Proposals for experimental projects

Graz, Nov. and Dec. 2025

Intro experiments

| intro | Experiments+ presentations | report |
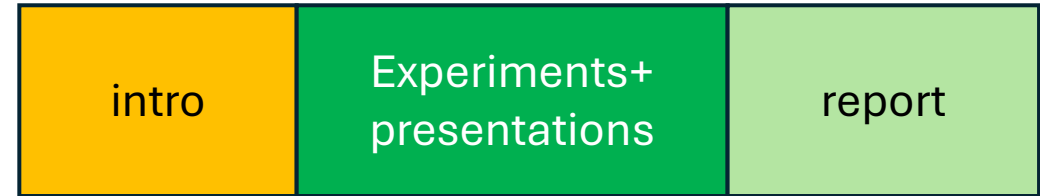
- Course set-up
  - Intro+background
  - Experiments (1 or 2 persons), brief presentations
  - Short report (1 person)
  - Assessment based on active participation and report

- ASR is way too broad to be addressed in just a few weeks
  - You might choose topic yourself or from precooked list
  - Keep the research question doable
  - Keep it feasible

Python scripts available for precooked options

- In github repo ltenbosch2/Graz_github

Three (four) proposals

- Bridging phonetics and ASR/deep learning
  - Locating phonetic structure in latent representations
- Prompting Whisper
  - ASR is context dependent
- Behavior of transformer based networks on small (user-build) data sets
  - E.g., boolean mappings
  - Regular expressions: https://aclanthology.org/2020.emnlp-main.576.pdf
- Reasoning
  - How to validate the outcome of reasoning models?

List of experimental options

1. ==Bridging phonetics and ASR/deep learning==
    - Start by using pretrained wav2vec2.0 via Huggingface
    - Script (github): ==try_Wav2Vec2_latent_logit.py==
        - Choose dataset
        - Process mono wave file (16 kHz)
        - Extract hidden layer k, and logit tensor
        - Select two vowels (for example) that minimally differ acoustically, e.g.
            - **Bahn** [baːn] – "track, railway" – **Bonn** [bɔn] – (city name)
            - **Beet** [beːt] – "garden bed" – **Bett** [bɛt] – "bed"
            - **fühlen** [ˈfyːlən] – "to feel" – **füllen** [ˈfʏlən] – "to fill"
            - **Ofen** [ˈoːfn] – "oven" – **offen** [ˈɔfn] – "open"
    - Use logit matrix to locate the corresponding time stamps
    - Use these time stamps to build two distributions of vectors
    - Try to measure the difference (as function of k) by LDA or similar method

List of experimental options (cont'd)

2. Whisper prompting
   - Present a wave file, keep this file fixed and investigate what effect text prompting has on Whisper output
   - Scripts (github):
   - try_Whisper_from_OpenAI.py # OpenAI API KEY required!!
   - OR: try_Whisper_from_HuggingFace_multiple_languages.py # this needs some further engineering to get it working

   - Follow the recipe in following link
   https://cookbook.openai.com/examples/whisper_prompting_guide
   - Helpful to prompt Whisper to obtain a certain transcription style

   - Keep experiments doable and transparent

List of experimental options (cont'd)

3. Behavior of transformer-based networks on small (user-build) data sets
- Small data sets (training, validation, test sets)
- Minimal functional transformer models

Various proposals – take them as starting point:
- **Scripts in .txt file (github):** Graz_regular_expressions_training_transformers.txt
  - Data are created in Graz_regular_expressions_create_training_test_sets.txt
  - MA_002 (data also in github) A*B versus B*A, in random **real** vector sequence
  - MA_003, ABC versus ACB in random **integer** vector sequence (dim = 5)
  - MA_003b, same as MA_003 but with dim = 3
  - MA_004, random integer sequence → Boolean (whether the last vector is 'between' first two vectors, or not)
- **Graz_various_transformer_studies.txt**
  - 4 specific cases (from vector sequence to vector, to Boolean, character sequences)

List of experimental options (cont'd)

4. Use of a Large Reasoning Model (LRM)

See (github): Reasoning_deepseek_simple.doc

- Select an LRM (minimum DeepSeek-v1, install ollama)
- Select say 10 (simple to hard) questions directly related to this course (speech recognition, AI, transformers)
- Let the LRM provide a reasoning for all questions
- Assess these answers in detail
  - Define your assessment method – based on knowledge/literature
  - Provide a conclusion about the quality of the LRM chosen
- Defend your choices in model selection, method and data selection

Reasoning: example (depends on # free GB)

- install ollama
- Run any of your preferred models, e.g. deepseek (many versions)
- ollama run deepseek-coder-v2

Q: Do you see how the towers of Hanoi problem can be solved for all N > 1 by using recursion?

A: "Thinking... Alright, so I'm trying to understand how to solve the Tower of Hanoi problem recursively for any number of disks greater than one. From what I remember, the classic Tower of Hanoi has three rods and three disks: two large disks and a small one. The goal is to move all the disks from the starting rod to another rod while following these rules:... [etc.]"

Short report

- Report of 3 pages proceedings format (2-column format)
  - Approx 2000 words
  - Intro + RQ
  - Method
  - Experimental data + set-up
  - Results
  - References

Computational platforms

Up to you:

Own laptop
- Windows, Linux

Cluster

Jupyter

SageMath

LaTeX

Cocalc:
https://cocalc.com/

For those interested in more involving experiments

- There are (too) many options!
- E.g. Compare wav2vec2.0 without lexicon versus with lexicon and LM
  - LM of small order (e.g., 3), via KenLM toolkit https://kheafield.com/code/kenlm/
  - finetuning XLSR model (Facebook, 2022) by fairseq toolkit (Ott et al., 2019)
  - Fairseq is a toolkit. With Fairseq you can train models that achieve state-of-the-art performance on machine translation and summarization tasks. It includes pre-trained models.