Special Topic: Advanced NLP

Class 5: The Transformer (Part 2)

Adam Faulkner

Februrary 24, 202

- Last week, we introduced most of the components of a transformer in the domain of language modeling: the transformer block, including multi-head attention, and the language modeling head
- The original transformer was structured as an encoder-decoder, but this architecture is not used in contemporary LLMs.
- We also introduced the dominant transformer architecture, the decoder-only architecture
- Today we'll cover **Encoder-only**, or **BERT**-style, Transformers
- This is an older architecture but still very widely used (we'll encounter BERT again when we discuss RAG systems in a later class)
- First, though, let's finish discussing sampling strategies for decoder-only models

The importance of sampling strategies

- The most common method for decoding in large language models is **sampling**
- Sampling from a model's distribution over words means to choose random words according to their probability assigned by the model
- A good sampling strategy will find the optimal tradeoff between **quality** and **diversity**
- Methods that emphasize the most probable words tend to produce generations that are rated by people as more accurate, more coherent, and more factual, but also more boring and more repetitive
- Methods that give a bit more weight to the middle-probability words tend to be more creative and more diverse, but less factual and more likely to be incoherent or otherwise low-quality

Greedy decoding

- **Greedy decoding**: Generate the most likely word given the context
- Make a choice that is locally optimal, whether or not it will turn out to have been the best choice with hindsight
- At each time step in generation, the output y_t is chosen by computing the probability for each possible output (every word in the vocabulary) and then choosing the highest probability word, (or the *argmax*):

$$\hat{w}_t = argmax_{w \in V} P(w|\mathbf{w}_{<t})$$

- The main problem with greedy decoding is that because the words it chooses are (by definition) extremely predictable, the resulting text is generic and often quite repetitive

Greedy decoding

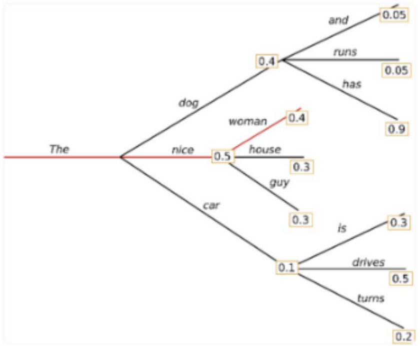

Figure 1: Generating the sentence *The nice woman* using greedy decoding

Beam search

- A popular alternative to greedy decoding is the **beam search** algorithm
- Instead of choosing the best token to generate at each timestep, we keep k possible tokens at each step
- k is called the **beam width**, on the metaphor of a flashlight beam that can be widened or narrowed

Beam search

The beam search algorithm

1. At the first decoding step, compute a softmax over the entire vocabulary, assigning a probability to each word
2. Select the k-best options from this softmax output. These initial k outputs are the **search frontier** and these k initial words are called **hypotheses**
3. At subsequent steps, each of the k best hypotheses is extended incrementally: a softmax is calculated over V to extend the hypothesis to every possible next token. Each of these $k \times V$ hypotheses is scored by $P(y_i|x, y < i)$: the product of the probability of the current word choice multiplied by the probability of the path that led to it.
4. Keep only the k best hypotheses, so there are never more than k hypotheses at the frontier of the search

Beam search

The beam search algorithm (continued)

5. This process continues until an EOS is generated indicating that a complete candidate output has been found
6. At this point, the completed hypothesis is removed from the frontier and the size of the beam is reduced by one
7. Repeat steps 1 - 6 until the beam size is 0

Beam search

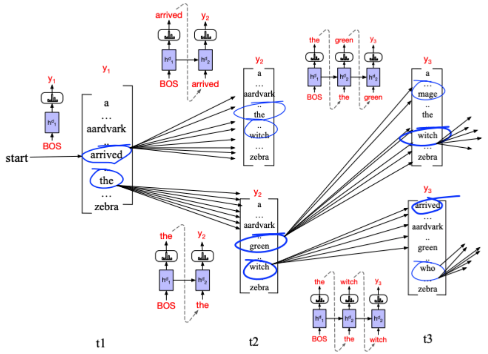

Figure 2: Beam search decoding of *the green witch*, beam size = 2. At each time step, we choose the k best hypotheses, form the V possible extensions of each, score those $k \times V$ hypotheses and choose the best $k = 2$ to continue.

Beam search

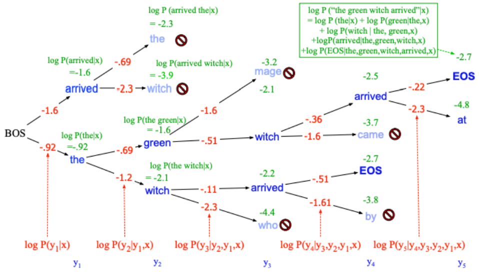
log P ("the green witch arrived"|x)
= log P (the|x) + log P(green|the,x)
+ log P(witch | the, green,x)
+logP(arrived|the,green,witch,x)
+log P(EOS|the,green,witch,arrived,x)
BOS
log P(arrived|x) =-1.6
log P(the|x) =-.92
log P (arrived the|x) = -2.3
log P(arrived witch|x) = -3.9
log P(the green|x) = -1.6
log P(the witch|x) = -2.1
arrived
the
green
witch
mage
came
EOS
at
by
who
log P(y1|x)
log P(y2|y1,x)
log P(y3|y2,y1,x)
log P(y4|y3,y2,y1,x)
log P(y5|y4,y3,y2,y1,x)
y1
y2
y3
y4
y5

Figure 3: Beam search decoding of *the green witch arrived*, beam size = 2.We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top k paths are extended to the next step.

Top-k sampling

- Basically a generalization of greedy decoding to k tokens (top-k sampling with $k = 1$ is just greedy decoding)
- The top-k sampling algorithm
  1. For each word in the vocabulary V, use the language model to compute the likelihood of this word given the context
  2. Sort the tokens by their likelihood; keep top k
  3. Renormalize the scores of the k words to be a legitimate probability distribution
  4. Randomly select a word from the new distribution

Top-p or nucleus sampling

- In **top-p sampling** or **nucleus sampling** , we keep the top p percent of the probability mass, rather than the top k words
- Sample tokens with the highest probability scores until a specified threshold p is reached
- Tends to generate text that is more varied and creative then greedy or top-k sampling

Temperature sampling

- Divide the logits by a temperature parameter τ before we normalize it by passing it through the softmax
- Higher values of τ lead to greater variability, more creative text
- Lower values of τ lead to boring, generic text
- Why this works:
  - Softmax pushes high values toward 1 and low values toward 0
  - The lower τ is, the larger the scores being passed to the softmax since dividing by a smaller fraction ≤ 1 results in making each score larger
  - When larger numbers are passed to a softmax the result is a distribution with increased probabilities of the most high-probability words and decreased probabilities of the low probability words

Temperature sampling

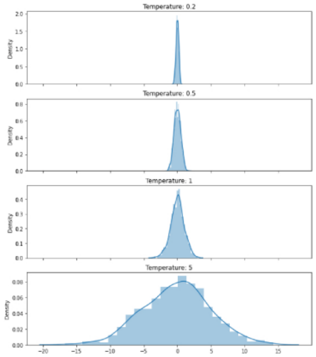

Figure 4: Temperature sampling: As the temperature nears 0, the probability of the most likely word approaches 1.

BERT

- Transformer-based models like ChatGPT are *decoder-only*; but what about *encoder-only* models?
- BERT (Bidirectional Encoder Representations from Transformers)
- BERT-style models are trained via the **masked language modeling** (MLM) task: a word in the middle of context window is masked and the model must predict the masked word given the words on *both* sides(hence the term *bidirectional*

Masked-language modeling

- MLMs are encoder-only since because they produce an encoding for each input token but don't generate text by decoding
- MLMs like BERT are not used for generation; instead they're generally used for semantic similarity tasks and classification (via finetuning, which we'll discuss in a bit)

Bidirectional training of MLMs

- So far we've only discussed next-word prediction, also called **causal**, transformer language models
- Instead of trying to predict the next word, an MLM learns to perform a fill-in-blank task, technically called the **cloze** task
- Instead of trying the predicting the next word given a sequence of previous words, we want predict a work given the surrounding context, e.g,

The ___ of Walden Pond is so beautifully...

Bidirectional training of MLMs

- Each token in the training corpus is used in one of three ways:
  1. It is replaced with the special vocabulary token named `[MASK]`.
  2. It is replaced with another token from the vocabulary, randomly sampled based on token unigram probabilities.
  3. It is left unchanged.
- In the original BERT paper 80% are replaced with `[MASK]`, 10% are replaced with randomly selected tokens, and the remaining 10% are left unchanged.

Bidirectional training of MLMs

- The MLM tries to predict the original inputs for each of the masked tokens using a bidirectional encoder (see below)
- As with the models we've discussed, the cross-entropy loss from these predictions drives the training process for all the parameters in the model

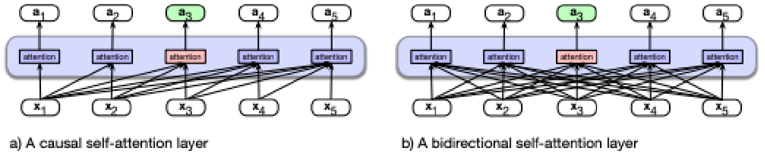
a_1 a_2 a_3 a_4 a_5
attention
x_1 x_2 x_3 x_4 x_5
a) A causal self-attention layer
b) A bidirectional self-attention layer

Figure 5: In the causal model (a) the attention computation for token 3 only used only information seen earlier in the context; in the bidirectional model (b) the model attends to all inputs on both left and right.

Bidirectional training of MLMs

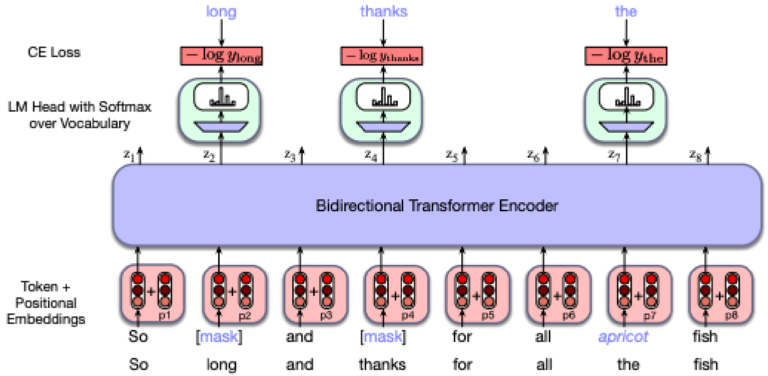
long
thanks
the
CE Loss
$-\log y_{\text{long}}$
$-\log y_{\text{thanks}}$
$-\log y_{\text{the}}$
LM Head with Softmax over Vocabulary
z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8
Bidirectional Transformer Encoder
Token + Positional Embeddings
p1 p2 p3 p4 p5 p6 p7 p8
So [mask] and [mask] for all apricot fish
So long and thanks for all the fish

Figure 6: Example of token replacement and bidirectional encoding: *long* and *thanks* are masked and *the* is replaced with randomly selected token *apricot*

Next sentence prediction

- MLM has the goal of producing effective word-level representations
- But many tasks in NLP involve the relationship between pairs of sentences.
- To learn the kind of knowledge required for these BERT includes a *second* learning objective: Next Sentence Prediction (NSP)

Next sentence prediction

- Several classic NLP tasks can be solved by applying the the sentence-level representations output by the NSP task
- **Paraphrase Detection**:

  Does *The company's stock dropped* = *The organization's stock slumped*?

- **Logical entailment**:

  Does *Fido is a mammal* logically entail *Fido is a dog*?

- **Discourse coherence**:

  Is the ordering of these two sentences coherent?

  Once upon a time there was a girl named Goldilocks

  2 + 2 is 4

Next sentence prediction

- NSP: Given a pair of sentences, predict whether each pair consists of an actual pair of adjacent sentences from the training corpus or a pair of unrelated sentences.
- The NSP loss is based on how well the model can distinguish true pairs from random pairs. In BERT, 50% of the training pairs consisted of positive pairs and in the other 50% the second sentence of a pair was randomly selected from elsewhere in the corpus.
- BERT introduces two special tokens to the input representation
    - The token `[CLS]` is prepended to the input sentence pair
    - The token `[SEP]` is placed between the sentences and after the final token of the second sentence
- During training, the output vector h^L_{CLS} from the final layer associated with the `[CLS]` token represents the next sentence prediction: *isNext* or *notNext*

Bidirectional training of MLMs

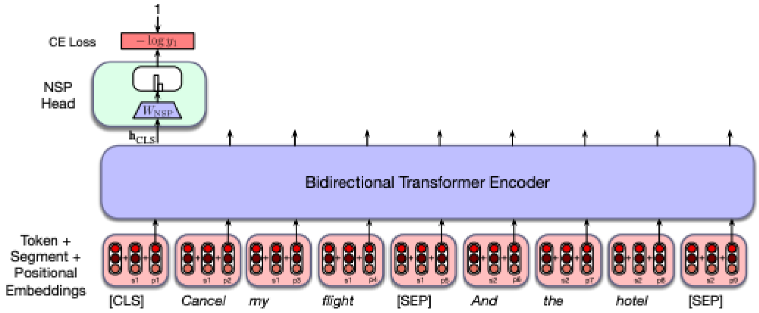

Figure 7: During training, the output vector h^L_{CLS} from the final layer associated with the `[CLS]` token represents the next sentence prediction: *isNext* or *notNext*

Finetuning (for BERT)

- The original excitement generated by BERT lay in its enablement of **transfer learning** via a technique called **finetuning**
- Rather than training task-specific models, a single model (BERT) can be used to solve multiple downstream tasks sequence classification, sentence-pair classification, and sequence labeling

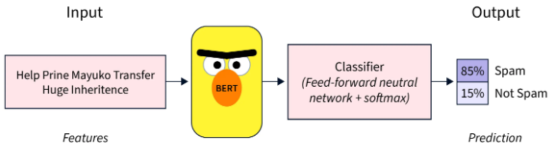
Input
Output
Help Prine Mayuko Transfer Huge Inheritence
BERT
Classifier (Feed-forward neutral network + softmax)
85% Spam
15% Not Spam
Features
Prediction

Figure 8: BERT can be finetuned for many different tasks, including spam detection.

Finetuning (for BERT)

- BERT-based classification is made possible with the addition of a unique token to the vocabulary called `[CLS]` which is prepended to the start of all input sequences, both during pretraining and encoding.
- The output vector in the final layer of the model for the `[CLS]` input represents the entire input sequence and serves as the input to a **classifier head**, a logistic regression or neural network classifier that makes the relevant decision.

Finetuning (for BERT)

Suppose we have a three-class (*positive, negative, neutral*) classification task and labeled data for this task, e.g. a dataset of tagged product reviews. Finetuning BERT on this task involves the following steps

1. Represent the first 512 tokens (BERT's token limit) of each product review as single vector by prepending `[CLS]` (the same token used in BERT's pretraining) to the start of the input
2. The output vector in the final layer of the model for the `[CLS]` input represents the entire input sequence and serves as the input to a classifier head, a logistic regression or neural network classifier that makes the relevant decision
3. Finetuning this classification head involves learning a set of weights W_c to map the output vector for the `[CLS]` token to a set of scores over the possible sentiment classes
4. To classify a review, we pass the input text through BERT to generate `[CLS]` multiply it by W_c and pass the resulting vector through a softmax.
5.

What is Domain Adaptation?

- **The Problem:** Machine Learning models assume that **training** data and **test** data come from the same distribution.
- **The Reality:** NLP models often encounter data in production that differs from their training set.
- **Definition:** Domain Adaptation is a sub-field of Transfer Learning that focuses on adapting a model trained on a **Source Domain** ($\mathscr{D}_S$) to perform well on a **Target Domain** ($\mathscr{D}_T$).

Examples of Different Domains

In NLP, a "domain" can be defined by several factors:

Domain Factor	**Source** ($\mathscr{D}_S$)	**Target** ($\mathscr{D}_T$)
Genre	Newswire (WSJ)	Social Media (Twitter)
Topic	Financial Reports	Medical Journals
Style	Formal Essays	Casual Dialogue
Sentiment	Electronics Reviews	Hotel Reviews

Specific Challenge: Vocabulary & Semantics

- **Lexical Shift:** Words like "check" mean different things in Banking vs. Sports.
- **Out-of-Vocabulary (OOV):** Technical domains (Legal, Bio-medical) contain specialized terminology not found in Wikipedia or Common Crawl.
- **Feature Divergence:** In sentiment analysis:
  - *Books:* "Predictable" is **negative**.
  - *Calculators:* "Predictable" is **positive**.

Common Strategies

1. **Supervised:** Fine-tuning on a small amount of labeled target data.
2. **Unsupervised (DA):**
   - **DANN:** Domain-Adversarial Neural Networks (making features domain-invariant).
   - **EasyAdapt:** Feature augmentation (Source-specific, Target-specific, and General features).
3. **Pre-training:** Continued Masked Language Modeling (MLM) on the target corpus (e.g., BioBERT).

Finetuning (for BERT): Adapting BERT to particular domains using MLM fine-tuning

- In addition to task-based finetuning using `[CLS]` we can perform another kind of finetuning that adjusts BERT’s weights to reflect a particular domain: **MLM fine-tuning**
- BERT is trained on a vast corpus incorporating hundreds of different subjects. During MLM pre-training it has domain-general weights. But what if we have a specific domain (movies, sports, medicine, finance) that we want to adapt the model to? BERT’s weights can be adjusted to orient toward this domain by finetuning on the domain’s corpus– the learning task is the same: predict ‘[MASK]‘ using the surrounding words.

The accompanying notebook demonstrates MLM fine-tuning for the *movies* domain

Next class: March 10

March 3: No class

Assignment 3 Due

Classification tasks

- Sentiment Analysis
- Argumentation Mining
- Entailment
- Intent Detection and Slot Filling
- Coreference resolution

Reading

- Jurafsky & Martin Chapter 4:Logistic Regression & Text Classification
- Deep Learning in Stance Detection: A Survey
- Argument Mining: A Survey
- Jurafsky & Martin Chapter 23: Coreference Resolution and Entity Linking