III. Model documentation and write-up

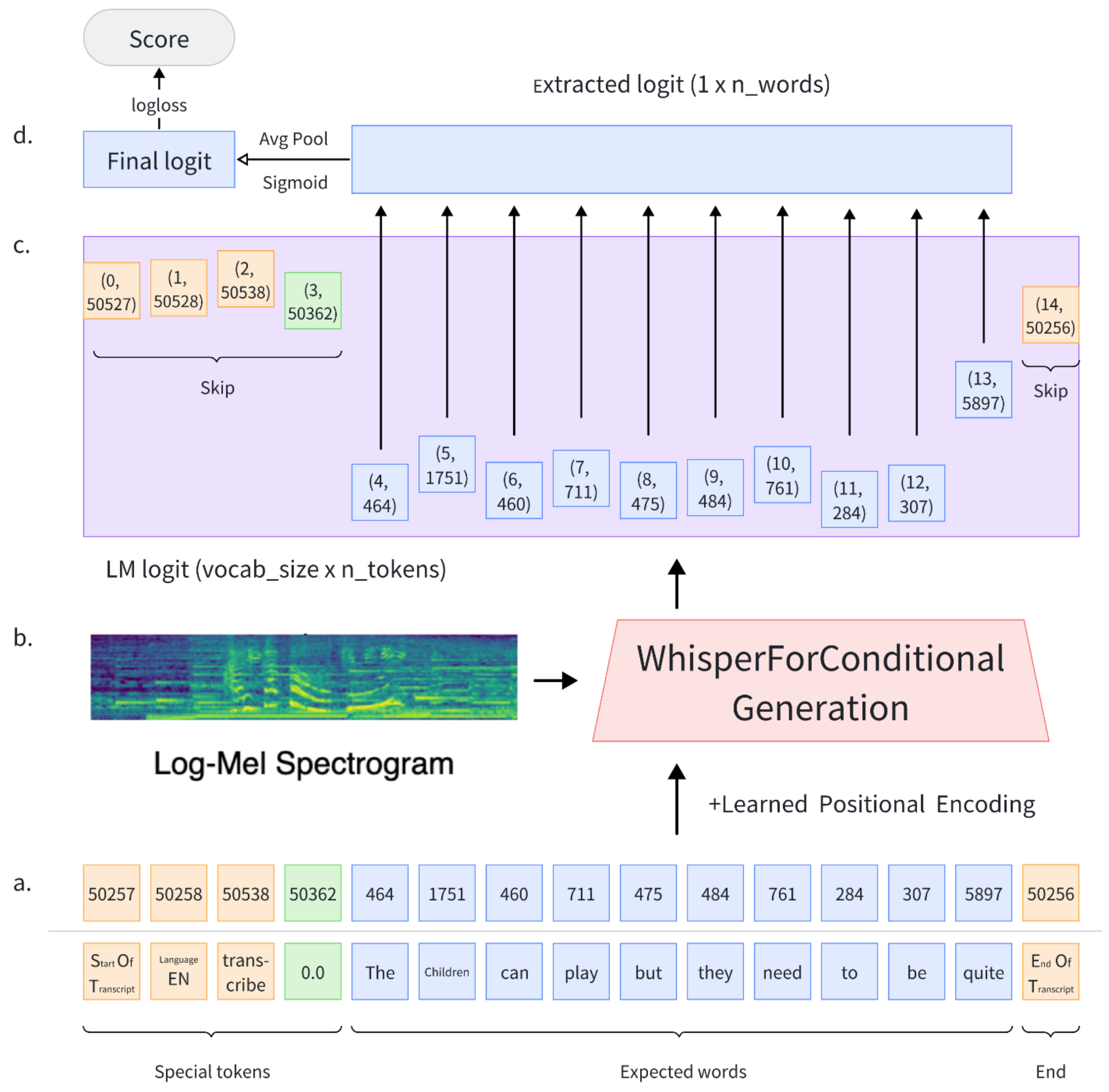
Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.
   I am an experienced data engineer and work in bioinformatic and computer vision field. I join many data science competition at both DrivenData and Kaggle, with a Kaggle GrandMaster tie.

2. What motivated you to compete in this challenge?
   First I am interested in both audio and multi-model problem, the emergence of openai's whisper provide a good opportunity to solve this kind of problem. Then the dataset of this challenge is will annotated and clean, only few has wrong label. This make this challenge a good benchmark for audio-text matching.

3. High level summary of your approach: what did you do and why?
   We fine-tune OpenAI's Whisper model with a custom loss function. For each word in `expected_text`, we input its token into the decoder and compute the binary cross-entropy (BCE) loss between the token's logit and its corresponding score. This approach enables us to evaluate the correctness of each word in a weakly supervised manner.

4. Do you have any useful charts, graphs, or visualizations from the process?
   Yes, the following page has the figure of full workflow and the loss function design.



5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   The most impactful part is the loss function design, the code is listed below (WhisperForConditionalGenerationMask.forward):

```
        loss_fct = BCEWithLogitsLoss(reduction="none")
        labels = labels.to(lm_logits.device)
        # extract the tokens (ignore first 4 special tokens)
        lm_logits = torch.gather(lm_logits, dim=2,
index=decoder_input_ids.unsqueeze(-1)).squeeze(-1)[:, 4:]
        mask = decoder_attention_mask[:, 4:]
        # only calculate those with words
        lm_logits = lm_logits.masked_fill(mask == 0, float('nan'))
        lm_logits = torch.nanmean(lm_logits, dim=1)
        loss = loss_fct(lm_logits, labels.float().reshape(-1)).mean()
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): AMD ryzen 3960x
   - GPU (model or N/A): RTX A6000 ada 48G
   - Memory (GB): 128GB
   - OS: Ubuntu.
   - Train duration:~12h.
   - Inference duration: ~4h (online submission)

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?
   You may take care of nonword performance, since this part is not pretrain by whisper, this part's accuracy may be decayed.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
   Nearly no, I only use potplayer to do error analysis.

9. How did you evaluate performance of the model other than the provided metric, if at all?
   I only use log loss, since this is the both metric and loss function, no other metric is used.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?
    I tried many methods, and listed below:
    - CLAP: nearly useless
    - Two tower model, CNN or transformers to extract audio feature; bert to extract text feature, this best CV is about 0.36
    - Hubert with CTC loss, very hard to make CTC loss penalty binary target

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?
    I may try to pretrain whisper with common voice and this dataset, I don't think any feature can help, the xgboost baseline only score 0.6 and our method score 0.2, this is a huge gap for logloss, I also tried groupkfold by expected_text, our model also perform very good and hadn't seen any decay on non-seen expected_text.

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?
    The inference speed is fast enough even for the large model. If we need more speed, the whisper small model is faster with abount 0.02 log loss decay.