

## **Call Center Co-Pilot: Enhancing Customer Service with Large Language Models**

### **ABSTRACT**

This study investigates the application of a large language model (LLM) system, Call Center Co-Pilot, to automate call interaction documentation within service centers. Manual documentation, a critical yet time-consuming process, hinders agent productivity. Co-Pilot addresses this by leveraging LLMs for sentiment analysis and transcript summarization, freeing agents to focus on customer interaction. We propose a five-step development strategy: (1) high-quality dataset preparation, (2) model selection for specific tasks, (3) comprehensive evaluation metrics, (4) rigorous testing and validation, and (5) deployment through a web application. Following extensive evaluation, DeciLM emerged as the most effective model for summarization, and Orca 2 demonstrated superior sentiment analysis capabilities. This research adds to the growing evidence that LLMs can significantly improve service operations, specifically by streamlining documentation and boosting agent productivity in call centers.

### **BODY**

#### **Section 1: Introduction**

The Call Center Co-Pilot project aims to enhance the efficiency of call center associates by leveraging a rigorously tested LLM to automate sentiment analysis and transcript summarization. The primary objective is to reduce the time needed for documenting client interactions, thereby improving overall productivity. Call centers handle a vast number of client interactions daily, requiring significant time and effort from associates to document each engagement manually. While this documentation is essential for record-keeping, understanding

customer sentiment, and maintaining service quality, it does not have to be so labor-intensive. That is where Co-Pilot comes in.

Our strategy consists of a five-step process: preparing high-quality datasets, choosing models specifically for sentiment analysis and summarization, utilizing comprehensive evaluation metrics, rigorously testing and validating models, and deploying the best-performing model through a web application. After putting various LLMs through their paces, we discovered that DeciLM excelled at summarization and Orca 2 was the champion of sentiment analysis.

## **Section 2: Related Work**

Recent advancements in natural language processing (NLP) have significantly improved sentiment analysis and summarization tasks, which are crucial for automating call center operations. Traditional methods like lexicon-based approaches and classical machine learning algorithms (SVM and Naive Bayes) often struggled with contextual understanding [1]. However, transformer-based models like BERT and GPT-3 revolutionized these tasks by leveraging large-scale pre-training and attention mechanisms to understand context and generate coherent text. By integrating advanced, transformer-based LLMs like Orca 2 and DeciLM, we aim to achieve efficient and accurate sentiment analysis and summarization, enhancing the automation and efficiency of call center operations.

## **Section 3: Dataset**

We evaluated our models on the DialogSum dataset [2] from HuggingFace. The dataset came with training, testing, and validation splits and for our evaluation, we focused on the

training split which consisted of 12,500 rows. Each row in this dataset includes a dialogue, an observed summary, and a topic corresponding to the dialogue. However, the dataset did not originally include observed sentiment for the dialogues, which we needed to evaluate our models' sentiment analysis capabilities. To address this, we created the *Actual Sentiment* column and labeled the dialogues' sentiment using the Flan-T5-XXL model, which has about 11 billion parameters. We use the sentiment analysis from this model as the baseline to evaluate our smaller models. Table 1 showcases three example rows from the dataset used in the study. One major limitation of our data is that the content of the dialogues varies greatly in terms of topic and does not accurately represent what conversations in a general call center context would look like. For example, many dialogues include more than two parties in the conversation and not many of them have the structure of one person assisting the other.

Dialogue ID	Dialogue Text	Observed Dialogue Summary	Actual Sentiment
1	<p>#Person1# (Doctor): Hi, Mr. Smith. Welcome to your annual check-up. How are you feeling?</p> <p>#Person2# (Mr. Smith): I've had trouble sleeping and my knee has been acting up. #Person1#: Let's take a look...</p>	Mr. Smith expresses some health concerns (trouble sleeping, knee pain) during his annual check-up with Dr. Hawkins...	neutral

2	<p>#Person1#: Excuse me, did you see a set of keys on a blue keychain? I think I dropped them.</p> <p>#Person2#: Oh, you mean these? I found them by the water fountain.</p> <p>#Person1#: That's them! Thank you so much!...</p>	<p>#Person1# has lost their keys and asks #Person2# if they've seen them. #Person2# found the keys and returns them to #Person1#, who is very grateful...</p>	positive
3	<p>#Person1#: I can't believe how rude the customer service rep was! They completely ignored my issue. #Person2#: That's awful! Did you try talking to a manager?</p> <p>#Person1#: Yeah, they were even worse!...</p>	<p>#Person1# is complaining to #Person2# about a negative customer service experience. They are frustrated and angry with the lack of help from both the representative and the manager...</p>	negative

Table 1: Example Dialogue dataset.

## **Section 4: Approach**

Our approach involves a five-stage pipeline:

### **1. Data Preparation**

- We sourced and refined our datasets from HuggingFace by eliminating redundancies and confirming the relevance of dialogue topics. This step ensured that the data fed into our models was high quality and relevant for subsequent

analyses. This data was stored in a dynamically updated database system (detailed in Section II).

## **2. Model Selection**

- We focused on models of up to seven billion parameters to ensure efficient use of resources and compatibility with A100 GPU Nvidia hardware. The models we selected also needed to perform text classification tasks, specifically sentiment analysis and summarization.

## **3. Evaluation Metrics**

- We selected appropriate evaluation metrics including BERTScore [3], METEOR [4], ROUGE, and accuracy scores to comprehensively assess our models' performance in both sentiment analysis and summary generation.

## **4. Model Evaluation**

- We evaluated seven open-source LLMs, including Mistral, LLaMA 3, DeciLM, Falcon, Gemma, Orca 2, and Zephyr Beta, based on key performance metrics such as BERTScore, METEOR, ROUGE, and accuracy scores (detailed in Section 4). This multi-week evaluation involved installing, testing, and selecting the most promising models based on these evaluation metrics.

## **5. Model Deployment**

- We created a web application using Heroku to deploy our model. Our application allows users to input call transcripts and output both sentiment analysis and summarization. This interface provides an accessible platform for call center associates to utilize the Co-Pilot system effectively.

## **Section 5: Software**

We utilized HPC3, UC Irvine’s high-performance computing resource, specifically an A100 GPU, to handle the computational demands of evaluating our models. The hardware resources provided by HPC3 were essential for efficiently processing large datasets and running complex large language models.

The primary programming language we used is Python, because of its versatility and support for various data processing and machine learning libraries. One of the core libraries used was *transformers* from HuggingFace, which provides state-of-the-art implementations for a range of natural language processing (NLP) models. Specifically, the *AutoModelForCausalLM* and *AutoTokenizer* classes were used which help to load the pre-trained large language models and preprocess text data into tokens that the model can understand. The *pandas* library was also used extensively for data exploration and manipulation, as well as post-inference data processing. A series of other libraries like *bert\_score*, *rouge*, and *nltk* were also used to evaluate the outputs of the models.

A PostgreSQL database hosted using Amazon AWS was used to manage and store a large volume of dialogue data. Structured Query Language (SQL) was used to construct, store, and retrieve data from the database. The *SQLAlchemy* library in Python was also used to establish a connection to the database and to execute SQL queries from within the Python environment. Lastly, Github was used as a platform for version control and collaboration. Our code was committed to a single repository, which ensured systematic project development and helped us to maintain a clean and well-documented codebase. The *dotenv* library was used during this process for environment variable management, so our database address and access tokens remain private.

## **Section 6: Model Evaluation**

Our experimentation and evaluation consisted of two major tasks: Sentiment Analysis and Summarization. We ran each model on our dataset for both tasks. This generated a high volume of data that consisted of valuable records for each model's classification and summarization tasks. However, due to the large size of our dataset, it should be noted that memory and time taken for both tasks were collected differently. This was due to the hardware constraints and time allotment we were given to perform both of these tasks using our university's computing platform.

### **Sentiment Analysis**

For sentiment analysis, we came up with three specific categories: negative, neutral, and positive. Each category represents the sentiment that a model would classify a given dialogue. By using these three categories specifically, we could gain insight into the tone and nature of how a given interaction took place. The neutral classification was added in case of the instance that a model could neither classify a dialogue negative nor positive. Thus, the neutral category acted as a reasonable solution to this problem to avoid misclassifying dialogue that might not necessarily be positive or negative.

For our evaluation process, we used two primary metrics to measure our models' performance: accuracy score and a weighted F1 score. Usage of these metrics gives us both a straightforward benchmark highlighting overall accuracy and one that gives us a more nuanced look into how each model performed across each category. This provided us with the benefit of

comparing the differences in each score across each model to further assist in our analysis for sentiment.

**Accuracy score:** is the most simplistic and easy to measure. This metric measures how many of the total predicted sentiments from a model were correct out of all its predicted sentiments. Below is the equation that highlights how this metric is calculated:

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Sentiment}}$$

**Weighted F1 score** gives us the harmonic mean of the precision and recall assessments for each model. In our context, it was incredibly valuable to use a metric that accounted for the uneven distribution of sentiment data in addition to being able to evaluate a model across each category.

$$Weighted\ F1 = 2 \times \frac{\sum_i^n (w_i \cdot Precision_i \cdot Recall_i)}{\sum_i^n (w_i \cdot Precision_i + Recall_i)}$$

$$W_i = \frac{\text{Total Actual Sentiment of category}_i}{\text{Total Actual Sentiment across all Categories}}$$

$$Precision = \frac{\text{True Positive}(TP)}{\text{True Positive}(TP) + \text{False Positive}(FP)}$$

$$Recall = \frac{\text{True Positive}(TP)}{\text{True Positive}(TP) + \text{False Negatives}(FN)}$$

The above showcases how the weighted F1 score is computed. The variable W is an adjusted value based on the current category being computed. This accounts for the difference in quantity for each category. Precision indicates how many of the predicted sentiments were correctly predicted whereas recall indicates how much of the actual sentiment was captured within the predicted sentiments.



Figure 1 highlights the distribution of predicted sentiment across each model compared to that of the actual sentiment that the dataset had labeled. The dotted line references the actual sentiment that came with the dataset whereas the bars, color encoded for each model shown in the top left, illustrate how much in each category a model predicted. This plot helps to justify why we included the weighted F1 score in conjunction with an overall accuracy score since we can see that the data isn't evenly split but that also, each model over- or under-classifies a category.

Figure 1 illustrates the performance of various models in classifying sentiment compared to the actual sentiment labels in the dataset. The dotted line represents the distribution of actual sentiments (negative, neutral, and positive), while the bars show the predicted sentiments for each model.

The inclusion of the weighted F1 score alongside the overall accuracy score is justified by the following observations from the plot:

1. **Uneven Distribution:** The actual sentiments are not evenly distributed across the three categories. This means that a model that simply predicts the majority class (positive in this case) could achieve a high accuracy without truly understanding the underlying sentiment.
2. **Over/Under-Classification:** Different models exhibit varying tendencies to over- or under-classify certain sentiments. For example, some models might frequently

misclassify neutral sentiments as positive, while others might struggle to correctly identify negative sentiments.

The weighted F1 score is particularly valuable in this context because it takes into account both precision (the accuracy of positive predictions) and recall (the ability to identify all positive instances) for each sentiment category. By weighting the F1 scores based on the class distribution, it provides a more balanced evaluation metric than accuracy alone.

In this specific scenario, the plot reveals that the Orca model demonstrates a strong ability to correctly classify positive sentiments (as indicated by the arrow), but it may also tend to over-classify neutral sentiments as positive. This highlights the importance of considering both the weighted F1 score and overall accuracy to get a comprehensive understanding of model performance.

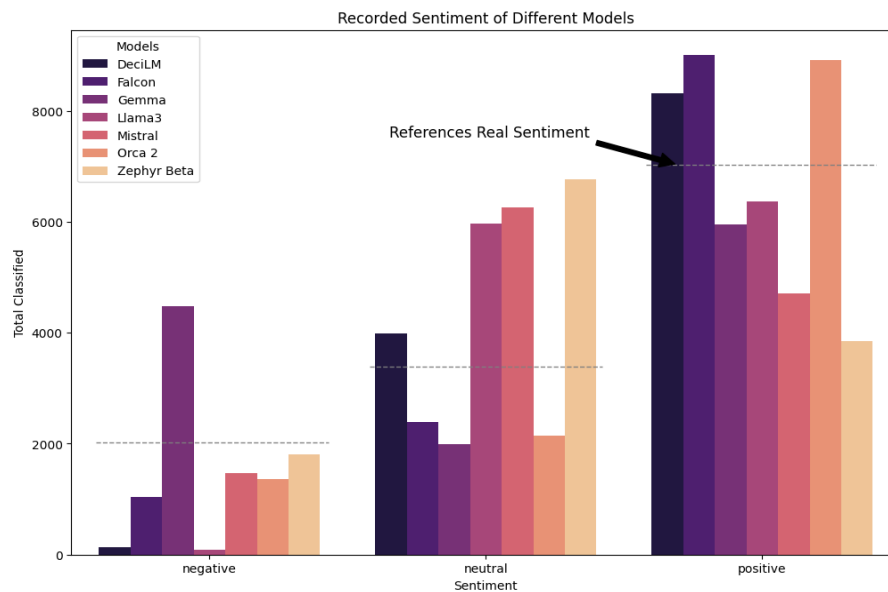


Figure 1: Recorded sentiment of different models.

Figure 2 highlights how each of these models performed in relation to their ability to correctly classify each dialogue. It should be first noted that the y-axis starts at 50 and ends at 70 and is reflected as a percentage.

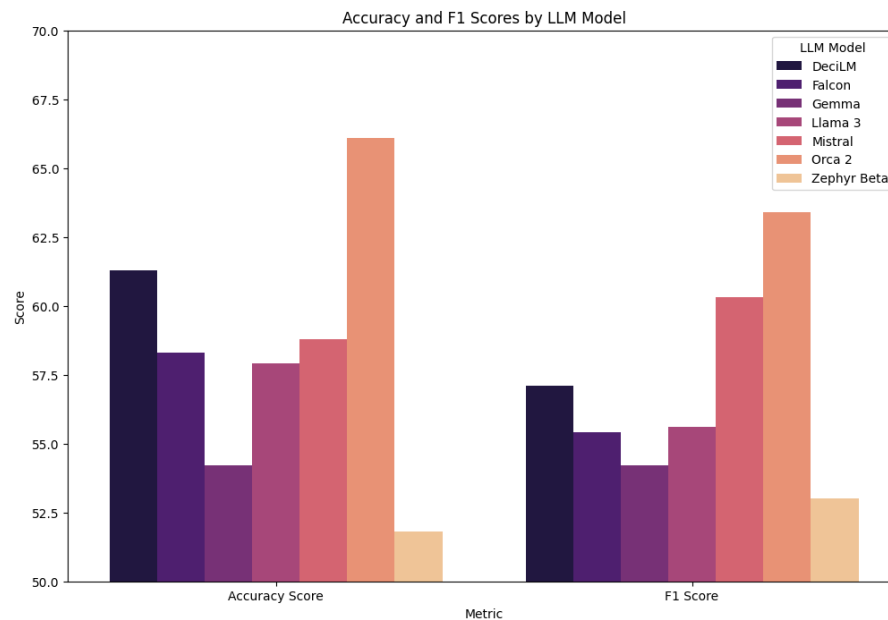


Figure 2: Sentiment analysis accuracy and F1 scores by LLM.

It is apparent the visible contrast that each model has with respect to their accuracy score versus that of the weighted F1 score. Two out of the seven models improved their score from accuracy to F1 score whereas the rest decreased. Nevertheless, the model with the overall highest accuracy score also had the highest F1 score, Orca 2. Thus, the top-performing model for sentiment had an accuracy score of 66.1% and a weighted F1 score of 63.4%. Despite this, both scores fall short of the typical 80% to 85% threshold for commercial viability. This indicates a need for further refinement and optimization of the models to meet industry standards for practical applications in sentiment analysis.

## Summarization

We have also analyzed the performance of these open-source large language models (LLMs) in the context of summarization tasks. We followed the following steps to conduct the evaluation:

1. **Dialogue Splitting:** Long dialogues from the DialogueSum were split into smaller segments to adhere to the maximum token limit of 1024 tokens. This splitting was done at the nearest new line to maintain coherence.
2. **Batch Processing:** The dialogues were processed in batches to optimize memory usage and computational efficiency. Each batch was summarized using the loaded model, and the time taken and memory usage for each batch were recorded.
3. **Summarization:** The model was prompted to generate summaries for each dialogue segment. The prompts were structured to explicitly ask for a summary, ensuring the model's output was focused and relevant.
4. **Post-Processing:** After summarization, the segments of each dialogue were combined back together. The results were aggregated and the total time and memory usage for each dialogue were calculated and stored in the database.
5. **Performance Metric Calculation:** Various performance metrics were used to evaluate the quality of the summaries, including memory usage and time taken, which provided insights into the computational efficiency of each model.

## Performance Metrics

Syntax focused:

ROUGE-N F1 Score: Recall-Oriented Understudy for Gisting Evaluation (ROUGE) that splits text up into n-gram and looks at the direct n-gram overlaps between the generated summary and the observed summary.

$$\begin{aligned} ROUGE-N_{recall} &= \frac{n\text{-gram generated} \cap n\text{-gram observed}}{|n\text{-gram observed}|} \\ ROUGE-N_{precision} &= \frac{n\text{-gram generated} \cap n\text{-gram observed}}{|n\text{-gram generated}|} \\ ROUGE-N_{F1} &= 2 \cdot \frac{recall \cdot precision}{recall + precision} \end{aligned}$$

ROUGE-L F1 Score: Calculate the longest common subsequence (LCS) between the generated and the observed summary.

$$\begin{aligned} ROUGE-L_{recall} &= \frac{LCS(\text{generated}, \text{observed})}{\# \text{ words in observed}} \\ ROUGE-L_{precision} &= \frac{LCS(\text{generated}, \text{observed})}{\# \text{ words in generated}} \\ ROUGE-L_{F1} &= 2 \cdot \frac{recall \cdot precision}{recall + precision} \end{aligned}$$

Semantic focused:

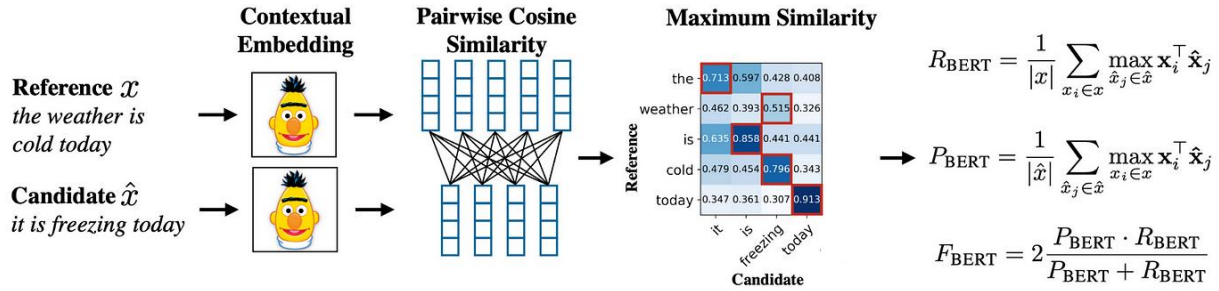
METEOR: Metric for Evaluation of Translation with Explicit Ordering: align unigrams based on matches, synonyms, and stemming.

$$\begin{aligned} METEOR_{recall} &= \frac{\# \text{ unigrams aligned}}{\# \text{ words in observed}} \\ METEOR_{precision} &= \frac{\# \text{ unigrams aligned}}{\# \text{ words in generated}} \\ METEOR_{F1} &= 2 \cdot \frac{recall \cdot precision}{recall + precision} \end{aligned}$$

BERTScore: A metric based on Bidirectional Encoder Representations from

Transformers (BERT): Generate contextual embeddings of each unigram using the BERT model and calculate their cosine similarity. See Figure 3 for a detailed illustration of how the score is computed.

# Introducing **BERTScore**



Source: Bertscore: Evaluating text generation with bert

Code for Bertscore is available at <https://github.com/Tiiiger/bert-score>

Figure 3: BERTScore calculation.

## Results

Table 2 describes the final (median) score of each LLM evaluated in this study. The low ROUGE and METEOR scores are largely because the LLMs used were not fine-tuned on dialogue datasets and the summarization task, thus the syntax-focused metrics do not give a good indication of the actual performance of the summarization task.

	Sentiment Analysis		Summarization				
	Accuracy	F1 Score	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	BERTScore

DeciLM-7B	0.613	0.571	0.345	0.115	0.316	0.303	0.886
Orca2-7B	0.661	0.634	0.312	0.093	0.282	0.326	0.884
Falcon-7B	0.583	0.554	0.264	0.075	0.237	0.278	0.880
Mistral-7B	0.587	0.603	0.267	0.082	0.245	0.325	0.878
Llama3-8B	0.579	0.556	0.287	0.095	0.264	0.341	0.878
Zephyr-7B	0.518	0.530	0.256	0.066	0.231	0.286	0.871
Gemma-7B	0.541	0.541	0.119	0.006	0.111	0.162	0.843

Table 2: Primary evaluation matrix results for sentiment analysis and summarization by model.

Therefore, our analysis of each model’s summarization performance will be focused on the BERT F1 Score. Figure 4 shows the Bert F1 Score distribution across the LLMs, with models ordered from the lowest Bert F1 Score to the highest. The plot shows that DeciLM demonstrates the best performance in terms of generating semantically accurate summaries of the dialogues without any further training, with Orca 2 and Falcon following immediately after.

To ensure these models are suitable for deployment in real-world environments with limited computational resources and require fast response time, we also analyzed the resource usage and the time each model takes to generate a summary. As shown in Figure 5, to our surprise, some of the best-performing models, such as Falcon and DeciLM, were also some of the least computationally demanding, requiring the least amount of memory usage. The results on time taken are similar to those observed in the memory usage graph: some of the best-performing models, such as Falcon and DeciLM, were also some of the fastest, taking the least amount of time to summarize the dialogues.

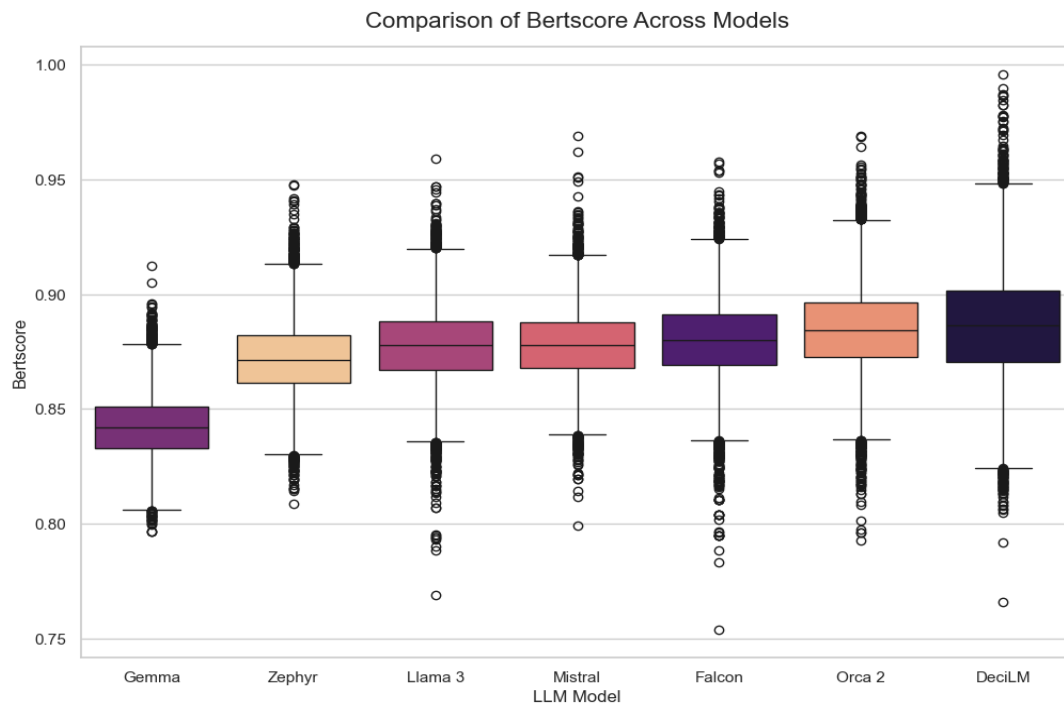


Figure 4: Comparison of Bert F1 Scores across LLMs.



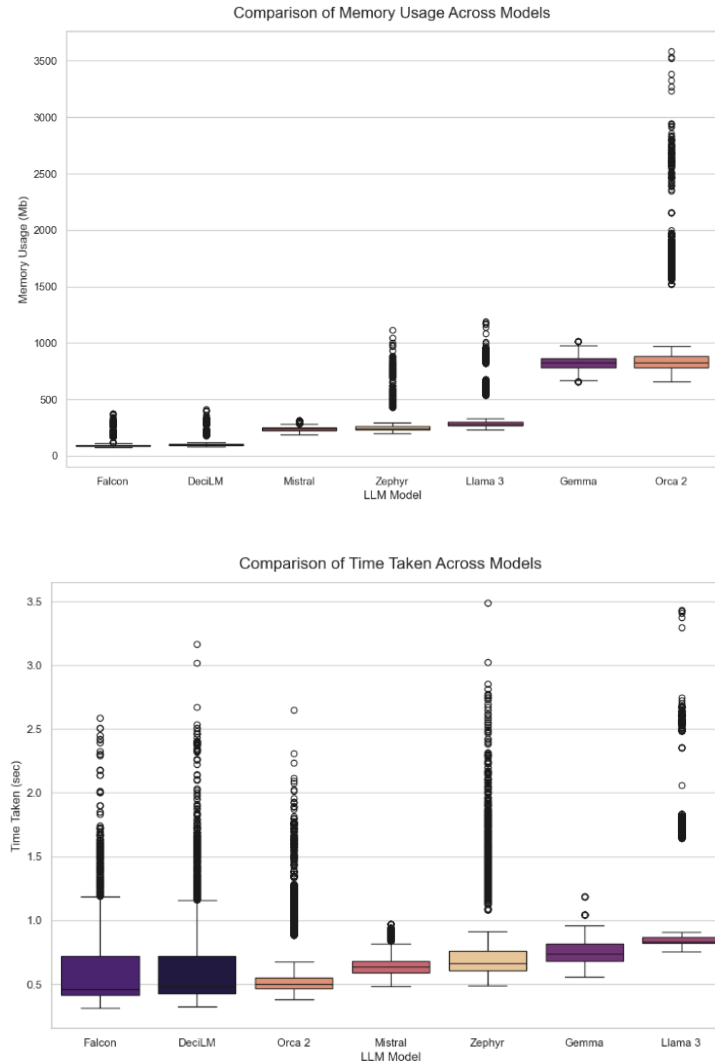


Figure 5: Comparison of memory usage and time taken to summarize per dialogue across LLMs.

## **Section 7: Notebook Description**

The sentiment-eval-models notebook provides a comprehensive analysis of sentiment scores across various models along with their corresponding visualization plots. It includes detailed assessments of models such as Orca 2 and DeciLM, illustrating the two key evaluation metrics we employed. The notebook highlights custom functions imported from our custom user modules library and documents the process of downloading data from our database server using this custom library. This notebook perfectly encapsulates the diverse aspects of our project,

demonstrating a multifaceted approach we utilized not only for sentiment analysis but for every process necessary in our project.

## **Section 8: Conclusion**

1. Through this project, we learned about the importance of selecting appropriate metrics to evaluate our models. Different metrics provide different perspectives on the model's capabilities, strengths and weaknesses. For instance:
  - **Strengths:**
    1. **BERTScore and METEOR:** Provided a deeper insight into the semantic and contextual accuracy of generated summaries.
    2. **F1 Score:** Accounts for each category and works best with uneven split of data.
  - **Weaknesses/Limitations:**
    1. **ROUGE:** Focuses on n-gram overlap and might miss the nuanced understanding, making it less effective in evaluating generated summaries, especially when the model uses synonyms or paraphrases.
    2. **Accuracy Score:** Simple and straightforward measurement but only works best when data is evenly split.
2. Setting up HPC3 with UCI and prompt engineering and string parsing were harder than expected. Both required extensive troubleshooting.
3. We learned how to use Hugging Face's transformers library for natural language processing tasks. We also gained experience in setting up and managing a relational database

service on AWS. Additionally, we learned how to leverage SQLAlchemy to connect and distribute data to our PostgreSQL server using the ORM library.

4. Given more time, we would invest in the following ideas:

- **Fine-Tuning:** Fine-tune the final model on the training split of the DialogSum dataset and reevaluate it on the testing split to improve the accuracy and relevance of generated summaries and sentiment labels.
- **Web App Design:** Enhance the call center software's user interface to be more intuitive and responsive, adding features like customizable dashboards and advanced search capabilities for better usability.

## TABLES AND FIGURES

Dialogue ID	Dialogue Text	Observed Dialogue Summary	Actual Sentiment
1	#Person1#: Hi, Mr. Smith. I'm Doctor Hawkins. ...	Mr. Smith's getting a check-up, and Doctor Haw...	neutral
2	#Person1#: Hello Mrs. Parker, how have you bee...	Mrs. Parker takes Ricky for his vaccines. Dr. P...	neutral
3	#Person1#: Excuse me, did you see a set of key...	#Person1#'s looking for a set of keys and asks...	positive

Table I: Example Dialogue dataset.

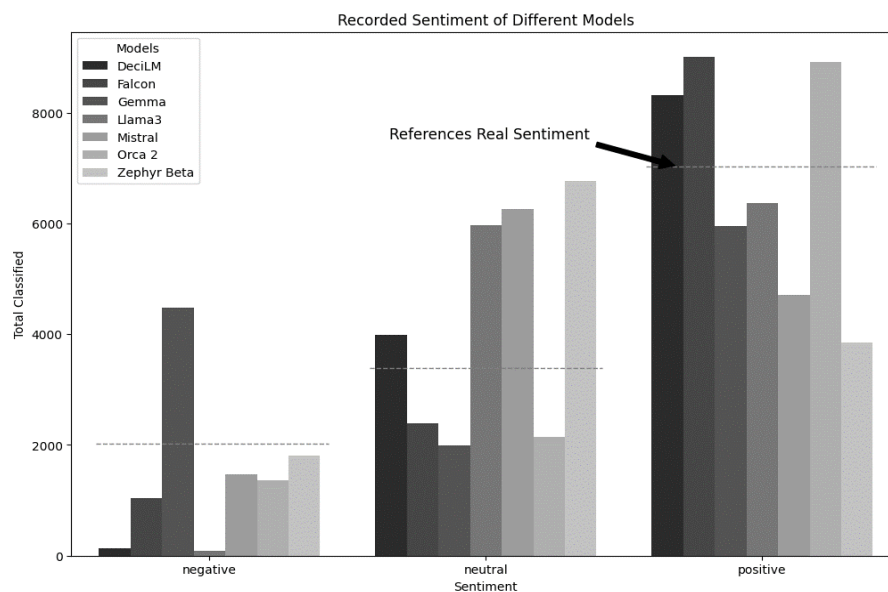


Figure I: Recorded sentiment of different models.

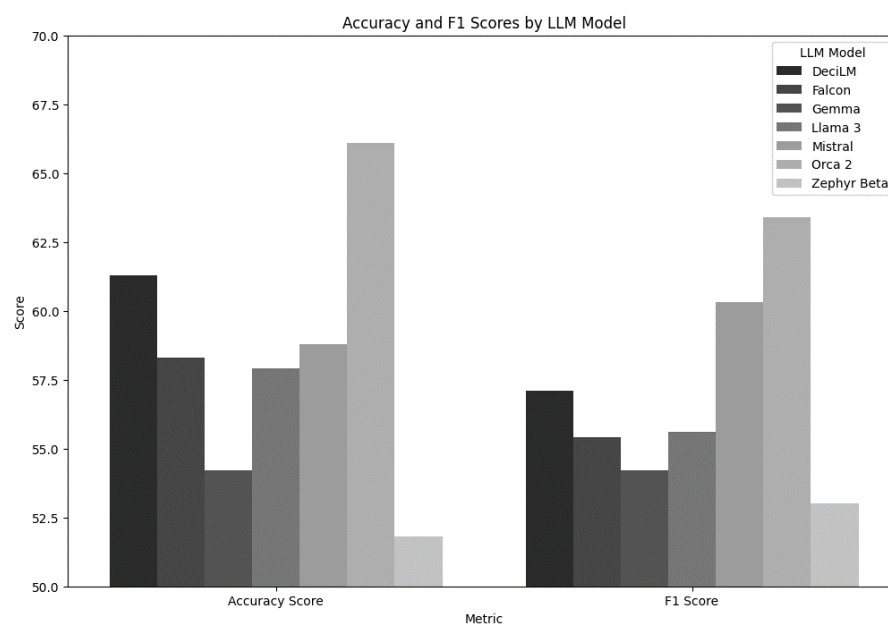
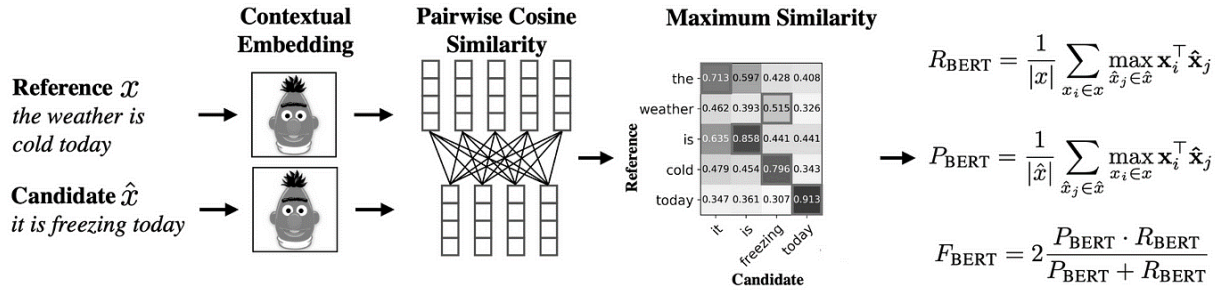


Figure II: Sentiment analysis accuracy and F1 scores by LLM.

# Introducing **BERTScore**



Source: Bertscore: Evaluating text generation with bert

Code for Bertscore is available at <https://github.com/Tiiiger/bert-score>

Figure III: BERTScore calculation.

	Sentiment Analysis		Summarization				
	Accuracy	F1 Score	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	BERTScore
DeciLM-7B	0.613	0.571	0.345	0.115	0.316	0.303	0.886
Orca2-7B	0.661	0.634	0.312	0.093	0.282	0.326	0.884
Falcon-7B	0.583	0.554	0.264	0.075	0.237	0.278	0.880

Mistral-7B	0.587	0.603	0.267	0.082	0.245	0.325	0.878
Llama3-8B	0.579	0.556	0.287	0.095	0.264	0.341	0.878
Zephyr-7B	0.518	0.530	0.256	0.066	0.231	0.286	0.871
Gemma-7B	0.541	0.541	0.119	0.006	0.111	0.162	0.843

Table II: Primary evaluation metric results for sentiment analysis and summarization by model.

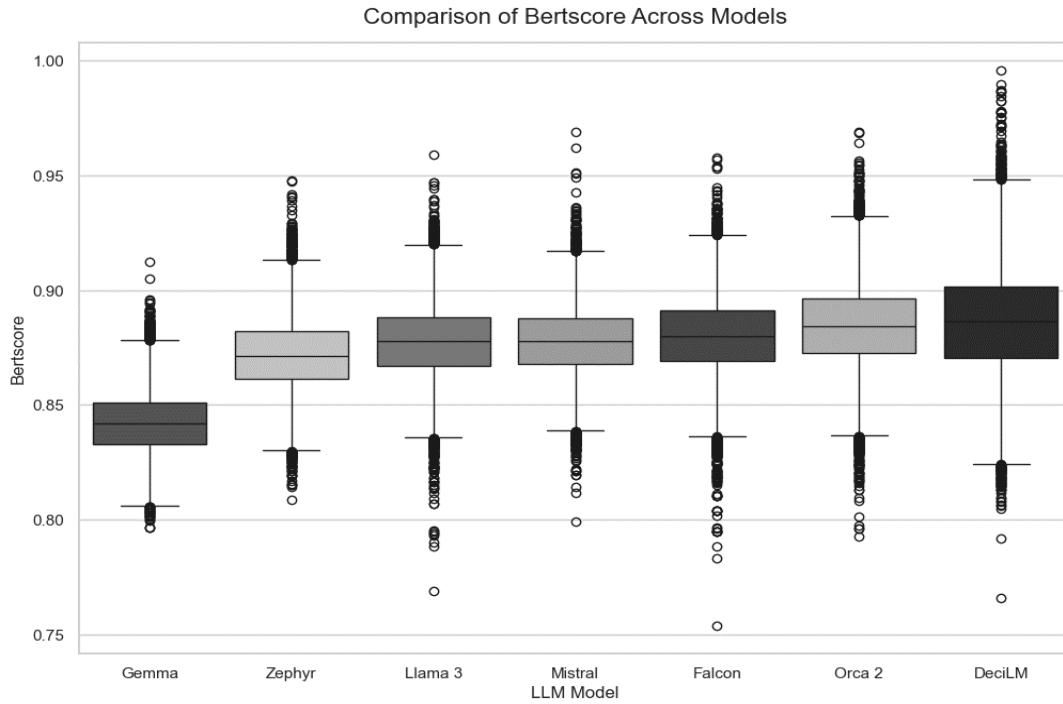


Figure IV: Comparison of Bert F1 Scores across LLMs.

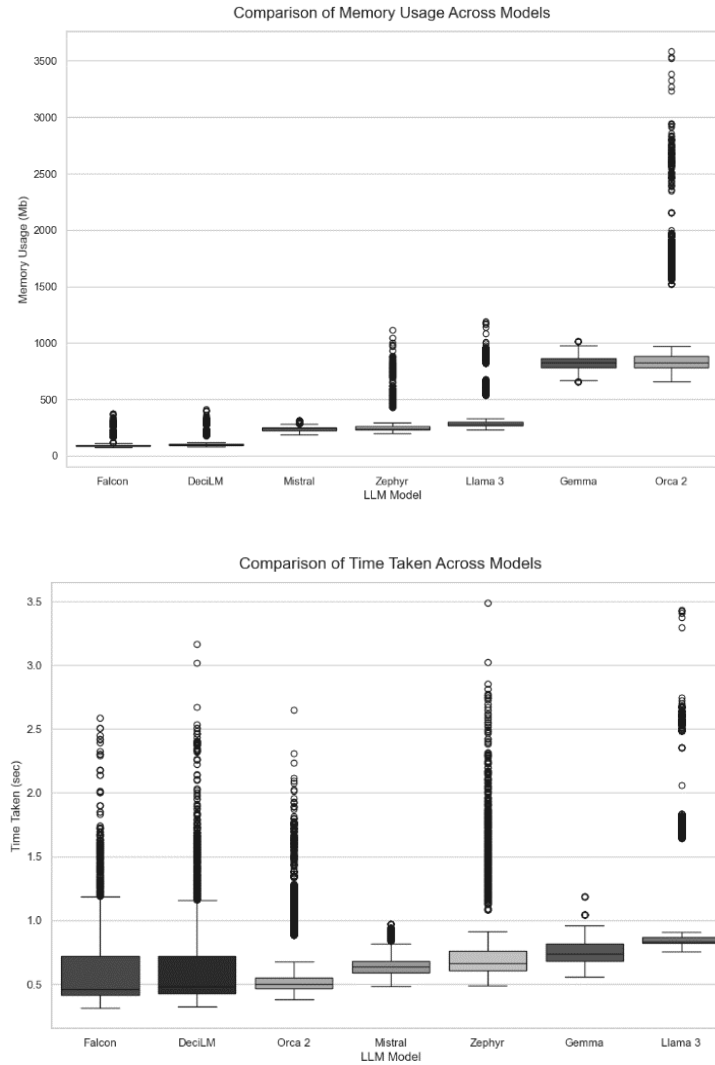


Figure V: Comparison of memory usage and time taken to summarize per dialogue across LLMs.

## REFERENCES

1. Zechner, N. (2013). The past, present and future of text classification. 2013 European Intelligence and Security Informatics Conference. <https://doi.org/10.1109/eisic.2013.61>
2. Chen, Y., Liu, Y., Chen, L., & Zhang, Y. (2021). Dialogsum: A real-life scenario dialogue summarization dataset. Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. <https://doi.org/10.18653/v1/2021.findings-acl.449>

3. Zhang, T., Kishore, V., Wu, F. F., Weinberger, K. Q., & Yoav Artzi. (2019).  
BERTScore: Evaluating Text Generation with BERT.  
<https://doi.org/10.48550/arxiv.1904.09675>
4. Lavie, A., & Denkowski, M. J. (2009). The Meteor metric for automatic evaluation of  
machine translation. *Machine Translation*, 23(2-3), 105–115.  
<https://doi.org/10.1007/s10590-009-9059-4>