GENERATIVE ARTIFICIAL INTELLIGENCE PART 1

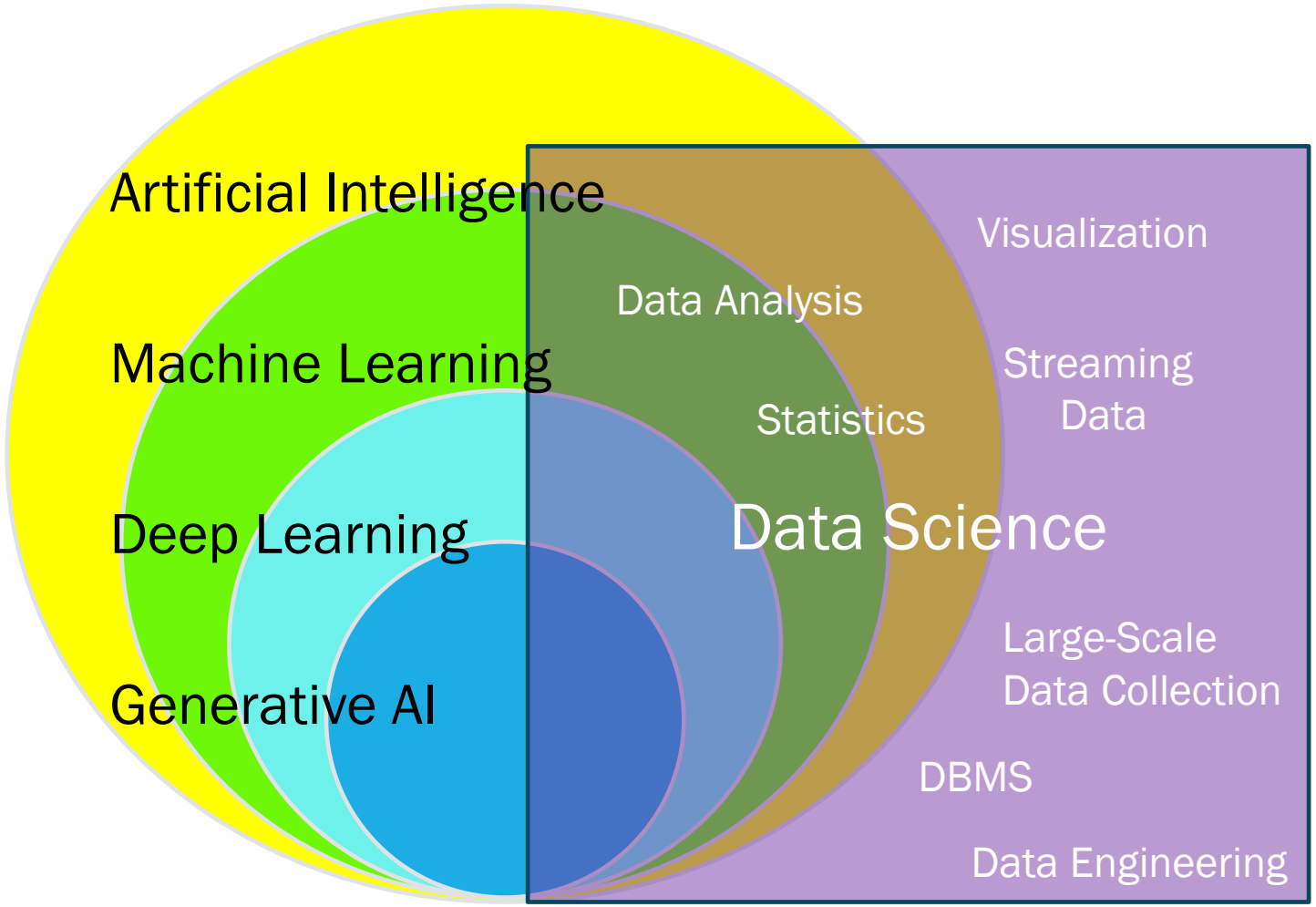
CS 180 Introduction to Data Science

AGENDA

- What is Generative Artificial Intelligence? Let's ask it!
- What are Large Language Models? Let's ask it!
- The process of building LLMs
- Prompt engineering
- Retrieval Augmented Generation
- Fine-Tuning
- Case study #1: unstructured text analysis for autism interviews
- Case study #2: depolarizing chatbots
- Quiz

BYU

WHERE DOES GENERATIVE AI FIT?



Generative AI:

- **Scope**: Use of multi-modal inputs (text, image, video, audio) to generate contextually relevant content.
- **End Goals**: Leverage deep learning and other neural network techniques to generate new, realistic, content.
- **Key Concepts**: Tokenization, Embedding, Large Language Models, Prompt Engineering, Retrieval Augmented Generation, Fine-Tuning, etc.

WHAT IS GENERATIVE AI?

- What does ChatGPT tell you? Try it!

BYU

WHAT GENERATIVE AI IS NOT...

1

NOT **A search engine.** Lots of people start interacting with AI like Google – inputting a question and expecting an immediate, accurate answer. You won't get value this way – instead think of AI as a colleague. You can ask questions, but you'll need to ask follow up questions, clarify, and provide feedback on responses to get value.

2

NOT **Thinking on its own**. Large language models (LLMs) like GPT by OpenAI and Claude by Anthropic aren't actually thinking. Instead, they're predicting. LLMs predict the next word in the conversation based on patterns they've learned from their training data set – huge volumes of information gleaned from the Internet.

3

NOT **A source of truth**. LLMs are really good at being creative with the data they have and they're biased toward pleasing us. Just like you would a coworker, take what you get with a grain of salt and verify.

WHERE AI CAN IMPROVE

- ☒ Occasionally makes surprising mistakes
- ☒ Makes up answers
- ☒ So eager to please, it may tell you what you want to hear
- ☒ Might perform great one day, and then terribly the next (on the same task)

BYU

ARE GENAI SYSTEMS JUST “STOCHASTIC PARROTS”?

- The phrase *"stochastic parrot"* was introduced in 2021 by computer scientists Emily M. Bender, et. al. in a paper titled *"On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?"*
- The term critiques large language models (LLMs), suggesting that, while these models mimic human language by predicting words based on patterns in their training data, but they lack true understanding or reasoning capabilities.
- Instead, they simply stitch together language patterns, akin to parrots that replicate speech without comprehending meaning.
- Advanced models can simulate reasoning by generating coherent responses or problem-solving steps, but this is not the same as human reasoning.
- Will GenAI systems eventually cross the threshold to human-like reasoning? This is an ongoing debate.

	Parrot	ChatGPT
		
Learns random sentences from random people	✓	✓
Talks like a person but doesn't really understand what it's saying	✓	✓
Occasionally speaks absolute non sense	✓	✓
Is a cute little bird	✓	✗

BYU

KEY IDEA:

Many language related problems can be framed as predicting the next word

BYU

PREDICTING NEXT WORD FORMULATED AS A PROBABILISTIC MODEL

Can you please come **here?**

History / context

Predicted word

$$P(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

Predicted word

History / context

Can you please come… and help me? I'm sorry, but I can't come and help you in person. However, depending on what kind of help you need, I may be able to provide advice or resources that can help.

to my house? I'm sorry, but I cannot come to your house.

up with a plan to reduce the cost of producing these products?

BYU

GENERATIVE AI APPLICATIONS

Text	Video	Image	Code	Music	Speech	3D
Marketing content	Video generation	Art & design	Code generation	Music composition	Text-to-speech	3D object modeling
Emails	Video editing	Marketing illustrations	Code completion	Sound effect design	Virtual assistance	Architectural visualization
Creative writing	Video game development	Photo editing	Refactoring & optimization	Arrangement & orchestration	Voice cloning	Animations for characters
Translation	Video summarization	Product design & prototyping	Bug detection & fixing	Remixing & mashups	Voice synthesis	Industrial design
Legal & technical texts	VR & AR	Fashion & apparel	Testing	Virtual instruments	Audiobook production	Gaming environments
News articles & summaries		Data visualization	Code formatting		Personalized voice interfaces	


Enterprise: Horizontal
SEARCH / KNOWLEDGE
glean
cohere
DUST
MARKETING
attentive
Mutiny
Jasper
copy.ai
Twain
WRITER
Rytr
anyword
copysmith
Hypotenuse AI
Writesonic
SALES
GONG
Clari
Apollo.io
tavus
Day.ai
unify
LAVENDER
Telescope
salesforce
HubSpot
CUSTOMER SUPPORT
Maven AGI
Slang.ai
CRESTA
zendesk
GENESYS
PRODUCTIVITY
tome
Microsoft 365 Copilot
Google Duet AI
RPA / AUTOMATION
zapier
Forge
DeepOpinion
ADEPT
rby AI
axiom.ai
UiPath
Google Duet AI
Kognitos
AUTOMATION ANYWHERE
DESIGN
Adobe
Figma
Recraft
Kittl
modyfi
Canva
Picsart
Visual Electric
DATA SCIENCE
ChatGPT Code Interpreter
seek
Wisdom AI
Hex Magic
turntable
tableau
Looker
Arcwise
SOFTWARE ENGINEERING / CODE GEN.
GitHub Copilot
replit
Cody
Cursor
FACTORY
CodeGen
codeium
tabnine
Magic
warp
Enterprise: Vertical
HEALTHCARE
Ambience
MEMORA HEALTH
DeepScribe
Hippocratic AI
abridge
Co:Helm
Nabla
ATROPOSHEALTH
LATENT
GLASS
Athelas
LEGAL
Harvey.
Ironclad
EvenUp
casetext
PINCITES
BIO
Inceptive
Cradle
FINANCIAL SERVICES
Hebbia
pilot
TRANSLATION
LILT

SOME COMMON GENAI PRODUCTS

Product	Category	Description	Use Cases
ChatGPT (OpenAI)	Conversational AI / Assistant	A large-language-model-powered chatbot accessible via web/mobile used by tens of millions of adults.	Asking questions, drafting writing (emails, essays), brainstorming ideas
Gemini (Google)	Multimodal AI assistant / model	Google's flagship multimodal AI (text, image, audio) with enhanced reasoning and coding in its 3.0 versions released in 2025.	Complex queries, image + text prompts, coding help, creative generation,
Midjourney	Text-to-image generation / Creative visuals	A popular AI image-generation tool that converts text prompts into high-quality, stylized visuals. Maintains strong usage among creators in 2025.	Creating illustrations, concept art, social-media visuals, mood boards, product mock-ups.
Suno AI	Music Generation	Creates full songs (lyrics, vocals, melody) from text prompts.	Music creation, jingles, personal projects, entertainment.
NotebookLM (Google)	Research / Knowledge-assistant	An AI tool aimed at helping with knowledge work: importing PDFs/websites/videos, generating summaries, mind-maps, audio overviews.	Academic research, studying, briefing documents, summarizing large content corpora, knowledge synthesis.

GENERATIVE PRE-TRAINED TRANSFORMER (GPT)

1. **Generative**: Model can generate new data—in this case, human-like text. Given an input (like a question or prompt), GPT predicts and generates a sequence of words to respond.
2. **Pre-trained**: Before it's fine-tuned for specific tasks, GPT undergoes a large-scale "pre-training" phase on massive amounts of text data. During this phase, it learns grammar, facts about the world, and common patterns of human language by predicting the next word in millions of text sequences.
3. **Transformer**: The "transformer" is the underlying architecture of GPT, which was introduced by researchers at Google in 2017. Transformers excel at handling sequential data and understanding context over long text. They use a mechanism called "attention" to weigh the importance of different words in a sequence, enabling the model to capture relationships between words, phrases, and sentences.

BYU

CHATGPT STATISTICS

- The tool set a **record** for having the **fastest-growing user base** in history for a consumer application, gaining **1 million users** in just **5 days**.
- The ChatGPT website received an estimated **1.8 billion** website visitors in April 2024 (an increase of around 200 million from 1.6 billion in February 2024) with an estimated **100 million** active users.
- In **May 2024**, OpenAI **announced ChatGPT-4o** which can accept and generate combinations of **text, images, audio, and videos**.
- ChatGPT contains **570 gigabytes of text data**, which is equivalent to roughly **164,129 times** the number of words in the entire Lord of the Rings series (estimated at over 1 trillion tokens).
- It is estimated that ChatGPT-4 was trained with **100 trillion parameters**, which is roughly equal to the human brain. (Source: Wired, Hix.ai)
- OpenAI used 1,023 A100 GPUs to train ChatGPT and it is estimated that training the model took **34 days**.
- The latest estimates suggest that ChatGPT **costs $700,000 per day or $21 million per month** to run, with **each query costing $0.36**.

BYU

LARGE LANGUAGE MODEL DEVELOPMENT PROCESS

- The process for training a Large Language Model (LLM) is complex and involves several stages, from data gathering to model deployment. Here's a high-level overview of each step:

Step	Description
Data Collection	Collect and clean large datasets from diverse sources.
Model Architecture	Select and configure the transformer-based architecture.
Pre-Training	Train on vast data to learn general language patterns.
Fine-Tuning	Adjust on domain-specific data for specialized tasks (optional).
Evaluation	Validate with automated and human metrics to check accuracy and coherence.
Deployment	Optimize and deploy the model for inference, monitoring its performance.
Iterative Improvement	Use feedback to refine and re-train the model as needed.

By following this end-to-end process, LLMs evolve from understanding basic language patterns to performing well-defined, complex tasks with human-like language abilities.

BYU

DATA COLLECTION AND PREPROCESSING

- **Data Collection**: LLMs require vast datasets to learn language patterns, so this step includes gathering data from sources like books, websites, scientific papers, forums, and more. The goal is to collect data that captures a broad range of vocabulary, syntax, and information.
- **Data Filtering and Cleaning**: Raw data often contains noise, irrelevant information, or duplicates. Filtering processes remove offensive or sensitive content, deduplicate entries, and exclude irrelevant parts, ensuring the model only learns from high-quality data.
- **Tokenization**: Text data is converted into smaller units called tokens, which can represent words, subwords, or characters. Tokenization helps the model process text numerically by turning each token into a sequence of numbers that the model can interpret and process.

BYU

TEXT TOKENIZATION

- Text tokenization is the process of breaking down text into smaller units, called **tokens**, that a machine learning model can process and understand.
- Tokens can represent words, subwords, or even individual characters.
- In natural language processing (NLP), tokenization is a foundational step, as it converts human language into numerical sequences that models can work with.
- On average, there are about **1.3 to 1.5 tokens per English word**

1. Word Tokenization: Splits text by words, often using spaces or punctuation as boundaries.

Example: "I love NLP." ⟶ ["I", "love", "NLP", "."]

2. Subword Tokenization: Splits text into meaningful subword units, which helps handle complex words and rare terms by breaking them into common roots and suffixes.

Example: "unbelievable" ⟶ ["un", "believ", "able"]

Subword tokenization is often used in transformer models (like BERT or GPT) and uses methods such as Byte-Pair Encoding (BPE) and SentencePiece.

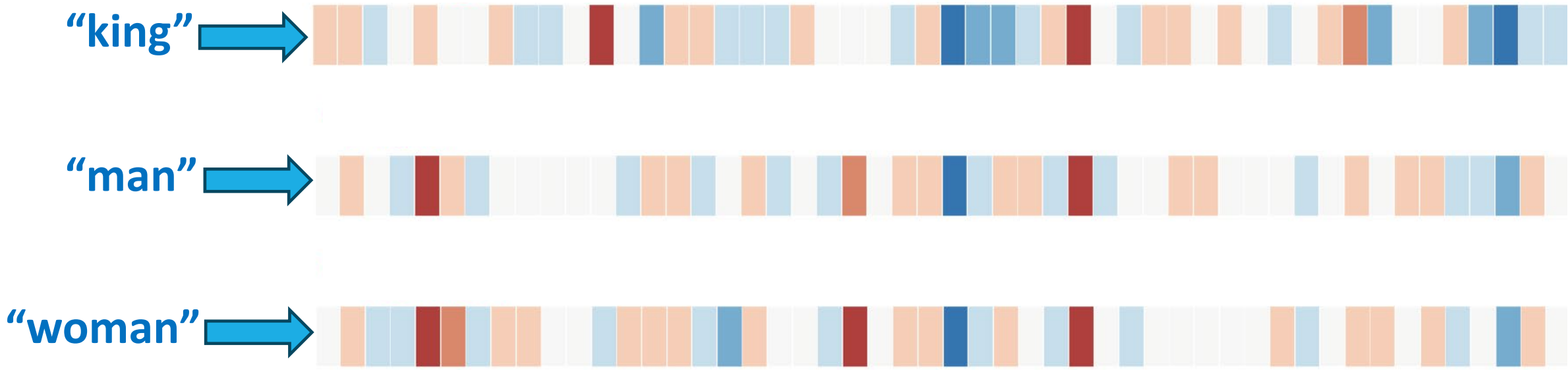
3. Character Tokenization: Splits text at the character level. This approach can be useful for handling misspellings or non-standard language but can result in longer token sequences.

Example: "cat" ⟶ ["c", "a", "t"]

BYU

TEXT EMBEDDING

Converts text tokens into dense numerical representations (vectors) that capture semantic meaning.

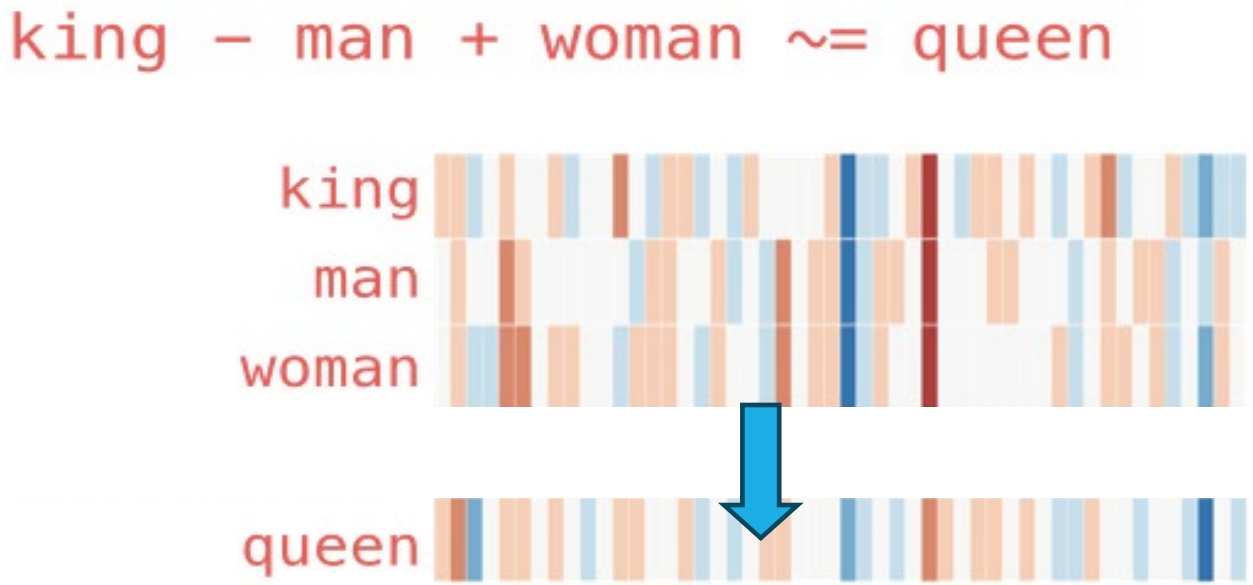


Typically 300-1000 features

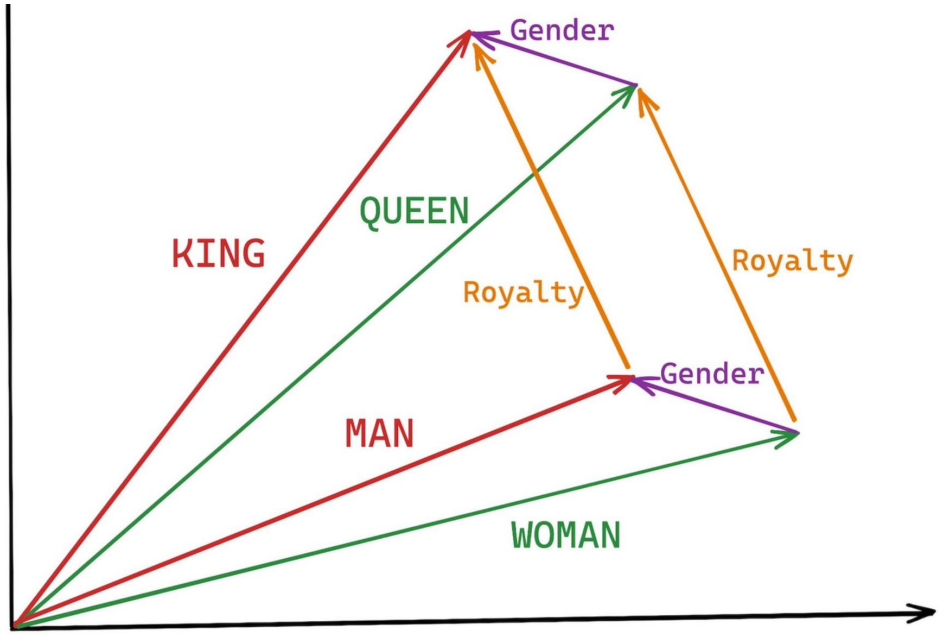
Big idea: represent the “meaning” of a word or phrase by a learned feature vector

BYU

TEXT EMBEDDING



Embedding Space



BYU

EMERGENT MATHEMATICAL RELATIONSHIPS

- big – bigger + small --> smaller
- France – Paris + Rome --> Italy
- au – gold + ag --> silver
- Einstein – scientist + Picasso --> painter
- Windows – Microsoft + Google --> Android
- Sushi – Japan + Germany --> bratwurst

BYU

MODEL ARCHITECTURE SELECTION

- **Choosing an Architecture**: The architecture of a model defines its structure and components. Transformers are the architecture of choice for LLMs due to their effectiveness in capturing long-range dependencies in text.
- Common architectures include BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pretrained Transformer), and others based on transformer layers.
- **Hyperparameter Selection**: Decisions on the number of layers, attention heads, hidden units, batch size, and other parameters are essential in defining the model's capacity and efficiency.
- These choices impact model performance, computational costs, and training time.

BYU

FOUNDATION MODEL TRAINING

- **Objective**: Teach the model language patterns and general knowledge by training it on large datasets with a self-supervised learning objective. Examples of self-supervised tasks include:
  - **Masked Language Modeling (MLM)**: Used in BERT, where certain words are masked, and the model learns to predict them based on context.
  - **Causal Language Modeling**: Used in GPT, where the model learns to predict the next word in a sequence.
- **Training Process**: Involves passing batches of tokenized text through the model and adjusting its weights to minimize prediction errors. This phase is computationally intense, often using powerful hardware (GPUs or TPUs) over several weeks or months.
- **Optimization**: Techniques like Adam optimization and learning rate schedules help the model converge to optimal performance while avoiding overfitting.

BYU

Andrej Karpathy

@karpathy

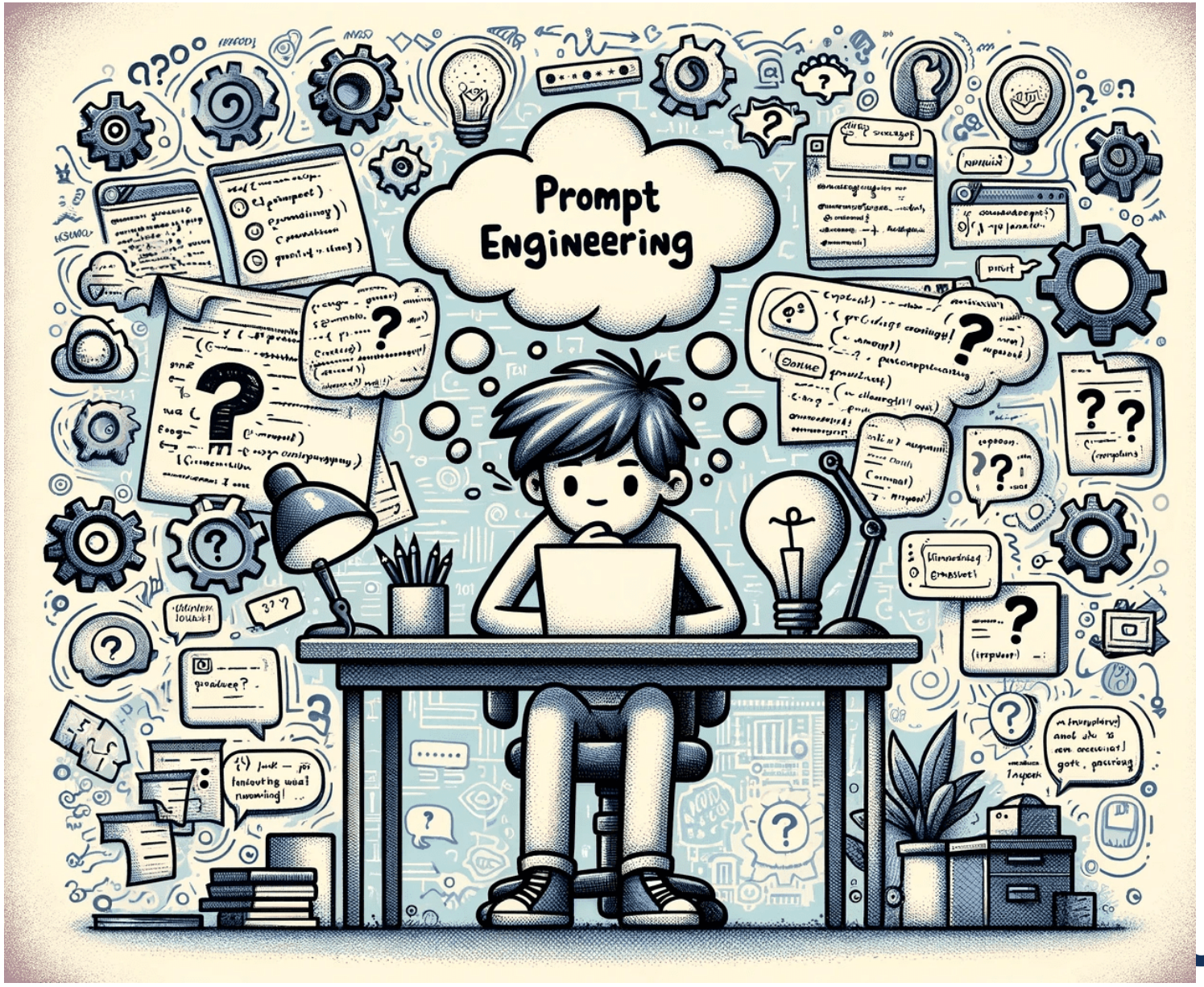
The hottest new programming language is English

1:14 PM · Jan 24, 2023 · **2.8M** Views

BYU

PROMPT ENGINEERING

- Prompt engineering is the process of crafting inputs, or "prompts," to get large language models (LLMs) like ChatGPT to produce useful and accurate responses.
- Effective prompt engineering helps shape how the model interprets and responds to your input, which can enhance its accuracy, creativity, and relevance.



WHAT'S THE BEST PROMPT?

You are rating movies based on reviews. After each review, rate each movie as "good" or "bad".

Review: <INSERT REVIEW HERE>

Rating: the movie was

Rate the following movie review on a scale of 1 to 10, where "1" is good, and 10 is "bad".

Review: <INSERT REVIEW HERE>

Rating:

BYU

1. UNDERSTAND YOUR GOAL

- Before writing a prompt, clarify your objective. Are you seeking:
  - **Direct information or an answer**?
  - **Creative content** (e.g., a story, poem, or artwork description)?
  - **Code generation** or **technical explanations**?
  - **Analysis** or **summarization** of a topic?
- Knowing your purpose helps you design the prompt to get closer to the desired outcome.

BYU

2. BASIC PROMPT TECHNIQUES

- Simple Prompting
  - Start with a straightforward question or command. If you want a direct answer, ask directly:
    - Example: "What are the main benefits of renewable energy?"
- Be Specific
  - Narrow down the topic or format to limit ambiguity.
    - Example: "List three environmental benefits of using solar energy over fossil fuels."
- Role-Playing
  - Ask the model to take on a persona or role. This can enhance responses for tasks needing domain-specific expertise.
    - Example: "As a financial analyst, explain the advantages of investing in index funds for a beginner."

BYU

3.1 ADVANCED TECHNIQUES

- Few-Shot Prompting
  - Give the model a few examples in the prompt so it can infer the pattern. This is especially helpful for creative writing, pattern-based tasks, and custom formatting.
    - Example:css
    - Copy code
    - **Write a short metaphor about a rainy day. - Example: "The sky wept softly over the silent city." - Example: "Rain tapped its fingers impatiently on the rooftops." - New:**
- Chain of Thought Prompting
  - Encourage the model to break down complex problems or provide step-by-step reasoning.
    - **Example: "Explain how photosynthesis works step-by-step, starting with how plants absorb sunlight."**
- Instructional Prompting
  - Frame the prompt as an instruction. This helps clarify what output you're looking for, especially in structured or technical tasks.
    - Example: "Generate three catchy social media post ideas for a coffee shop's new seasonal drinks."

BYU

3.2 ZERO-SHOT PROMPTING

Idea: models have been fine-tuned to follow instructions. Just put your instructions in the prompt

Prompt: Classify the text into neutral, negative or positive.

Text: I think the vacation is okay.

Sentiment:

BYU

3.3 FEW-SHOT PROMPTING

Prompt:

Below are examples of analogies. For each pair, complete the analogy in a similar format.

1. King is to Queen as Man is to Woman.
2. Sun is to Day as Moon is to Night.
3. Cat is to Kitten as Dog is to Puppy.
4. Tree is to Forest as Flower is to ?

Answer:

In this first prompt, we've provided three examples of analogies that show how pairs of related words are structured. After a few examples, the model is expected to follow the same pattern for the fourth analogy by inferring that the answer should be "Garden" or a similarly suitable word.

Prompt:

Translate the following English phrases into Spanish.

1. Hello -> Hola
2. Thank you -> Gracias
3. Good night -> Buenas noches
4. How are you? -> ?

Answer:

In this 2nd prompt, the model is primed with a few translation examples and is expected to complete the last translation accordingly with "¿Cómo estás?".

BYU

3.4 CHAIN OF THOUGHT PROMPTING

Problem: Calculating Remaining Inventory After Daily Sales and Restocking
Question: "A bookstore has 200 books in stock. Every day, they sell 30 books and restock 20 books. How many books will they have in stock after 10 days?"

- **Without Chain-of-Thought Prompting**
- **Prompt:** "A bookstore has 200 books. They sell 30 books and restock 20 books each day. How many books will they have after 10 days?"
- **Response:** "After 10 days, the bookstore will have 500 books."
- **Explanation of Error:** This answer is incorrect. The model incorrectly multiplies the number of days by the net change (30 - 20) and adds it to the initial amount without considering the day-by-day changes in inventory.

- **With Chain-of-Thought Prompting**
- **Prompt with CoT:** "Let's break this down step-by-step. Start with 200 books. Each day, calculate how many books are sold and restocked, then find the net daily change. Repeat this for 10 days to get the final count."
- **Response with CoT:**

1. "Starting with 200 books in stock."
2. "Each day, they sell 30 books and restock 20, so the net change is -10 books per day."
3. "Over 10 days, the total change is 10 days × -10 books = -100 books."
4. "Starting with 200 books, they will have 200 - 100 = 100 books after 10 days."

- **Final Answer:** "After 10 days, the bookstore will have 100 books."

BYU

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

- **Definition**: RLHF combines human feedback with reinforcement learning, where the model receives rewards based on human feedback. In this setup, the model iterates and optimizes its responses to maximize "reward," which correlates with producing responses that align with human expectations.
- **How It Works**:
  - The model generates multiple responses to a prompt.
  - Humans rank these responses by quality, typically using preferences like "best," "okay," or "poor."
  - A reward model is trained based on this ranking data to assign scores to generated responses.
  - Reinforcement learning techniques like Proximal Policy Optimization (PPO) use this reward model to fine-tune the LLM, encouraging it to produce responses with higher reward scores in the future.
- **Example in Practice**: In ChatGPT, human reviewers might evaluate responses to complex, open-ended questions like, "Explain quantum computing." They rank outputs based on depth, clarity, and accuracy. The reward model is then trained to favor these higher-ranked answers, and the LLM is subsequently optimized to produce similarly high-quality responses.

BYU

4. CONTROLLING OUTPUT LENGTH AND STYLE

- You can ask for specific lengths, tones, or formats:
  - **Length**: "Provide a brief summary of climate change solutions in 2-3 sentences."
  - **Tone**: "Explain the importance of data security in a friendly, conversational tone."
  - **Format**: "Write a bulleted list of key features of electric vehicles."

BYU

5. ITERATIVE REFINEMENT

- Experiment by adjusting the prompt based on the response quality:
  - If responses are too vague, add more details or examples.
  - If too verbose, specify a shorter format.
- For instance:
  - Initial Prompt: "What is quantum computing?"
  - Revised Prompt: "In two sentences, explain quantum computing to a high school student."

BYU

6. USING CONSTRAINTS FOR SPECIFICITY

- If your prompt needs factual accuracy or topical specificity, guide the model with constraints:
  - Example: "List 5 main events in American history after 1900 with brief descriptions."

BYU

7. PROMPT TEMPLATES FOR COMMON USE CASES

- Here are some template ideas for frequently used prompt types:
  - **Summarization**: "Summarize the following article in 100 words: [paste text]."
  - **Paraphrasing**: "Rewrite the following sentence to be more formal: [insert sentence]."
  - **Comparative Analysis**: "Compare the features of [Product A] and [Product B] in bullet points."

BYU

8. EXPERIMENTING AND ITERATION

- Iterating on your prompts is essential because even small changes can have significant impacts on responses. Here's a process for refining prompts:
  - **Analyze the Response**: Look for clarity, relevance, and accuracy.
  - **Adjust Specificity**: Add or remove details based on whether the response is too general or overly complex.
  - **Rephrase for Tone**: If the tone doesn't match your goal, try rephrasing to influence the style.
- Try it with any of your previous prompts. How did it change?

BYU

HALLUCINATIONS

- LLMs are trained on vast amounts of unstructured data, often from the internet, where factual accuracy varies.
- They generate text based on probability, not verification, meaning that if certain phrases or patterns are statistically associated—even if inaccurate—the model might generate them confidently.
- Hallucination can also occur if the model lacks relevant context or if the user prompt is ambiguous or open-ended.

Real-World Examples of Hallucination:

1. Medical Advice: If asked about a rare disease, an LLM might fabricate treatments, symptoms, or drug interactions that don't exist.

2. Historical or Biographical Information: An LLM might confidently state incorrect details about a historical figure. For example, it may claim that a prominent figure won an award they never received or participated in an event they weren't involved in.

3. Legal Information: LLMs can also generate fabricated legal information, such as non-existent laws or incorrect legal precedents. For example, a model might cite a court case that sounds legitimate but does not actually exist, which could lead to serious misinformation in legal contexts.

HALLUCINATION BENEFITS

- Creative solutions
- Unique or personalized perspectives
- Insightful ideas and thought partnership

HALLUCINATION DOWNSIDES

- Requires verification before publication
- Difficult to verify output if you aren't an expert
- Bad or costly decisions due to face answers

BYU

MITIGATING HALLUCINATIONS

- **Grounding Models in Real-Time Data**: Integrating live or real-time data can help models provide more accurate information.
- **Retrieval-Augmented Generation (RAG)**: This approach allows models to reference external knowledge bases, like Wikipedia or specific databases, during generation, which can help ensure answers are accurate.
- **Human Feedback and Reinforcement Learning**: Fine-tuning models with human feedback, especially in critical areas, can reduce the likelihood of hallucination in sensitive fields.

BYU

MODEL FINE-TUNING

- Model fine-tuning for large language models (LLMs) involves adjusting a pre-trained model on a more specific or smaller dataset to optimize its performance for a particular application or domain.
- Fine-tuning is typically conducted after the initial large-scale training phase, where the model is exposed to vast datasets (e.g., Common Crawl, books, and Wikipedia) and learns general language patterns, vocabulary, and syntax.
- Fine-tuning helps adapt general-purpose LLMs to niche requirements.
- For instance, fine-tuning an LLM for customer support allows it to understand brand-specific language and provide more consistent and on-brand responses.

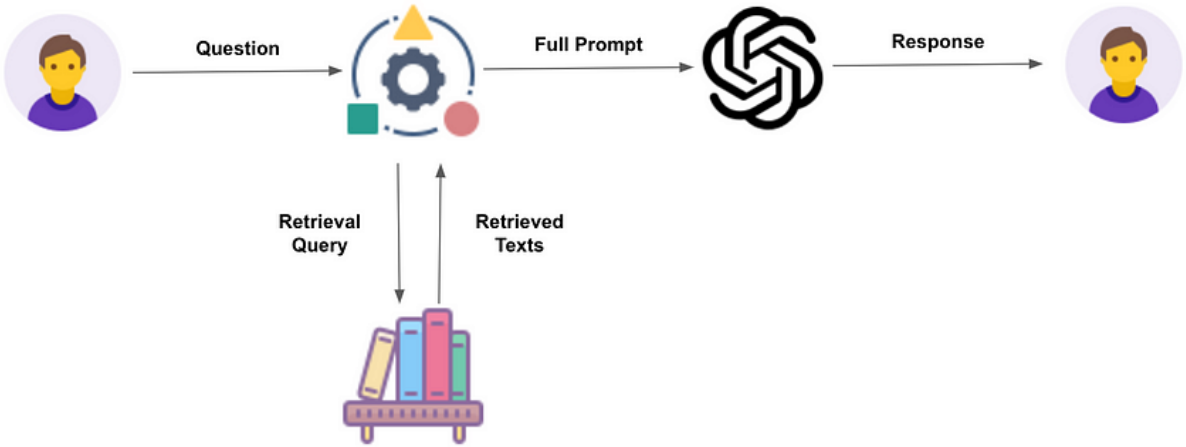
Key Differences

Aspect	Initial Training	Fine-Tuning
Objective	Learn general language patterns	Specialize in a specific task or domain
Data Size	Massive, diverse datasets	Smaller, domain-specific datasets
Scope	Broad, general-purpose understanding	Targeted understanding for specific use
Duration	Long, requiring extensive resources	Shorter, lower computational cost
Outcome	General-purpose language model	Customized, domain-optimized language model

BYU

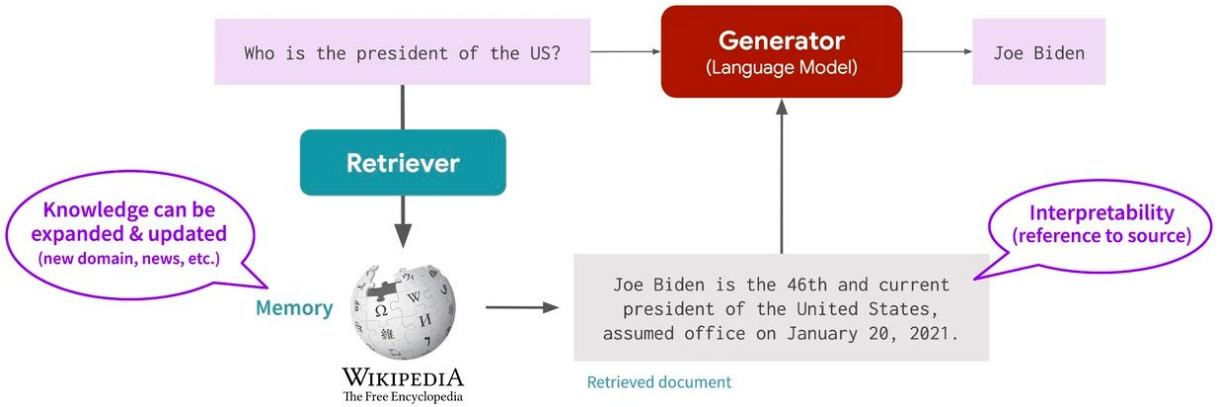
RETRIEVAL AUGMENTED GENERATION (RAG)

- Retrieval-Augmented Generation (RAG) is a technique that enhances the ability of a language model to produce accurate and relevant answers by incorporating external knowledge retrieved in real time.
- This approach is particularly valuable in applications where up-to-date or domain-specific information is needed, as it combines the generative strengths of language models with the factual reliability of external data sources.



- RAG involves two main components:

1. **Retriever**: This module searches a knowledge base or document store to find relevant information in response to a user query.
2. **Generator**: Using the retrieved documents as context, the language model generates a response, effectively "grounding" its answer in the provided, factual content.



STEPS TO IMPLEMENT A RAG PIPELINE

Step 1: Prepare the Document Store

- A document store is a database of all the text documents you want the model to retrieve information from. Some common options include:
  - **Elasticsearch**: A search engine for efficiently querying large volumes of text.
  - **Pinecone or FAISS**: Vector databases designed for fast similarity search, useful for finding semantically similar documents.
  - **Custom Knowledge Base**: A curated set of articles, manuals, or other data that the retriever can query.
- Each document should ideally be chunked into smaller, meaningful sections to improve retrieval quality, such as paragraphs or individual answer sections.

Step 2: Choose or Build the Retriever

- The retriever finds relevant documents based on a user query. There are several retriever types:
  - **Keyword-based Retrievers** (like BM25): Perform searches based on exact keywords and are common in traditional search engines like Elasticsearch.
  - **Dense Vector Retrievers**: Use embeddings (numerical vector representations) generated from deep learning models (e.g., BERT) to find semantically similar documents. Dense retrieval is more flexible and context-aware but requires more computation.
- For dense retrieval, common libraries are:
  - **FAISS**: Facebook's similarity search library.
  - **Haystack**: An NLP library that integrates with both traditional and vector-based retrievers.

Step 3: Choose or Fine-Tune the Generator

- The generator is typically a transformer-based model, like GPT-3 or T5, that can process retrieved content and generate a grounded response. Fine-tuning the generator on examples that mimic the RAG format—using documents and related questions and answers—can improve the specificity and accuracy of its responses.

Step 4: Configure the RAG Pipeline

1. **Receive Query**: The user inputs a question or prompt.
2. **Retrieve Documents**: The retriever searches the document store for the most relevant documents and returns them.
3. **Generate Answer**: The retrieved documents are concatenated with the user query and sent to the generator, which synthesizes a response based on both the query and the supporting documents.

Step 5: Handle Post-Processing

- Post-processing helps refine the output by:
  1. **Filtering Out Irrelevant Content**: Checking if the generated response aligns closely with the query.
  2. **Confidence Scoring**: Using models or rules to ensure responses meet a quality threshold.
  3. **Fallback Mechanisms**: Implementing alternatives, like “I’m not sure” responses, if no relevant information is found.

BYU

REFERENCES

Reference	Focus / What you'll learn
"Suno: potential, prospects and trends" (Yu et al., 2024)	A high-level survey of Suno's capabilities, outlining a general technical framework for AI music generation (symbolic vs audio, model categories, etc.).
"A Fourier Explanation of AI-music Artifacts" (2025)	Focused on artifact analysis in AI-music (including Suno). Very useful for understanding the signal-processing / architecture tradeoffs (e.g., decoder/layout choices causing artefacts).
"Inside Suno v5 – Model Architecture & Technical Mechanics" by Jack Righteous	A non-peer-reviewed but detailed write-up claiming to explain how Suno v5 uses a hybrid of transformer sequencing + diffusion for audio generation. Good for architectural intuition.
API & developer docs: the official "Suno API Documentation" site	Less about deep model internals, but covers system architecture, endpoints, workflows and how the system is deployed/integrated — helpful for applied engineering.
Case studies / data-driven analyses: "Data-Driven Analysis of Text-Conditioned AI-Generated Music: A Case Study with Suno and Udio" (2025)	Provides empirical evaluation of usage, prompting strategies, and generated output quality — useful for linking architecture to behavior in the real world.

REFERENCES

Reference	Summary & What You'll Learn
"Suno: potential, prospects and trends" (Zhang et al., 2024)	A peer-reviewed article summarizing AI-music generation in general, then focusing on Suno. Covers text-to-song pipelines (lyrics, voice synthesis, audio generation) and discusses Suno's technical framework. Great for high-level "what & why".
"Data-Driven Analysis of Text-Conditioned AI-Generated Music: A Case Study with Suno and Udio" (Casini et al., 2025)	Empirical study of how Suno and its peer Udio are used: prompt distributions, genres, and user behavior. Helpful in seeing how the architecture is used in practice and what kinds of outputs result.
"Suno AI: Advancing AI-Generated Music with Deep Learning" (TechRxiv)	A preprint/essay-style piece focusing on the deep-learning backbone of Suno (GANs, Transformers, diffusion models). Not peer-reviewed, but gives helpful pointers to model types and architecture choices.
"SunoCaps: A novel dataset of text-prompt based AI-generated music"	Research on a dataset generated with Suno: good for understanding data issues, prompt→audio mapping, evaluation. Helpful if you are interested in training/benchmarks.
"The Use of AI in Creating Music Compositions: A Case Study on Suno ..." (Atlantis-Press)	Conference-style paper that describes the application of Suno for non-musicians; contains technical walkthroughs of the pipeline. Useful as bridging theory → practice.

BYU