**FROM STRACTH TRANSLATION MODELS**

**TEAM 28**

**1. Project summary & deliverables**

**Goal:** build, train and analyze a Transformer sequence-to-sequence model (encoder–decoder) from first principles to translate between **English** and **Telugu**. Produce: dataset pipeline, clear architecture writeup, training/validation/eval procedures, plots & metrics (BLEU / sacreBLEU / chrF), comparison to at least one pre-trained baseline, error analysis, and a reproducible repo.

**2. Background theory**

**2.1 Transformer building blocks (short, precise)**

* **Token embeddings:** map token IDs → dense vectors E∈RV×dmodel\mathbf{E} \in \mathbb{R}^{V\times d\_{model}}E∈RV×dmodel​.
* **Positional encoding:** add deterministic or learned position vectors so model knows token order.
  + Sinusoidal formula commonly used in original Transformer:

PE(pos,2i)=sin⁡(pos/100002i/dmodel),PE(pos,2i+1)=cos⁡(pos/100002i/dmodel).\text{PE}\_{(pos,2i)}=\sin(pos/10000^{2i/d\_{model}}),\quad \text{PE}\_{(pos,2i+1)}=\cos(pos/10000^{2i/d\_{model}}).PE(pos,2i)​=sin(pos/100002i/dmodel​),PE(pos,2i+1)​=cos(pos/100002i/dmodel​).

* **Scaled dot-product attention:** for queries QQQ, keys KKK, values VVV,

Attention⁡(Q,K,V)=softmax⁡ ⁣(QK⊤dk)V.\operatorname{Attention}(Q,K,V) = \operatorname{softmax}\!\left(\frac{QK^\top}{\sqrt{d\_k}}\right)V.Attention(Q,K,V)=softmax(dk​​QK⊤​)V.

* **Multi-head attention:** split dmodeld\_{model}dmodel​ into hhh heads, compute attention in parallel, concat and linearly project:

MultiHead(Q,K,V)=Concat(head1,…,headh)WO.\text{MultiHead}(Q,K,V)=\text{Concat}(\text{head}\_1,\dots,\text{head}\_h)W^O.MultiHead(Q,K,V)=Concat(head1​,…,headh​)WO.

* **Position-wise feed-forward:** two linear layers with a ReLU (or GELU) in between per token.
* **Residual connections + LayerNorm** after sublayers for training stability.
* **Encoder–Decoder:** encoder stacks only self-attention + FFN; decoder has masked self-attention, encoder-decoder cross-attention, and FFN.

**2.2 Loss & training objective**

* **Cross-entropy** with teacher forcing: model predicts next token given previous ground truth tokens. Optionally use **label smoothing** (e.g., ε=0.1).
* **Masking**: create padding masks for attention; in decoder use causal mask to prevent peeking at future tokens.

**2.3 Optimization & LR schedule**

* Use **Adam / AdamW**. The original Transformer used the “Noam” schedule: warmup for NNN steps then decay ∝ 1/step1/\sqrt{\text{step}}1/step​. Also recommend gradient clipping, mixed precision (FP16) for speed.

**2.4 Evaluation metrics**

* **BLEU** and **sacreBLEU** (for reproducible BLEU).
* **chrF** for morphologically rich languages like Telugu.
* Optionally use **COMET** for learned, quality-aware evaluation (if compute/time permits).

**3. Data: sources, cleaning & alignment (English ↔ Telugu)**

**3.1 Best public parallel sources to check & use**

* **Samanantar** — very large English↔Indic parallel collection (mined + public corpora). Excellent to start; contains many English–Indic pairs. [arXiv](https://arxiv.org/pdf/2104.05596?utm_source=chatgpt.com)
* **OPUS / Tatoeba / OPUS-MT collections** — many small aligned corpora and test/dev pairs (Tatoeba is good for small, high-quality test pairs). Use Hugging Face OPUS/Tatoeba dataset variants for quick loading. [Hugging Face+1](https://huggingface.co/datasets/Helsinki-NLP/tatoeba_mt?utm_source=chatgpt.com)
* **FLORES-101** — use as an evaluation / held-out benchmark (professionally translated dev/test sentences for many languages, including Telugu). Good for reporting comparable metrics. [arXiv](https://arxiv.org/abs/2106.03193?utm_source=chatgpt.com)

(If you find domain-specific parallel corpora for education/news/行政, include them; domain matching helps performance.)

**3.2 Data pipeline steps (detailed)**

1. **Collect & index raw corpora** (keep raw copies; respect licenses).
2. **Normalize encodings:** UTF-8, remove control chars, normalize punctuation.
3. **Sentence segmentation & filtering:** use language-appropriate sentence splitters; remove extremely long/short sentences (<3 tokens, >250 tokens).
4. **Deduplicate & length ratio filter:** remove pairs with extreme length ratios (e.g., >3×).
5. **Script issues:** Telugu uses its own script; ensure transliteration steps *only if* you plan to use romanized pipelines (usually avoid for translation).
6. **Align / clean noisy mined pairs:** use lexical heuristics or fast alignment scoring (BLEU/round-trip checks) to discard low-quality pairs.
7. **Split:** train / dev / test (e.g., 98/1/1 for huge corpora; ensure dev/test are held-out and diverse). Consider reserving FLORES as an external test.
8. **Create monolingual corpora** (Telugu & English) for back-translation and LM pretraining if you plan to augment data.

**4. Model design decisions (from-scratch)**

Document each choice and *why*:

* **Architecture sizing options**
  + *Research / baseline (small):* 4 encoder / 4 decoder layers, dmodel=256d\_{model}=256dmodel​=256, dff=1024d\_{ff}=1024dff​=1024, heads=4 — trains faster for experiments.
  + *Standard:* 6 / 6, dmodel=512d\_{model}=512dmodel​=512, dff=2048d\_{ff}=2048dff​=2048, heads=8 — stronger but heavier.
  + *Large:* 12/12 etc. (requires more compute).
* **Tokenization:** use **SentencePiece** (unigram or BPE) trained jointly on concatenated English+Telugu (shared vocabulary) or separate vocabularies. Shared vocab helps transfer but be careful with scripts. Document pros/cons.
* **Embedding tying:** tie decoder input embeddings and final linear layer for parameter efficiency.
* **Regularization:** dropout in embeddings/attention/FFN, label smoothing, weight decay.
* **Batching strategy:** token-based batching (pack sentences to fixed token count per batch) is more efficient than sentence-based.
* **Low-resource tricks:** back-translation (generate synthetic English from Telugu monolingual or vice versa), fine-tuning from multilingual checkpoints (see below), transfer learning.

**5. Training loop**

Explain the full training loop conceptually — no code:

1. **Epoch loop**: shuffle training data per epoch.
2. **Batch prep**: tokenize, pad sequences, build input/target tensors and attention masks, and decoder causal mask.
3. **Forward pass**: compute logits from decoder; apply log-softmax / cross-entropy.
4. **Compute loss**: average across non-pad tokens; apply label smoothing if used.
5. **Backward pass & optimizer step**: gradient backward, gradient clipping, optimizer.step(), scheduler.step() (LR warmup + decay).
6. **Logging**: track training loss, token-per-second, learning rate, and sample translations. Log to TensorBoard / W&B.
7. **Validation**: after N steps or epochs, run inference on dev set with greedy / beam search and compute sacreBLEU/chrF.
8. **Checkpointing**: save best model by dev BLEU and also periodic saves (last N).
9. **Early stopping** if dev score does not improve for M validations.

Also include: **inference details** — beam search (width 4–8), length penalty, and handling of unknown tokens.

**6. Experiments & ablations (what to run & why)**

Design experiments to show what you learned:

1. **From-scratch small vs standard vs large** (scaling impact).
2. **Shared vs separate SentencePiece vocab** (does sharing help transfer?).
3. **Label smoothing on/off; different dropout rates.**
4. **Back-translation:** add synthetic pairs — measure gains.
5. **Pre-trained baseline vs scratch:** fine-tune a public model (Hugging Face) and compare. (See references below.)
6. **Ablation:** remove positional encoding, reduce attention heads, etc., to show effect.

**7. Baselines & recommended pre-trained models (for comparison)**

Compare your from-scratch model against one or two existing models:

* **OPUS-MT / Marian models (Helsinki-NLP)** — OPUS-MT provides many language pairs and is a common baseline for many language pairs. [GitHub](https://github.com/Helsinki-NLP/Opus-MT?utm_source=chatgpt.com)
* **AI4Bharat IndicTrans / IndicBART family** — multilingual models targeting Indic languages; strong baselines for English↔Indic transfer. [GitHub+1](https://github.com/AI4Bharat/indicTrans?utm_source=chatgpt.com)

(Load these via Hugging Face and fine-tune or evaluate out-of-the-box as baselines.)

**8. Datasets & benchmarks (quick reference)**

* **Samanantar** — large mined+public English–Indic parallel corpus. Useful to bootstrap training data. [arXiv](https://arxiv.org/pdf/2104.05596?utm_source=chatgpt.com)
* **Tatoeba / OPUS** — small, clean sentence pairs for dev/test or data augmentation. [Hugging Face+1](https://huggingface.co/datasets/Helsinki-NLP/tatoeba_mt?utm_source=chatgpt.com)
* **FLORES-101** — high-quality evaluation benchmark (use for final reporting). [arXiv](https://arxiv.org/abs/2106.03193?utm_source=chatgpt.com)

*(Cite exact dataset pages when you assemble your data and note licenses for reuse.)*

**9. Practical considerations & compute**

* **Hardware:** GPU recommended (16GB+ for moderate models). For limited GPUs: use smaller model, gradient accumulation, or mixed precision (FP16).
* **Estimate:** small transformer can train in hours on a single GPU with ~1–10M sentence pairs; larger models and corpora need multiple GPUs / days.
* **Reproducibility:** fix random seeds, log package versions, and save checkpoints & tokenizer config.

**10. Folder structure (copy this into your repo / project document)**

project-root/

├── README.md # project overview & how to reproduce results

├── data/

│ ├── raw/ # raw downloads, keep originals (do NOT modify)

│ │ ├── samanantar/

│ │ └── opus/

│ ├── processed/ # cleaned, filtered, aligned pairs

│ │ ├── en-te.train.tsv

│ │ ├── en-te.dev.tsv

│ │ └── en-te.test.tsv

│ └── tokenized/ # tokenized datasets (SPM models, vocab files)

│ ├── spm.model

│ └── en-te.train.bpe

├── src/

│ ├── data/ # data loaders, preprocessing & tokenization scripts

│ ├── model/ # transformer components, layers, attention, positional encodings

│ ├── train/ # training loop, schedulers, checkpoints

│ ├── eval/ # inference, beam search, metrics (sacreBLEU, chrF)

│ └── utils/ # logging, seed, config parsing

├── experiments/ # experiment configs / notes (yaml/json)

│ ├── exp\_small.yaml

│ └── exp\_standard.yaml

├── notebooks/ # EDA and small reproducible experiments

├── checkpoints/ # saved model checkpoints (large files ignored by git)

├── logs/ # TensorBoard / W&B logs

├── docs/ # writeups: architecture, data pipeline, results

└── scripts/

├── run\_train.sh

├── run\_eval.sh

└── prepare\_data.sh