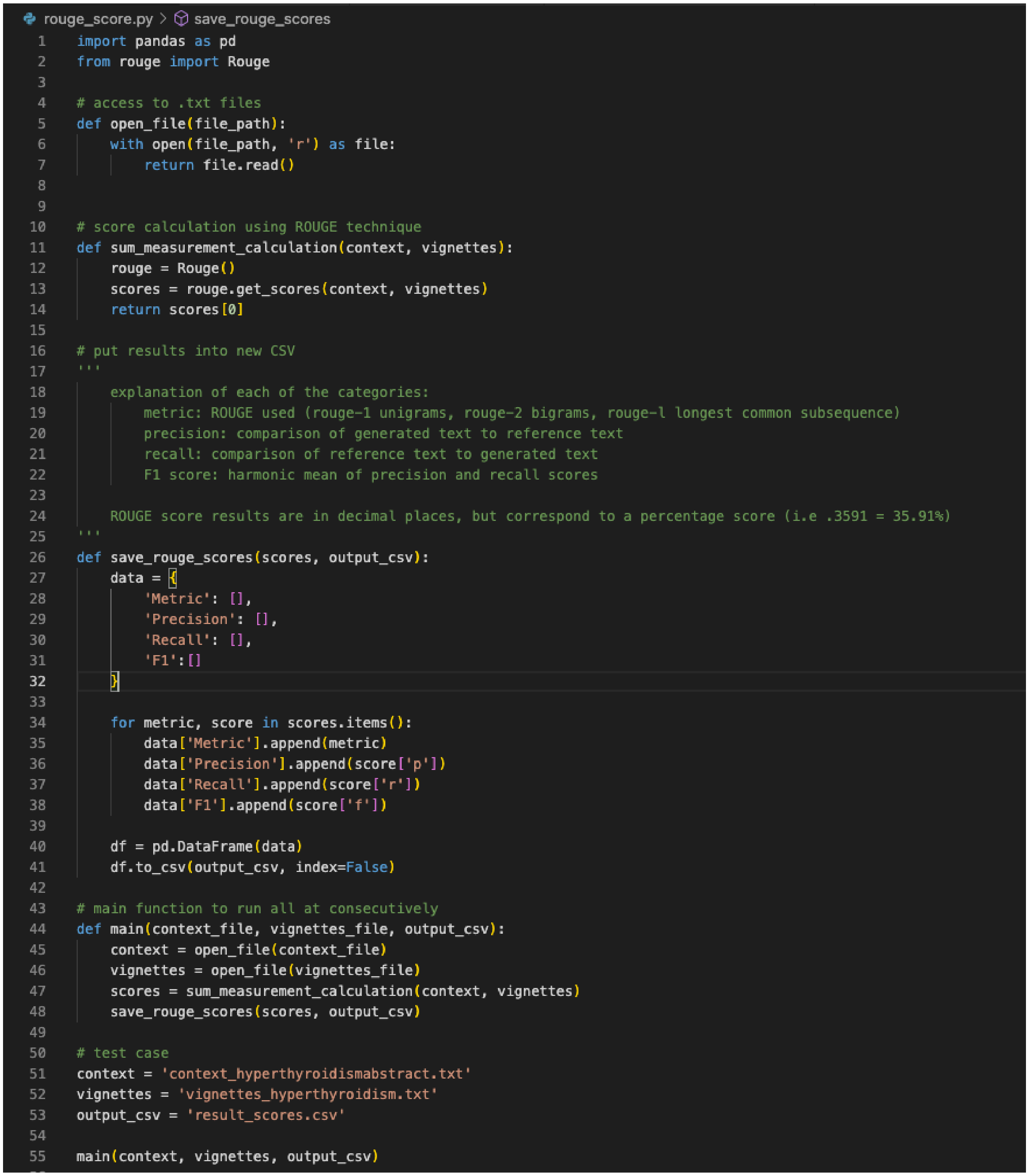
Key:

$Count_{match}(gram_n)$ - number of n-grams in the reference text that match the n-grams in the generated text.

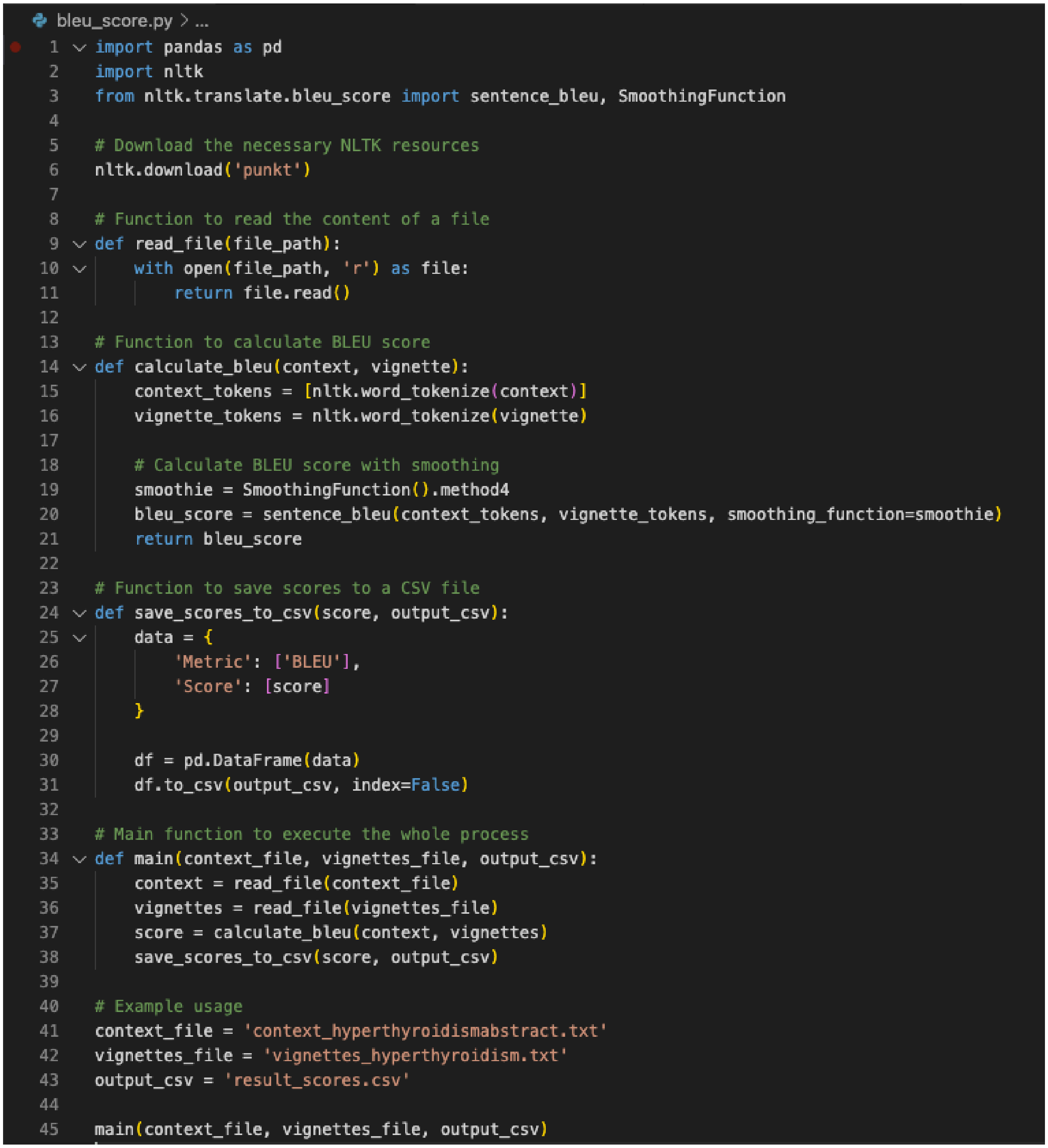
$Count(gram_n)$ - total number of n-grams in the reference text.

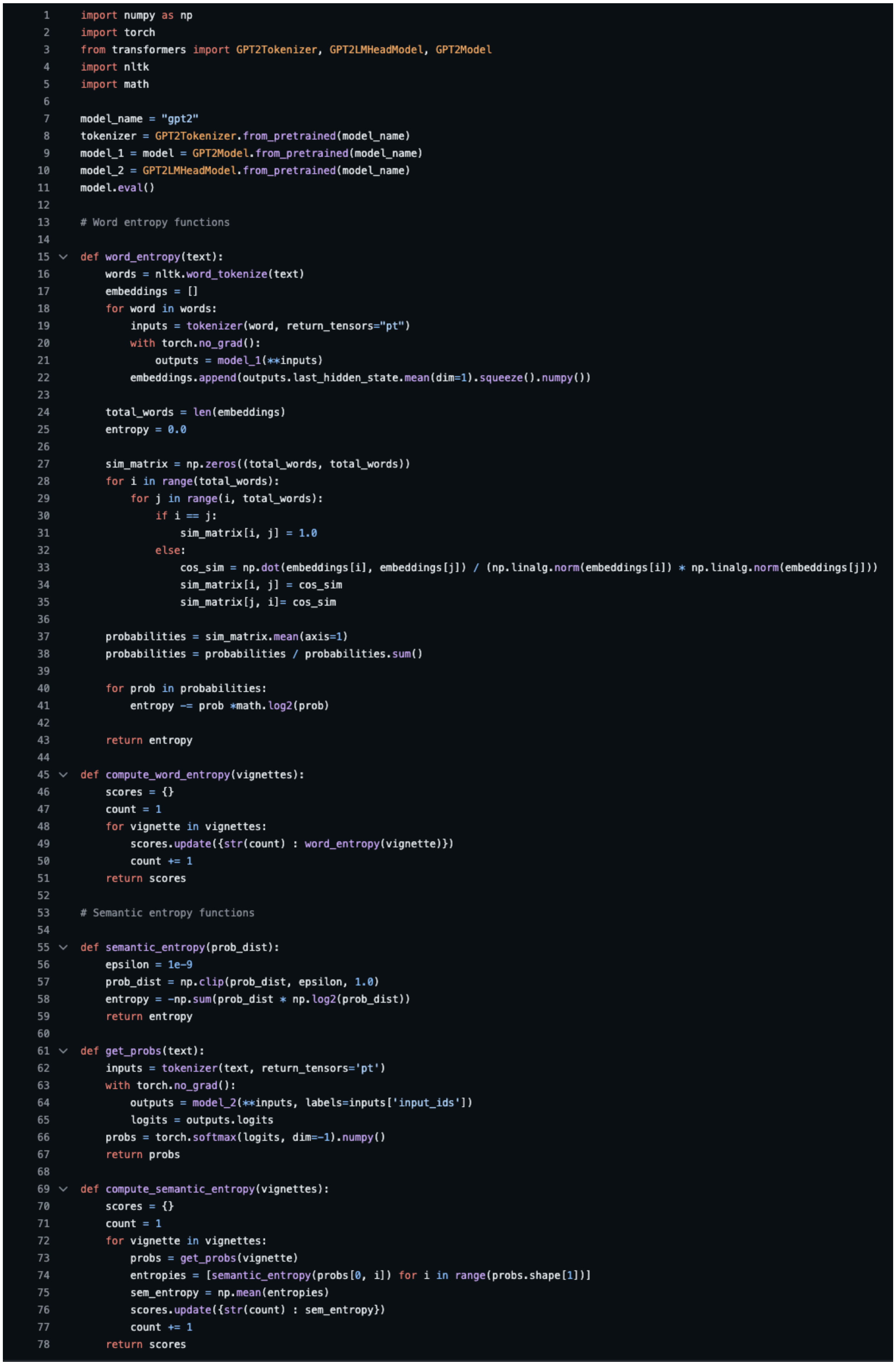
3. **Semantic Entropy**
   - *Motivation*: While not explicitly shown in your diagram under the name "Semantic Entropy," if we consider metrics like METEOR or newer embeddings-based approaches, these are often motivated by a desire to capture semantic similarity more effectively than simple n-gram matching. Metrics like METEOR, which incorporate synonyms and stemming, or embeddings-based metrics, aim to understand the underlying meanings rather than just the surface string similarity.
   - *Utility*: These metrics are useful when the precise words used are less important than the conveyed meaning, such as in dialogue systems, question answering, and more nuanced translations.

$$SE(x) = -\sum_{c} p(c \mid x) \log p(c \mid x) = -\sum_{c} \left(\left(\sum_{\mathbf{s} \in c} p(\mathbf{s} \mid x) \right) \log \left[\sum_{\mathbf{s} \in c} p(\mathbf{s} \mid x) \right] \right).$$



```python
import pandas as pd
from rouge import Rouge

# access to .txt files
def open_file(file_path):
    with open(file_path, 'r') as file:
        return file.read()


# score calculation using ROUGE technique
def sum_measurement_calculation(context, vignettes):
    rouge = Rouge()
    scores = rouge.get_scores(context, vignettes)
    return scores[0]

# put results into new CSV
'''
    explanation of each of the categories:
        metric: ROUGE used (rouge-1 unigrams, rouge-2 bigrams, rouge-l longest common subsequence)
        precision: comparison of generated text to reference text
        recall: comparison of reference text to generated text
        F1 score: harmonic mean of precision and recall scores

    ROUGE score results are in decimal places, but correspond to a percentage score (i.e .3591 = 35.91%)
'''
def save_rouge_scores(scores, output_csv):
    data = {
        'Metric': [],
        'Precision': [],
        'Recall': [],
        'F1':[]
    }

    for metric, score in scores.items():
        data['Metric'].append(metric)
        data['Precision'].append(score['p'])
        data['Recall'].append(score['r'])
        data['F1'].append(score['f'])

    df = pd.DataFrame(data)
    df.to_csv(output_csv, index=False)

# main function to run all at consecutively
def main(context_file, vignettes_file, output_csv):
    context = open_file(context_file)
    vignettes = open_file(vignettes_file)
    scores = sum_measurement_calculation(context, vignettes)
    save_rouge_scores(scores, output_csv)

# test case
context = 'context_hyperthyroidismabstract.txt'
vignettes = 'vignettes_hyperthyroidism.txt'
output_csv = 'result_scores.csv'

main(context, vignettes, output_csv)
```


bleu_score.py > ...
```python
import pandas as pd
import nltk
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction

# Download the necessary NLTK resources
nltk.download('punkt')

# Function to read the content of a file
def read_file(file_path):
    with open(file_path, 'r') as file:
        return file.read()

# Function to calculate BLEU score
def calculate_bleu(context, vignette):
    context_tokens = [nltk.word_tokenize(context)]
    vignette_tokens = nltk.word_tokenize(vignette)

    # Calculate BLEU score with smoothing
    smoothie = SmoothingFunction().method4
    bleu_score = sentence_bleu(context_tokens, vignette_tokens, smoothing_function=smoothie)
    return bleu_score

# Function to save scores to a CSV file
def save_scores_to_csv(score, output_csv):
    data = {
        'Metric': ['BLEU'],
        'Score': [score]
    }

    df = pd.DataFrame(data)
    df.to_csv(output_csv, index=False)

# Main function to execute the whole process
def main(context_file, vignettes_file, output_csv):
    context = read_file(context_file)
    vignettes = read_file(vignettes_file)
    score = calculate_bleu(context, vignettes)
    save_scores_to_csv(score, output_csv)

# Example usage
context_file = 'context_hyperthyroidismabstract.txt'
vignettes_file = 'vignettes_hyperthyroidism.txt'
output_csv = 'result_scores.csv'

main(context_file, vignettes_file, output_csv)
```

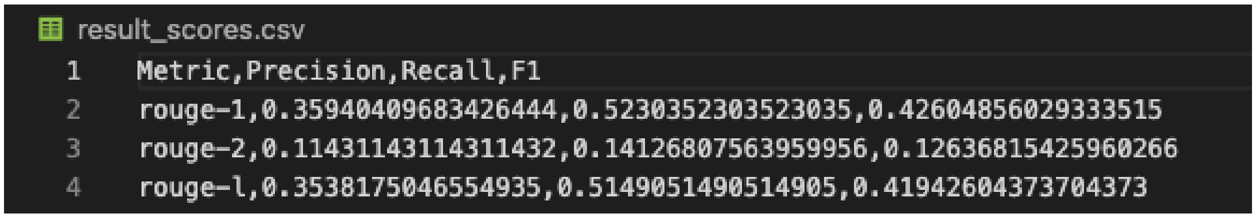


```python
import numpy as np
import torch
from transformers import GPT2Tokenizer, GPT2LMHeadModel, GPT2Model
import nltk
import math

model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model_1 = model = GPT2Model.from_pretrained(model_name)
model_2 = GPT2LMHeadModel.from_pretrained(model_name)
model.eval()

# Word entropy functions

def word_entropy(text):
    words = nltk.word_tokenize(text)
    embeddings = []
    for word in words:
        inputs = tokenizer(word, return_tensors="pt")
        with torch.no_grad():
            outputs = model_1(**inputs)
        embeddings.append(outputs.last_hidden_state.mean(dim=1).squeeze().numpy())

    total_words = len(embeddings)
    entropy = 0.0

    sim_matrix = np.zeros((total_words, total_words))
    for i in range(total_words):
        for j in range(i, total_words):
            if i == j:
                sim_matrix[i, j] = 1.0
            else:
                cos_sim = np.dot(embeddings[i], embeddings[j]) / (np.linalg.norm(embeddings[i]) * np.linalg.norm(embeddings[j]))
                sim_matrix[i, j] = cos_sim
                sim_matrix[j, i]= cos_sim

    probabilities = sim_matrix.mean(axis=1)
    probabilities = probabilities / probabilities.sum()

    for prob in probabilities:
        entropy -= prob *math.log2(prob)

    return entropy

def compute_word_entropy(vignettes):
    scores = {}
    count = 1
    for vignette in vignettes:
        scores.update({str(count) : word_entropy(vignette)})
        count += 1
    return scores

# Semantic entropy functions

def semantic_entropy(prob_dist):
    epsilon = 1e-9
    prob_dist = np.clip(prob_dist, epsilon, 1.0)
    entropy = -np.sum(prob_dist * np.log2(prob_dist))
    return entropy

def get_probs(text):
    inputs = tokenizer(text, return_tensors='pt')
    with torch.no_grad():
        outputs = model_2(**inputs, labels=inputs['input_ids'])
        logits = outputs.logits
    probs = torch.softmax(logits, dim=-1).numpy()
    return probs

def compute_semantic_entropy(vignettes):
    scores = {}
    count = 1
    for vignette in vignettes:
        probs = get_probs(vignette)
        entropies = [semantic_entropy(probs[0, i]) for i in range(probs.shape[1])]
        sem_entropy = np.mean(entropies)
        scores.update({str(count) : sem_entropy})
        count += 1
    return scores
```

Results

When running tests, an example set of patient contexts and their corresponding vignettes were used for BLEU and ROUGE, each with 50 patients with a variety of differences in symptoms and medical and physical history.

Rouge CSV Context/Output:



```
result_scores.csv
Metric,Precision,Recall,F1
rouge-1,0.35940409683426444,0.5230352303523035,0.42604856029333515
rouge-2,0.11431143114311432,0.14126807563959956,0.12636815425960266
rouge-l,0.3538175046554935,0.5149051490514905,0.41942604373704373
```

metric: ROUGE used (rouge-1 unigrams, rouge-2 bigrams, rouge-l longest common subsequence)

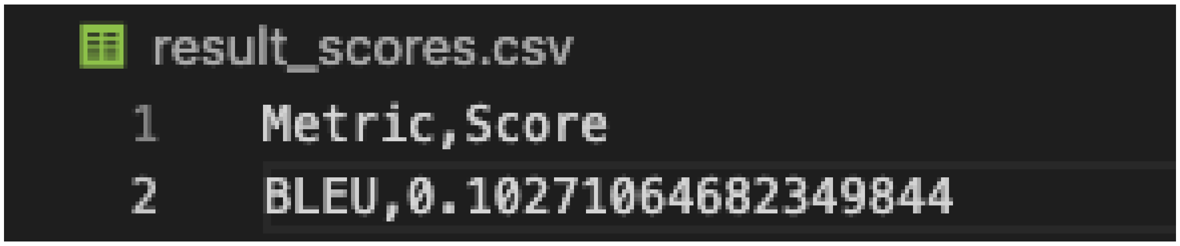
precision: comparison of generated text to reference text

recall: comparison of reference text to generated text

F1 score: harmonic mean of precision and recall scores

ROUGE score results are in decimal places, but correspond to a percentage score (i.e .3591 = 35.91%)

BLEU CSV Context/Output:

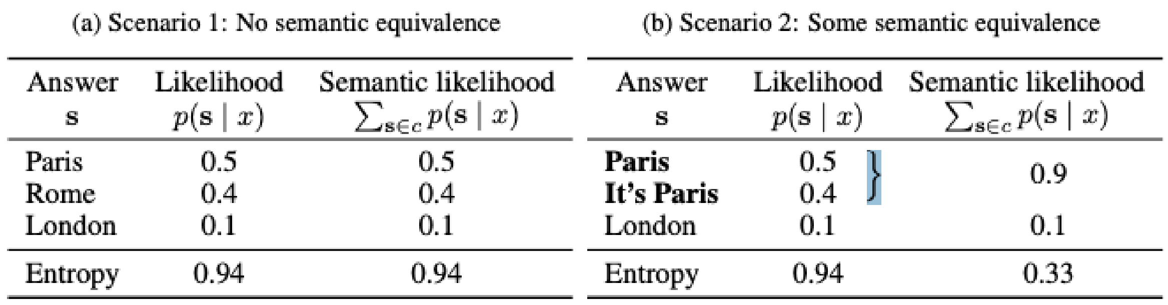


```
result_scores.csv
Metric,Score
BLEU,0.10271064682349844
```

score: ranges from 0 to 1, higher score indicates better quality of summary

Semantic Entropy Context/Example Output:

Unable to conduct tests with semantic entropy, here is an example output file:



(a) Scenario 1: No semantic equivalence

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$	Semantic likelihood $\sum_{\mathbf{s}\in c} p(\mathbf{s} \mid x)$
Paris	0.5	0.5
Rome	0.4	0.4
London	0.1	0.1
Entropy	0.94	0.94

(b) Scenario 2: Some semantic equivalence

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$	Semantic likelihood $\sum_{\mathbf{s}\in c} p(\mathbf{s} \mid x)$
Paris	0.5	0.9
It's Paris	0.4	
London	0.1	0.1
Entropy	0.94	0.33

Output: A numerical value representing the entropy of the text, typically ranging from 0 (low entropy) to higher positive values (high entropy).

Meaning: This value quantifies the level of unpredictability or diversity in the semantic content of the generated text.

Discussion

In the provided scenarios comparing "No Semantic Equivalence" and "Some Semantic Equivalence," the Semantic Entropy metric effectively quantifies the difference in semantic consistency of potential answers. In the first scenario, Semantic Entropy does not change, reflecting no consideration for semantic similarity among answers like Paris, Rome, and London. However, in the second scenario, acknowledging semantic equivalents like "It's Paris" alongside "Paris" significantly reduces entropy, demonstrating this metric's utility in measuring the nuanced understanding required in contexts like medical vignette generation, where capturing semantic depths is crucial for accuracy.

The results from the ROUGE and BLEU evaluation metrics as depicted in the uploaded images reveal insights into the quality of machine-generated medical vignettes compared to reference texts. ROUGE scores, represented in decimal form but interpreted as percentages, show a moderate level of n-gram overlap between the generated texts and the references, indicating a reasonable but not perfect capture of necessary information. For instance, the highest ROUGE-L score suggests that the longest common subsequences align fairly well, yet the overall scores suggest there is room for improvement in capturing finer details and nuances of the reference texts. On the other hand, the BLEU score, which is on a scale from 0 to 1, reflects a more critical evaluation, focusing on the precision of n-gram matching, and indicates that the exact wording and phrasing have lower correspondence. This suggests that while the gist of the medical content might be captured, the specific phrasing and terminology usage requires closer alignment with standard medical language to enhance accuracy and reliability in clinical settings.

References

We would like to thank Professor Rahmat Beheshti and his team in the University of Delaware heAIthy lAIfe Lab for their mentorship and incredible support.