Language Modeling

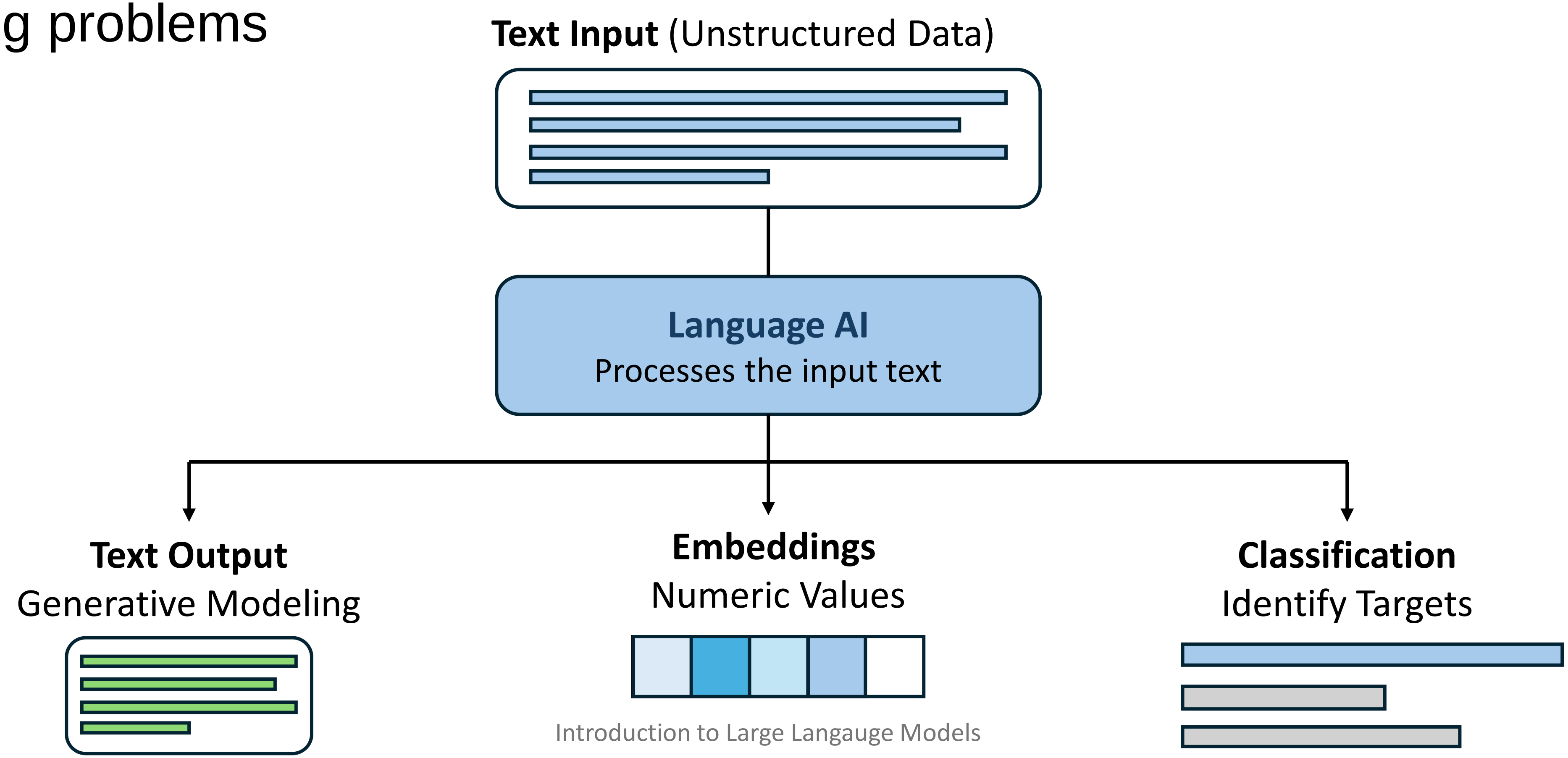
CS 335: Introduction to Large Language Models

Abdul Samad, Fisal Alvi Habib University

Contents

Language AI (NLP)

- Language AI refers to a subfield of AI that focuses on developing technologies capable of understanding, processing, and generating human language.
- Language AI can often be used interchangeably with natural language processing (NLP) with continued success of machine learning methods in tackling language processing problems



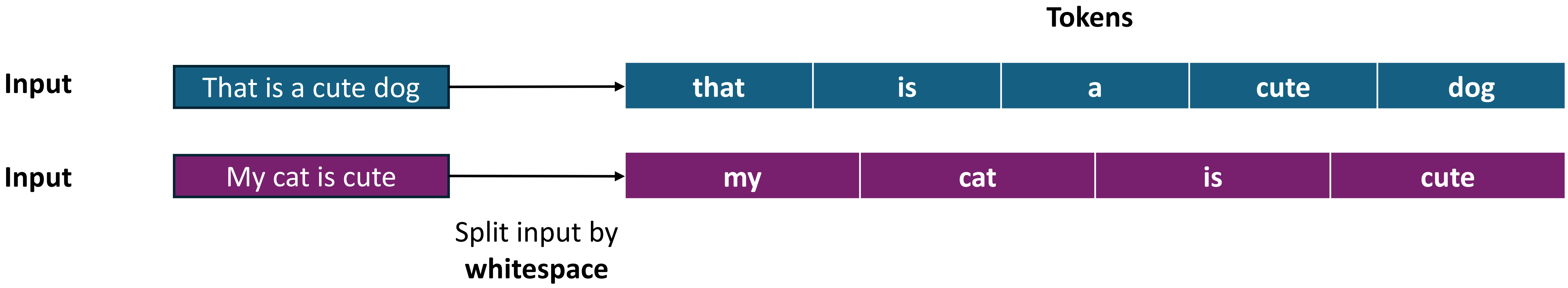
Language AI

- RECALL
  - Supervised Learning: Given a collection of labeled examples (each x paired with y) learn a function from x to y.

  - Language tasks commonly tackled in supervised setting:
    - **Sentiment analysis:** map a product review to a sentiment label (positive or negative)
    - **Question-answering:** given a question about a document, provide the location of the answer within the document
    - **Textual entailment:** given two sentences identify whether the first sentence entails or contradicts the second one
    - **Machine translation:** given a sentence in a source language, produce a translation of that sentence in a target language.

Representation of Language

- Bag of Words
  - Our history of Language AI starts with a technique called bag-of-words, a method for representing unstructured text.
  - Bag-of-words works as follows: let's assume that we have two sentences for which we want to create numerical representations. The first step of the bag-of-words model is tokenization, the process of splitting up the sentences into individual words or subwords (tokens).
  - The most common method for tokenization is by splitting on a whitespace to create individual words.



Representation of Language

- Bag of Words
  - After tokenization, we combine all unique words from each sentence to create a **vocabulary** that we can use to represent the sentences.

that	is	a	cute	dog	my	cat	is	cute

Vocabulary (V)
that
is
a
cute
dog
my
cat

Vocabulary size (s)

Representation of Language

- Bag of Words
  - **One-hot Encoding:** Each token in the vocabulary is assigned an index. The corresponding numerical representation of the token is a vector of dimension V with a value of 1 at the position corresponding to the token's index in the vocabulary, while all other positions are set to 0.

Vocabulary size (s)

	Vocabulary (V)	
0	that	<**1**, 0, 0, 0, 0, 0, 0>
1	is	
2	a	<0, 0, **1**, 0, 0, 0, 0>
3	cute	<0, 0, 0, **1**, 0, 0, 0>
4	dog	
5	my	
6	cat	<0, 0, 0, 0, 0, 0, **1**>

Representation of Language

- Bag of Words
  - Using our vocabulary, we simply count how often a word in each sentence appears, quite literally creating a bag of words

Input My cat is cute → my | cat | is | cute

my: <0, 0, 0, 0, 0, **1**, 0>

cat: <0, 0, 0, 0, 0, 0, **1**>

is: <0, **1**, 0, 0, 0, 0, 0>

cute: <0, 0, 0, **1**, 0, 0, 0>

A bag of words is created by counting individual words.

Vector representation

that	is	a	cute	dog	my	cat
0	1	0	1	0	1	1

<0, 0, 0, 0, 0, **1**, 0> + <0, 0, 0, 0, 0, 0, **1**> + <0, **1**, 0, 0, 0, 0, 0> + <0, 0, 0, **1**, 0, 0, 0>

= <0, **1**, 0, **1**, 0, **1**, **1**>

Representation of Language

- Word Embedding
  - Given some text, create a representation of that text (usually real-valued vectors) that capture its linguistic properties (syntax, semantics)

Word	dim0	dim1	dim2	dim3
today	0.35	-1.3	2.2	0.003
cat	-3.1	-1.7	1.1	-0.56
sleep	0.55	3.0	2.4	-1.2
watch	-0.09	0.8	-1.8	2.9

vector representation of "today"

  - Word2Vec, GloVe, FastText are popular algorithms used to generate word embeddings.

Representation of Language

- Word Embedding

  - Size of Embeddings:

    Smaller embeddings (e.g., 50-100 dimensions): Capture basic word relationships, faster to train, suitable for simple tasks.

    Larger embeddings (e.g., 300-1000+ dimensions): Capture more detailed semantic features, better for complex language understanding.

  - Syntax Level:

    Captures grammatical relationships and word usage patterns.

    Example: "run" and "runs" may have similar embeddings based on their syntactic role.

  - Semantics Level:

    Captures the meaning and context of words.

    Example: "king" and "queen" having similar embeddings due to related meanings.

Self Supervised Learning

Given a collection of just text (no extra labels), create labels out of the text and use them for representation learning or generating text.

- **Language Modelling:** given the beginning of a sentence or document, predict the next word
- **Masked Language Modelling:** given an entire document with some words or spans masked out, predict the missing words

Probabilistic Language Modeling

- The classic definition of a language model (LM) is **a probability distribution over sequences of tokens**. Suppose we have a **vocabulary** $\boldsymbol{V}$ of a set of tokens. A language model p assigns each sequence of tokens $x_1, \dots, x_L \in V$ a probability (a number between 0 and 1):

$$p(x_1, \dots, x_L).$$

- The probability intuitively tells us how "good" a sequence of tokens is. For example, if the vocabulary is $V = \{\mathrm{ate}, \mathrm{ball}, \mathrm{cheese}, \mathrm{mouse}, \mathrm{the}\}$, the language model might assign:

$$p(\mathrm{the}, \mathrm{mouse}, \mathrm{ate}, \mathrm{the}, \mathrm{cheese}) = 0.02,$$

$$p(\mathrm{the}, \mathrm{cheese}, \mathrm{ate}, \mathrm{the}, \mathrm{mouse}) = 0.01,$$

$$p(\mathrm{mouse}, \mathrm{the}, \mathrm{the}, \mathrm{cheese}, \mathrm{ate}) = 0.0001.$$

Probabilistic Language Modeling

- Goal: Compute the probability of a sentence or a sequence of tokens

$$p(x_{1:L}) = p(x_1, x_2, x_3, \dots, x_L)$$

- Related Task: Probability of an upcoming word

$$p(x_4 | x_1, x_2, x_3)$$

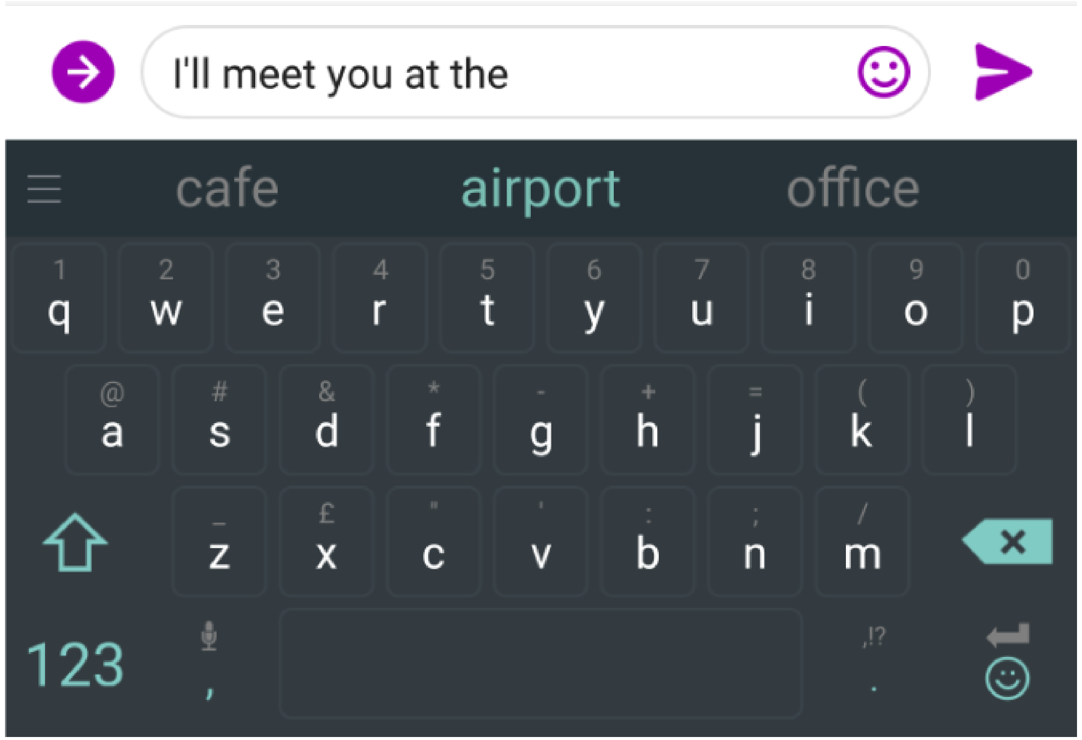
- A model that computes either of these

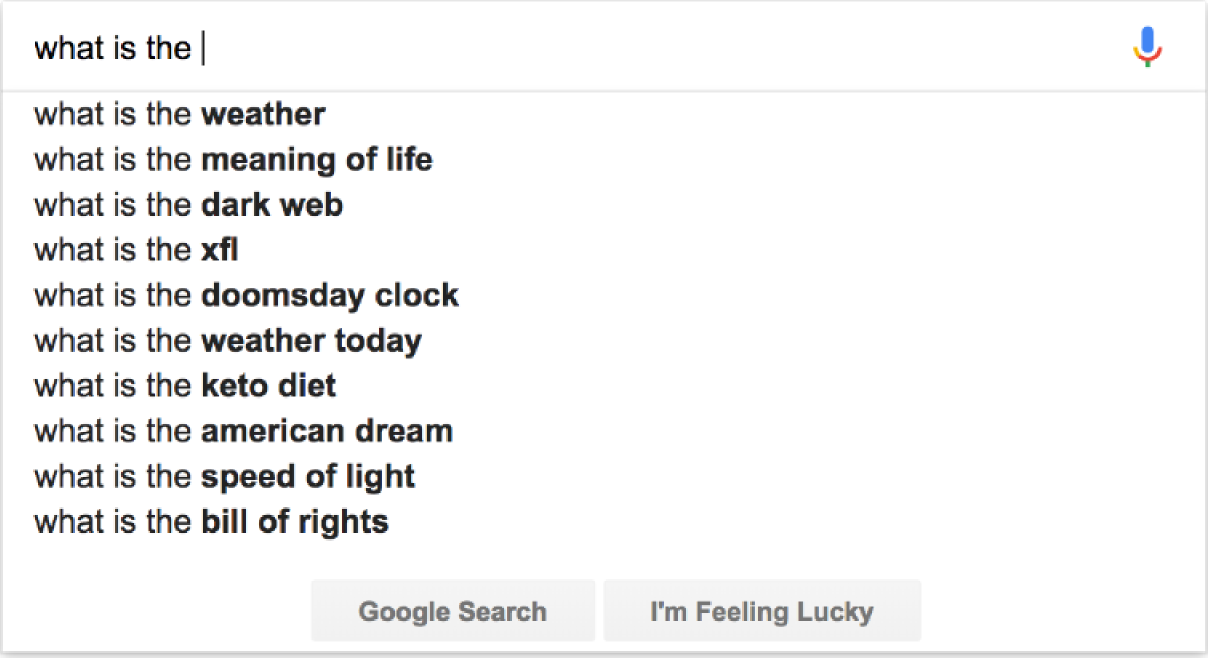
$p(x_{1:L})$ or $p(x_4 | x_1, x_2, x_3)$ is called a Language Model (LM)

Probabilistic Language Modeling

- Language models assign a probability to a piece of text

You use Language Models every day!





Probabilistic Language Modeling

- How to compute $p(x_{1:L})$

$$p(\text{the}, \text{mouse}, \text{ate}, \text{the}, \text{cheese})$$

Probabilistic Language Modeling

- Let's rely on the chain rule of probability

$$p(x_{1:L}) = p(x_1)p(x_2|x_1)p(x_3|x_1,x_2)\cdots p(x_L|x_{1:L-1}) = \prod_{i=1}^{L} p(x_i|\overbrace{x_{1:i-1}}^{\text{Prefix sequence}})$$

- For example:

$$\begin{aligned} p(\text{the}, \text{mouse}, \text{ate}, \text{the}, \text{cheese}) = {} & p(\text{the}) \\ & p(\text{mouse} \mid \text{the}) \\ & p(\text{ate} \mid \text{the}, \text{mouse}) \\ & p(\text{the} \mid \text{the}, \text{mouse}, \text{ate}) \\ & p(\text{cheese} \mid \text{the}, \text{mouse}, \text{ate}, \text{the}). \end{aligned}$$

- In particular, $p(x_i \mid x_{1:i-1})$ is a **conditional probability distribution** of the next token x_i given the previous tokens $x_{1:i-1}$.

Probabilistic Language Modeling

- How to estimate these probabilities
  - Count?

$$p(\text{cheese } | \text{the}, \text{mouse}, \text{ate}, \text{the}) = \frac{\text{count}(\text{the mouse ate the cheese})}{\text{count}(\text{the mouse ate the})}$$

  - NO!! We will never see enough data for accurately estimating these

Probabilistic Language Modeling

- How to estimate these probabilities
  - Count?

$$p(\text{cheese }|\text{the}, \text{mouse}, \text{ate}, \text{the}) = \frac{\text{count(the mouse ate the cheese)}}{\text{count(the mouse ate the)}}$$

  - NO!! We will never see enough data for accurately estimating these
  - Markov Assumption

$$p(\text{cheese }|\text{the}, \text{mouse}, \text{ate}, \text{the}) \approx p(\text{cheese}|\text{the})$$

$$p(\text{cheese }|\text{the}, \text{mouse}, \text{ate}, \text{the}) \approx p(\text{cheese}|\text{ate the})$$

Probabilistic Language Modeling

- How to estimate these probabilities

$$p(x_{1:L}) = \prod_{i=1}^{L} p(x_i | x_{i-k:i-1}) = \prod_{i=1}^{L} p(x_i | \overbrace{x_{i-k} \cdots x_{i-1}}^{\text{Instead of starting from 1 we start from some } i-k})$$

- In other words, we approximate each component in the product

$$(\text{cheese } | \text{the}, \text{mouse}, \text{ate}, \text{the}) \approx p(\text{cheese} | \text{mouse ate the})$$

N-gram Models

- In an **n-gram model**, the prediction of a token x_i only depends on the last $i - k$ words where $k = n - 1$:

$$p(x_i \mid x_{1:i-1}) = p(x_i \mid x_{i-(n-1):i-1}).$$

- For example, a trigram ($n = 3$) model would define:

$$p(\text{cheese} \mid \text{the}, \text{mouse}, \text{ate}, \text{the}) = p(\text{cheese} \mid \text{ate}, \text{the}).$$

- These probabilities are computed based on the number of times various n-grams (e.g., ate the mouse and ate the cheese) occur in a large corpus of text

N-gram Models

- Example: Estimating Bi-gram probabilities

$$p(x_i|x_{i-1}) = \frac{p(x_{i-1}, x_i)}{p(x_{i-1})}$$

- Text:

“<s>I am Sam</s><s>Sam I am</s><s>I do not like green eggs and toast</s>”

$$p(\text{I}|\text{<s>}) = \frac{2}{3} \qquad p(\text{Sam}|\text{<s>}) = \frac{1}{3} \qquad p(\text{am}|\text{I}) = \frac{2}{3}$$

$$p(\text{</s>}|\text{Sam}) = \frac{1}{2} \qquad p(\text{Sam}|\text{am}) = \frac{1}{2} \qquad p(\text{do}|\text{I}) = \frac{1}{3}$$

N-gram Models

- Fitting n-gram models to data is extremely **computationally cheap** and scalable. As a result, n-gram models were trained on massive amount of text. For example, Brants et al. (2007) trained a 5-gram model on 2 trilling tokens for machine translation. In comparison GPT-3 was trained on only 300 billion tokens. However, an n-gram model was fundamentally limited. Imagine the prefix:

Stanford has a new course on large language models. It will be taught by ____

- If n is too small, then the model will be incapable of capturing long-range dependencies, and the next word will not be able to depend on **Stanford**. However, if n is too big, it will be *statistically infeasible* to get good estimates of the probabilities (almost all reasonable sequences show up 0 times even in “huge” corpora):

Count(Stanford, has, a, new, course, on, large, language, models) = 0

Generation

- As defined, a large language model p takes a sequence and returns a probability to assess its goodness. We can also generate a sequence given a language model.
- to generate an entire sequence $x_{1:L}$ from a language model p, we sample one token at a time given the tokens generated so far:

$$\text{for } i = 1, \ldots, L:$$
$$x_i \sim p(x_i \mid x_{1:i-1})^{1/T},$$

  where $T \geq 0$ is a **temperature** parameter that controls how much randomness we want from the language model:

    - $T = 0$: deterministically choose the most probable token x_i at each position i
    - $T = 1$: sample “normally” from the pure language model
    - $T = \infty$: sample from a uniform distribution over the entire vocabulary V

Generation

- Conditional generation

we can perform conditional generation by specifying some prefix sequence $x_{1:L}$ (called a **prompt**) and sampling the rest $x_{i+1:L}$ (called the **completion**). For example, generating with $T = 0$ produces:

$$\underbrace{\text{the, mouse, ate}}_{\text{prompt}} \rightsquigarrow \underbrace{\text{the, cheese.}}_{\text{completion}}$$

If we change the temperature to $T = 1$, we can get more variety, for example, **its house** and **my homework**.

Evaluation

- How good is our model?
  - A good LM prefers real sentences:
    - Assigns higher probability to “real” or “frequently observed” sentences.
    - Assigns lower probability to “word salad” or “rarely observed” sentences.

Evaluation

- How good is our model?
  - A good LM prefers real sentences:
    - Assigns higher probability to “real” or “frequently observed” sentences.
    - Assigns lower probability to “rarely observed” sentences.
  - We train parameters of our model on a **training set**.
  - We test the model’s performance on data we haven’t seen. An **evaluation metric** tells us how well our model does on the **test set**.

Evaluation

- Evaluation Metric: Perplexity
  - Perplexity is the inverse probability of the test set, normalized by the number of words

$$PP(x_{1:L}) = \exp\left(\frac{1}{L}\sum_{i=1}^{L} -\log\left(\frac{1}{p(x_i|x_{1:i-1})}\right)\right)$$

  - Minimizing Perplexity is the same as maximizing the probability of the test set, therefore, the lower the perplexity the better the model.

References

- Jurafsky, D., & Martin, J. H. Speech and Language Processing. Stanford University
- Stanford University. CS324 - Large Language Models. Course materials.
- https://jalammar.github.io/illustrated-word2vec/