Tokenization

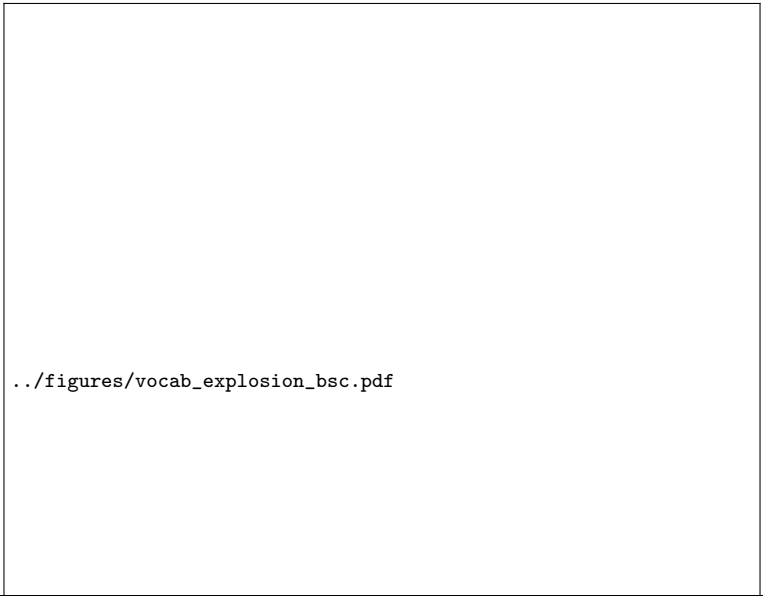
Week 8 - From Bytes to Subwords

NLP Course 2025

October 27, 2025

Two-Tier BSc Discovery

The Vocabulary Explosion Problem



The Trilemma

Character-Level
Vocab: 256
Memory: Tiny
Sequences: 5× longer
Speed: Slow

Word-Level
Vocab: 100K+
Memory: Huge
Sequences: Short
OOV: Can't handle "COVID"

Subword (Solution)
Vocab: 30K
Memory: Right-sized
Sequences: Reasonable
OOV: Handles everything

Subwords are the Goldilocks zone - not too big, not too small

What Are Subwords?

../figures/subword_concept_bsc.pdf

Three Subword Methods

BPE
Byte-Pair Encoding
Bottom-up merging
Most common

WordPiece
Likelihood-based
BERT uses this
Similar to BPE

SentencePiece
Unigram LM
Language-agnostic
Google's standard

All three work well - BPE most widely used

Why Subwords Work

Key Advantages:

- Fixed vocabulary size (30K typical)
- Handle rare/unknown words via composition
- Capture morphology ("play" in "playing", "player")
- Language-agnostic (same algorithm for all languages)
- Balance sequence length vs vocab size

Example: "COVID-19" (unseen)

- Word-level: UNK (fails!)
- Subword: ["CO", "VI", "D", "-", "19"] (works!)

Compositionality solves the OOV problem

BPE: The Core Idea

Byte-Pair Encoding (Sennrich et al., 2016)

Algorithm:

1. Start with characters
2. Find most frequent pair
3. Merge into single token
4. Repeat until desired vocabulary size

Example:
Corpus: "low low low lowest lowest"

- Most frequent pair: ("l", "o") appears 5 times
- Merge: "lo"
- Result: "lo w lo w lo w lo west lo west"
- Repeat...

Greedy algorithm - simple but effective

BPE Algorithm Flowchart

Worked Example: BPE Merge Steps

Corpus: "low low low low lowest lowest"

Initial: Characters = l, o, w, e, s, t

Step 1: Count pairs
(l,o): 6 times
(o,w): 4 times
(e,s): 2 times
Most frequent: (l,o)
Merge: "lo" added to vocabulary

Step 2: Corpus now "lo w lo w lo w lo w lo west lo west"
Count pairs:
(lo,w): 4 times
(w,e): 2 times
Merge: "low"

Continue until 30,000 tokens...

Real BPE runs millions of merges - this shows the pattern

WordPiece: BERT's Tokenizer

Similar to BPE but chooses merges by likelihood increase

Key Difference:
BPE: Max frequency
WordPiece: Max log P(corpus) increase

Example: "unhappiness" $\rightarrow$ ["un", "##happiness"]
indicates continuation

Used By: BERT, DistilBERT, ALBERT (30,522 tokens)

Likelihood-based selection slightly better empirically

Key Takeaways

1. Subword tokenization solves vocabulary explosion
2. BPE: Greedy merging of most frequent pairs
3. 30K vocabulary balances coverage and efficiency
4. Handles rare/OOV words via composition
5. Universal across all modern transformers

Tokenization is foundational - all models use it

Technical Appendix

Appendix A1: BPE Algorithm Complete

Formal Algorithm:
vocab ← all characters **while** —*vocab*— ¡ *target_size* **do**
end
pairs ← count all adjacent pairs in corpus best_pair ← argmax pairs vocab ← vocab ∪ {best_pair} Replace all occurrences of best_pair in corpus

Complexity: $O(N \times V)$ where N = corpus size, V = vocab size

Stopping: When vocab reaches 30K-50K (empirically optimal)

Simple greedy algorithm with strong empirical performance