Large Language Models I

HSS 510 / DS 518: NLP for HSS

Taegyoon Kim

May 27, 2025

Agenda

Things to be covered

- What generative language models are, and how they generate text
- How attention and decoding work in generative models
- Decoding strategies: greedy, top-k, top-p, temperature
- Using LLMs for text classification
- Guided coding

Generative LLMs

What do we mean by LLMs?

- Overall, LLMs tend to refer to generative models
  - GPT, LLaMA, Claude, Gemini, etc.
- Various terminology: autoregrsesive, left-to-right, or causal models
- Autoregressive models
  - Use their earlier predictions to make later predictions (e.g., the model's first generated token is used to generate the second token)
- They are primarily decoder-only models
  ↔ encoder-only (representation) models (e.g., BERT)
  ↔ encoder-decoder (seq-to-seq) models (e.g., T5, BART)

Generative LLMs

Autoregressive models do not generate all at once

- They generate one token at a time (just like ChatGPT)
- Each token generation step is one forward pass through the model
  - The input tokens go into the model and flow through the computations to produce an output
- After each token generation, the generated token is appended to the end of the original input from the previous run
- So the model is run in a loop to sequentially expand the generated text until completion

Generative LLMs

Autoregressive models do not generate all at once

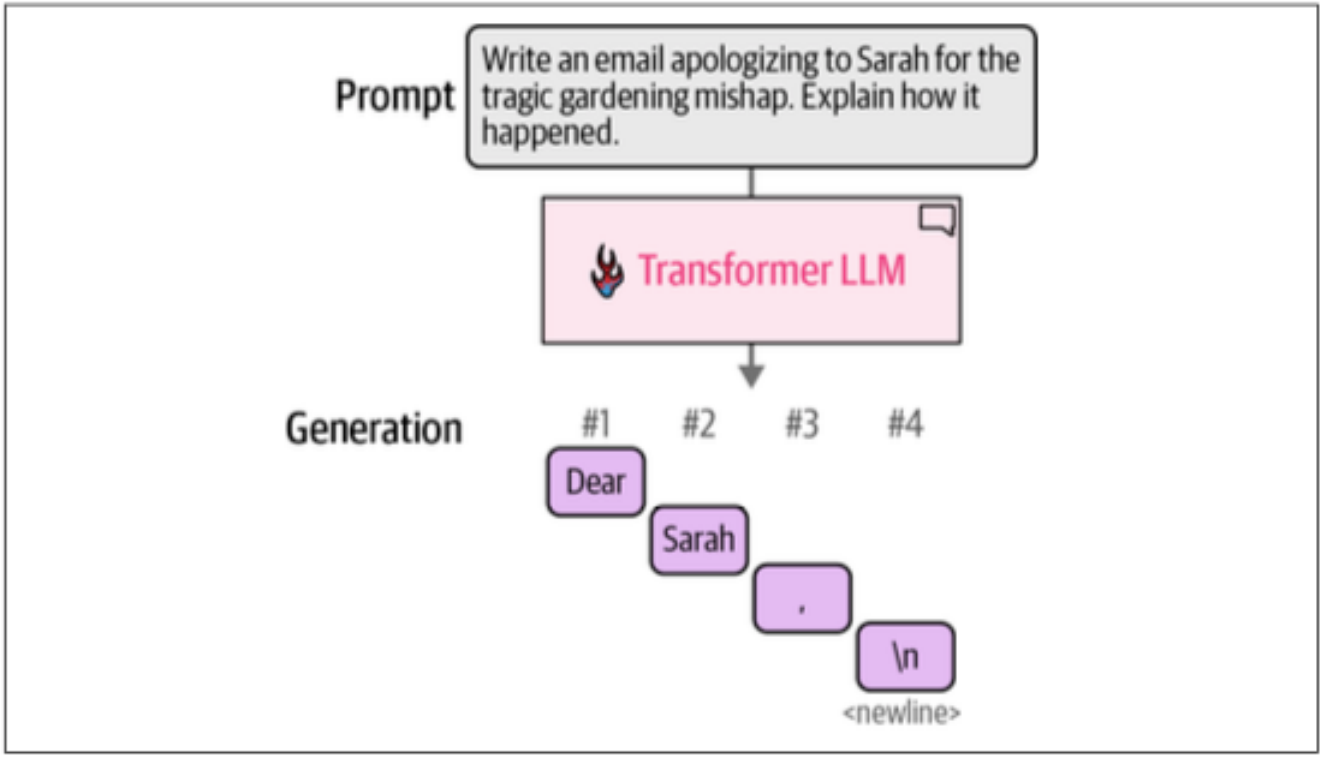

Figure 3-2. Transformer LLMs generate one token at a time, not the entire text at once.

Generative LLMs

Autoregressive models do not generate all at once

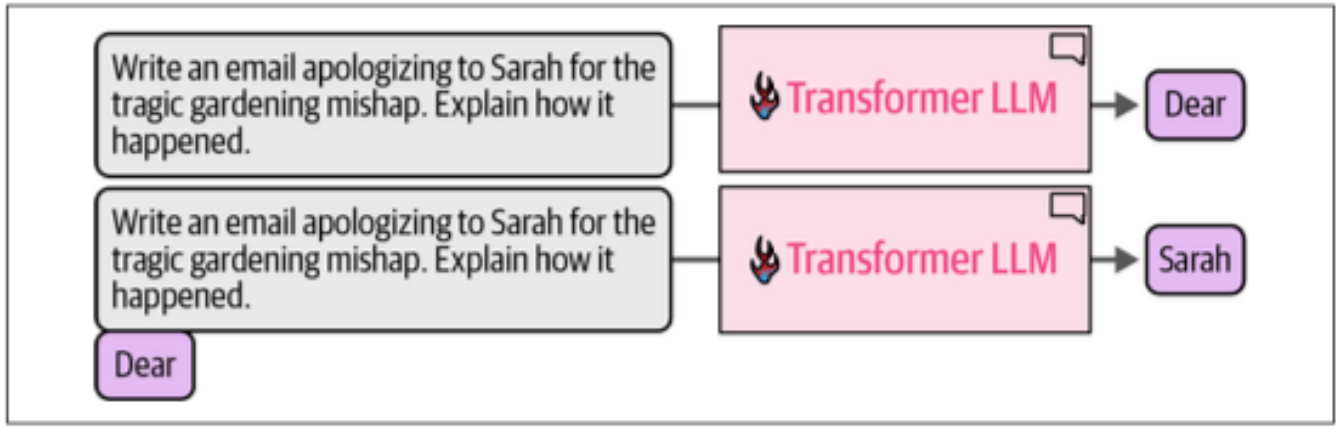

Figure 3-3. An output token is appended to the prompt, then this new text is presented to the model again for another forward pass to generate the next token.

Generative LLMs

How is the next token predicted?

- Only the final token's output vector (= vector representation) is passed into the final layer (language model head) to predict the next token
- This head produces a probability distribution over the vocabulary
- Then, why compute representations for all tokens if we discard all but the last?
  - The final output vector incorporates contextual information from all preceding tokens
  - This allows the vector to reflect the overall meaning of the input

Generative LLMs

How is the next token predicted?

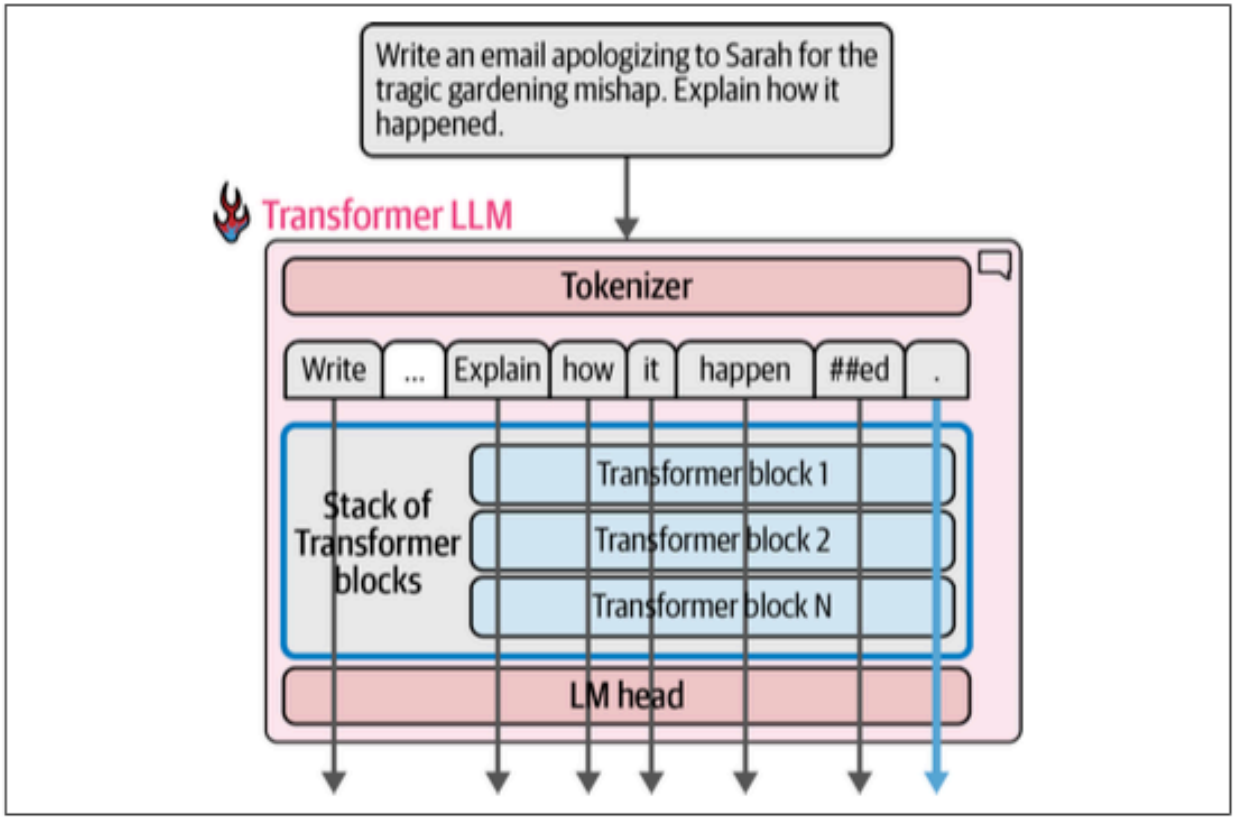

Figure 3-8. Each token is processed through its own stream of computation (with some interaction between them in attention steps, as we'll later see).

Generative LLMs

How is the next token predicted?

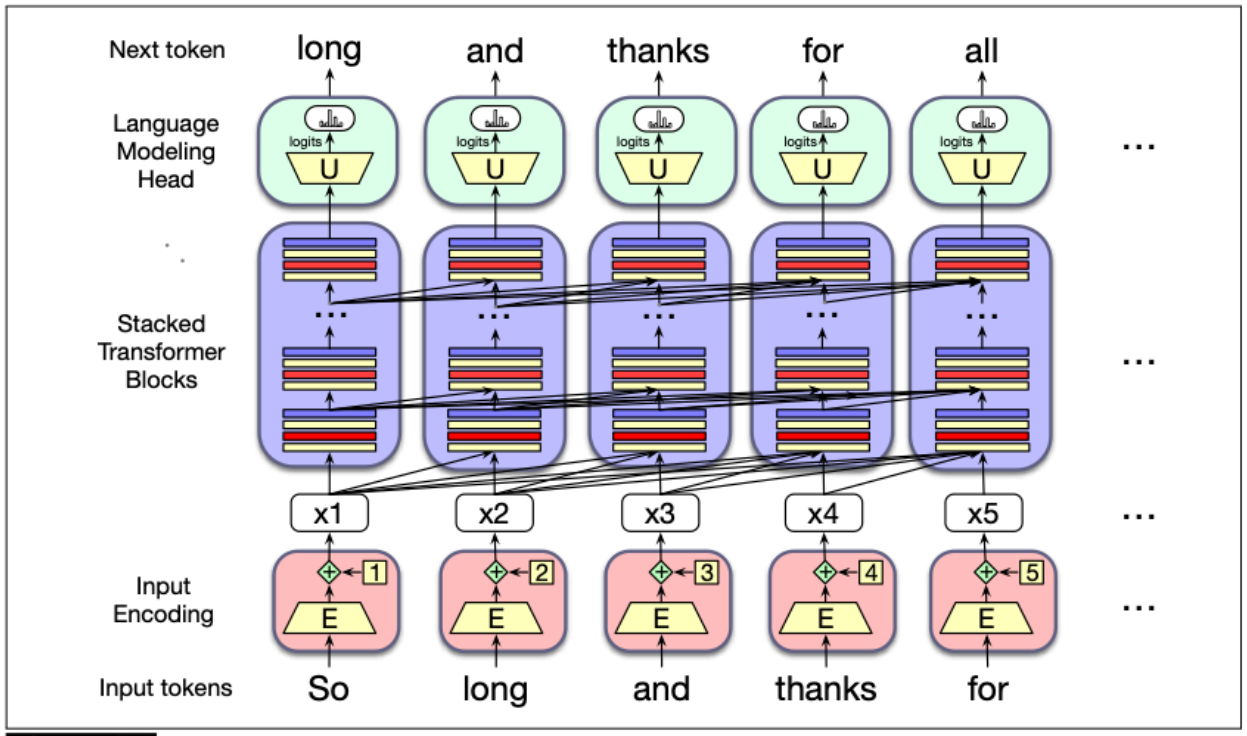

Figure 9.1 The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.

Attention in Generative Models

Simplified illustration

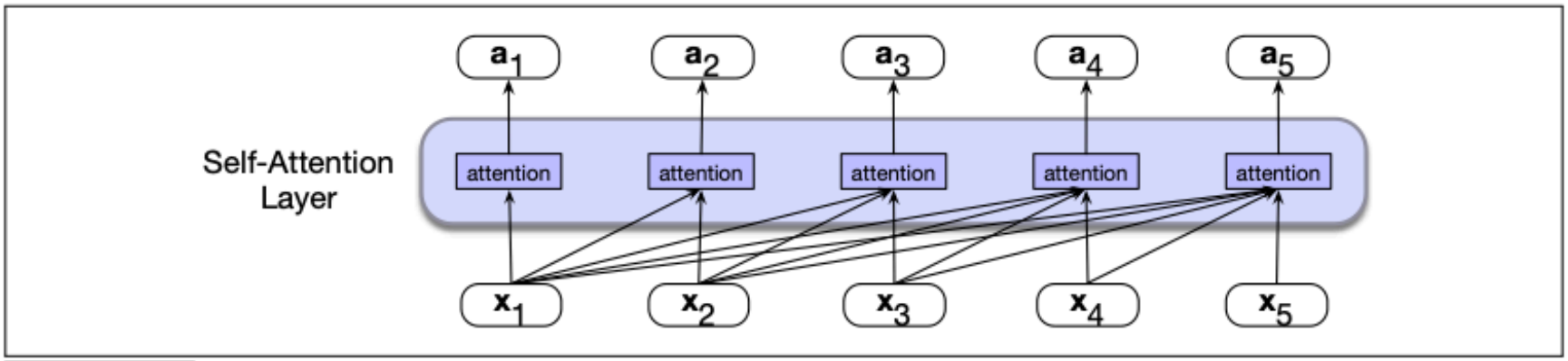

Figure 9.3 Information flow in causal self-attention. When processing each input $\mathbf{x}_i$, the model attends to all the inputs up to, and including $\mathbf{x}_i$.

Attention in Generative Models

Simplified illustration

$$a_i = \sum_{j \leq i} \alpha_{ij} \cdot x_j$$

Where:

- a_i is the updated representation for token i
- x_j is the input representation of token j
- α_{ij} is the attention weight from token i to token j

Attention in Generative Models

Self-attention

- The weights are computed using similarity scores between tokens
- The dot product is used to measure how similar tokens i and j are
- Normalize these scores across all tokens using softmax
- This results in a probability distribution over all tokens j, indicating how much attention token i pays to each

$$\alpha_{ij} = softmax(\text{score}(x_i, x_j)) \forall j \leq i$$

Attention in Generative Models

This is a simplified view, and actual attention involves learned projection matrices

- In practice, each attention head uses three distinct learned weight matrices:
  - W^Q (query), W^K (key), and W^V (value)
- Each input vector x_i is projected into three new vectors:
  - Query: $q_i = x_i W^Q$
  - Key: $k_i = x_i W^K$
  - Value: $v_i = x_i W^V$
- The attention weights are calculated using the dot product between the current token's query q_i and all previous keys k_j (for j $\leq$ i)

- These scores are normalized via softmax to produce weights α_{ij}, which are used to compute a weighted sum of the value vectors:

$$a_i = \sum_{j \leq i} \alpha_{ij} \cdot v_j$$

- For a more detailed explanation, see pp. 5–7 in Ch. 9 of [JM]

Language Modeling Head

Once we have a vector representation for the last token

- This should be mapped to a probability distribution over $|V|$
- The task of the language modeling head is to take the output of the final transformer layer from the last position and use it to predict the upcoming word at the next position
- It takes the output of the last token at the last layers and produces a probability distribution over $|V|$
- Details can be found pp. 16–18 [JM]

Language Modeling Head

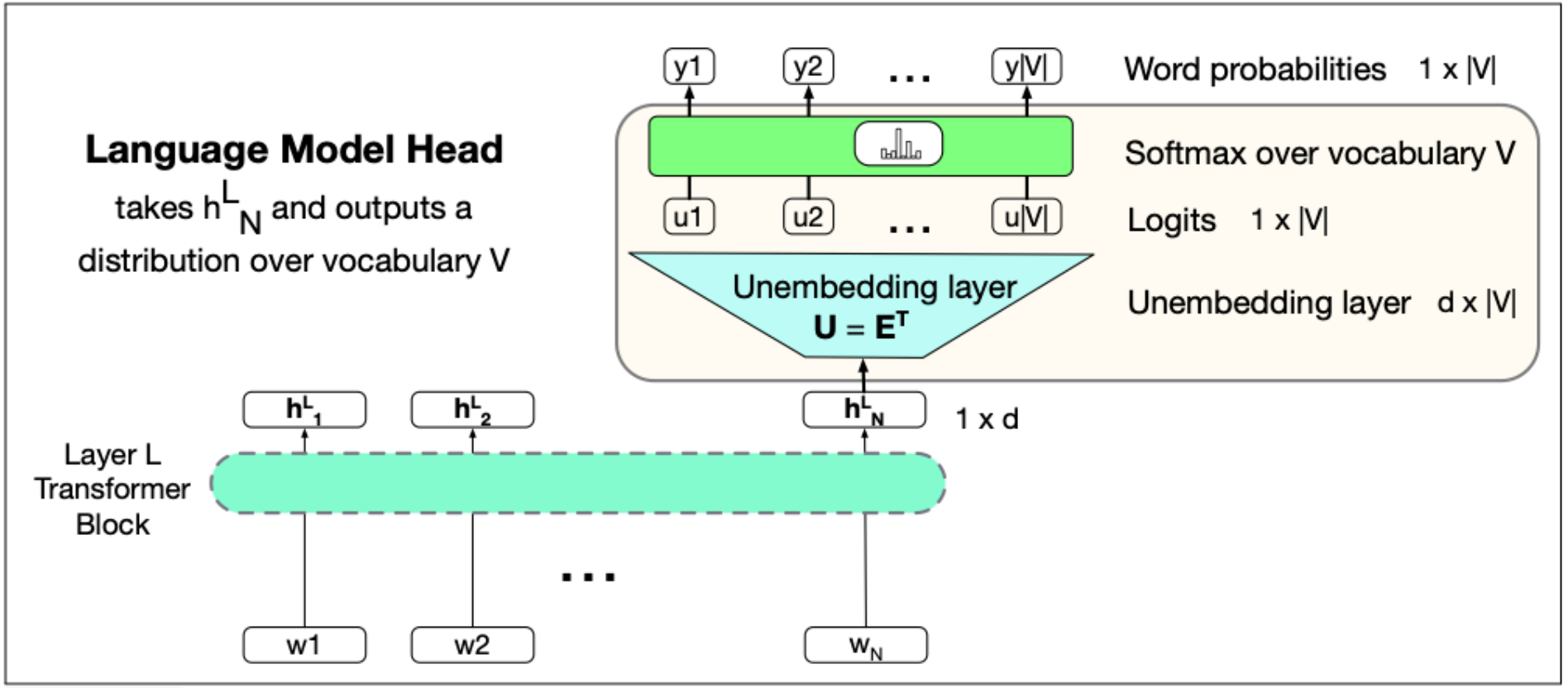

Figure 9.14 The language modeling head: the circuit at the top of a transformer that maps from the output embedding for token N from the last transformer layer ($\mathbf{h}_N^L$) to a probability distribution over words in the vocabulary V.

Decoding

The task of choosing a word to generate based on the model-generated probabilities

- Greedy decoding
  - Always selects the token with the highest probability at each step
    - Simple and fast, but can lead to bland outputs
    - Deterministic: given the same prompt, always produces the same output
- Sampling-based decoding
  - Selects the next token by sampling from the probability distribution
  - Allows for more diverse and creative text generation
  - Stochastic: same prompt may produce different outputs each time

Decoding

Varieties of sampling-based decoding

- Pure/random sampling
  - Sample directly from the probability distribution
  - *How can this go wrong?*
- Top-k sampling
  - We truncate the distribution to the most likely tokens, re-normalize to produce a legitimate probability (= adding up to 1), and then randomly sample form within these k words
  - When k=1, top-k sampling = greedy decoding
  - When $k > 1$, it leads to choosing a token that is not necessarily the most probable

Decoding

Varieties of sampling-based decoding (cont'd)

- Top-p sampling
  - Depending on the distribution, the most probable tokens from top-k sampling will only include a fraction in the probability distribution
  - Top-p sampling (a.k.a. nucleus sampling) keeps the top p percent of the probability mass
  - Likely more robust with various shapes of of the probability distribution
  - Higher p yields more randomness

Decoding

Varieties of sampling-based decoding (cont'd)

- Temperature sampling
  - Originates from thermodynamics (a system at a high temperature is flexible)
  - Instead of truncating the distribution, we reshape it
  - For more randomness, we make the distribution flatter
  - Higher temperature parameter, τ, indicates more randomness ($\tau = 0$: greedy decoding)

Generative LLMs for text classification

Promises

- No need for labeled training data (zero-shot/few-shot prompting)
- High performance in various tasks (classification, sentiment, ideology)
- Little programming or machine learning expertise required
- Multi-lingual and adaptable across domains

Generative LLMs for text classification

Pitfalls

- Transparency and replicability issues with closed models (e.g., GPT-4)
- Computational power (open-source models) or API usage fees (closed-source, proprietary models)
- Lack of validation framework: ↔ Traditional ML tools (e.g., train-test splits, CV) do not directly apply
- Small prompt changes can produce drastically different results

Generative LLMs for text classification

Recommended pipeline (Törnberg et al. 2024)

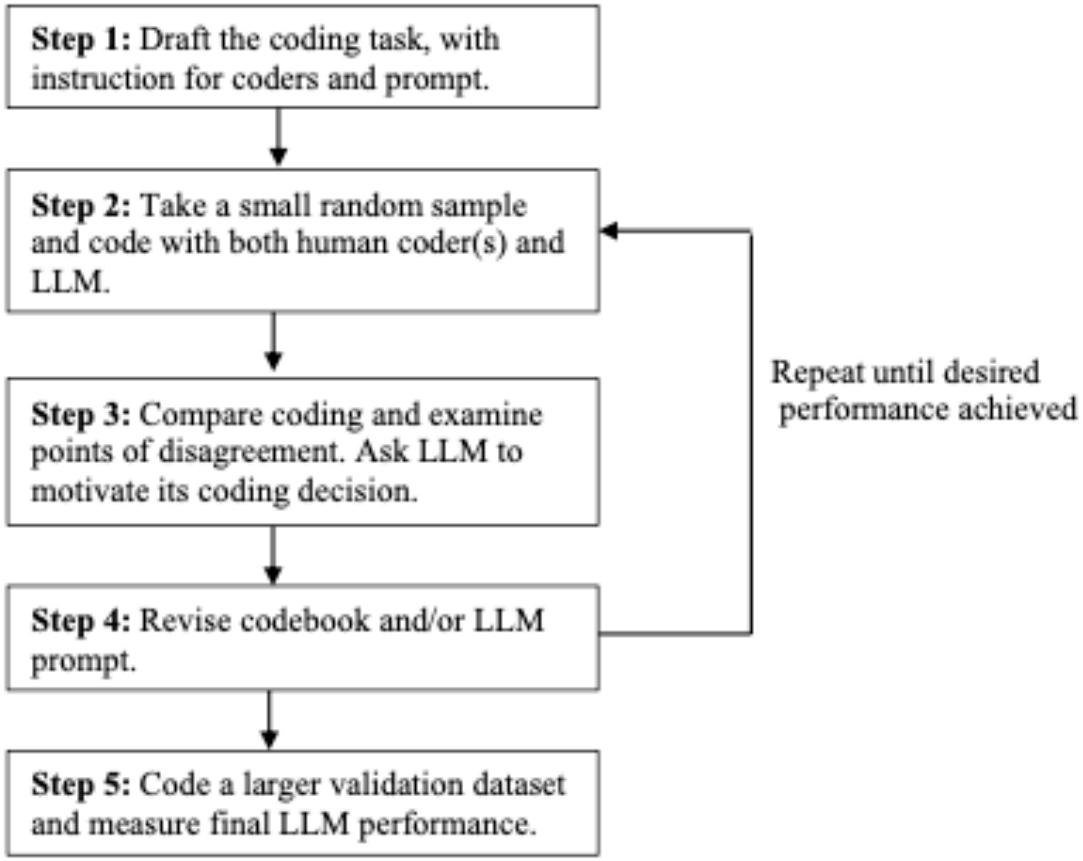

Figure 1: Example of a systematic coding procedure.

Summary

- Generative LLMs produce text token-by-token using attention and learned representations
- Decoding strategies shape the creativity and consistency of outputs
- These models enable powerful text classificadstion with minimal training data
- But challenges remain: transparency, replicability, validation, and prompt sensitivity
- A thoughtful pipeline can help researchers use LLMs responsibly and reproducibly

Guided Coding

- Introduction to text classification with generative models: link
- Stance detection with tweets: link