Module 5---chapter 3, 8, 9

Text Generation---The Challenge With Generating Coherent Text

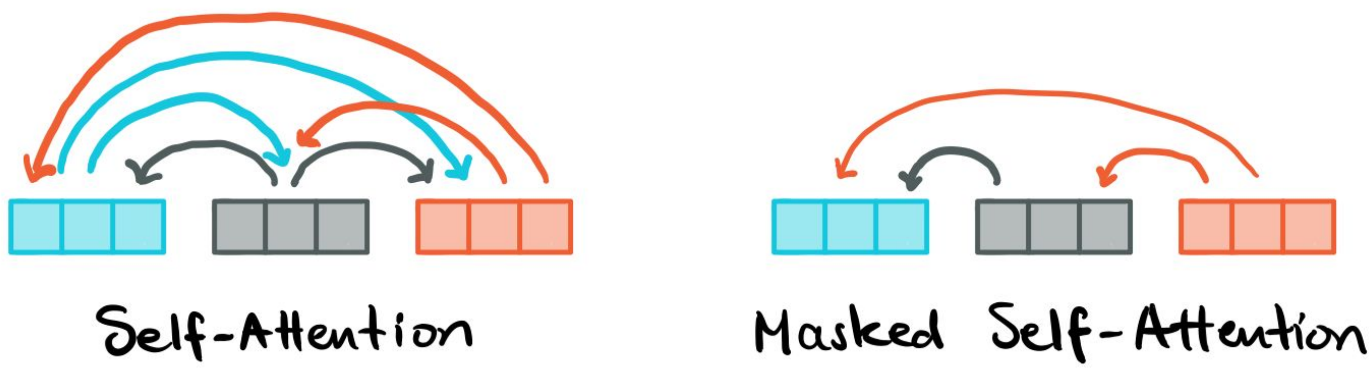
- converting the model's probabilistic output to text requires a *decoding method* which introduces a few challenges that are unique to text generation:
- The decoding is done *iteratively* and thus involves significantly more compute than simply passing inputs once through the forward pass of a model.
- The *quality* and *diversity* of the generated text depends on the choice of decoding method and associated hyperparameters.

how this decoding process works:

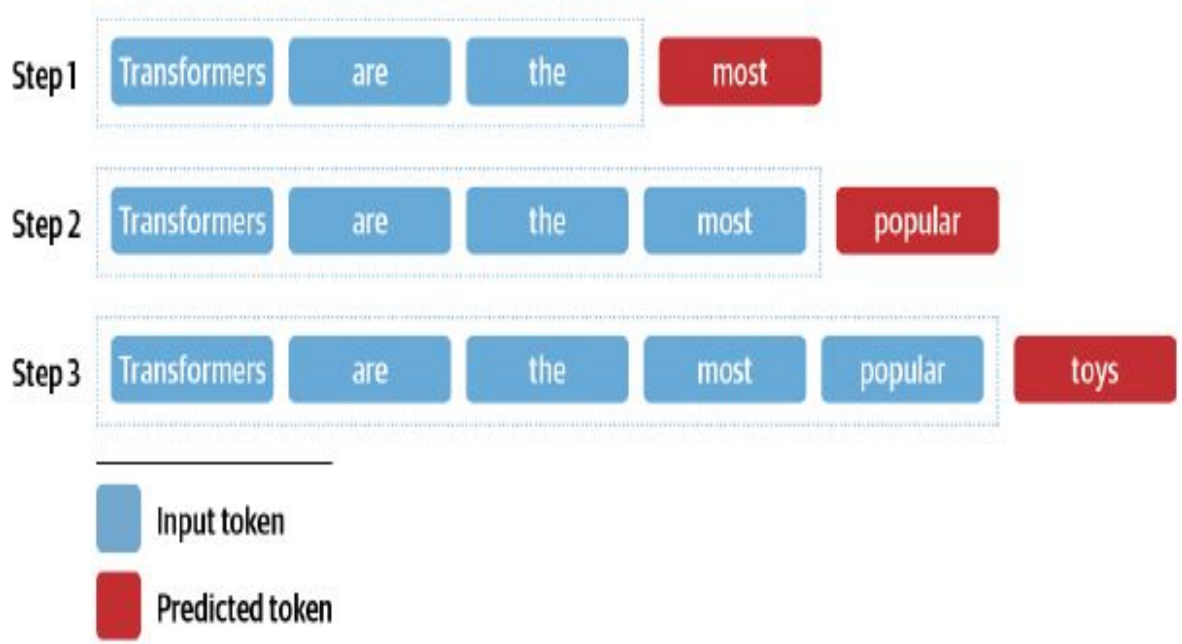
$$P\left(y_1, \ldots, y_t | \mathbf{x}\right) = \prod_{t=1}^{N} P\left(y_t | y_{<t}, \mathbf{x}\right),$$

where $y_{<t}$ is a shorthand notation for the sequence $y_1, \ldots, y_{t-1}$.

Difference between the self-attention mechanisms of BERT (left) and GPT-2 (right) for three token embeddings. In the BERT case, each token embedding can attend to all other embeddings. In the GPT-2 case, token embeddings can only attend to previous embeddings in the sequence.



Generating text from an input sequence by adding a new word to the input at each step.



- The goal of most decoding methods is to search for the most likely overall sequence by picking a ˆy such that

$$\widehat{\mathbf{y}} = \underset{y_t}{\operatorname{argmax}} \; P\left(y_t | y_{<t}, \mathbf{x}\right).$$

- Since there does not exist an algorithm that can find the optimal decoded sequence in polynomial time, we rely on approximations instead.

1. Greedy Search Decoding

- The simplest decoding method to get discrete tokens from a model's continuous output is to greedily select the token with the highest probability at each timestep:

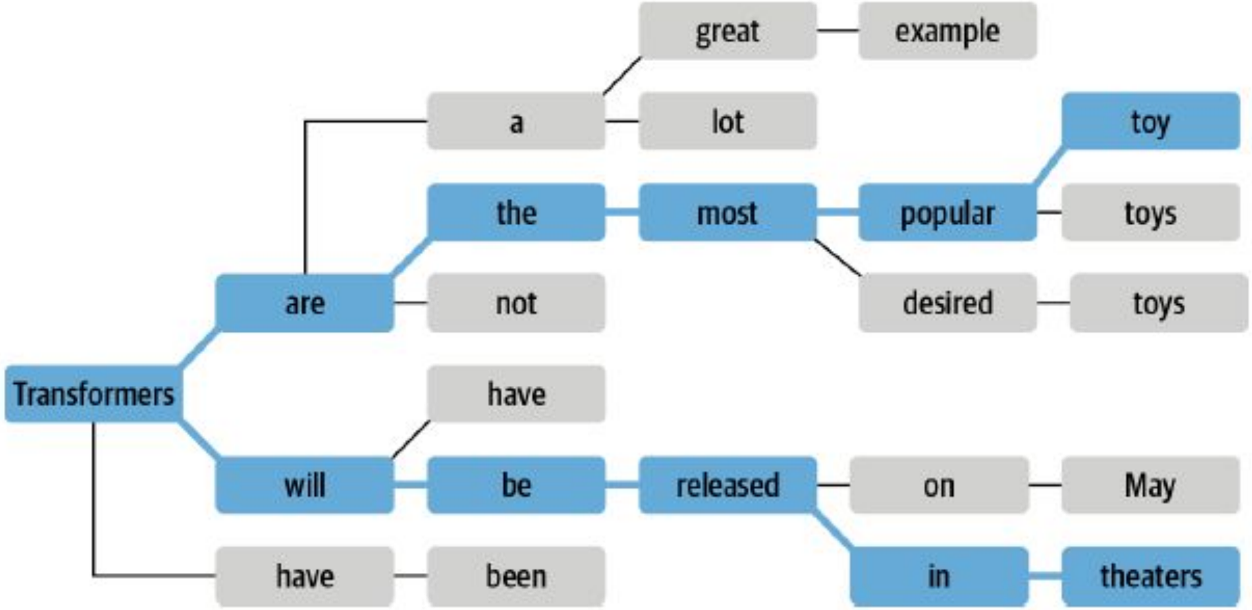
$$\hat{y}_t = \underset{y_t}{\operatorname{argmax}} \; P\left(y_t | y_{<t}, \mathbf{x}\right).$$

Input	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5
Transformers are the	most (8.53%)	only (4.96%)	best (4.65%)	Transformers (4.37%)	ultimate (2.16%)
Transformers are the most	popular (16.78%)	powerful (5.37%)	common (4.96%)	famous (3.72%)	successful (3.20%)
Transformers are the most popular	toy (10.63%)	toys (7.23%)	Transformers (6.60%)	of (5.46%)	and (3.76%)
Transformers are the most popular toy	line (34.38%)	in (18.20%)	of (11.71%)	brand (6.10%)	line (2.69%)
Transformers are the most popular toy line	in (46.28%)	of (15.09%)	, (4.94%)	on (4.40%)	ever (2.72%)
Transformers are the most popular toy line in	the (65.99%)	history (12.42%)	America (6.91%)	Japan (2.44%)	North (1.40%)
Transformers are the most popular toy line in the	world (69.26%)	United (4.55%)	history (4.29%)	US (4.23%)	U (2.30%)
Transformers are the most popular toy line in the world	, (39.73%)	. (30.64%)	and (9.87%)	with (2.32%)	today (1.74%)

- main drawbacks with greedy search decoding: it tends to produce repetitive output sequences, which is certainly undesirable in a news article.
- This is a common problem with greedy search algorithms which can fail to give you the optimal solution; in the context of decoding, it can miss word sequences whose overall probability is higher just because high probability words happen to be preceded by low probability ones.

2. Beam search decoding

- keeps track of the top-b most probable next-tokens, where b is referred to as the number of *beams* or *partial hypotheses*.
- The next set of beams are chosen by considering all possible next-token extensions of the existing set and selecting the b most likely extensions.
- The process is repeated until we each the maximum length or an EOS token, and the most likely sequence is selected by ranking the b beams according to their log-probabilities.



$$\log P(y_1, \ldots y_t | \mathbf{x}) = \sum_{t=1}^{N} \log P(y_t | y_{<t}, \mathbf{x}).$$

- we get a better log-probability (higher is better) with beam search than we did with simple greedy decoding.
- However we can see that beam search also suffers from repetitive text.
- More number of beams increases precision but reduces the performance (increased processing time and power)

- **Top-K Sampling**
- Top-K sampling is used to ensure that the less probable words should not have any chance at all. Only top K probable tokens should be considered for a generation.

- **Nucleus Sampling/ Top-p sampling**
- Nucleus sampling is similar to Top-K sampling. Instead of focusing on Top-K words, nucleus sampling focuses on the smallest possible sets of Top-V words such that the sum of their probability is ≥ p.

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p.$$

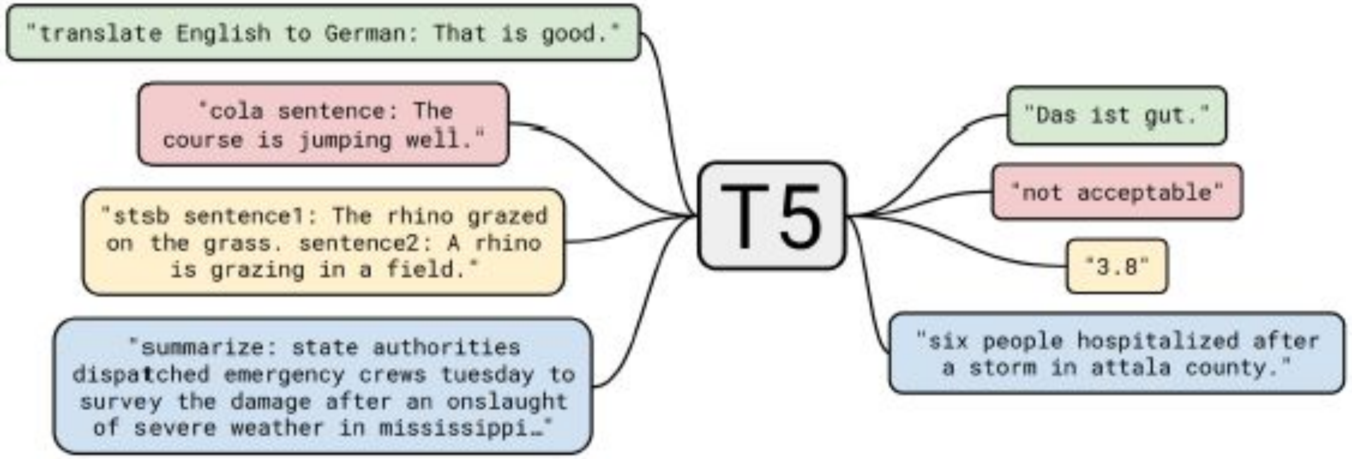
- **When to Use Top-K Sampling**
- Top-K sampling is often used when you want a balance between randomness and relevance in the generated text. It allows the model to explore a bit, potentially generating more creative and diverse text while still being more coherent than random sampling.
- **Limitations and Considerations**
- Hyperparameter Tuning: The choice of K can significantly influence the results. A smaller K will make the output more focused but less creative, while a larger K will make the output more diverse but potentially less relevant.
- Not Adaptive: The value of K remains constant, meaning the method isn't adaptive to the context of the text being generated. This limitation has led to the development of more advanced sampling techniques like nucleus sampling.

- When to Use Top-P Sampling
- Top-P sampling is particularly useful when you want more adaptive and context-sensitive text generation. Unlike Top-K, which has a fixed number of candidates, Top-P allows for a variable number of candidates based on the context, making it more flexible.

- Limitations and Considerations
- Computational Cost: The sorting operation increases the computational cost slightly compared to Top-K sampling.
- Hyperparameter Sensitivity: The choice of P can significantly influence the generated text. A smaller P will make the text more random, while a larger P will make it more deterministic.
- Top-P sampling provides an adaptive method for balancing the trade-off between diversity and informativeness in generated text. It has gained popularity in several NLP applications, from automated customer service to creative writing aids.

Summarization

- T5 (Text-To-Text Transfer Transformer) is a variant of the transformer architecture designed for text-to-text tasks.
- T5 formulates text summarization as a text-to-text task, where both input and output are text sequences.
- It uses a unified architecture for various NLP tasks, including summarization, by conditioning on a textual task description.



- PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization) is a transformer-based model specifically designed for abstractive text summarization.
- PEGASUS uses a gap-sentence generation objective during pre-training, which encourages the model to generate summaries.
- It leverages a mixture of pre-training tasks, including gap-sentence generation, sentence shuffling, and document permutation.

- GPT-2 (Generative Pre-trained Transformer 2) is a large-scale unsupervised language model based on the transformer architecture.
- GPT-2 is primarily designed for generating coherent and contextually relevant text given a prompt.
- While not explicitly designed for summarization, GPT-2 can be used for abstractive summarization by conditioning the generation process on a context or input document.

- BART (Bidirectional and Auto-Regressive Transformers) is a sequence-to-sequence model based on the transformer architecture.
- BART uses a combination of autoencoder and autoregressive pre-training objectives, making it suitable for various natural language processing tasks, including text summarization.
- It utilizes a pre-training method called denoising autoencoder, where it learns to reconstruct corrupted text.

- BART and PEGASUS are specifically designed for abstractive summarization tasks and often outperform other models in this domain.
- T5's flexibility makes it suitable for various NLP tasks, including summarization, although it may not achieve the same level of performance as specialized models like PEGASUS or BART.
- GPT-2, while not specialized for summarization, can still generate summaries, especially when fine-tuned on summarization datasets. However, it may lack the precision and coherence of models explicitly designed for summarization tasks.