Let's create a system that supports both transcription and editing-by-voice, allowing users to flexibly interleave dictation and editing. How does this system work?

**Task Overview**

The system consists of a series of modules: segmentation, ASR repair, normalization, and execution. The process begins with segmenting the input speech into dictation and command segments. Then, ASR repair steps are applied to fix ASR errors. The normalized segments are then used as input to the execution engine, which generates the final output.

**Dataset: TERTiUS**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Dialogues</th>
<th>Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicate doc</td>
<td>Exactly recreate an email</td>
<td>372</td>
<td>1495 192</td>
</tr>
<tr>
<td>Elaborate doc</td>
<td>Expand a terse description of an email to a full email</td>
<td>343</td>
<td>347 929</td>
</tr>
<tr>
<td>Replicate segment</td>
<td>Exactly recreate the effect of a single command segment sampled from demonstrations of the previous two objectives</td>
<td>605</td>
<td>139 1299 1438</td>
</tr>
</tbody>
</table>

**Table:** Number of ways to invoke various commands, in terms of number of distinct first tokens used to invoke that command. Comparing TERTiUS to prior systems (Dragon NaturallySpeaking).

**Models**

- $M_{SEG}$: T5 encoder trained to perform BIOES tagging to identify command boundaries.
- $M_{NOR}$: T5 encoder-decoder model trained to map noisy ASR segments into normalized ASR segments (repairing ASR/speech errors).
- $M_{INT}$: Maps normalized command ASR segments into either: (1) prog programs which get executed by an execution engine into the end-state; or (2) state: the end-state directly.

**Experiments**

- **Metric**
  - F1
  - 90.9%
  - 85.3%
  - 41.9%
  - 65.9%

- **Runtime**
  - 1.28
  - 4.36
  - 5.32
  - 6.92

We evaluate segmentation (top) and the ASR repair and interpretation components jointly (bottom), reporting accuracy metrics (F1, EM) as well as runtime (in seconds per example). For ASR repair and interpretation, we experiment with a fine-tuned T5 vs. a prompted GPT3 model, each outputting either the end state (state) or a program to carry out the command (prog).