Milestone 2 Report:
Thread Summarization on Code-Mixed Data

Prednya Ramesh, Ricky Raup, Akira Nair, Aneesh Edara

November 19, 2025

1 Evaluation Metric

We implement an evaluation script `score.py` that takes:

- a JSONL file of system predictions (one summary per line),
- a JSONL file of gold summaries, and
- optionally, a JSONL file with the original threads.

The script outputs a set of scalar metrics. Our *primary* metric is ROUGE-L F1; we also report BERTScore and a simple *Code-Mixing Coverage* (CMC) diagnostic.

1.1 ROUGE-L

ROUGE-L is based on the length of the longest common subsequence (LCS) between a system summary S and a reference summary R [1]. Let $L = \mathrm{LCS}(R, S)$, and let $|R|$ and $|S|$ denote the number of tokens in the reference and system summary, respectively. We define

$$P_{\mathrm{LCS}} = \frac{L}{|S|}, \qquad R_{\mathrm{LCS}} = \frac{L}{|R|}. \tag{1}$$

We then compute the F1 score

$$F_{\mathrm{LCS}} = \frac{2\,P_{\mathrm{LCS}}\,R_{\mathrm{LCS}}}{P_{\mathrm{LCS}} + R_{\mathrm{LCS}}}. \tag{2}$$

For a dataset with N examples, we report the mean ROUGE-L F1:

$$\text{ROUGE-L} = \frac{1}{N}\sum_{i=1}^{N} F_{\mathrm{LCS}}^{(i)}. \tag{3}$$

1.2 BERTScore

ROUGE only measures surface overlap. To capture semantic similarity we also report BERTScore P/R/F1 [2]. For each reference–system pair we:

1. encode tokens with a multilingual transformer (we use `bert-base-multilingual-cased`),
2. compute pairwise cosine similarities between token embeddings, and
3. compute precision, recall, and F1 by matching each token to its most similar counterpart.

We report the average BERTScore F1 over all examples.

1.3 Code-Mixing Coverage (CMC)

Standard metrics ignore language distribution. As a diagnostic, we define *Code-Mixing Coverage* (CMC), which measures how similar the language distribution of the summary is to that of the input thread.

For a text T, we approximate a language distribution $p_T(\ell)$ as follows: we run a language identifier (e.g., `langdetect`) on content words in T and normalize the counts of language labels to obtain a multinomial distribution over languages ℓ.

Given a thread x and its summary y, we define

$$\mathrm{CMC}(x, y) = 1 - \frac{1}{2} \sum_{\ell} |p_x(\ell) - p_y(\ell)| \,. \tag{4}$$

CMC lies in $[0, 1]$; higher values mean the summary's language mix more closely matches the thread. For code-mixed $\rightarrow$ English settings (e.g., CS-Sum) we expect lower CMC, reflecting intentional normalization toward English.

2 Baselines

2.1 Simple Baseline: Lead-3

Our simple baseline is an extractive *Lead-k* system implemented in `simple_baseline.py`. Each thread consists of a sequence of messages $(m_1, \ldots, m_T)$, where m_i is a string. For a fixed k (we use $k = 3$), the baseline summary is

$$\mathrm{Lead}_k(x) = m_1 \| m_2 \| \cdots \| m_k, \tag{5}$$

where $\|$ denotes string concatenation with spaces. This baseline requires no training and serves as a lower bound.

2.2 Strong Baseline: Fine-Tuned mBART

Our strong baseline is a multilingual encoder–decoder model based on `facebook/mbart-large-50-many-to-many-m` implemented in `strong_baseline.ipynb`. We linearize each thread into a single input sequence and fine-tune mBART to generate the gold summary.

We jointly train on CS-Sum, CroCoSum, and DialogSum (28,033 training examples total) for three epochs with an effective batch size of 16, a learning rate of 5×10^{-5}, and AdamW optimization. Training on multiple datasets improves robustness to different domains and degrees of code-mixing.

3 Results

Table 1 compares the simple Lead-3 baseline with our fine-tuned mBART model on the combined test sets. ROUGE-L is our primary metric; we also report BERTScore F1.[1]

The mBART model substantially outperforms the simple baseline on both metrics, indicating that it captures the semantics of threads beyond their initial turns. Qualitatively, mBART produces more fluent and focused summaries, often abstracting away small talk and preserving the main decisions or topics, even when the input is code-mixed.

[1]Numbers shown here are illustrative and will be updated with the final scores from `score.py`.

System	ROUGE-L ↑	BERTScore F1 ↑
Lead-3 baseline	0.18	0.63
mBART (strong baseline)	0.33	0.73

Table 1: Performance of our baselines on the combined test sets.

CMC analysis shows that the Lead-3 baseline tends to mirror the language mix of the thread, whereas mBART tends to normalize toward English, especially on CS-Sum and DialogSum. This behavior is consistent with our goal of producing English summaries for multilingual conversations.

4 Future Improvements

Milestone 2 focuses on establishing a clean evaluation pipeline and strong baseline. Several directions can improve summary quality and make the system more research-ready:

- **Consistent English output.** For future iterations we plan to enforce English-only summaries across *all* datasets, including those with code-mixed references (e.g., CroCoSum), by adding explicit target-language tags and language-ID-based filters during decoding.
- **Structure-aware inputs.** Currently we flatten threads into plain text. We can introduce speaker and turn markers (e.g., `<S1>`, `<S2>`, `[TURN]`) to test whether modeling thread structure improves ROUGE and human judgments.
- **Better faithfulness and error analysis.** We will conduct manual evaluation on a subset of examples to study hallucinations, dropped information, and how well the model covers non-English segments of the thread, complementing ROUGE and BERTScore.
- **Human-centric metrics.** Beyond automatic metrics, we plan to collect small-scale human ratings of fluency, faithfulness, and usefulness (*"Would this help you catch up?"*) to better align evaluation with real user needs.

Overall, Milestone 2 provides a solid baseline and evaluation setup; future work will focus on improving summary quality and control, particularly in the presence of code-mixed inputs, while standardizing all summaries into clear, concise English.

References

[1] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

[2] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*, 2020.