Assignment III: Unsupervised part-of-speech tagging

Objective

- In this assignment, you will explore unsupervised methods for discovering word categories directly from raw text. Your task is to build and compare several models for part-of-speech (POS) induction.
- You will implement and evaluate multiple models, each representing a different approach to clustering words into syntactic categories without labeled supervision.

Data

- The data is from Penn TreeBank.
- Training and evaluate your model using word sequences in `ptb-train.conllu`.
- Report results separately for two tag sets:
  - Coarse-grained POS tags (column 4)
  - Fine-grained POS tags (column 5)

Models to implement

Model	Description	Part II	ACS
Model 1	HMM trained with EM variants	✓	✓
Model 2	Neural HMM (Tran et al. 2016)	✓	✓
Model 3	Neural HMM (Chiu and Rush. 2020)	–	✓
Model 4	K-means over BERT embeddings	✓	–

Table: Models required for each student group.

- Part-II students implement Models 1 (40%), 2 (45%) and 4 (15%).
- ACS Part-III/MPhil students implement Models 1 (40%), 2 (45%) and 3 (15%).

Ben Zhang from last year kindly shares his work. The skeleton code is based on his implementation.

Model 1: HMM with EM Training

Implement an HMM tagger and experiment with three EM training variants:

- Hard EM
- Online EM
- Standard EM

The skeleton code provides the full pipeline. You must complete the marked sections, including:

- Viterbi decoding for HMM:
  - `utils.py` (line 62)
  - `hmm.py` (lines 194 and 210)
- Training functions
  - Standard EM: `train_EM_log` (line 148)
  - Hard EM: `train_EM_hard_log` (line 161)
  - Online/Streaming EM: `train_sEM` (line 173)

For background on these EM variants, refer to Liang & Klein (2009): Online EM. `https://aclanthology.org/N09-1069/`

Model 2: NHMM (Tran et al., 2016)

- Implement the base Neural HMM model as described in Section 4 of Tran et al. (2016): Unsupervised Neural Hidden Markov Models.
- Do not implement any Conv or LSTM layers.
- `https://arxiv.org/pdf/1609.09007`.

Model 3: NHMM (Chiu and Rush, 2020)

- Implement and the basic neural HMM model from Chiu and Rush (2020): scaling neural hidden markov models.
- The paper introduces

  > *three techniques: a modeling constraint that allows us to use a large number of hidden states while maintaining efficient exact inference, a neural parameterization that improves generalization while remaining faithful to the probabilistic structure of the HMM, and a variant of dropout that both improves accuracy and halves the computational overhead during training.*

  You are only asked to apply the neural HMM model to a small number of hidden states, so you don't need to implement the first technique (block).
- Their code is available at `https://github.com/harvardnlp/hmm-lm`, but you must implement your own version with the given skeleton code.
- `https://aclanthology.org/2020.emnlp-main.103/`

Model 4

- Conduct K-means clustering on word embeddings of the Penn TreeBank sentences.
- Use BERT-base-uncased (`https://huggingface.co/google-bert/bert-base-uncased`) to generate word embeddings.

Submission requirements

Submit the following:

- Code + README
  - Include clear instructions for installation and execution.
- Report (max 2000 words)
  - Describe your implementation approach for each of the required models.
  - Present your experimental results in a clear and organized manner (e.g., using tables).
  - Provide a thorough analysis of your results, including a comparative discussion of the models' performance, strengths, and weaknesses.
  - Ensure that all reported results can be precisely reproduced by running your submission's execution script.
  - Align your reported results with the experiments runnable from your code.
  - Ben Zhang's report is included as a reference.
- Execution Environment & Scripts
  - Provide a virtual environment specification.
  - Include a shell script that runs all experiments reported in your paper.

Important: We will randomly execute $\sim$30% of submissions.

If the reported results cannot be reproduced, the case will be forwarded to examiners.

Marking scheme

Your final grade will be determined based on the following criteria.

Implementation Correctness (70%)

- The majority of the grade is allocated to the correct and functional implementation of the required models.

Written Report and Analysis (30%)

- Clarity, Structure, and Reproducibility (10%):
  - The report is well-structured, clearly written, and adheres to the word limit.
  - The readme and scripts are clear and allow for the exact reproduction of all reported results.
- Presentation of Results (10%):
  - Experimental results for all models are presented clearly and professionally, using tables where appropriate.
  - Results for both coarse-grained and fine-grained tag sets are required.
- Comparative Analysis and Discussion (10%):
  - A comparative discussion of the models' performance, highlighting their theoretical strengths and weaknesses.