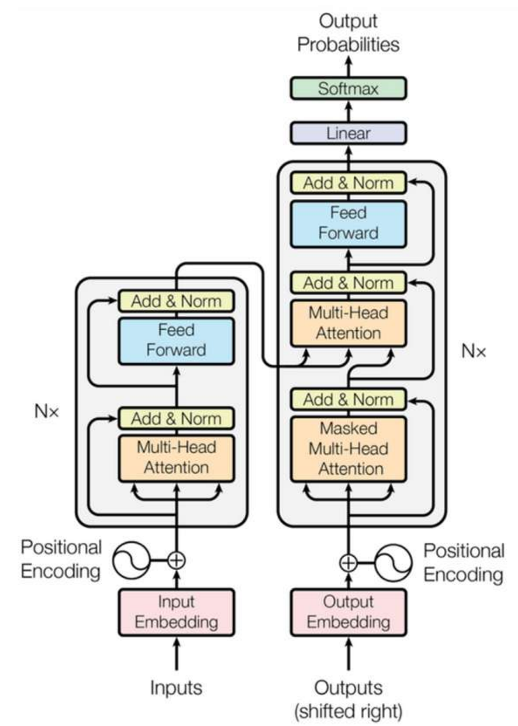
LLM Evaluation
and Preference Measurement

Minha Hwang



TWO TYPES OF LARGE LANGUAGE MODEL (LLM): DIFFERENT EVAL

Base LLM: Pre-Training

predict next word, based on text training data

- Self-supervised

Once upon a time, there was a unicorn
that lived in a magical forest with
all her unicorn friends

What is the capital of France?
What is France's largest city?
What is France's population?
What is the currency of France?

Instruction Tuned LLM: Post Training

Tries to follow instructions

Fine-tune on instructions and good response pairs

- **Human Labeled Data**: Instruction – Response Pair
- **SFT** (Supervised Fine-tuning) vs. **RLHF** (Reinforcement Learning with Human Feedback) or **DPO** (Direct Preference Optimization)

What is the capital of France?
The capital of France is Paris.

	Base LLM	Instruction Tuned LLM
Eval	Predictive Accuracy: Perplexity, Cross-Entropy, BPC, BPB	• Subjective Eval: Satisfied, Truthful, Fresh Functional Correctness: Pass@k • Semantic Similarity: Embedding-Based

Source: DeepLearning.AI – ChatGPT Prompt Engineering

PRE-TRAINED EVAL - PERPLEXITY: PREDICTIVE ACCURACY (1/2)

Pre-Training: How well a language model predicts a sequence of words.

- **Low perplexity: better predictive performance, more capabilities**
- **Not a good measure** to evaluate model that have been **post-trained**
- Perplexity of 3 – Interpretation: This model has a 1 in 3 chance of predicting next token correctly.
- More structured data: more predictable, lower perplexity
- Bigger vocabulary: higher perplexity
- Longer context window: lower perplexity
- Exponentiation of the average negative log-likelihood of the predictive probability of each word in a test dataset

$$\text{Perplexity}(PP) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid w_{<i})\right)$$

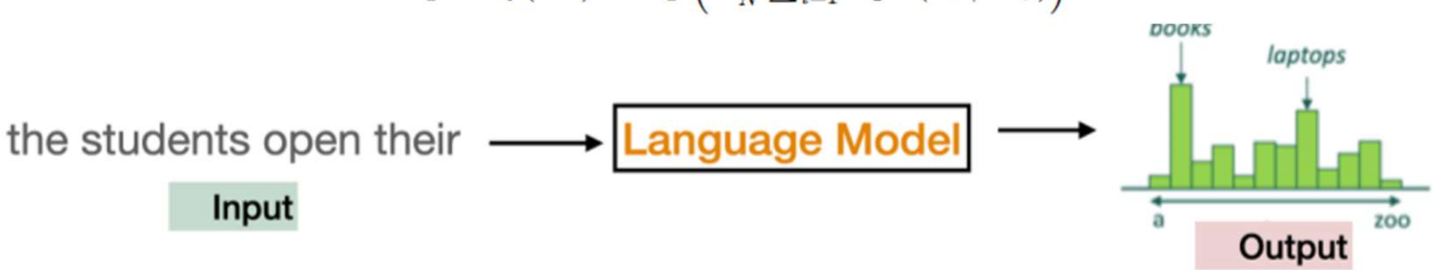


Image source: https//web.Stanford.edu/class/cs224n, AI Engineering by Chip Huyen

PRE-TRAINED EVAL - PERPLEXITY: PREDICTIVE ACCURACY (2/2)

Example Calculation: "The cat sat on the mat"

- P("The") = 0.2
- P("cat"|"The") = 0.1
- P("sat"|"The cat") = 0.15
- P("on"|"The cat sat") = 0.3
- P("the"|"The cat sat on") = 0.25
- P("mat"|"The cat sat on the") = 0.05

$$\text{Perplexity}(PP) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid w_{<i})\right)$$

First, calculate the average negative log probability:

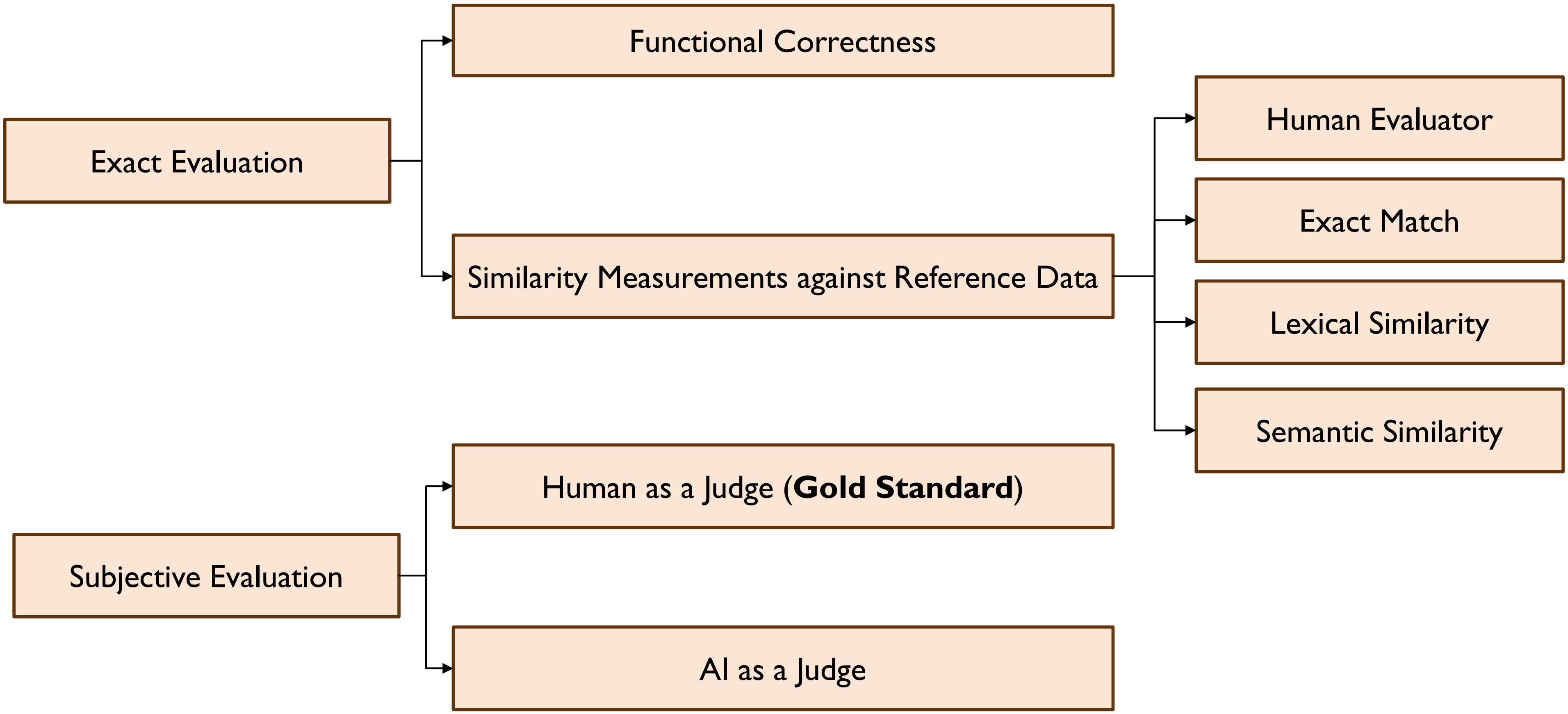
$$-\frac{1}{6}\left(\log(0.2) + \log(0.1) + \log(0.15) + \log(0.3) + \log(0.25) + \log(0.05)\right) \approx 1.8992$$

Then, exponentiate to find perplexity:

$$PP = \exp(1.8992) \approx 6.68$$

This means the model, on average, considers about 6.68 possible next words, indicating its uncertainty in prediction.

POST-TRAINED MODEL EVALUATION: OVERVIEWS



Source: AI Engineering by Chip Huyen, Minha's Synthesis

POST-TRAINED MODEL EVALUATION (1/3): EXACT EVALUATION

Functional Correctness

What it is

- Measures whether your application does what it is intended to do
- Not always straightforward to measure
- Can be automated for certain tasks: code generation, math

Examples

- **Code generation capabilities**: Automated functional correctness measurements (e.g., unit tests) – OpenAI's HumanEval, Google's MBPP, Spider (text-to-SQL)
- **Mathematical problem-solving**, **Fact-based Q&A**

How to measure

- For each coding problem, a number of code samples (i.e., k) are generated
- **pass@k:** A fraction of solved problems out of all problems

Similarity Measurements against Reference Data

What it is

- Evaluate AI's outputs against reference data – ground truths (reference-based)
- Reference data: (input, reference responses) - multiple reference responses possible
- Generated responses that are more similar to the reference responses are considered better
- Bottlenecked by how much and how fast reference data can be generated (human or AI)

Examples

- **BLEU, ROUGE, METEOR++, TER, CIDEr**
- Common for **translation tasks**

How to measure

- 4 ways to measure similarity: **human evaluator**, **exact match**, **lexical similarity** (fuzzy matching, edit-distance, counting how many tokens overlap), **semantic similarity** (embedding-based)

Source: AI Engineering by Chip Huyen

POST-TRAINED MODEL EVALUATION (2/3): EXACT EVALUATION

Functional Correctness

- Pass@k: for code evaluation
- 10 problems and a model solves 5 with k=3: pass@3 score of 50%
- The more code samples a model generates, the more chance the model has at solving each problem, hence the greater the final score
- In expectation, pass@1 score should be lower than pass@10
- Unit tests: run automated tests against generated outputs to check correctness
- Game bots: what score it gets in playing "Tetris"
- Logical validation: Ensure responses follow logical consistency (e.g., in reasoning tasks)

Similarity Measurements against Reference Data

BLEU (Bilingual Evaluation Understudy)

- Measures n-gram overlap between generated and reference text.
- Used in machine translation and text generation tasks.
- Formula:

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

- Example: If the reference text is "*The cat is on the mat*" and the generated text is "*The cat is sitting on the mat,*" BLEU calculates overlapping words and assigns a score.

BP: Brevity Penalty

Source: AI Engineering by Chip Huyen

POST-TRAINED MODEL EVALUATION (3/3): EXACT EVALUATION

Similarity Measurements against Reference Data

Edit Distance (Levenshtein Distance)

- Measures the number of insertions, deletions, and substitutions needed to convert one string to another.
- Example: "hello" → "helo" (1 edit) vs. "hello" → "world" (5 edits).

Embedding-based Similarity (Cosine Similarity)

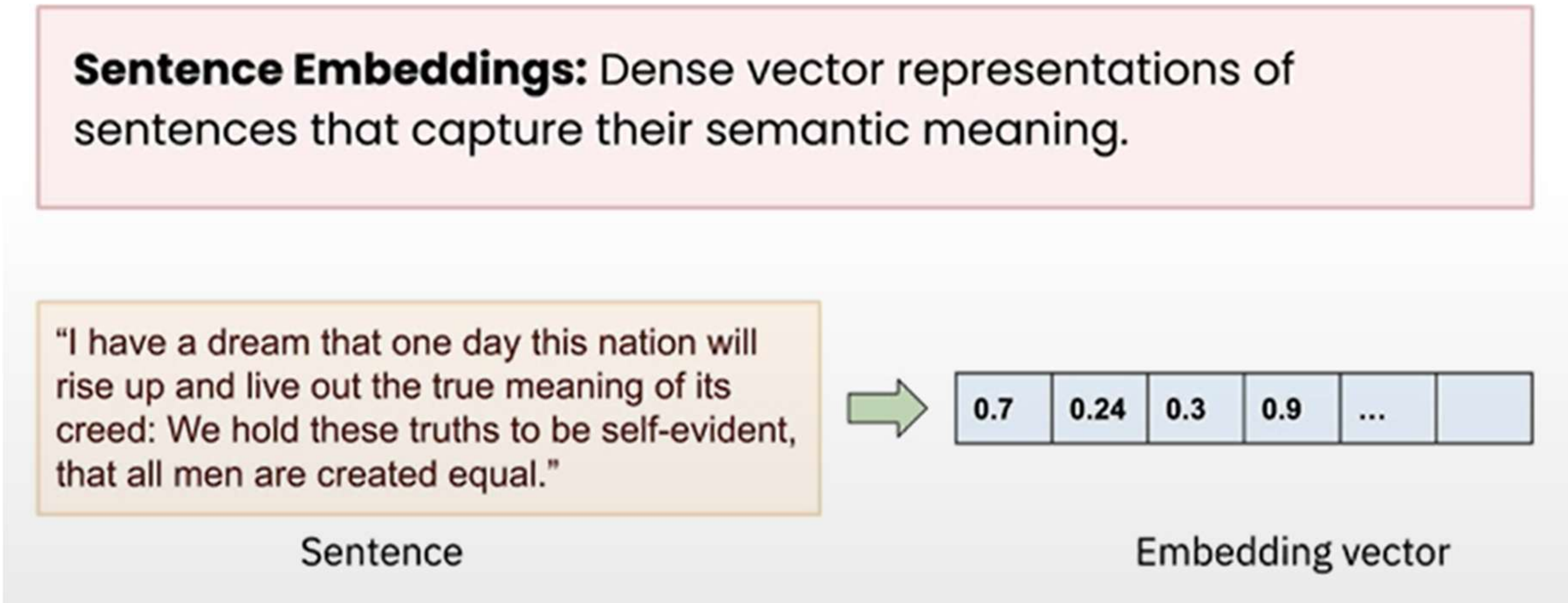
- Measures semantic similarity by comparing word embeddings in vector space.
- Used in paraphrasing and open-ended text generation.
- Formula:

$$\cos(\theta) = \frac{A \cdot B}{||A|| \, ||B||}$$

- Example: If "*The cat is on the mat*" and "*A feline is resting on a rug*" have similar vector representations, they are considered semantically similar.

Source: AI Engineering by Chip Huyen

SENTENCE EMBEDDING CAPTURES SEMANTIC MEANING AND CONTEXT-AWARE



Source: Deeplearning.ai – Embedding Models from Architecture to Implementation
https://learn.deeplearning.ai/courses/embedding-models-from-architecture-to-implementation

POST-TRAINED MODEL EVALUATION: SUBJECT EVALUATION (HUMAN PREFERENCE)

Example Criteria

Satisfaction

- **Satisfaction**: Which response makes the user satisfied with the results?
  - **Task Completion (+):** Was the user able to complete his/her tasks?
  - **Efforts (-):** How much efforts did the user put?
- **Helpfulness**: Which response answers the query and follow the specified instructions better?

Factuality (Against Hallucination)

- **Factuality**: Which response provides more accurate answers without hallucinations, even for questions that require very precise or niche knowledge?

Freshness (Against Stale Knowledge)

- **Freshness (inspired by FreshLLMs):** Which response contains more up-to-date information? A model excels in this criterion if it is able to answer queries with "fresh" information.

Source: Introducing PPLX Online LLMs
Perplexity enhances LLMs with holistic quality evaluation

Other Criteria: Relevance, Coherence, Fluency, Conciseness, Harmfulness, Maliciousness, Controversiality, Misogyny, Insensitivity, Criminality

APPENDIX

AUTOREGRESSIVE LANGUAGE MODELS: PROBABILITY KERNEL

- A language model is a **probability kernel μ** given a prefix of words: $\mu: X \to Pr(Y)$
  - Stochastic in nature: **A same prefix X** can give a **random output** sampled from a probability distribution μ_X(i.e., generative) → A key reason for factual inaccuracy, inconsistency or hallucination (making stuff up)

- A language model calculates $Pr(s)$ given a sequence of words: $s = (w_1, w_2, \ldots\ldots., w_{T-1}, w_T)$

- An autoregressive language model calculates this **conditional on a previous sequence of words**:

$$Pr(s) = Pr(w_1, w_2, \ldots\ldots., w_{T-1}, w_T)$$

$$= \prod_{t=1}^{T} Pr(w_t | w_1, w_2, \ldots\ldots., w_{t-1})$$

  - **Next-word prediction**: Given a prefix $(w_1, w_2, \ldots\ldots., w_{t-1})$, calculate the probability of the next word w_t (Conceptually same to time series with path dependence)

Source: Prof. Kyunghyun Cho

AUTOREGRESSIVE LANGUAGE MODELS: SIMPLE EXAMPLE

- 4-word sentence example: "I am a student"

$$\boldsymbol{Pr(s) = Pr(w_1, w_2, w_3, w_4) = Pr(w_1) \times Pr(w_2|w_1) \times Pr(w_3|w_1, w_2) \times Pr(w_4|w_1, w_2, w_3)}$$

- All you need is **"counting"** (if there are large amounts of data)

$$\boldsymbol{Pr(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)}} \longrightarrow \text{2-grams (Bigrams)}$$

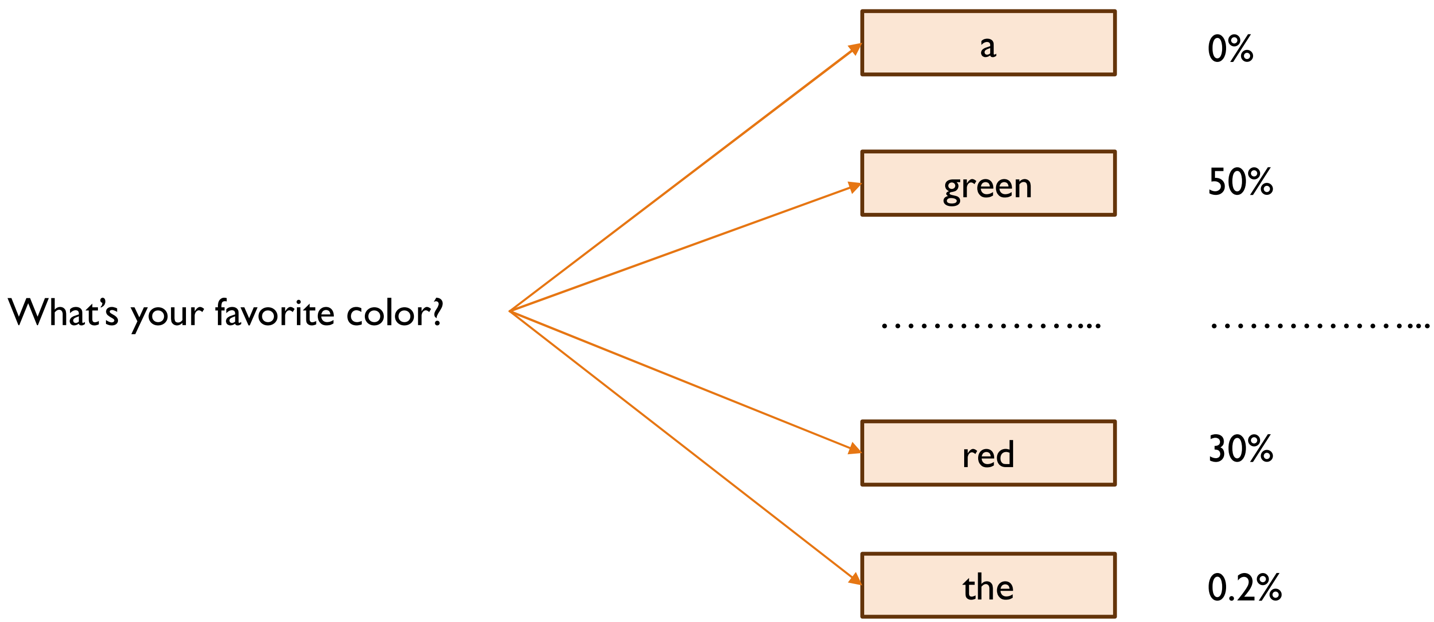
$$\boldsymbol{Pr(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{count(w_1, w_2)}} \longrightarrow \text{3-grams (Trigrams)}$$

$$\boldsymbol{Pr(w_4|w_1, w_2, w_3) = \frac{count(w_1, w_2, w_3, w_4)}{count(w_1, w_2, w_3)}} \longrightarrow \text{4-grams}$$

- Problems:
  - This requires **a lot of space (RAM)**
  - Count-based language models **cannot generalize**: A certain sentence **does not appear** in the corpus

Source: Prof. Kyunghyun Cho

SAMPLING: KEY TO UNDERSTAND PARAMETERS FOR LLM



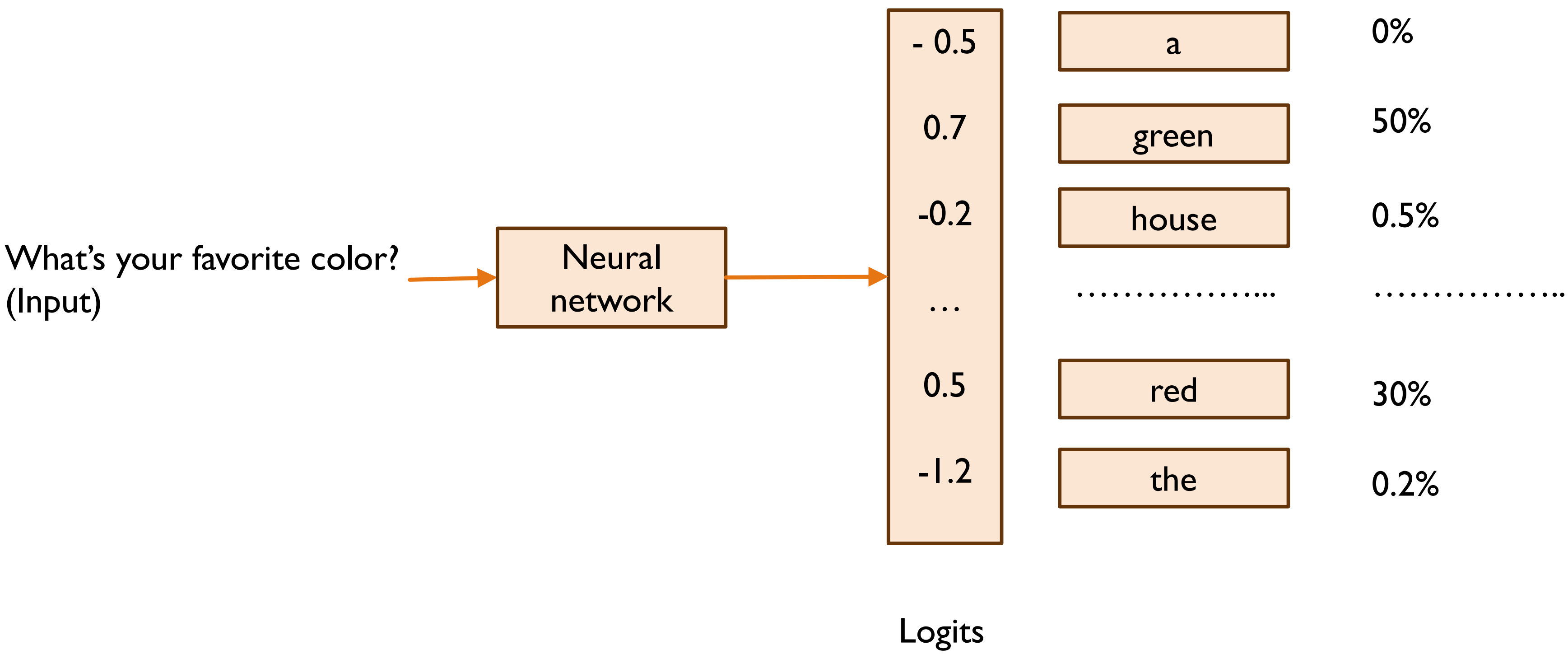
Probability distribution over the vocabulary

<u>SoftMax:</u>
Multiple classes

- **<u>Greedy sampling:</u>** Always pick the outcome with highest probability (boring outcome)
- Sample the next token **according to the probability distribution** over all possible values

Source: AI Engineering

UNDER THE HOOD



Source: AI Engineering