

Homework 3: Tokenization, Long Context, and Pretraining Data

CS 388: Natural Language Processing • Fall 2026

Instructions. Please transcribe your answers/work into the provided L^AT_EXfile and submit the PDF. Please use the `\B{}` command to wrap your solutions, making them blue (this will help us when grading).

1. Question 1: Byte-Pair Encoding (35 points)

In class we saw byte-pair encoding (BPE), which builds a subword vocabulary by repeatedly merging the most frequent pair of adjacent symbols. In this question you will run BPE by hand on the small corpus below, which is given as a list of word types with their counts.

Training corpus $\mathcal{D}$			
talking	8	walks	4
talked	5	king	2
walk	6	thing	3
walking	3		

Setup. Every word is first split into characters, and a special end-of-word symbol `_` is appended, so `walk` starts out as `w a l k _`. The initial vocabulary is the set of all characters that occur in $\mathcal{D}$, plus `_`. Merges do not cross word boundaries.

Learning. At each step, count every pair of adjacent symbols across the corpus (weighting each word by its count), merge the most frequent pair everywhere it occurs, and add the merged symbol to the vocabulary.

Tie-breaking. If several pairs are tied for most frequent, merge the pair that comes first in alphabetical order, comparing first symbols and then second symbols, and treating `_` as coming *after* every letter. For example, `(d, s)` comes before `(e, a)`, and `(e, d)` comes before `(e, _)`.

Tokenizing. To tokenize a new word, split it into characters plus `_`, then apply the learned merges *in the order they were learned*, each one everywhere it applies.

- (a) **(20 pts)** Write down the initial vocabulary and its size. Then run 9 merges. For each merge, report the pair that was merged and its count. Finally, report the size of the vocabulary after all 9 merges and the segmentation of each of the 7 training words.

Example (merges)

Step	Merged pair	Count
1	(?, ?)	?
...	...	...

- (b) **(10 pts)** Using your merges from (a), tokenize each of the following words. Show the final segmentation of each.

walked talks kings stalking thinking sled

- (c) **(5 pts)** Which of the words in (b) are tokenized “well” (i.e. the tokens line up with meaningful units of the word) and which are tokenized poorly? For each poorly-tokenized word, explain in one sentence what about the training corpus or the BPE procedure caused it.

2. Question 2: Long-Context Models (40 points)

- (a) **(15 pts) Length generalization.** Suppose we pre-train a decoder-only Transformer only on sequences of length exactly 1024, and then, at test time, give it sequences of length 2048. For each of the two positional encoding schemes below, describe what the model “sees” at positions 1025 through 2048, and whether you expect its predictions there to remain good. Explain your reasoning in a few sentences for each question.
- (i) Sinusoidal (cosine) absolute positional embeddings, as in the original Transformer, where a fixed vector p_i whose entries are $\sin(i \omega_k)$ and $\cos(i \omega_k)$ for a range of frequencies ω_k is added to the token embedding at position i .
 - (ii) Rotary positional embeddings (RoPE), where the query at position m and the key at position n are each rotated by angles proportional to their positions, so that the attention score $q_m^\top k_n$ depends on the positions only through the offset $m - n$.
- (b) **(15 pts) Attention over long contexts.** Consider a single attention head at query position n , attending over all positions $1, \dots, n$ with scores $s_1, \dots, s_n$ and weights $\alpha_i = \exp(s_i) / \sum_{j=1}^n \exp(s_j)$. Suppose exactly one position i^* (the “needle in the haystack”) holds the information the query needs. For example, imagine we are looking for a particular name in a long document. The name is the “needle” and the document is the “haystack”. Call the other $n - 1$ positions *distractors*. Write $s = s_{i^*}$ for the needle’s score, and suppose the head has learned to rank the needle first by a margin of *at least* $\Delta > 0$: every distractor j scores $s_j \leq s - \Delta$. Some distractors may score much lower than $s - \Delta$; we only need to assume that none of them comes within Δ of the needle.

Under this assumption we can lower-bound the needle’s attention weight. Dividing the numerator and denominator by e^s ,

$$\alpha_{i^*} = \frac{e^s}{e^s + \sum_{j \neq i^*} e^{s_j}} = \frac{1}{1 + \sum_{j \neq i^*} e^{s_j - s}}.$$

Since $s_j - s \leq -\Delta$ for every distractor, each term satisfies $e^{s_j - s} \leq e^{-\Delta}$, so the denominator is at most $1 + (n - 1)e^{-\Delta}$, and therefore

$$\alpha_{i^*} \geq \frac{1}{1 + (n - 1)e^{-\Delta}} = \frac{e^\Delta}{e^\Delta + n - 1}.$$

The bound holds with equality only when every distractor scores exactly $s - \Delta$.

- (i) With $\Delta = 5$, compute the lower bound on α_{i^*} for $n = 128$, $n = 1024$, and $n = 8192$ (to 3 decimal places). What happens to this guarantee as $n \rightarrow \infty$? If α_{i^*} itself shrank in the same way, what would that mean for the output of the head, $o = \sum_i \alpha_i v_i$?
- (ii) A lower bound that goes to 0 does not by itself show that α_{i^*} goes to 0: in principle the distractors could score far below $s - \Delta$ and keep the needle’s weight high. Explain why this does not rescue us in a trained Transformer, so that α_{i^*} really does get washed out as n grows. Your answer should refer to the form of the softmax, and should address two points: (1) why the distractors cannot all be pushed arbitrarily far below the needle, and

- (2) why a trained model cannot simply fix the problem by making Δ very large. (*Hint for both:* the queries and keys are computed from layer-normalized hidden states with fixed weights, so every score lies in some bounded range $[-B, B]$. Use this to give an *upper* bound on α_{i^*} .)
- (c) **(10 pts)** Propose one modification to the attention computation that would mitigate the problem from (b). Describe it precisely (an equation is welcome), and explain how it changes your answer to (b)(i).

3. Question 3: Pretraining Data (25 points)

C4 (the “Colossal Clean Crawled Corpus”) is a common pretraining dataset derived from Common Crawl, available at <https://huggingface.co/datasets/allenai/c4>. Besides the filtered `en` split, it includes an `en.noclean` split containing the raw English web text *before* C4’s cleaning filters were applied. Each datapoint has a `text` field, a `timestamp` field, and a `url` field giving the web page it was scraped from.

Look through the `en.noclean` split (downloading a single shard file from the `en.noclean/` directory is more than enough) and find **3** datapoints you would want to exclude from a pretraining corpus. Your three examples should illustrate three *different* reasons for exclusion. Fill in the table below. Snippets should be copied from the `text` field and be at most 400 characters; if a snippet contains offensive language, you may replace those words with *******.

Snippet (400 chars max)	URL to datapoint	Reason for excluding