Assignment 3

email: damiano.ficara@usi.ch

Deadline: 05 Dec 2024 - 11.59pm

Language models with LSTM

Data (35 points)

1. The code from lines 390 to 392 implements the data access and download functionality:
    - `datasets` is a Hugging Face package designed for efficient data loading and preprocessing of public datasets, particularly optimized for Audio, Computer Vision, and Natural Language Processing (NLP) tasks.
    - The data loads into a `DatasetDict` object, organizing content into training and testing splits, with each split containing dataset objects structured similarly to pandas DataFrames.
    - The dataset contains six columns: `link`, `headline`, `category`, `short_description`, `authors` (all as strings), and `date` (as timestamps), providing comprehensive article metadata for NLP processing.
2. From 396 to line 397, it is possible to use a lambda function to filter the training dataset, retaining only articles categorized as "POLITICS". This approach results in the expected 35,602 items, as shown in the question.
3. From lines 402 to 406, the code implements tokenization by converting headlines to lowercase, splitting whitespace, and appending the `<EOS>` token to each sequence, following the assignment's recommendation for simple tokenization.
4. From line 402 to 433 there is the vocabulary analysis task , it was decided to implement a vocabulary construction and special token handling, taking inspiration by Exercize 4. The implementation creates two dictionaries `word_to_int` and `int_to_word` that establish a bidirectional mapping between words and integers. This approach positions special tokens strategically (`<EOS>` at index 0 and `<PAD>` at the final index) while maintaining a frequency-based vocabulary mapping, for the encoding and decoding operations in the LSTM model. The analysis revealed:
    - The 5 most common words in political news dataset, ranked by frequency: to: 10701 occurrences the: 9619 occurrences trump: 6896 occurrences of: 5536 occurrences in: 5250 occurrences

- The final vocabulary size,that includes all words plus special tokens with a total of `33207` tokens.

5. From lines 33 to 59, the dataset class implementation contains three components following Exercise 4's structure: an initialization method converting tokenized sequences to integer indices via `word_to_int`, a length method tracking the total sequence count, and a `getitem` method that generates training pairs by creating tuples of sequences.

6. From lines 60 to 74, the data loading implementation introduces a collate function to handle variable-length sequences by padding with `PAD` tokens to match batch maximum length. The implementation (lines 436 to 438) configures a DataLoader with batch size 32 as suggested in the final remarks.

Model definition (10 pts)

From lines 78 to 119, the implementation of the model follows three core architectural components mandated by the assignment specifications. The embedding layer at the input stage converts token indices into dense vectors, incorporating a dedicated padding index for consistent processing. The LSTM layer serves as the primary sequential processor, configured with adjustable hidden size and layer count parameters for flexibility in model capacity. A final fully connected layer maps the LSTM outputs to logits matching the vocabulary size. The key architectural distinction from RNNs lies in the LSTM's cell state mechanism, which maintains long-term dependencies through controlled information flow, enabled by specialized gating operations.

Evaluation - part 1 (10 points)

- The sampling mechanisms required three main implementations with specific modifications from Exercise 4: It's possible to find the code from line 124 to 178

  - `random_sample_next`: This function samples the next word based on the probability distribution p of the top k most likely candidates.
  - `sample_argmax`: A new function, not present in Exercise 4, that implements a deterministic selection method. Choose the word with the maximum probability from p using `Torch.argmax`, providing a greedy selection alternative.
  - `sample`: Enhanced from Exercise 4's version by adding a `sample_method` parameter for switching between *random* and *argmax* strategies. This unified function handles both sampling approaches while maintaining the core generation loop until either reaching `max_seqlen` or encountering the `<EOS>` token. In the end, it was decided to put a `max_seqlen` of 18, because generating the `<EOS>` token proved challenging for the untrained model. This limitation also aligns well with the nature of news headlines, which are typically concise and rarely exceed this length. Furthermore, this choice

helps prevent the model from generating overly long or run-on sequences that would be unrealistic for headline generation. The parameter also serves as a safeguard against potential infinite generation loops in cases where the model fails to naturally terminate with an `<EOS>` token.

- After implementing the necessary generation functions, the untrained model was tested. As expected, the generated sentences lack coherence and meaning, serving primarily to verify the functionality of the implementation and to establish a starting point for the next steps. Using the prompt **"the president wants"** and the helper function `keys_to_values` from Exercise 4 for token conversion, both sampling strategies were tested. The sampling process involves converting the text prompt into indices using `keys_to_values`, then feeding these indices into the `sample` function with appropriate parameters for each strategy. For the random sampling approach, `sample_method='random'` was set with `topk=5`, while for the greedy strategy, `sample_method='argmax'` was used. The generated indices are then converted back into readable text using the same `keys_to_values` function with the `int_to_word` mapping. mapping.

Strategy	**Generated Sequences**
Random Sampling	"the president wants bridge cellphones rejoin bomb-sniffing bridge slept ruthless sign!" "the president wants bridge worldview principle? regimes accrued suspending aliens'" "the president wants bridge thousands listens bungles york buddies, warning star"
Greedy Strategy	"the president wants nuclear disappointed stay-at-home players. over. away: november?"

Table 1: Generated sequences from untrained model using different strategies

These results demonstrate that while random sampling produces varied but incoherent outputs, the greedy approach, being deterministic, generates the same sequence each time.

Training (35 points)

1. From lines 183 to 253, the training process was implemented by leveraging prior experiences and integrating task-specific enhancements. The assignment required achieving a loss threshold of 1.5, prompting the implementation of early stopping once this target was met. Additionally, to monitor qualitative progress, sentence generation was conducted at each epoch, offering insights into the model's developmental trajectory. Perplexity was calculated following Lesson 8, using the exponential function applied to cross-entropy loss, which proved more reliable compared to alternatives such as *torch-metrics* that it creates some problems.

    - The training metrics showcase clear learning trends. The loss function (left plot in Figure 1) consistently decreases, reaching the tar-

get threshold of 1.5 by epoch 6, despite final remarks suggested 12, which activated the early stopping mechanism. The sharp decline in the initial epochs reflects efficient pattern learning, with subsequent refinements in later epochs. The perplexity plot (right) aligns closely with this improvement in predictive accuracy.

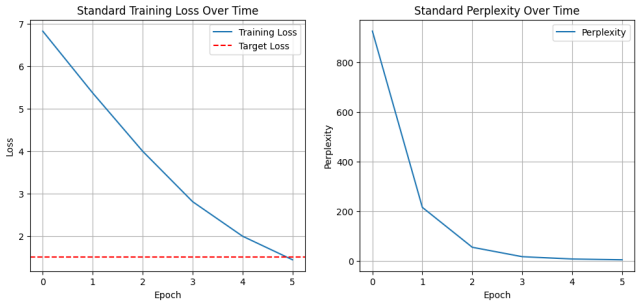

Figure 1: Loss and Perplexity curves over training epochs.

- The model's sentence generation capability significantly evolves during training, as depicted in Table 2:

Training Stage	**Generated Sentence**
Beginning	the president wants to know about the u.s. <EOS>
Middle	the president wants to help the long-term jobless. how's <EOS>
End	the president wants to solve domestic violence against women <EOS>

Table 2: Generated sentences at different epochs of training.

These sentences, using greedy strategy, illustrate a clear progression: from basic syntactic structures to relevant political headlines. Early-stage outputs show simple grammatical competence, mid-stage examples incorporate specific politics terms, while the sentence resemble professional news headlines in style and content.

- The architecture was chosen to balance performance and efficiency, aligning with the assignment guidelines. Key settings include a hidden state dimension of 1024 and a single LSTM layer, to capture complex headline patterns while avoiding overfitting. An embedding dimension of 150 was selected to represent word semantics compactly, enhancing the model's ability to generalize. The Adam optimizer with a learning rate of 0.001 facilitated stable and efficient convergence, while gradient clipping at 1.0 mitigated exploding gradients.

2. Truncated Backpropagation Through Time (TBBTT), implemented from lines 257 to 349, introduced sequence segmentation for memory-efficient training. Adjustments included increasing the LSTM size to 2048 and

setting a new parameter named chunk length of 35. For training stability, gradient clipping was initially set to 0.5 to handle the expanded model capacity and prevent gradient explosions in early training phases. As training progressed and the model stabilized, the clipping threshold was increased to 1.0, enabling more precise weight adjustments while maintaining stable convergence.

- TBBTT differs from standard training by processing sequences in smaller chunks of 35 tokens rather than processing entire sequences at once, which enabled an increase in the LSTM hidden state size from 1024 to 2048 units while maintaining computational efficiency. The chunked processing led to faster convergence, with TBBTT reaching the target loss threshold in 5 epochs compared to standard training's 6 epochs. While TBBTT's larger model capacity improved the text's ability to capture broader context, this came with a tradeoff in grammatical precision.
- The loss and perplexity plots for TBBTT training are shown in Figure 2. The loss function descends rapidly, reaching the target threshold of 1.5 by epoch 5. The steeper initial decline highlights the benefits of chunked processing and enhanced model capacity.

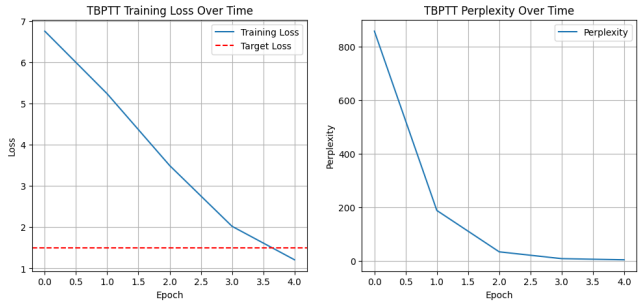

Figure 2: Loss and Perplexity curves for TBBTT training showing convergence at epoch 5.

- The progression in sentence generation during TBBTT training is summarized in Table 3:

Training Stage	**Generated Sentence**
Beginning	the president wants to be a big problem <EOS>
Middle	the president wants to make sure all eligible voters can they do it<EOS>
End	the president wants to make sure every a lot of things to be better president <EOS>

Table 3: Generated sentences during TBBTT training.

The generated sentences, using greedy strategy, demonstrate an interesting progression throughout the training process. Initial outputs establish a basic structure, while mid-training examples show

increased complexity and thematic exploration, albeit with occasional inconsistencies. In the final stage, the model exhibits the ability to incorporate political topics and produce linguistically coherent sentences, though semantic precision still leaves room for improvement.

Evaluation (5 points)

Using the prompt "the president wants", I evaluated both sampling strategies for generating political headlines.

Strategy	**Generated Examples**
Random Sampling	the president wants a citizenship now and a bond with a democratic lawmakers look <EOS> the president wants a citizenship and how the media was actually made it was legal <EOS> the president wants a lot more and it can prove help the women in puerto election <EOS>
Greedy	the president wants to solve domestic violence against women <EOS>

Table 4: Comparison of generation strategies using prompt "the president wants".

The greedy strategy demonstrates the model's ability to produce grammatically sound outputs that consistently align with professional news standards.In contrast, random sampling showcases the model's creative potential by generating diverse headlines exploring various political themes like citizenship and media relations, even though it sometimes trades grammatical precision for variety. Both approaches highlight promising aspects of the model's learning - structured coherence from greedy sampling and creative flexibility from random sampling .

Bonus Question (5 pts)

From line 541 to 569 ,it was chosen to start from the example relationship demonstrated by Word2Vec, such as king - woman = queen, the goal was to test a similar approach in the embeddings made, exploring relationships like republican - trump + biden. However, the results differed significantly: the closest terms were biden (17.7445) and republican (17.9412), rather than democratic.

This difference is due to distinct training objectives and datasets. Unlike Word2Vec, designed for semantic analogies, the embeddings of the assignment focus on predicting subsequent words in specific contexts, using a narrower dataset. Consequently, the analogical properties observed in Word2Vec do not emerge, reflecting methodological differences rather than errors.

The similar distances suggest a compressed embedding space for co-occurring political terms, unlike Word2Vec's more distributed space, where analogies are clearer.