Fine-tuning LLMs

CS 685, Fall 2025

Advanced Natural Language Processing

https://people.cs.umass.edu/~brenocon/cs685_f25/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

- project proposals!

Talks this week, for EC

- Wed: at the NLP seminar
  - https://nlp.cs.umass.edu/seminar/
  - Alisa Liu, University of Washington, "**Between Language and Models: Rethinking Algorithms for Tokenization**."

  - Abstract: Language models operate over real numbers, while users of language models interface with human-readable text. This is made possible by tokenization, which encodes text as a sequence of embeddings and decodes real-valued predictions back into generated text. Despite its foundation importance to language modeling, the algorithms for tokenization have remained largely unchanged in the era of LLMs. In this talk, I will discuss my recent work in improving algorithms for tokenization. The first half presents SuperBPE, a superword tokenizer that extends traditional subword tokenization to include tokens that span multiple words. We motivate superword tokens from a linguistic perspective, and demonstrate empirically that models pretrained from scratch with SuperBPE achieve stronger performance on downstream tasks while also being significantly more efficient at inference-time. The second half revisits a fundamental limitation of tokenizer-based LMs: models trained over sequences of tokens cannot, out of the box, model the probability of arbitrary strings. I discuss the practical implications of this in domains such as Chinese and code, and then present an inference-time algorithm that converts LM-predicted probabilities over tokens into probabilities over characters, while preserving the sampling distribution at the text level. I will conclude by discussing open questions on the future of tokenization.
  - Related papers: SuperBPE: Space travel for language models (Liu et al., COLM 2025) and Sampling from your language model one byte at a time (Hayase et al., arxiv 2025).

Talks this week, for EC

- Fri: visitor **Tal Linzen** at the linguistics colloq.!

---------- Forwarded message ---------
From: Carla Spellerberg <cspellerberg@umass.edu>
Date: Mon, Oct 13, 2025 at 10:02 PM
Subject: UMass Linguistics Colloquium - Tal Linzen - Friday, October 17, 2025 (party RSVP attached)
To: <ling-fac@linguist.umass.edu>, <ling-grad@linguist.umass.edu>, <ling-visitor@linguist.umass.edu>, <ling-colloq@linguist.umass.edu>
Cc: <linzen@nyu.edu>

Dear all,

This **Friday, October 17**, we will have the second talk in our Fall colloquium series. Our speaker will be Tal Linzen, who is an Associate Professor of Linguistics and Data Science at New York University and a Research Scientist at Google.
The talk will take place in **ILC S211 at 3:30pm**.

[...]

To model human linguistic cognition, make LLMs less superhuman

When people listen to or read a sentence, they actively make predictions about upcoming words: words that are less predictable are generally read more slowly than predictable ones. The success of large language models (LLMs), which, like humans, make predictions about upcoming words, has motivated exploring the use of these models as cognitive models of human linguistic prediction. Surprisingly, in the last few years, as language models have become better at predicting the next word, their ability to predict human reading behavior has declined. This is because LLMs are able to predict upcoming words much better than people can, leading them to predict lower processing difficulty in reading than observed in human experiments; in other words, mainstream LLMs are ‘superhuman’ as models of language comprehension. LLMs’ superhumanness is primarily driven by two factors: compared to humans, LLMs have much stronger long-term memory for facts and training examples, and they have much better short-term memory for previous words in the text. In this talk, I'll survey some of the work from my group and others that supports these hypotheses, and will outline some ongoing and future work possible directions for creating more human-like LLMs.

Using labeled data with LLMs

- Today: gradient-based learning for LLMs with labeled data
- Thursday: human labeling and natural language understanding tasks

- Why LM at all? Why not supervised data only for training?
  - RNNs/LSTMs, BoE: sure (e.g. Stanza: Qi et al. 2020)
  - Transformers: no, needs LM pretraining

Stanza

Overview

Usage

Neural Pipeline

Models

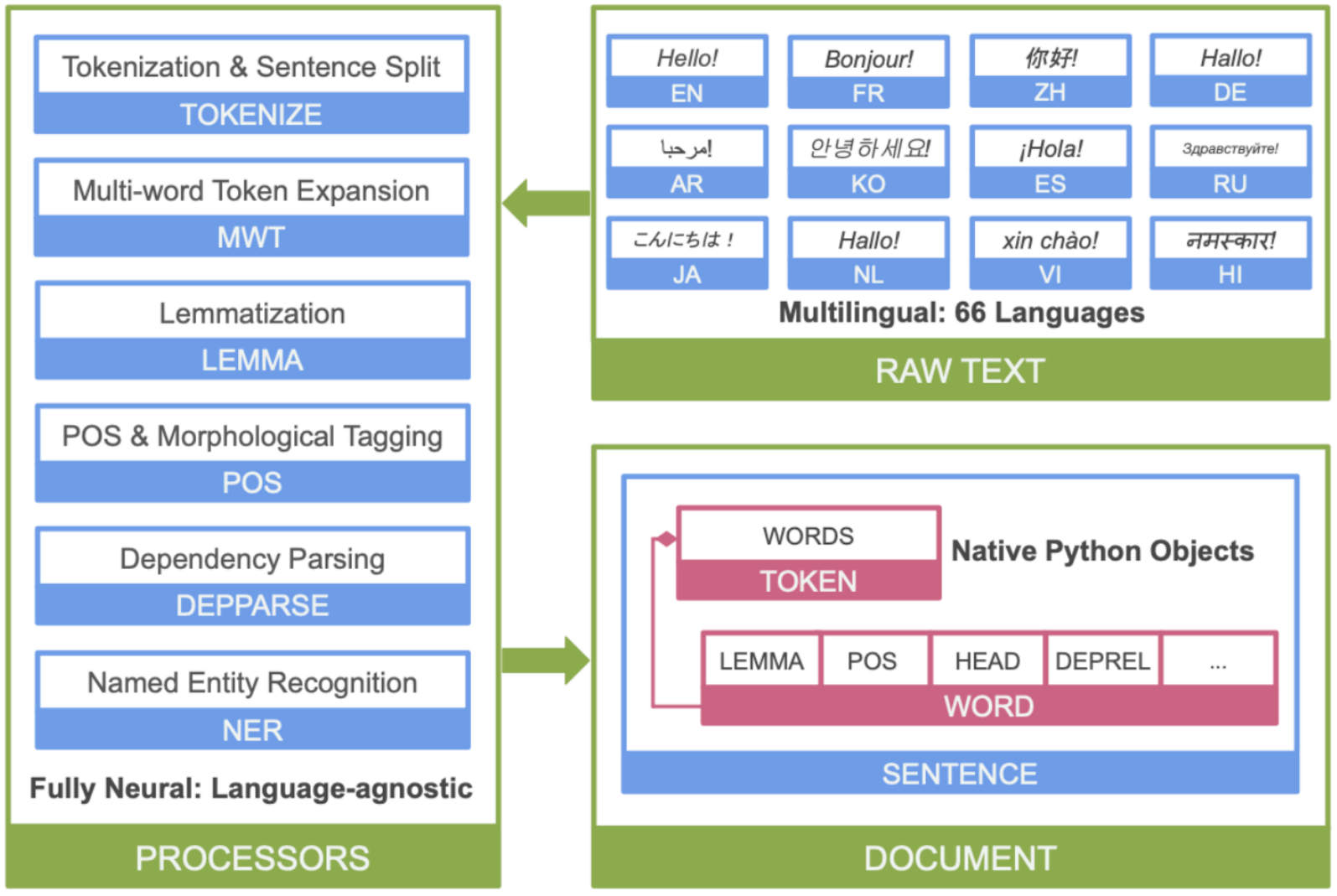
Biomedical Models

Training

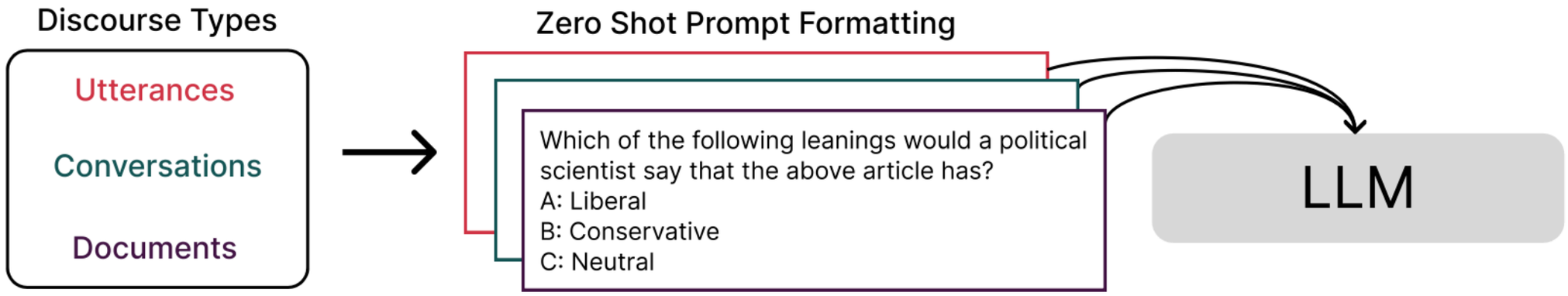
Stanford CoreNLP Client

Resources

Below is an overview of Stanza's neural network NLP pipeline:



- Why used labeled data at all? Why not zero-shot prompts?



Effective Prompt Guideline	Reference	Guideline Example
When the answer is categorical, enumerate options as alphabetical **multiple-choice** so that the output is simply the highest-probability token ('A', 'B').	Hendrycks et al. (2021)	{$CONTEXT} Which of the following describes the above news headline? ⏎ **A:** Misinformation ⏎ **B:** Trustworthy ⏎ {$CONSTRAINT}
Each option should be separated by a newline (⏎) to resemble the natural format of online multiple choice questions. More natural prompts will elicit more regular behavior.	Inverse Scaling Prize	
To promote instruction-following, **give instructions *after* the context** is provided; then **explicitly state any constraints**. Recent and repeated text has a greater effect on LLM generations due to common attention patterns.	Child et al. (2019)	{$CONTEXT} **{$QUESTION}** **Constraint:** Even if you are uncertain, you **must pick either "True" or "False"** without using any other words.
Clarify the expected output in the case of uncertainty. Uncertain models may use default phrases like "*I don't know*," and clarifying constraints force the model to answer.	No Existing Reference	
When the answer should contain multiple pieces of information, `request responses in JSON format`. This leverages LLM's familiarity with code to provide an output structure that is more easily parsed.	MiniChain Library	{$CONTEXT} {$QUESTION} `JSON Output:`

Table 1: **LLM Prompting Best Practices** to generate consistent, machine-readable outputs for CSS tasks. These techniques can help solve overgeneralization problems on a constrained codebook, and they can force models to answer questions with inherent uncertainty or offensive language. See full example prompts in the Appendix.

[Ziems et al., 2023]

- Why used labeled data at all? Why not zero-shot prompts?

- how to select good prompts??
  - tricks like this? https://www.promptingguide.ai/
- prompt choice is tightly interleaved with the LLM
  - e.g. sensitivity whitespace and punctuation, wordings/ phrasings
- how do you know if the LM classifier is doing what you want?
  - need human labels for evaluation!
  - but if you have labels, might as well train.
    - typically, this improves the model.

- Fine-tuning, general paradigm
  - Use LM pretraining to initialize weights
  - Use supervised objective to update
    - inductive bias: hopefully don't move too far away
- Today: methods for FT
  - "Vanilla" fine-tuning
    - Classification
    - More LM
  - Parameter efficient FT
    - Soft prompts
    - Low-Rank Adaptation

Fine-tuning for classification

- From a BERT encoder-only model:
  - Add new classification head, typically attached to "[CLS]" token's last layer
  - Gradient update ALL Transformer params (and from the new classif. head) from classification cross-entropy

Fine-tuning for tagging

- From a BERT encoder-only model:
  - Add a new tag classification head, attached to each token's last layer
  - Gradient update ALL Transformer params (and from the new classif. head) from tag classification cross-entropy

Fine-tuning for similarity: SentenceBERT

- There are many released BERT-likes tuned for specific tasks - sentiment, toxicity, etc.
- **SentenceBERT** is worth mentioning: designed to encode sentences to embedding vectors
  - f(sentence) --> vector
  - The model is trained to give high cosine similarity to human-annotated pairs of similar, paraphrased, sentences
- Document retrieval, clustering, etc.
- https://sbert.net/

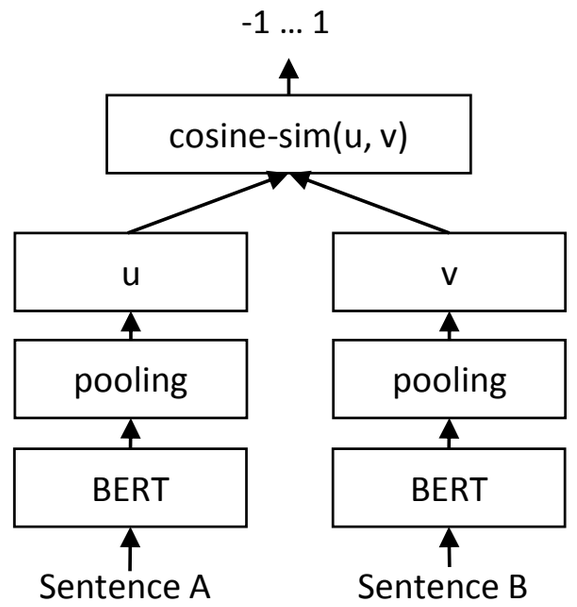


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

[Reimers and Gurevich, 2019]

Using BERT

- Fine-tuned BERT is an accurate, widely appliable way to train a text classifier or tagger with at least a moderate (>100) amount of labeled data
- Param efficient, compared to FT genLLM
  - Typically BERT-likes are ~100M parameters
    - Trained on "only" ~3B tokens (Wikipedia + (pirated) BooksCorpus)
    - Compare to >=7B for most generative LLMs
- Many pretrained BERT-like, MLM-trained models are available
  - Naming: "BERT" sometimes means the original release, but sometimes means the general class of models (!)
  - RoBERTa, DeBERTa, "ModernBERT", ...
  - Many languages
    - mBERT and XLM-R: multilingual models
    - or specific languages or language families (AfriBERTa, LatinBERT, ...)
  - Many domains (LegalBERT, BERTweet, SciBERT, ...)

Fine-tuning gen. LLMs

- Mechanism: FT with the next-word prediction head

- continued pretraining
  - help LM adapt to new domain or language

- alignment/safety: give examples of desired generative outputs
  - "Supervised FT for instruction-tuning" in this context; alternatives exist (next week)

- do a specific task
  - assume your labeled task is cast as generation

Parameter-Efficient Finetuning

Adapting to a new domain by continued pretraining (finetuning) is a problem with huge LLMs.

- Enormous numbers of parameters to train
- Each pass of batch gradient descent has to backpropagate through many many huge layers.
- Expensive in processing power, in memory, and in time.

Instead, **parameter-efficient fine tuning** (PEFT)

- Efficiently select a subset of parameters to update when finetuning.
- E.g., freeze some of the parameters (don't change them),
- And only update some a few parameters.

[slide: SLP3]

Soft prompts

- *(Lester et al., EMNLP 2021)*

- [for classification,] assume your labeled task is cast as generation

- a textual prompt selects input embeddings, used for generation

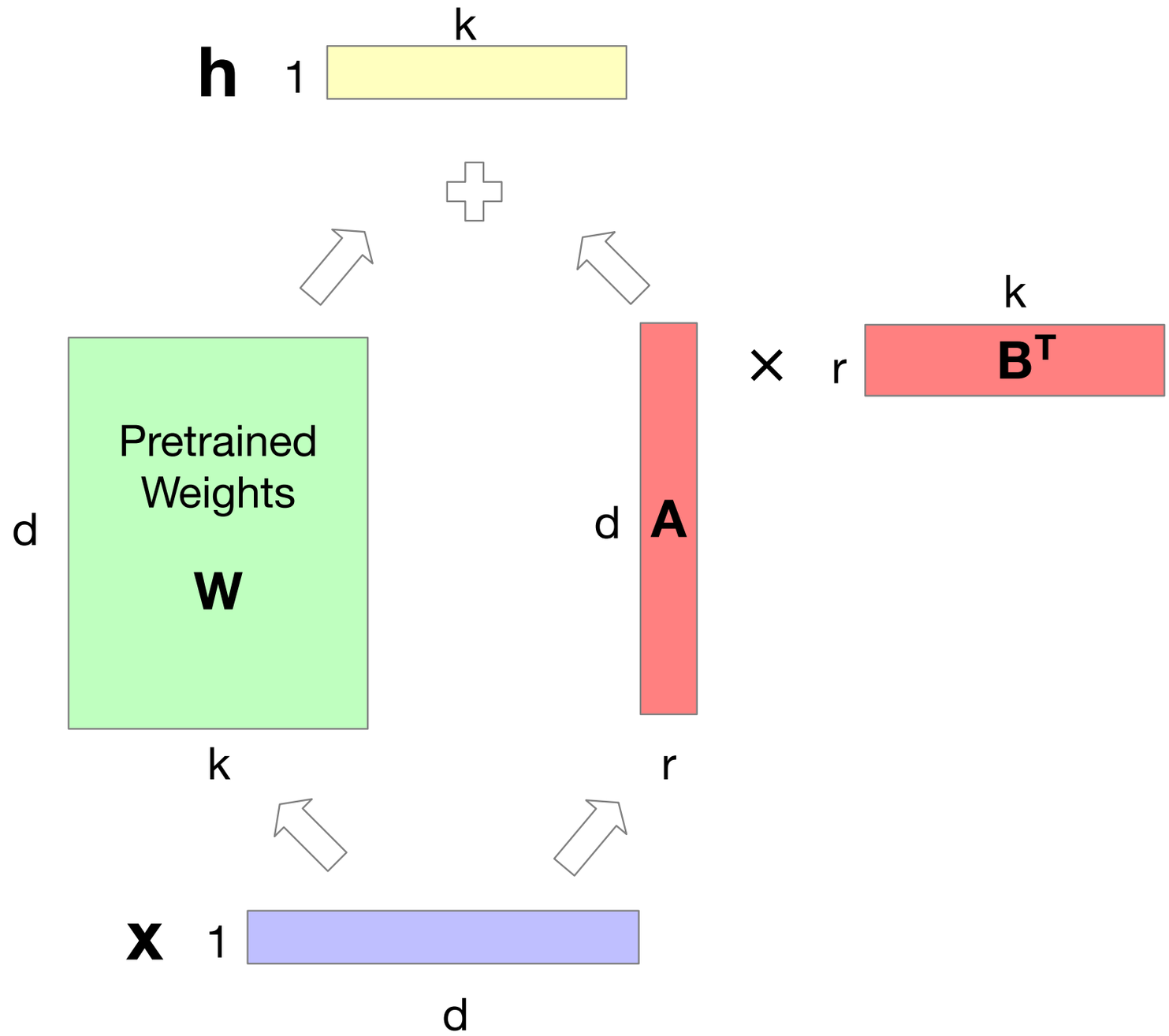
- use labeled predictive loss to gradient descend the input embedding, to encourage the LLM to generate desired outputs
  - ...this "soft prompt" can be easily stored shared

LoRA (Low-Rank Adaptation)

- Trransformers have many dense matrix multiply layers
  - Like $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$, $\mathbf{W}^O$ layers in attention
- Instead of updating these layers during finetuning,
  - Freeze these layers
  - Update a low-rank approximation with fewer parameters.

[slide: SLP3]

LoRA



[slide: SLP3]