Homework 3

Fatemeh Moradihaghighi

Clemson University

Contents

I uploaded the project in the GitHub link below. The file Bert.py is the source code and run_output folder shows all the figures and Json files generated during preprocessing evaluation.

https://github.com/fgmoradi/DeepLearning_HW3/tree/main

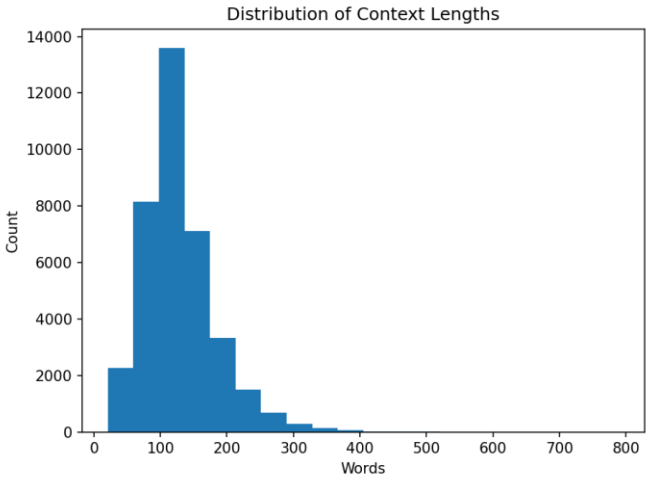
Introduction

The goal of this project is to build an extractive question answering (QA) system for spoken content: given a question and an ASR-transcribed passage (Spoken-SQuAD), the model must extract the contiguous text span that answers the question.
Key challenges vs. classic SQuAD: (i) ASR noise (insertions, deletions, substitutions), (ii) potential answer boundary drift under tokenization, and (iii) longer contexts.

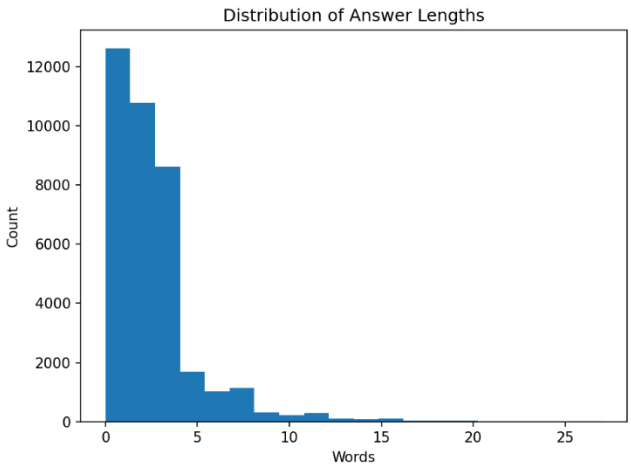
Dataset

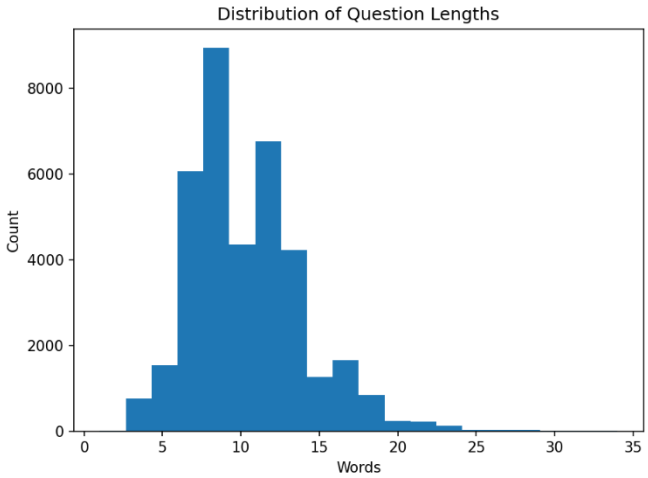
The Spoken-SQuAD split used in this homework contains 37,111 train and 15,875 validation QA pairs. Each item has a lower-cased ASR transcript as context, a question, and a ground-truth answer text with character offsets. I tokenize with BertTokenizerFast using a maximum sequence length of 512 to minimize truncation. The model input is the standard pair [question, context] with segment/type IDs. Gold start/end labels are obtained by sub-token matching of the answer text inside the tokenized pair. The "span-match failure" rate—cases where the gold answer cannot be aligned after tokenization—is 1.48% for training and 1.64% for validation, which is low enough to trust the supervision signal.

To characterize the data, I plot length distributions. Contexts cluster around 100–200 words, questions around 8–14 words, and answers are strongly right-skewed with the vast majority under 5 words, consistent with extractive QA assumptions. Span-length histograms (in subword tokens) on both train and validation confirm that most gold spans are short.

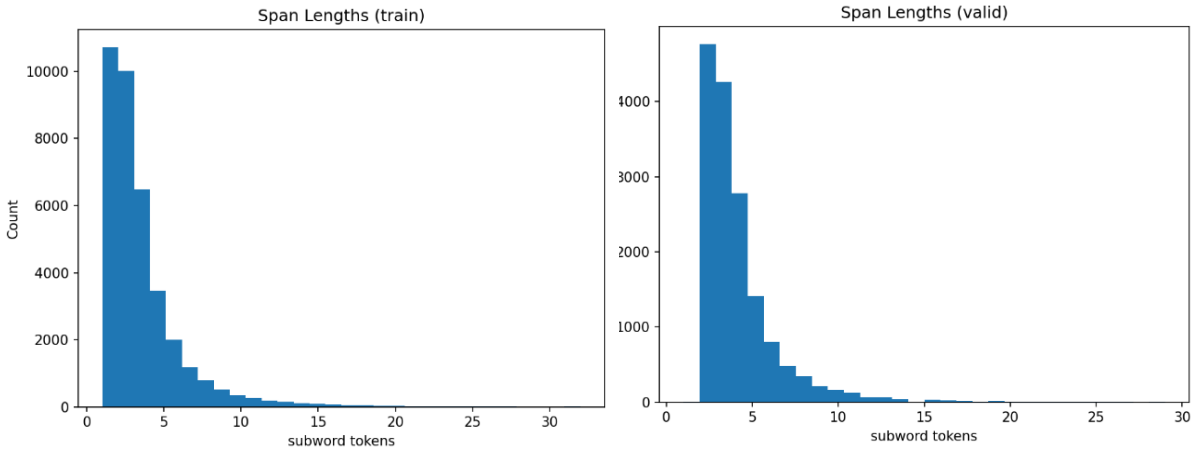


These diagnostics justify using a single-pass, windowed encoding at length 512 and a simple span decoder.





Also below are the Span lengths for training and test .



1-2 Model

I fine-tune bert-base-uncased for span prediction. The QA head concatenates the last and third-to-last hidden layers at each token position, feeds them through a dropout-MLP with LeakyReLU, and produces two logit vectors over the sequence: one for start and one for end. At inference time, for each example I take the argmax of start logits and end logits independently and decode the corresponding token span (clamping the order if needed). This is deliberately simple decoding; the homework fixes parameters and leaves decoding tricks out so we can analyze the core behavior cleanly.

1-1 Training

I fine-tune bert-base-uncased end-to-end with AdamW using an initial learning rate of 2×10^{-5} and weight decay **0.**02. AdamW decouples weight decay from the adaptive moments in Adam, which is now standard for Transformer fine-tuning: the decoupled L2 regularization controls overfitting on small, noisy datasets without interfering with Adam's per-parameter step sizes. The chosen learning rate is in the typical range for BERT-base with full fine-tuning and a small batch size; at 2e-5 it is high enough to adapt the encoder but conservative enough to avoid catastrophic forgetting of the pretraining signal.

I use a linear warmup/decay schedule with 0 warmup steps, implemented as a linear decay from the initial learning rate to zero across the total number of training steps. In practice this behaves like a gentle annealing schedule: early updates occur at the full 2e-5 rate, and the step size shrinks steadily as the model converges. Because the starting rate is already conservative and the loss (see below) damps unstable early gradients, an explicit warmup phase is not required here. The benefit of linear decay is predictable late-epoch stability: as the rate falls, the model refines span boundaries rather than chasing large parameter updates.

The objective is focal loss with gamma =1, applied independently to the start- and end-position logits and averaged. Extractive QA exhibits extreme class imbalance at each position (one correct index versus hundreds of negatives). With standard cross-entropy, the many easy negatives dominate the gradient early in training; focal loss down-weights well-classified positions by a factor $(1-p)^{\gamma}$, preserving gradient signal on the harder positions near the true boundary. With $\gamma = 1$the loss is only mildly "focal," which balances two goals: keeping the optimization close to cross-entropy (stable, well-understood behavior) while still reducing the tendency to over-fit trivial negatives or ASR filler tokens. Empirically this shows up as smooth loss decay and steadily rising start/end index accuracy across epochs.

Inputs are tokenized as [question] + [context] with a maximum sequence length of 512. This setting minimizes truncation on Spoken-SQuAD while staying within GPU memory limits for BERT-base. A batch size of 8 is used for training and 1 for validation. Small batches are common in QA fine-tuning because the sequence length dominates memory; they also introduce a modest amount of gradient noise, which can improve generalization on noisy ASR text. Validation with batch size 1 avoids OOM during decoding and simplifies span reconstruction. I train for 3 epochs, which is sufficient for convergence under these settings; longer runs risk overfitting to transcription quirks without architectural regularizers.

All experiments are conducted with seed 42 on CUDA. The full run of three epochs over 37,111 training examples comprises roughly $\lceil 37111/8 \rceil \times 3 \approx 13{,}917$optimization steps and completes in 4102.59 **s** on the available GPU. During training I monitor the mini-batch focal loss and the accuracy of the argmax start/end indices; at the end of each epoch I evaluate on the validation set and report EM**,** F1, and WER. The observed dynamics—monotonic loss reduction (1.59 → 0.80 → 0.49) and increasing start/end index accuracy (start: 0.50 → 0.76; end: 0.53 → 0.82)—indicate that the optimizer, schedule, and loss interact as intended: early steps explore with a full learning rate, then progressively fine-tune boundary decisions as the rate decays.

Below is the log from epochs

```
Train sizes: 37111 37111 37111
Valid sizes: 15875 15875 15875
[train] span-match failed: 548 / 37111 (1.48%)
[valid] span-match failed: 261 / 15875 (1.64%)
Train Epoch 1: 100%| | 4639/4639 [19:26<00:00,  3.98it/s, e_acc=0.53, loss=1.59, s_acc=0.495]
Eval: 100%| | 15875/15875 [03:22<00:00, 78.52it/s]
and ch 1] loss=1.5931 s_acc=0.4954 e_acc=0.5303 | WER=1.9293 EM=0.4612 F1=0.6412
Train Epoch 2: 100%| | 4639/4639 [19:24<00:00,  3.98it/s, e_acc=0.724, loss=0.802, s_acc=0.667]
Eval: 100%| | 15875/15875 [03:21<00:00, 78.60it/s]
[Epoch 2] loss=0.8025 s_acc=0.6665 e_acc=0.7238 | WER=1.6399 EM=0.4827 F1=0.6619
Train Epoch 3: 100%| | 4639/4639 [19:24<00:00,  3.98it/s, e_acc=0.817, loss=0.494, s_acc=0.758]
Eval: 100%| | 15875/15875 [03:21<00:00, 78.80it/s]
[Epoch 3] loss=0.4941 s_acc=0.7585 e_acc=0.8169 | WER=1.7509 EM=0.4876 F1=0.6670

Done. Files written to: run_outputs
```

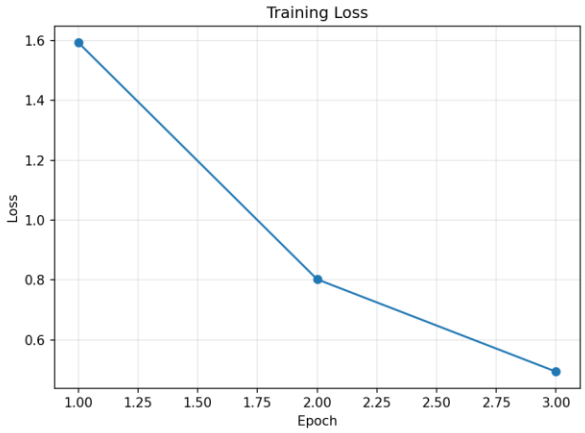
1-2 Evaluation

I report three complementary metrics:

1. Exact Match (EM): after whitespace normalization, 1 if predicted text equals the gold span exactly, else 0.
2. Token-level F1: harmonic mean of precision and recall over token sets of the predicted and gold spans; robust to small boundary and determiner differences.
3. Word Error Rate (WER): normalized edit distance between predicted and gold strings, commonly used in ASR. WER is sensitive to insertions, so returning an over-long span can hurt WER substantially even when the answer overlap (F1) is good

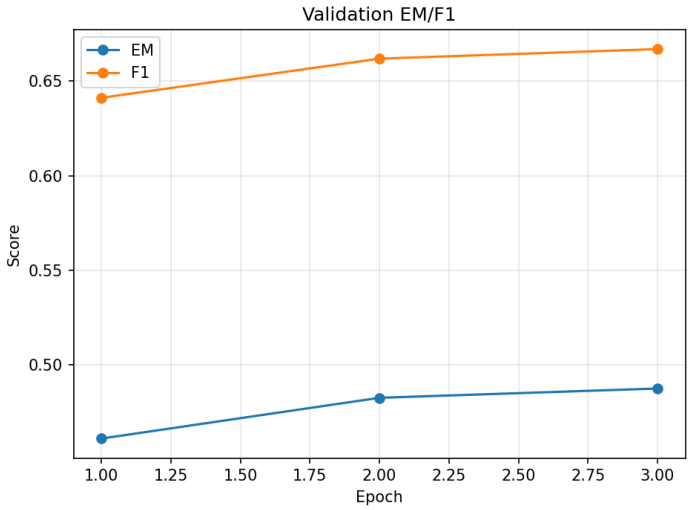
1-3 Results and Observations

Training behaves as expected. The average loss decreases monotonically from 1.593 , 0.802 , 0.494 across epochs 1–3, and the token-index accuracies improve in tandem (start: 0.495 , 0.667 , 0.758; end: 0.530 , 0.724 ,0.817). The curve is smooth, with no sign of divergence or plateaus. Insert the training curve:

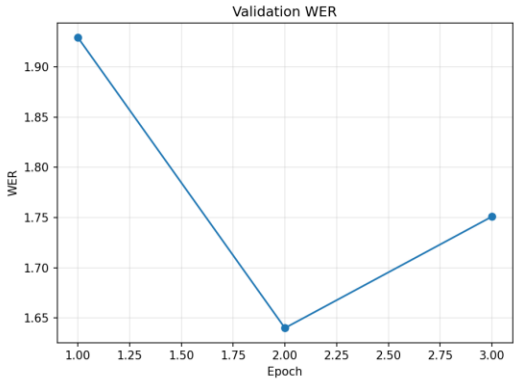


This is the signature of a well-conditioned optimization regime under the chosen schedule and focal loss.

On the validation set, EM improves from 0.461 to 0.488, and F1 improves from 0.641 to 0.667 over three epochs. These gains are monotonic and modest but consistent, which is typical for a BERT-base model on noisy transcripts without decoding constraints. Insert the span-metric plot:



WER shows a different pattern: 1.929 (epoch 1) , 1.640 (epoch 2, best) , 1.751 (epoch 3)**.**



This divergence—F1 continuing to rise while WER bottoms out at epoch 2—is consistent with occasional over-long predictions at epoch 3. Qualitatively, when the model selects a long clause that contains the answer (for example, predicting an entire sentence that includes “golden anniversary” as a substring), token overlap remains high (so F1 increases), but WER becomes worse because insertions are heavily penalized. On spoken text, especially with ASR artifacts, span-based QA and ASR-style WER are not perfectly aligned objectives.

It is therefore appropriate to select a checkpoint by the downstream preference**:** if the target application is QA grading by span overlap, epoch 3 is the best (EM/F1 highest). If the goal were minimizing a transcript-string distance (WER), epoch 2 is preferable.