Prompt Engineering:

DSPy, LLM, and Chatbots

## Prompt engineering

Prompt engineering is a recently developed skill set in the usage of generative AI. It involves crafting precise instructions or queries that guide AI models to produce desired outputs.

This process requires understanding the model's capabilities and limitations, as well as the specific requirements of the task at hand. Effective prompt engineering can support the performance and the reliability of AI-generated content in chatbots, NLP’s, image generation, and various other models.

## DSPy

DSPy represents an approach to working with Large Language Models. DSPy is an open-source framework that streamlines the process of creating LLM applications.

**DSPy**:

* **Task definition**: Users can easily specify the objectives they want to achieve.
* **Pipeline construction**: It allows for the creation of complex workflows involving multiple LLM interactions.
* **Automatic prompt optimization**: DSPy can fine-tune prompts to improve performance, reducing the need for manual tweaking.

## Large Language Models (LLMs)

LLMs are natural language processing technology.

LLM AI models are:

* **Massive training datasets**: Models are trained on vast amounts of text data from diverse sources.
* **Versatility**: LLMs can perform a wide range of language tasks, from text generation to translation.
* **Deep learning architecture**: They utilize advanced neural network structures to process and generate human-like text.

## Chatbots

Chatbots can be explained as two main types of AI:

* **Rule-based chatbots**: Simple systems that follow predefined scripts.
* **AI-powered chatbots**: More advanced systems that use machine learning and natural language processing to understand and respond to user inputs.

## Prompt Engineering Concepts?

### What Does Prompt Engineering Do?

Prompt engineering is a specialized technique for designing precise instructions that guide AI language models to generate accurate, relevant, and high-quality responses. It is communicating effectively with AI to achieve specific outcomes within your LLM.

### How Prompt Engineering Works

* Create clear, structured instructions for AI models.
* Use specific language and context-setting techniques.
* Refine prompts through iterative testing and optimization.
* A diagram of a product

  Description automatically generatedApply strategic frameworks to improve AI response quality.

### What Prompt Engineering Does

## **Performance Optimization**:

## Improves AI model accuracy.

## Enhances response relevance.

## Reduces errors and inconsistencies.

## **Customization Capabilities**:

## Tailors’ outputs to specific tasks

## Adapts responses for different domains.

## Enables precise control.

## **Advanced Problem Solving**:

## Supports complex tasks.

## Enables multi-step reasoning.

## Facilitates nuanced communication.

## Practical Applications for Automation with prompt engineering

Topics of what a person can make while incorporating prompt-based engineering strategies:

* Content creation
* Code generation
* Customer service automation
* Research assistance
* Language translation
* Creative writing
* Technical documentation

## Technical Techniques

Techniques you can use when creating your project:

* Zero-shot prompting
* Few-shot prompting.
* Chain-of-thought prompting.
* Contextual framing
* Role-playing scenarios
* Systematic instruction design

## What is Prompt Chaining?

Prompt chaining is a technique where multiple prompts are used in sequence to guide an AI through a complex task.

It is like breaking a big job into smaller steps, where each step builds on the previous one.

### How Does Prompt Chaining Work?

* **Task Breakdown**: A complex task broken down and divided into smaller, manageable subtasks.
* **Sequential Prompts**: For each subtask, a specific prompt is created to be used in sequence with one another.
* **Step-by-Step Execution**: The AI processes these prompts in order, one after another.
* **Linking Outputs and Inputs**: The result from one prompt becomes the starting point for the next prompt.

### Example:

* Prompt 1: "Summarize this article about climate change."
* Prompt 2: "Based on the summary, list three main challenges."
* Prompt 3: "For each challenge, suggest a potential solution."

## What Does Prompt Chaining Do?

* **Tackles Complex Tasks**: It helps AI handle complicated jobs by breaking them down.
* **Improves Accuracy**: By focusing on one small task at a time, the AI can be more precise.
* **Allows for Corrections**: You can check and adjust the AI's work at each step.
* **Provides Better Control**: You can guide the AI's thinking process more closely.
* **Makes AI Work More Transparent**: You can see how the AI arrived at its final output by following the chain of prompts and responses.
* **Enhances Flexibility**: You can easily modify or add steps in the process as needed.

## Input & Output of prompting within Language Models

**Input**: The Prompt

The input is the text you provide to the language model. This can be:

* A question: "What are the main causes of climate change?"
* An instruction: "Write a short story about a time traveler."
* A statement to complete: "The three branches of the U.S. government are..."
* A conversation starter: "Hello, I'd like to know more about renewable energy."

**Processing**: How the Model Works

When you input a prompt, the language model:

* **Analyzes the text**: It breaks down the prompt to understand its structure and meaning.
* **Accesses its knowledge**: The model draws upon its vast training data to find relevant information.
* **Predicts the next words**: Based on patterns in its training, it determines the continuation of the text.

**Output**: The Generated Response

The model then produces an output.

This can be:

* An answer to your question
* A continuation of your statement
* A creative piece based on your instruction.
* A response to your conversation starter

### Note:

* **Context Matters**: The model considers the entire prompt, not just individual words.
* **Probability-Based**: The output is based on statistical likelihood, not a predefined set of answers.
* **No Real-Time Learning**: The model does not learn from your interactions; it uses pre-existing knowledge.
* **Versatility**: The same model can manage various tasks just by changing the prompt.
* **Limitations**: The model can only work with information it was trained on and may sometimes produce inaccurate or biased results.

## Large Language Models and Nodes

**LLMs are:**

* Trained in massive text datasets.
* Capable of understanding and generating human-like text
* Process input prompts and produce relevant outputs.
* GPT-3, GPT-4, BERT, and T5

**Nodes are functional components:**

* Function as building blocks in complex AI systems.
* Perform specific tasks or operations.
* Can connect to create advanced AI applications.

**LLM nodes typically:**

* Interfaces with a language model
* Accepts custom prompts as input.
* Returns the model's response as output.

## Relationship between LLMs, Nodes, within Prompt Engineering

* LLMs provide the core language processing capabilities for the model.
* Nodes create interaction within the LLMs in a structured workflow.
* Prompt engineering concepts are applied in the code and use techniques within nodes to optimize LLM performance.

### Workflow Steps:

* **Input Node**: Receives user query.
* **Preprocessing Node**: Cleans and formats the query.
* **LLM Node**: Sends optimized prompt to the language model.
* **Postprocessing Node**: Refines the LLM's output.
* **Output Node**: Delivers the final response to the user.

## Incorporation of databases into prompt chaining?

### How It Works:

**Question Asked**: A user asks the AI a question.

**Database Check**: Before answering, the AI looks up relevant information in the database.

**Information Added to Prompt**: The AI includes this database information in its thinking process.

**AI Responds**: The AI then answers the question, using both its general knowledge and the specific information from the database.

**Next Step**: The AI repeats the process, repeat checking the database for relevant info.

### What It Uses:

**Databases**: Large collections of organized information.

**Search Tools**: Software that helps the AI find the right or closest to right information within the database.

**Prompt Templates**: Pre-written text that helps AI include database information in its algorithmic searching process.

**AI Language Model**: The AI understands questions and generates the best possible chosen answer.

**Example:**

User: **"*What's the weather like in New York today?*"**

1. AI checks a weather database for current New York weather.
2. AI includes this data in its prompt: "Current temperature in New York is 75°F, partly cloudy."
3. AI responds with a complete answer using this information.

## What is a signature?

A **signature** is a way to define the input and output structure for interactions with Large Language Models.

It specifies the expected format of “inputs and outputs” when communicating with the AI model.

1. **Input**: Specifying the type and structure of data the model should receive.
2. **Output**: Describing the desired format and content of the model's response.
3. **Internal translation**: The signature is changed by the user into a prompt template that guides the LLM's behavior as it continues to be prompted.
4. **Consistency**: Signatures help maintain a standardized to interacting with a LLM for different tasks and applications.

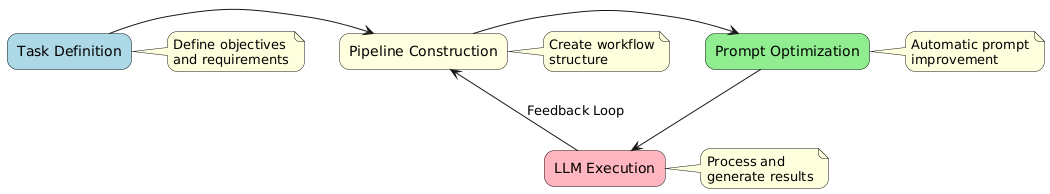
### Define a signature.

1. **Input Fields**: Specifies the type of input the model will receive.
2. **Output Fields**: Defines the expected format and type of response.
3. **Optional Instruction**: Provides additional context or guidance for the task.

### DSPy Characteristics

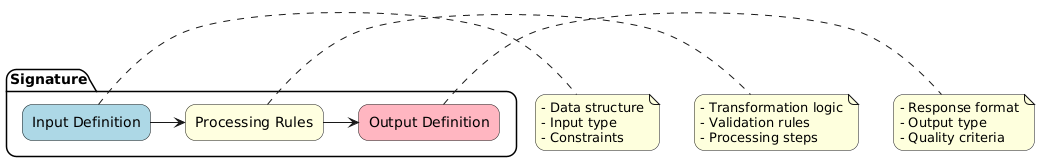
A DSPy signature is specifically:

* A specification of a text transformation
* A **tuple** comprising “input and output” fields.
* Focused on defining the system's behavior.
* (question → answer, document → translation)



## Reasons to use a signature?

* **Standardization**: Signatures provide a consistent structure for defining inputs and outputs, making it easier to create and maintain prompts across different tasks or applications.
* **Improved efficiency**: By clearly defining input and output formats, signatures help streamline the interaction between users and AI models, reducing the time spent on manual tasks.
* **Enhanced control**: Signatures give developers more control over how users interact with AI models, allowing them to refine outputs and present information in the desired format.
* **Better user experience**: Well-defined signatures help users obtain relevant results more easily, often on the first attempt, by guiding the AI to understand user intentions even with minimal input.
* **Increased flexibility**: Signatures with domain-neutral instructions can be reused across different contexts, allowing organizations to scale their AI investments more effectively.
* **Predictability**: Signatures help ensure consistent and predictable responses from AI models, aligning outputs with specific requirements or expectations
* **Error reduction**: By providing clear structures for inputs and outputs, signatures can help minimize misunderstandings or errors in AI-generated responses.



## Doc-Strings

Attaching information to a docstring is important for advancing and improving code readability and assists in providing clear documentation for your classes and methods.

### Docstring in DSPy

How to use:

* Use triple quotes (**"""**) for multi-line docstrings.
* Place the docstring immediately after the class or function definition.
* Include a brief description of the class or method's purpose.
* Document parameters, return values, and any exceptions raised.

## PDF’s As Tokens in DSPy

PDFs are used as data sources in DSPy and how this information becomes the foundation for language model predictions.

**Extracting Content from PDFs**

* + Extract the text content from our PDF files, use libraries like PyPDF2, pdfminer, or pdf2image and **Optical Character Recognition** for scanned documents.

**Preprocessing the Extracted Text**

* + Once we have the raw text, next is to clean and preprocess it.
    - Removing special characters or formatting artifacts
    - Normalizing whitespace
    - Splitting the text into manageable chunks

**Indexing the Preprocessed Text**

* + Index our preprocessed text chunks so they can be retrieved later.
  + Tools like Elasticsearch, Pinecone, or a vector database can be used for this.

**Creating a Retrieval System**

* + The PDF content is indexed next we retrieve relevant information based on queries.
  + This is done using semantic search or similarity matching.

**Integrating with DSPy**

* + Use DSPy to create a module that retrieves information to generate predictions and answers.

## How This Becomes Data for the Language Model

### When you use this setup:

* The user asks a question.
* The system retrieves relevant chunks from the indexed PDF content.
* These chunks, along with the question, are formatted into a prompt for the language model.
* The language model processes this information-rich prompt to generate a prediction or answer.

### What are we trying to do?

* We want to use information from PDF documents to help AI language models (like ChatGPT) give better answers.
* We are using a tool called **DSPy** to make this happen.

### How we USE DSPy:

**Getting Information from PDFs**

* We use special computer programs to read the text from PDF files.
* We clean up this text, removing any weird formatting or symbols.
* We break the text into small pieces called "**tokens.**"

bite-sized chunks of information.

**Organizing the Information**

* We store these tokens in a way that makes them easy to find later.
* Creating a smart index for our PDF content.

**Using DSPy to Work with the Information**

* DSPy is used as a toolkit that helps us interact with AI language models.
* We use it to create a system that can:
  + Find the right pieces of information from our PDFs when needed.
  + Give this information to the AI language model.
  + Ask the AI to use this information to answer questions or complete tasks given by the users input.

### How it works

When someone asks a question, our system:

* 1. Searches through the PDF information we have stored.
  2. Picks out the most relevant bits.
  3. Gives these bits to the AI along with the question.
  4. The AI then uses this extra information to give a better answer.
* It gives the AI a preset information of relevant facts before it answers a question.
* This helps the AI work towards and develop more accurate and informed answers, especially about specific topics covered within the PDFs.

## What is Predict in Prompt Engineering?

Predicting is the process of anticipating how AI language models will respond to different instructions.

* Prompts guide the AI towards the best expected outputs.

### What is Prediction?

**Understanding Model Behavior**

* How will the AI interpret, and process instructions given by the user?
* How can we design prompts that maximize results for accuracy and relevance?
* How can we guide the AI's reasoning and response generation to give the best possible output?

**Prompt Design Strategies**

* Break complex tasks into **step-by-step** instructions.
* Provide **minimal context** for efficient understanding.
* Structure prompts to elicit **specific responses**.

**Prediction Techniques**

* **Zero-shot prompting**: AI solves tasks without specific examples.
* **Few-shot prompting**: Learning from limited demonstration.
* **Chain of Thought**: Breaking problems into logical steps.
* Multi-solution generation for improved accuracy.

### The goal of prediction is to:

* Enhance the AI model’s comprehension of input to output results.
* Improve response quality of the models results.
* Create an intelligent and adaptable interaction between users and the AI systems for better results.

## Chain Of Thought

Chain of Thought (**CoT**) is a technique in prompt engineering that helps…

“AI to think more like humans.”

### What is Chain of Thought?

* It is a way to guide AI to break down complex problems into smaller, logical steps.
  + We give the AI a prompt that encourages step-by-step thinking.
  + The AI then explains its reasoning process as it solves the problem.
  + This helps the AI tackle more complex tasks and makes its thinking clearer to us.
* Instead of just giving a definitive answer, the AI shows its work, like a student solving a math problem.

1. Give the AI a problem to solve or predict.
2. Ask the AI to think "**step by step**" about the problem.
3. The AI breaks down its thinking into clear, logical steps.
4. Each step builds on the previous one, leading to a solution.
5. The AI provides its answer or prediction with the reasoning behind it.

### How do we use it in prompt-based engineering?

* It is a tool for making AI responses accurate and understandable.
* Using CoT, we create prompts that lead to better “*thought-out*” and detailed answers.

## Markov’s Chain in Relation to LLM’s

Markov chains and Large Language Models are related in the use of AI based language processing.

### What are the Markov Chains?

* A way to predict what comes next based on what just happened.
* In verbal or written language, it is guessing the next word based on the current word.
* LLMs are like super-advanced Markov chains.
  + They look at more words to make better predictions.
  + Both use probabilities to guess what comes next
  + LLMs can understand longer patterns and broader context.
  + Markov chains are simple, LLMs are more complex.

### How does Markov Chains relate to LLM’s?

**Define states**: Identify all states in the system you are modeling.

**Create a transition matrix**: Build a matrix showing the probabilities of moving from one state to another.

**Start with an initial state**: Choose a starting point in your system.

**Calculate probabilities**: Use the transition matrix to determine the likelihood of moving to each possible next state.

**Move to the next state**: Based on the calculated probabilities, determine the next state.

**Repeat**: Continue steps 4 and 5 to predict future states.

**Analyze patterns**: Over time, observe how the system evolves and identify any long-term trends or stable states.

### How do LLM’s use algorithms?

* **Transformer Architecture**: This is the “**brain”** of LLMs. It helps the AI understand context and relationships between words.
* **Neural Networks**: Interconnected "**thinking cells**" in the AI's brain. Neural networks work to process information and generate responses.
* **Word Embeddings**: “**Digital maps of words**.” They help the AI understand word meanings and how they relate to each other.
* **Zero-shot and Few-shot Learning**: Allows the AI model to “**understand”** new tasks with little or “**no examples.”** It is like being able to figure out a new game just by reading the rules.
* **Chain of Thought Prompting**: AI “**thinking step-by-step.”** It is like showing your work in a math problem.
* **Reinforcement Learning**: The AI learns from “**feedback**,” like a person learning from practice. Valuable feedback trains the model.
* **Retrieval-Augmented Generation**: Helps AI find “**specific information**” when needed. The AI has a reference to look specific information up.

## Signatures in DSPy:

A signature tells you what you are making based on the **input and output**, but not the exact steps to create it.

* It is a way to tell the AI what you want it to do, without saying how to do it.
* Example Input:
  + "*Take a* ***question****, give an* ***answer***.”
  + "***Read*** *a long text,* ***make*** *a short summary*.”

## Modules in DSPy:

Modules are pre-made components that help the AI do specific tasks.

* A "**Chain of Thought**" module helps the AI think **step-by-step**, like solving a math problem.

### How they work together:

Instead of writing exact instructions “**prompts”** for the AI, signatures say what you want done. Modules use signatures to figure out the best way to talk to the AI. Model.

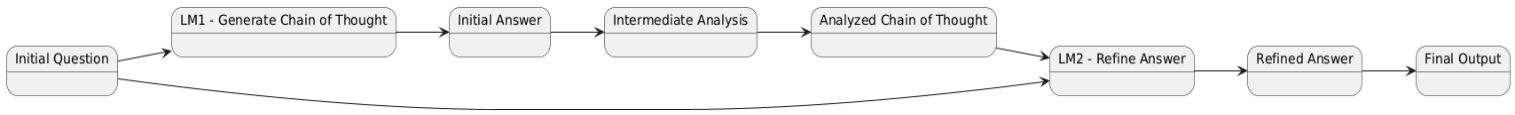
* The signature focuses on what you want to **achieve**, not on writing perfect instructions.
* A **signature** can be used to create complex AI tasks without getting stuck on details, we can improve the tasks through methods such as CoT.

### Why use Signatures and Modules:

* You can easily change or improve your AI project without rewriting everything and the system gets smarter over time, learning the best ways to talk to the AI model. Something like pre-conditioned training.

## Double Chain of Thought

Double **Chain of Thought** is a prompting technique that enhances the reasoning capabilities of LLM’s by using two separate language models in sequence.



* **Initial question**: A user creates a query or problem.
* **First Language Model** (LM1): Receives the original question, generates **chain of thought** which breaks down the problem into logical steps then it provides an initial answer based on this reasoning.
* **Intermediate analysis**: The **chain of thought** from LM1 is examined. Reasoning for decisions, process steps, and potential issues are found in this phase.
* **Second Language Model** (LM2): Receives the original question to process separately for LM1 where it also receives the analyzed chain of thought from LM1. The second language model then generates a **refined answer**, considering both the original query/problem and the previous reasoning.
* **Final output**: The second language model produces a “**better**” more accurate and well-reasoned response because of the initial thought process of LM1’s attempts.
* This method is preferred because of:
  + Improved problem-solving by leveraging multiple reasoning paths.
  + Better error detection and correction of queries.
  + Reliable answers, especially for complex problems.
  + Reduces chance for incorrect results, also AI Error, and unsafe content.

## What are Modules?

**Modules** have specific jobs or “**functions**,” making it easier to create complex systems.

### How are Modules Used?

**Chain of Thought (CoT):**

* In CoT, there is one main module that helps the AI think step-by-step.
* If you ask, "***How many cookies does each person get if there are 24 cookies and 6 people?***" the module would break this down into steps.
  + It recognizes that you need to divide 24 by 6.
  + Then it calculates that each person gets 4 cookies.
  + This step-by-step thinking helps the AI provide clearer answers.

**Double Chain of Thought (Double CoT):**

* Double CoT uses two modules in a row to improve the answer for better “**refined**,” results.
* The first module (LM1) thinks through the question and gives an initial answer.
* The second module (LM2) takes both the **original question** and **LM1's** thought process and checks it again:
  + "Okay, LM1 said each person gets 4 cookies.”
  + “Let's confirm that by doing the math again."
* This two-step approach helps **catch mistakes**, creating better answers.

## Stacking Multiple Modules in LLM Programs

**Creating Complex Systems:**

* By stacking multiple modules together, developers can build complex programs. Each module can focus on a different task,
  + reasoning
  + understanding language
  + analyzing information.

**Emergent Intelligence:**

* Modules, when they are used collaborating with each other can solve basic and complex problems better than one single model would.
  + One module might be great at math.
  + Another excels at understanding context.
  + Together, they make a smarter system.

**Performance Improvement:**

* Smaller, specialized modules can perform just as well as large models.
  + Developers can create an effective system without needing one huge overly complex super refined model.

## Reasons to use Modules?

* **Flexibility:**
  + Modules allow developers to easily change or add new features without starting from nothing.
  + This is an important skill when programming in general and prompting modules is no different.
  + If a task needs more reasoning, they can just add another reasoning module for effectiveness and efficiency.
* **Scalability:**
  + As tasks get more complex, new modules can be added without disrupting the existing system.
  + Creating an ecosystem within a project can be effective in refining a project, keep in mind as a warning, **Composition** >>> **Inheritance**.
* **Better Performance:**
  + Using **specialized modules** means the AI can perform **specific tasks** better than if it were using a single model used for everything.
  + **Note**: Specialized and specific go hand in hand, this is refinement.
* **Easier Debugging:**
  + If something goes wrong, it is easier to find the problem within one module rather than searching through a large “**complex and repetitive”** system.

## How Modules Work in DSPy

1. **Building Blocks**: Each DSPy module acts as a building block for programs that use LLMs. They abstract various prompting techniques, allowing developers to implement complex functionalities without starting from nothing.
2. **Composability**: Multiple modules can be combined into larger programs. This means developers can create sophisticated applications by stacking different functionalities, like how LEGO blocks can be combined to build various structures.
3. **Learnable Parameters**: Each module contains learnable parameters, including those related to prompts and model weights. This allows modules to be fine-tuned based on performance feedback, enhancing their effectiveness over time.

**Benefits of Using Multiple Modules**

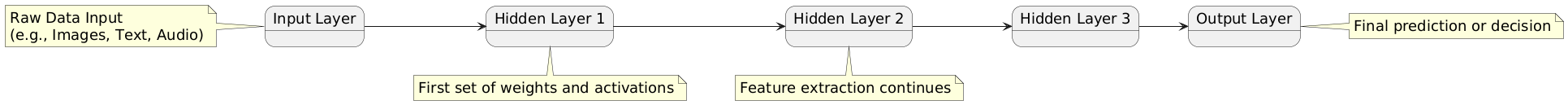
1. **Flexibility and Scalability**: Composing multiple modules allows for easy adjustments as requirements change. New features can be added without disrupting existing functionality.
2. **Enhanced Performance**: By leveraging specialized modules for different tasks, applications can achieve better performance compared to using a single monolithic model.
3. **Simplified Maintenance**: If an issue arises, it is easier to isolate problems within specific modules rather than searching through a large codebase.
4. **Collaboration Between Modules**: Different modules can work together to solve complex problems more effectively than individual modules alone.

## What are Deep Learning Modules?

Within **Deep learning modules**, each module has a specific job and works together with other modules to help the language model **learn** from data and make well trained predictions.

### Structure of a Deep Learning Module:

* **Neurons**: “***Tiny decision-makers***.”
  + Each neuron takes in information, processes it, and sends out a result.
* **Layers**: Neurons are organized into layers:
  + The **Input Layer**:
    - This is where the model first receives data.
    - a picture or text.
  + The **Hidden Layers**: “***more than one***”
    - Middle layers where the learning happens.
    - A deep learning model usually has multiple layers to help it understand complex patterns.
  + The **Output Layer**:
    - This is the final layer that gives the answer or prediction.
    - An image is a cat or a dog.
* **Connections**: Neurons in one layer are connected to neurons in the next layer through lines of connections.
  + - Each connection has a “**weight**” that determines how much more or less influence **one** neuron has on another.

****

## Types of Deep Learning Modules

**Convolutional Neural Networks (CNNs)**: used to work on **Images**, they help the model recognize patterns in pictures (like identifying objects).

[YouTube link for this](https://www.youtube.com/watch?v=cAkMcPfY_Ns&t=111s)

**Recurrent Neural Networks (RNNs)**: Used for **Sequential data**, like sentences or time series data.

* **RNN** remembers previous information, this is useful for tasks like language translation.

[YouTube link for this](https://www.youtube.com/watch?v=jPPWc0y-08Q)

**Transformers**: Used for understanding language. Self-attention to focus on important parts of the data. “***Can get a bit complex but very useful.***”

[YouTube link for this](https://www.youtube.com/watch?v=zxQyTK8quyY)

**Multi-Layer Perceptron’s (MLPs)**: neural network where every **neuron** in **one layer** connects to **every neuron** in the **next layer**.

* Predicting numbers or classes.

[YouTube link for this](https://www.youtube.com/watch?v=u5GAVdLQyIg)

## How Deep Learning Modules Work?

**Data Processing**: When you give data to a deep learning model, it goes through all the layers of neurons.

* Each layer changes the data slightly, making it easier for the model to understand.

**Learning Patterns**: During its training, the model learns by adjusting how strong each connection is based on the mistakes made.

* This process helps it improve the module over time.

**Making Predictions**: After training, when you give new data, it can process **new information** and make predictions based on what it has learned.

## Why do we use Deep Learning Modules?

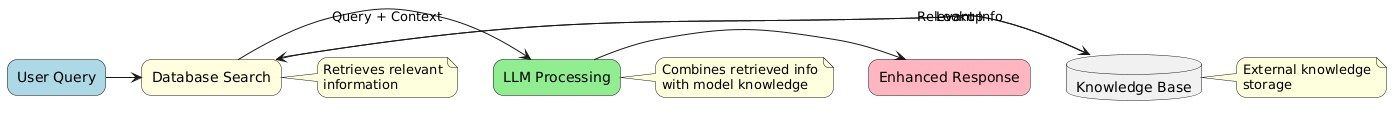
* **Flexibility**: Modules can be combined to create models for specific tasks.
* **Scalability**: You can easily add new modules or change existing ones without beginning from nothing.
* **Reusability**: Once a module is created, it can be used in different projects, saving time and effort.

TOPICS LEFT TO COVER FOR **Learning This:**

……………………………………………………………………………………….

……………………………………………………………………………………….

* DSPy is the pytorch for foundation models.
* Output non-string values in LLM.
* Dspy: relation to pydantic.
* pydantic what is it what is the confidence.
* how sure is the model about its answer.
* how to use prediction of the object and pydantic.
* the usage of Chain of thought and pydantic.
* complex predictors
* object, question, list.
  + I want to Discuss this… “very important.”
* Json and following format are it explicit and does it need to follow
* RAG: Define, Explain, details, how to use, how DSPy.
* Question, LLM, query, DB context back to LLM, retrieve context LLM generate answer.



* Colbertv2 retrieval server
* What is it? Data. Wiki, DSPy. Retrieve
* ask a question with normal chain of thought, may not provide a correct answer.
* information on retrieve and chain of thought
* relevance and retrieval process, how it works.
* When retrieve is called it should not be the question it can result in incorrect answer.