**Project Documentation — Task Classification with Multimodal Event Logs**

**1. Problem Overview**

In task mining and business process analysis, **event logs** record user interactions, actions, and system events over time.  
Each event belongs to a **session** (a continuous block of activity) and is associated with a **step name** that identifies the stage of a task.

The **goal** is:

Given a sequence of events in a session, predict the correct step name for each event.

This is **sequence labelling** — the output is a label for every element in a sequence.

**2. Challenges**

* **Sequential nature of data**: Events are temporally ordered; predictions must consider context.
* **Multimodal features**:
  + **Event name**: Discrete tokens, often domain-specific.
  + **Categorical**: App names, file extensions, etc.
  + **Numeric**: Time deltas, path depth, etc.
  + **Text**: Free-form text descriptions.
  + **Images**: Screenshots or icons from UI.
* **Class imbalance**: Some step names appear rarely.
* **Noisy logs**: Event names can be inconsistent; similar steps may look alike.
* **High accuracy requirement**: Must often prefer *not* predicting (abstaining) over making a wrong prediction.

**3. Methodology**

The pipeline has five stages:

**Stage 1: Data Preparation**

* **Sorting** events within each session by timestamp.
* **Feature extraction**:
  + Event name tokens.
  + Derived categorical & numeric features (e.g., file extension, elapsed time).
  + Optional text features (encoded via a pretrained language model).
  + Optional image features (encoded via a pretrained vision model).
* **Handling missing values**:
  + Fill categorical NAs with "unknown".
  + Fill numeric NAs with sentinel values (-1).
* **Encoding**:
  + Event names → vocabulary indices (<PAD>=0, <UNK>=1).
  + Categoricals → OrdinalEncoder (with PAD & UNK slots).
  + Numerics → StandardScaler for normalization.
  + Text → SentenceTransformer embeddings.
  + Images → ResNet-50 embeddings.
* **Session-level grouping** so each training sample is a complete sequence.

**Stage 2: Train/Validation/Test Split**

* **Split by session IDs**, not by events, to avoid leakage.
* Default: 60% train, 20% validation, 20% test.

**Stage 3: Sequence Modelling**

Two model architectures are available:

**Option A: BiLSTM Tagger**

* **Why LSTM?**
  + Handles variable-length sequences.
  + Good at modeling temporal dependencies.
  + Bidirectional variant captures both past and future context.

**Option B: Transformer Encoder Tagger**

* **Why Transformer?**
  + Self-attention can model long-range dependencies without recurrence.
  + Often outperforms RNNs on large datasets with complex interactions.

**Stage 4: Fusion of Modalities**

All embeddings are concatenated or combined via a **gated fusion mechanism**:

1. **Event embedding** — from learned lookup table.
2. **Categorical embeddings** — one per categorical feature.
3. **Numeric projection** — a small linear layer to map normalized numerics into embedding space.
4. **Text embedding** — from pretrained SentenceTransformer, optionally projected to a smaller size.
5. **Image embedding** — from pretrained ResNet-50, optionally projected.

Fusion mode:

* **Concat**: Simple concatenation of all modality vectors.
* **Gated**: Learnable gates weight each modality’s contribution.

**Stage 5: Output & Loss**

* **Per-token linear classifier** maps sequence encoder outputs to logits for each step name.
* **CrossEntropyLoss** with ignore\_index for padded positions.
* Optional **selective evaluation**: Only consider predictions above a confidence threshold (e.g., 95%).

**4. Why These Methods Work for This Problem**

**Sequential Models**

Event logs are inherently **sequential**. The meaning of an event often depends on:

* The events before it (context).
* The events after it (future context in offline setting).

**BiLSTM**:

* Reads the sequence forward and backward.
* Captures dependencies both ways.
* Robust for smaller datasets where self-attention might overfit.

**Transformer Encoder**:

* Uses self-attention to compare every position to every other position.
* Captures long-range dependencies without decay.
* Works well if we have enough data and diverse patterns.

**Multimodal Fusion**

Different features carry different types of information:

* **Event tokens**: Identity of the action.
* **Categoricals**: Domain-specific hints (e.g., which app is open).
* **Numerics**: Timing and structural signals.
* **Text**: Descriptive cues.
* **Images**: Visual context.

By embedding all modalities into the same vector space and **fusing them**, the model:

* Gains complementary signals.
* Can disambiguate steps that are textually or visually similar but contextually different.

**Pretrained Text & Image Encoders**

Why not learn from scratch?

* Text: Domain vocabulary can be large and sparse. Pretrained SentenceTransformers already encode semantic meaning and generalize better.
* Images: Training a CNN from scratch is costly; using a ResNet pretrained on ImageNet gives strong feature extractors immediately.

**Selective Evaluation**

For business processes, **false positives** are costly.  
By only predicting when the model is ≥95% confident:

* Precision increases.
* Coverage decreases, but errors are reduced.
* Mimics a human-in-the-loop scenario where low-confidence events are sent for manual review.

**5. Model Layer-by-Layer Explanation**

Let’s take the **BiLSTM** example (Transformer is similar but with attention layers):

1. **Event Embedding Layer** (nn.Embedding):
   * Maps event token IDs to dense vectors.
   * Learns representation of event semantics.
2. **Categorical Embedding Layers** (nn.Embedding per categorical):
   * Each categorical feature gets its own embedding table.
   * Handles discrete domain features efficiently.
3. **Numeric Projection Layer** (nn.Linear):
   * Maps scaled numeric features to embedding space.
   * Allows model to learn how numerics interact with other features.
4. **Text Projection Layer** (nn.Linear):
   * Maps SentenceTransformer embeddings to a smaller vector size.
   * Reduces dimensionality and prevents text embeddings from dominating.
5. **Image Projection Layer** (nn.Linear):
   * Maps ResNet-50 features to embedding space.
   * Aligns modality dimensions for fusion.
6. **Fusion Layer**:
   * **Concat**: [event; cats; nums; text; img].
   * **Gated**: Learnable gates adjust contribution per modality.
7. **BiLSTM Encoder** (nn.LSTM bidirectional):
   * Processes the fused sequence representation.
   * Hidden size doubled due to bidirectionality.
   * Captures temporal dependencies.
8. **Dropout Layer**:
   * Reduces overfitting by randomly zeroing features during training.
9. **Classification Layer** (nn.Linear):
   * Maps encoder output at each timestep to logits over step name classes.

**6. Why It Works**

* **Sequence encoders** capture temporal context crucial for event interpretation.
* **Multimodal embeddings** enrich the representation with complementary signals.
* **Pretrained encoders** boost generalization for text/image modalities.
* **Selective prediction** allows balancing accuracy vs. coverage for business-critical use cases.
* The architecture is **modular**: You can turn on/off modalities depending on available data.