UNIVERSITI MALAYA

LANGUAGE MODELLING

NATURAL LANGUAGE PROCESSING

WQF7007

ASSOCIATE PROF. DR. NORISMA IDRIS

AND

DR. MOHAMED LUBANI

WHAT IS A LANGUAGE MODEL?

- A language model is a **statistical or neural model** that **assigns probabilities** to sequences of words helping predict the likelihood of the next word in a sentence based on the previous words.
- The goal is to model **naturalness**: make machine output similar to human language.

- Every time your phone suggests the next word or your email autocompletes a phrase, that's a language model at work!

IMPORTANCE OF LANGUAGE MODELS

- Language is probabilistic and **ambiguous**. To process it automatically, we need a model that can **predict and evaluate** the likelihood of words or phrases.

- Language models help machines choose the most likely and meaningful interpretation based on context.
- Without a language model, machines **cannot effectively "guess"** or evaluate what makes sense in human language.

NEXT WORD PREDICTION

- **Why predict the next word given preceding words?** ☐ To **resolve ambiguity**, improve accuracy, and ensure fluent language output in NLP systems. Example Applications:

1. **Speech Recognition**
   - Predicting words helps **disambiguate similar-sounding phrases**.
   - The model assigns a **higher probability** to the phrase that makes **linguistic sense**.
   - P("I saw a van") ≫ P("eyes awe of an")
2. **Machine Translation**
   - A language model helps ensure the **output is fluent and grammatically correct** in the target language.
   - The model ranks possible translations and chooses the one that is **most probable and natural-sounding** in the target language.
3. **Spelling Correction**
   - Predicting words in context helps correct typos that are otherwise valid words.
   - P("about fifteen **minutes** from") ≫ P("about fifteen **minuets** from")

TYPES OF LANGUAGE MODELS

- There are different types of language models, depending on how they learn and predict language:

1. **Statistical Language Models (N-gram models)**
   - Based on counting word frequencies
2. **Neural Language Models**
   - Use neural networks to learn word sequences and patterns e.g., LSTM
3. **Transformer-based Language Models**
   - Use attention mechanisms for better context understanding e.g, BERT

Statistical Language Models (N-gram models)

Statistical Language Models (N-gram models)

- Statistical language models **use probability theory** to predict the likelihood of word sequences.
- These models **learn from large corpora** to estimate the probability distribution over word sequences based on their observed frequency.
- An **N-gram** model is the most basic form of statistical LM.
- **N-gram models** estimate the probability of a word based on the **previous n–1 words.**
- Definitions:
  - **Unigram (1-gram):** individual words
  - **Bigram (2-gram):** pairs of consecutive words
  - **Trigram (3-gram):** triples of consecutive words
  - **N-gram:** any n-word sequence

Statistical Language Models (N-gram models)

- **Example Sentence:** *The car is blue.*

N-gram	**Examples**
Unigram (1-gram)	'The', 'car', 'is', 'blue'
Bigram (2-gram)	'The car', 'car is', 'is blue'
Trigram (3-gram)	'The car is', 'car is blue'
4-gram	'The car is blue'

Statistical Language Models (N-gram models)

- **How N-gram models work?**
- The model estimates: ***$P(wn \mid wn-1,...,wn-(n-1))$***
- N-gram models **approximate the true probability** by looking only at the **last n–1 words**.

- **Example for trigram**:
  - P(“blue” ∣ “car is”)= Count(“car is blue”) / Count(“car is”)

   This probability is based on **frequency counts in a training corpus**.

Statistical Language Models (N-gram models)

- **Example 4-gram language model**: *“you lost the ____”*
  - □ A 4-gram LM only considers the **previous 3 words** to predict the next word.
  - □ We want to compute the probability of the next word ***w*** after **“you lost the”**:
  - □ P(w | you lost the)= Count(you lost the w) / Count(you lost the)
  - □ Suppose in our training corpus:

4-gram phrase	Count
“you lost the”	10,000
“you lost the game”	2,000
“you lost the bet”	1,000
“you lost the argument”	500

Statistical Language Models (N-gram models)

- **Example 4-gram language model**: *“you lost the ____”*
    - P(game | you lost the) = 2000/10000 = **0.2**
    - P(bet | you lost the) = 1000/10000 = 0.1
    - P(argument | you lost the) = 500/10000 = 0.05

- The model predicts the next word as **“game”** because it has the **highest probability (0.2)** given the previous 3 words.

N-grams Vs Collocations

- **N-grams count all sequences of n words, whether or not they form meaningful expressions.**
- An N-gram is purely **statistical and positional.**
- Example bigram from a corpus:
  → “go to”, “a trip”, “trip with”, “with my” → all are counted.

- **Collocations focus on meaningful, frequent word pairs** or phrases that tend to occur together in a natural, conventional way.
- Example collocations:
  → “make a suggestion”, “take a seat”, “strong tea” (not “powerful tea”)

- Collocations are useful in **machine translation, lexicon building, and idiomatic phrase detection**, while N-grams are more general-purpose in language modelling.

N-grams Vs Collocations

Example: *“I took a seat and made a suggestion.”*

- N-grams:
  - “I took”, “took a”, “a seat”, “seat and”, “and made”, “made a”, “a suggestion”
  - all are equally counted.

- Collocations:
  - “took a seat”, “made a suggestion”
  - only these two are identified as collocations because they carry **conventional meaning together.**

Collocations in Language Modelling

- **Language models aim to predict the next word in a sequence.**
- Certain word pairs (collocations) occur **much more frequently together.**
- Identifying collocations **improves prediction accuracy**, captures natural language patterns, and **enhances the understanding of idiomatic expressions**.
- **Example:** “make a decision” vs. “do a decision”

 A bigram model might assign similar probabilities if both appear a few times, but **only collocation detection** knows "make a decision" is idiomatic.

 identifying collocations will **improve the statistical model’s understanding** of natural language.

Statistical Language Models Limitations

1. **Data Sparsity**
    - Rare or unseen word combinations get **zero probability**
    - The model struggles with generalization beyond the training data
2. **Fixed Context Window**
    - N-gram models can only see a limited number of previous words (e.g., bigram sees only 1, trigram sees 2)
    - Long-range dependencies in text are ignored
3. **Lack of Semantic Understanding**
    - Models operate mostly on frequency, not meaning
    - → Cannot detect synonyms, analogies, or context shifts

Neural Language Models

Neural Language Models

- A **Neural Language Model** is a type of language model that uses **neural networks** to learn the probability distribution of word sequences.
- Predicts the likelihood of the next word in a sequence by representing words as **dense vectors** (embeddings) and using a neural network to model context.

- Advantages over Statistical Models:
  - Can **learn general patterns**, not just memorized frequencies
  - Represents words as **dense vectors** that capture meaning
  - Captures **long-range dependencies** (especially with Transformers)
  - Supports **transfer learning**
  - Handles **unseen word combinations** gracefully

Neural Language Models Components

1. Word Embeddings

- **Typically *learned* during training**
- The neural model has an embedding layer which is **initialized randomly** and updated via **backpropagation** along with the rest of the model during training.
- This allows the embeddings to be **tailored to the specific language modelling task**.

 The embedding layer can be initialized with pre-trained embeddings, which is often done when **training data is limit**ed or **faster convergence is desired**.

Neural Language Models Components

2. Contextual Modeling

- Neural architectures (e.g., Feedforward, LSTM, Transformer) **model how previous words influence the next word**
- Evolution of Neural Language Models (NLMs): feedforward □ Recurrent/Memory e.g., LSTM □ Transformers (modern LLMs)
- **Feedforward NLMs:** Fixed context window and no memory but better than N-gram
- **Recurrent NLMs**: Sequential, hard to parallelize, limited long-range dependency handling
- **Transformer-based LMs**: Handles full context (via self-attention), fully parallel, best for long-range dependency, *better contextualized embeddings*

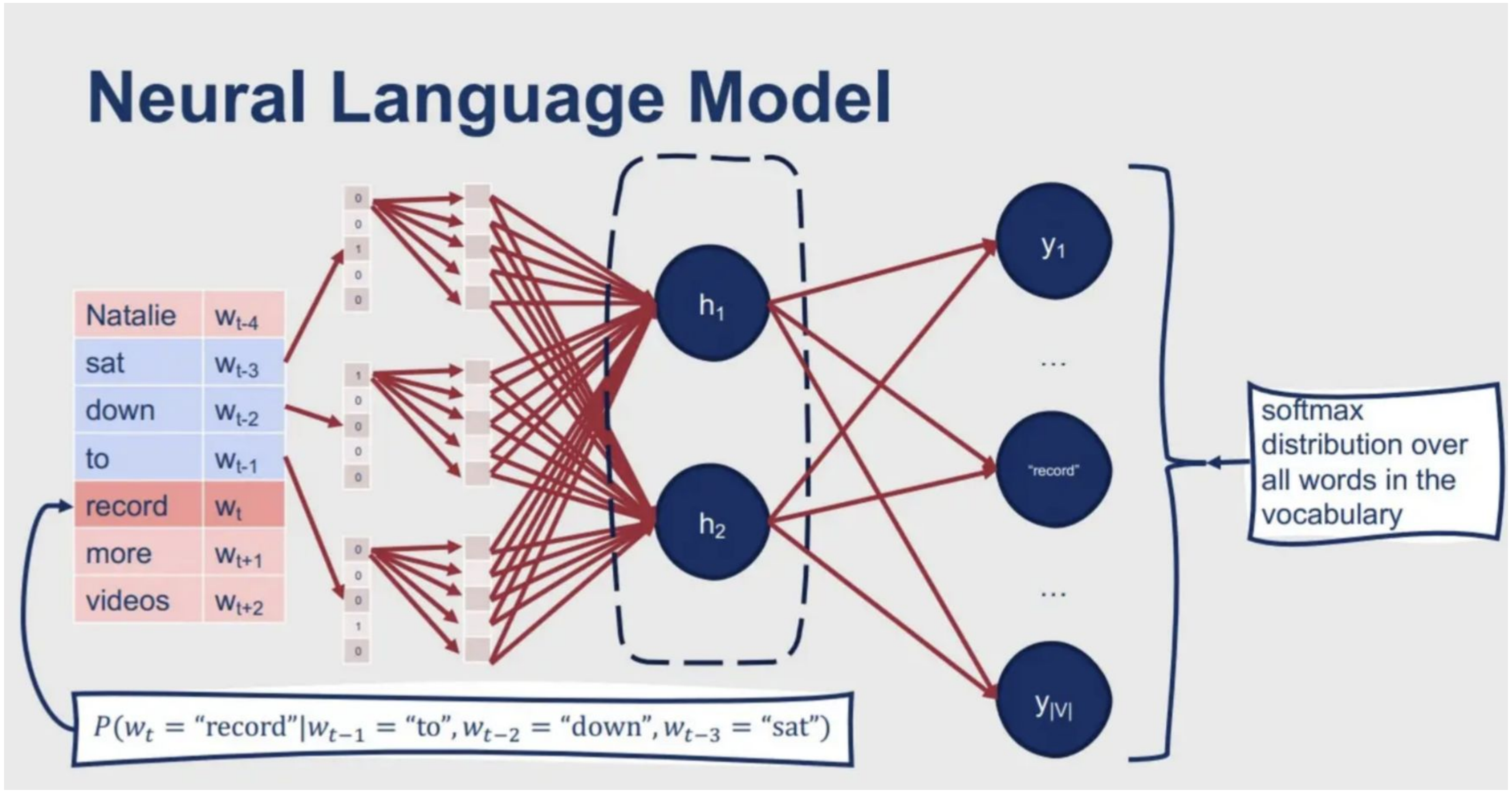
□ RNNs and LSTMs *were dominant before Transformers*, but are now largely replaced in NLP due to **limitations in parallelization** and performance on **long sequences**.

Neural Language Models Components

3. Probability Output

- The final layer outputs a **probability distribution over the vocabulary** for the next word
- Helps the model predict the next word by **converting raw scores into a probability** distribution.
- The word with the **highest probability** is chosen as the prediction

Feedforward Neural Language Model


Neural Language Model
Natalie
w_{t-4}
sat
w_{t-3}
down
w_{t-2}
to
w_{t-1}
record
w_t
more
w_{t+1}
videos
w_{t+2}
h_1
h_2
y_1
"record"
$y_{|V|}$
softmax distribution over all words in the vocabulary
$P(w_t = \text{"record"} | w_{t-1} = \text{"to"}, w_{t-2} = \text{"down"}, w_{t-3} = \text{"sat"})$

Feedforward Neural Language Model

- **Input Context Words**
  - The model takes a fixed-size **window of context words** before the target word
- **One-Hot Encoding → Embedding Layer**
  - Each word is initially a **one-hot vector**
  - These are projected to **dense embeddings** through a *learned embedding* matrix
- **Feedforward Hidden Layers**
  - The embeddings are concatenated and passed through **one or more hidden layers**
- **Output Layer with Softmax**
  - Predicts the **next word** *wt*=record
  - The softmax layer outputs a **probability distribution over the vocabulary**

Transformer-based Language Models

What are transformers?

- Transformers are deep learning models designed to handle sequences using **self-attention**:
  - Introduced in 2017 (Vaswani et al., ”***Attention is All You Need***”) most of the authors were affiliated with **Google Brain and Google Research** at the time
  - Originally designed for **machine translation** now used in almost **all state-of-the-art NLP models**
- Transformers replaced RNNs and LSTMs in NLP due to their **faster parallel processing**, **greater scalability, and *superior context modelling***

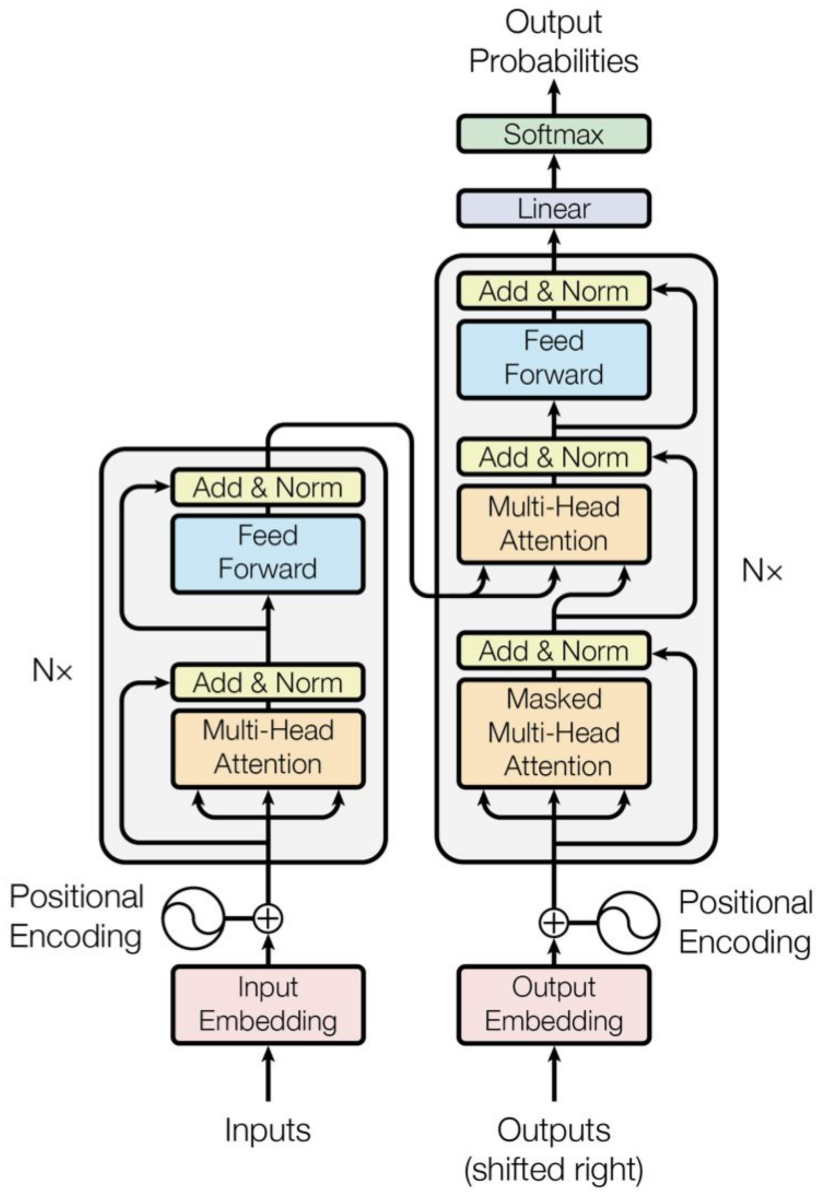
- **Key Innovations:**
  - **Self-Attention:** Helps the model **focus on relevant words** in a sentence regardless of their position**,** captures **long-range dependencies**
  - **Positional Encoder:** Adds word order information
  - **Context-aware embeddings:** Represent each word based on surrounding words

Transformers for Language Modelling

- Transformers provide **richer, dynamic word representations** that adapt to meaning, enabling superior performance on tasks like translation, summarization, and question answering.
- Enables models to **differentiate meanings** and **track relationships** over long distances
- Supports **parallel computation** → faster training
- Ideal for ***large-scale pretraining*** **and long text understanding**

Transformer Model Architecture

- The Transformer architecture is built around an **encoder-decoder structure**
- **The encoder (Left Side)** processes the input sequence creating meaningful representations
- **The decoder (Right Side)** generates the output sequence based on the encoder representations



Transformer Model Architecture

1. Encoder Steps:

- **Input Embedding (learned embeddings)**: Words are converted into numerical vectors that capture their meanings. These embeddings are stored as **weights in the transformer's first layer** and are learnt during model training.
- **Positional Encoding**: Adds a vector *(element-wise addition)* to each word embedding that encodes its position in the sentence (1st, 2nd, etc.).
- **Self-Attention Layer**: Each word looks at all other words to understand the full context. Can be split into ***multiple independent heads*** to let the model learn different relationships (e.g., subject-verb, adjective-noun).
- **Feed-Forward Layer**: Each word's output from the attention layer is passed through a small neural network to ***learn more complex representations*** beyond attention.
- Repeat ***N*** times (identical layers).

- **Output:** A set of contextualized representations corresponding to input tokens.

Transformer Model Architecture

2. Decoder Steps:

- **Output Embedding**: The **expected output sentence** is **shifted one position to the right by adding a** special <start> at the beginning. This *prevents the model from seeing the correct next word* ahead of time. These shifted tokens are then **converted into embedding vectors** and passed into the decoder.
- **Masked Self-Attention**: Each word only sees earlier words in the output to avoid peeking at future words.
  - A way **to implement autoregressive** models□ generating tokens *one at a time using previous context*.
- **Cross-Attention**: Decoder looks at (attends to) encoder outputs to guide generation. Helps the decoder ***brings in knowledge*** from the input sentence.
- **Feed-Forward Layer**: Each output token is refined again in a simple neural network to extract higher-level features.
- Repeat ***N*** times (identical layers).

- **Output:** A sequence of vectors turned into words (a sequence of output tokens).

Large Language Models (LLM)

So what happens when we train a transformer on massive amounts of text? We get Large Language Models.

Large dataset: Trained on internet-scale corpora (web, books, code)

Large model: Billions of trainable parameters (GPT-3 has 175B)

Large capabilities: Reasoning, Question Answering, Summarization, Code Generation

LLMs are general-purpose models: Trained once, useful for many tasks with little or no fine-tuning.

Examples of Large Language Models (LLMs)

Generative Pre-trained Transformer (GPT-2/3)

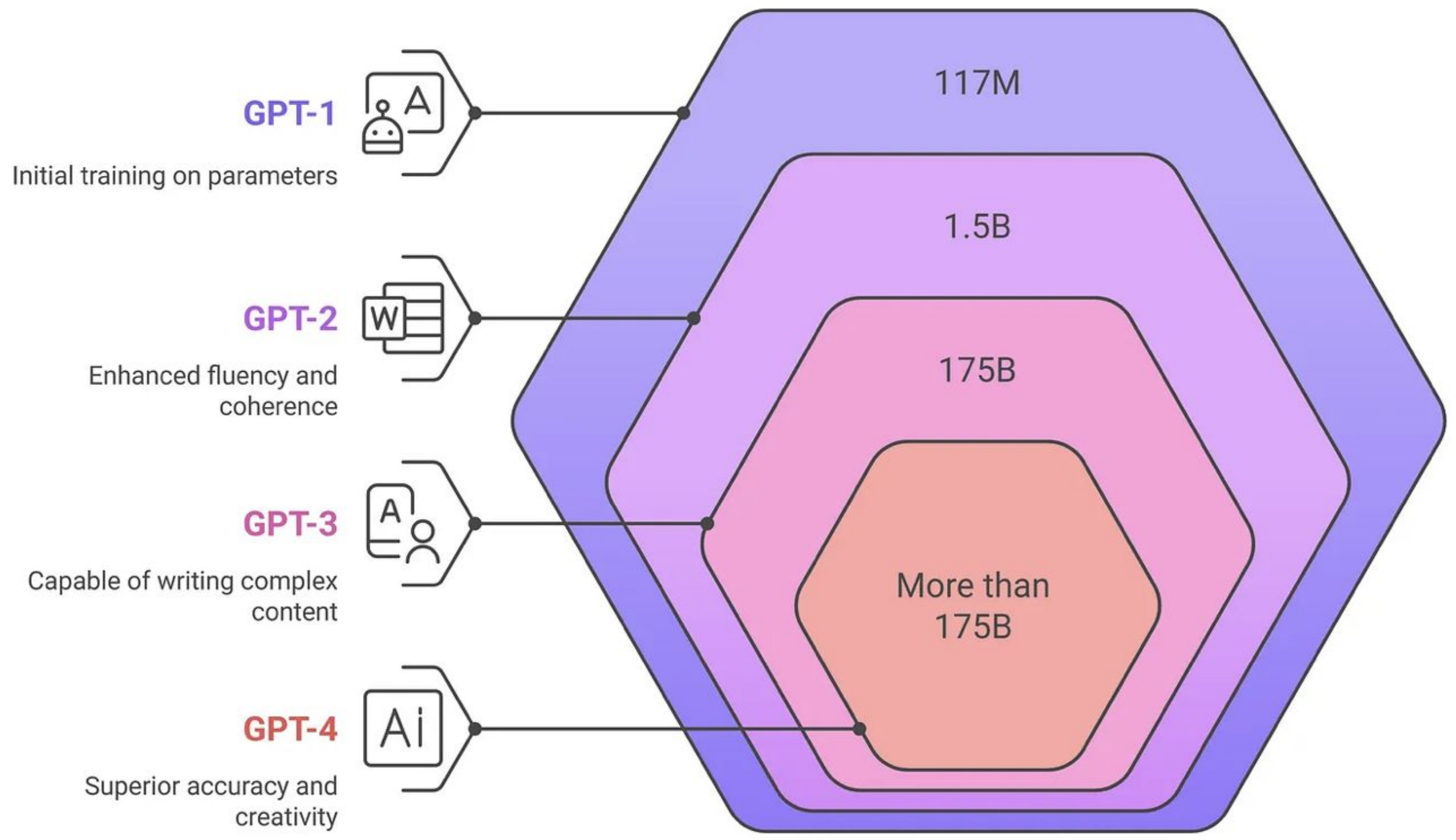
- The model’s primary task is **text generation (generative) and it was pre-trained** on **massive text corpora** (books, websites, etc.)
- It is based on a **decoder-only Transformer architecture** trained as a **causal language model (CLM)** to **predict the next token** in a sequence.

Bidirectional Encoder Representations from Transformers (BERT)

- BERT is designed to understand the **context of a sentence** by looking at words in **both directions**
- BERT uses only the **encoder part** of the Transformer, making it ideal for **understanding** tasks rather than generation.

GPT is used in generative tasks (e.g., completion, summarization), while BERT is used in understanding tasks (e.g., classification, question answering).

Examples of Large Language Models (LLMs)


GPT-1
Initial training on parameters
GPT-2
Enhanced fluency and coherence
GPT-3
Capable of writing complex content
GPT-4
Superior accuracy and creativity
117M
1.5B
175B
More than 175B

Parameter growth GPT-1 to GPT-4. Source.

Fine-tuning Vs. Pre-training

LLMs are initially trained using self-supervised learning, where no manual labels are needed. The model learns by **predicting missing or next tokens** in vast amounts of unlabeled text (e.g., books, articles, web pages).

In pre-training the model learns to understand syntax, semantics, and world knowledge by observing text patterns.

After pre-training, the model is **fine-tuned on smaller, labeled datasets** for specific tasks like sentiment analysis, summarization, or medical Q&A.

Transfer Learning

Transfer learning allows a model to **leverage knowledge learned from one task** and **apply it to another** with minimal additional data.

Pre-training captures general language understanding ⟶ Fine-tuning specializes it.

Cross-Language Transfer Learning: Models pre-trained in one language (often English) are fine-tuned or adapted to work in other languages for various tasks.

Task Transfer E.g., BioBERT**:** Pre-trained on biomedical text (PubMed), then fine-tuned for tasks like drug–disease relation extraction, entity recognition.

Real-World Applications of LLMs

Language Translation

LLMs like **mBART**, and **GPT** power multilingual translation systems

Question Answering

Models such as **BERT** and **GPT-4** are fine-tuned on QA datasets

Text Summarization

Applications include news aggregation, academic digesting, and email summarization tools

Sentiment Analysis

Used in social media monitoring, product review analysis, and market trend prediction

Real-World Applications of LLMs

Text Generation

Creative writing, content generation

Code Completion

Tools like **GitHub Copilot**

Named Entity Recognition

Extracting people, organizations, and locations

Speech-to-Text + Text-to-Speech

Used in assistants like Siri, Alexa

Thank You