LANGCHAIN COMPONENTS

```
┌─────────────┐                                    ┌──────────────┐
│ 6. Agents   │                                    │ 1. Models    │
└─────────────┘ ↖                              ↗   └──────────────┘
┌─────────────┐   ↖        ⎛          ⎞     ↗      ┌──────────────┐
│ 5. Indexes  │ ←──────────│ LangChain │──────────→│ 2. Prompts   │
└─────────────┘   ↙        ⎝          ⎠     ↘      └──────────────┘
┌─────────────┐ ↙                              ↘   ┌──────────────┐
│ 4. Memory   │                                    │ 3. Chains    │
└─────────────┘                                    └──────────────┘
```
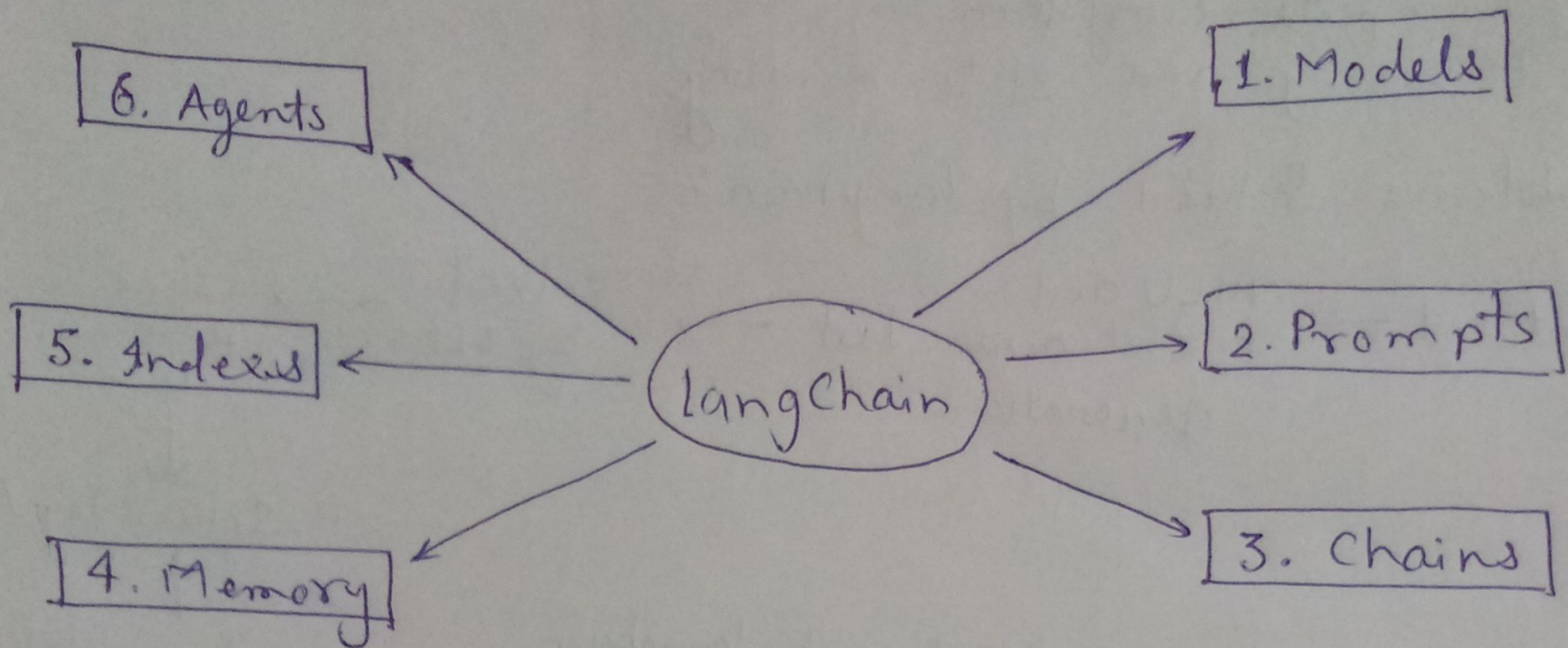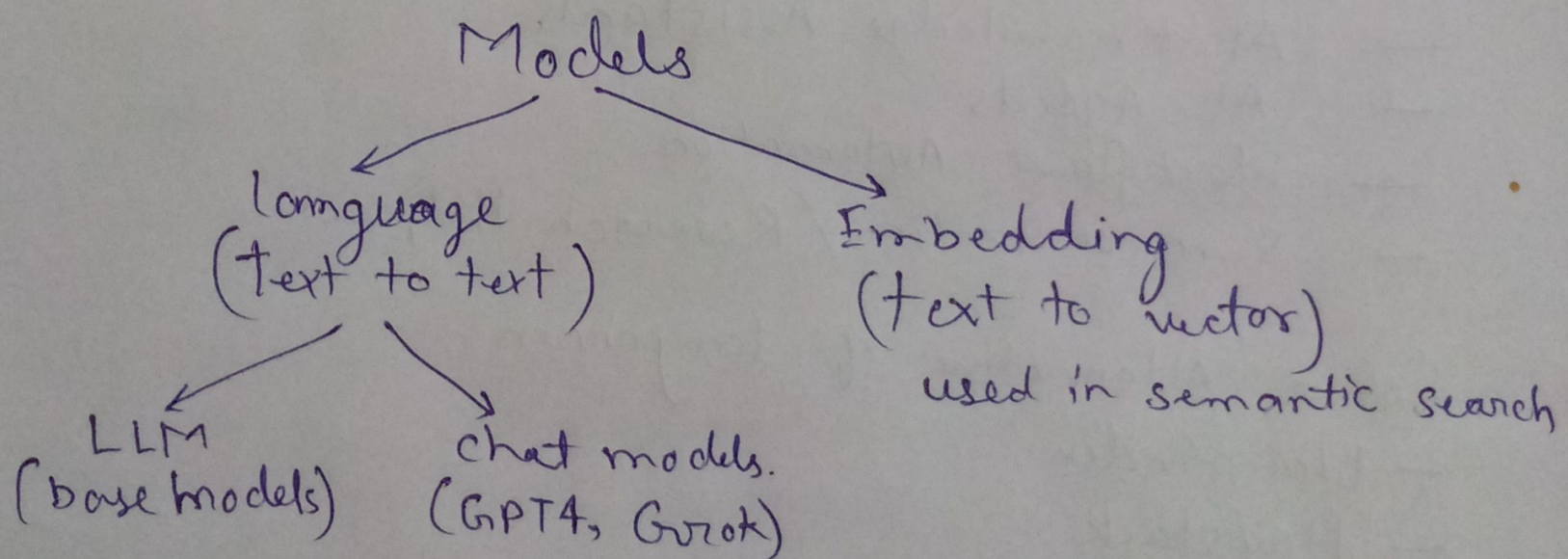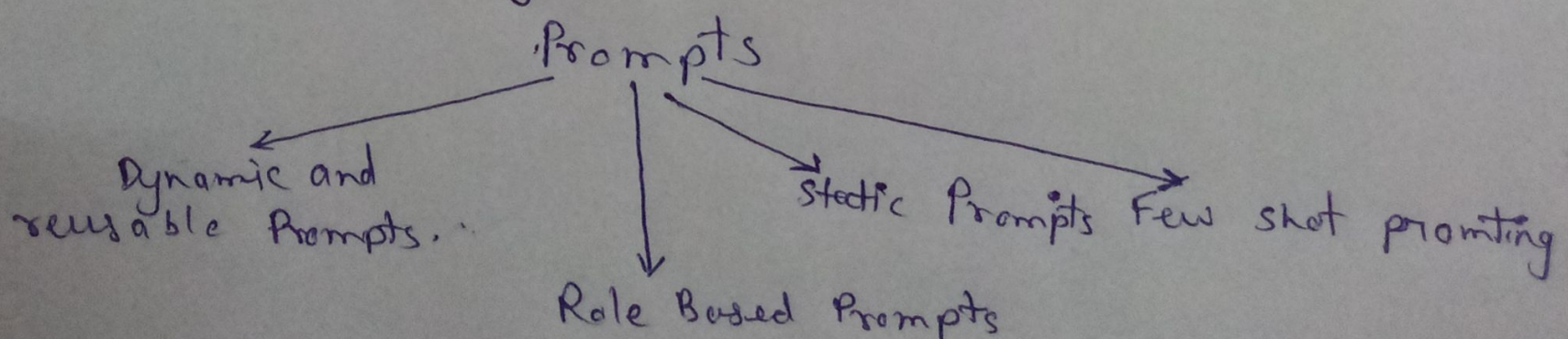
1. Models:

- In LangChain "models" are core interfaces through which you interact with AI models
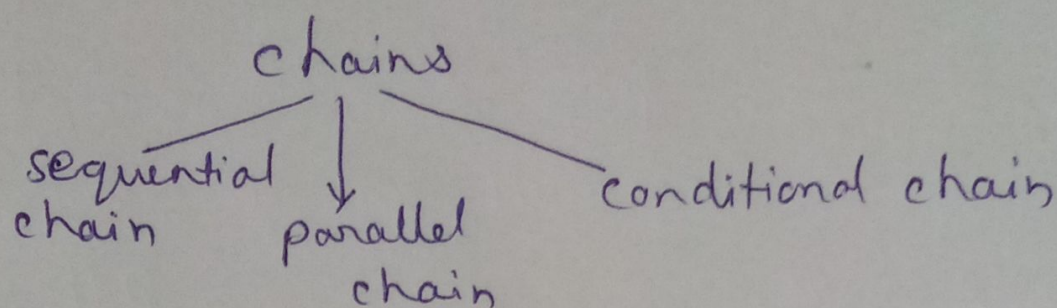- it designed to interaction with language and embedding models

<div align="center">

Models

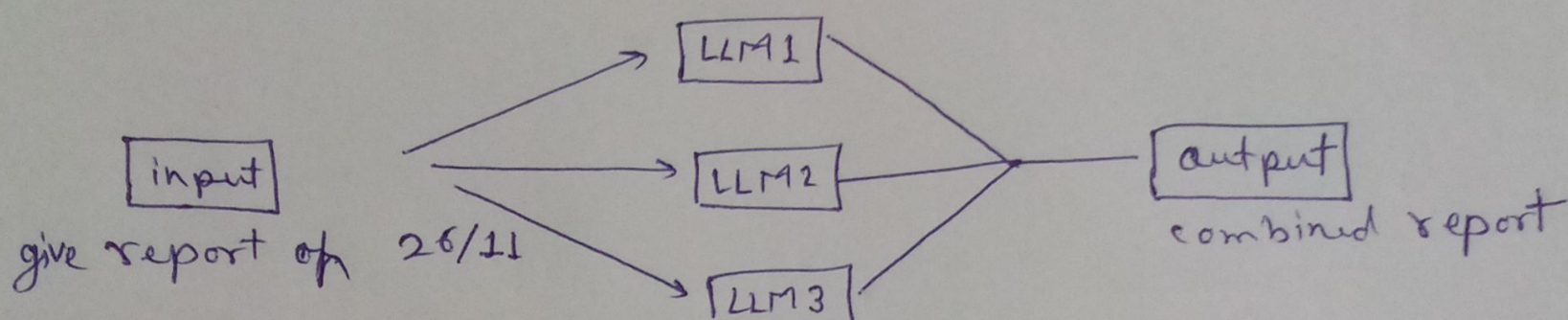</div>

```
              Models
            ↙        ↘
      language        Embedding
    (text to text)   (text to vector)
      ↙      ↘        used in semantic search
    LLM      chat models.
 (base models) (GPT4, Grok)
```

2. Prompts:

Input provided by user.

```
              Prompts
          ↙      │      ↖
  Dynamic and    │    Static Prompts  Few shot prompting
 reusable Prompts.. │
                  ↓
            Role Based Prompts
```

3. Chains:

- it is used for build the pipeline of the project.
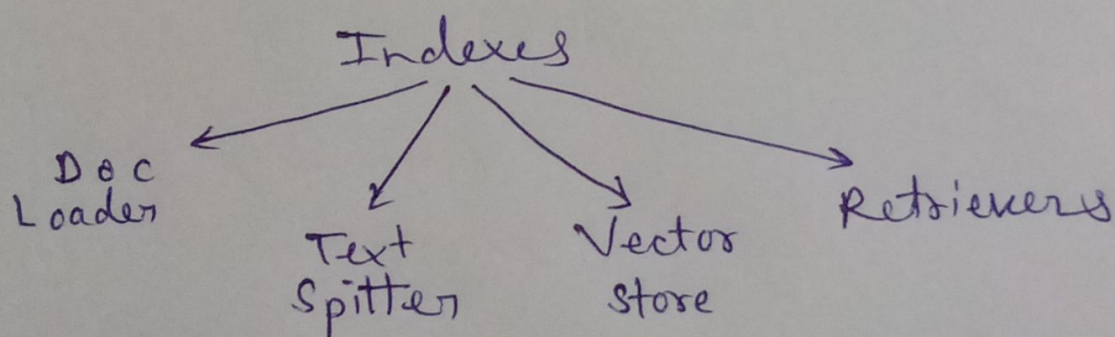- Here the stage of one output become other's input

```
                    chains
          _____|_____
         /            |            \
  sequential       parallel      conditional chain
   chain           chain
```
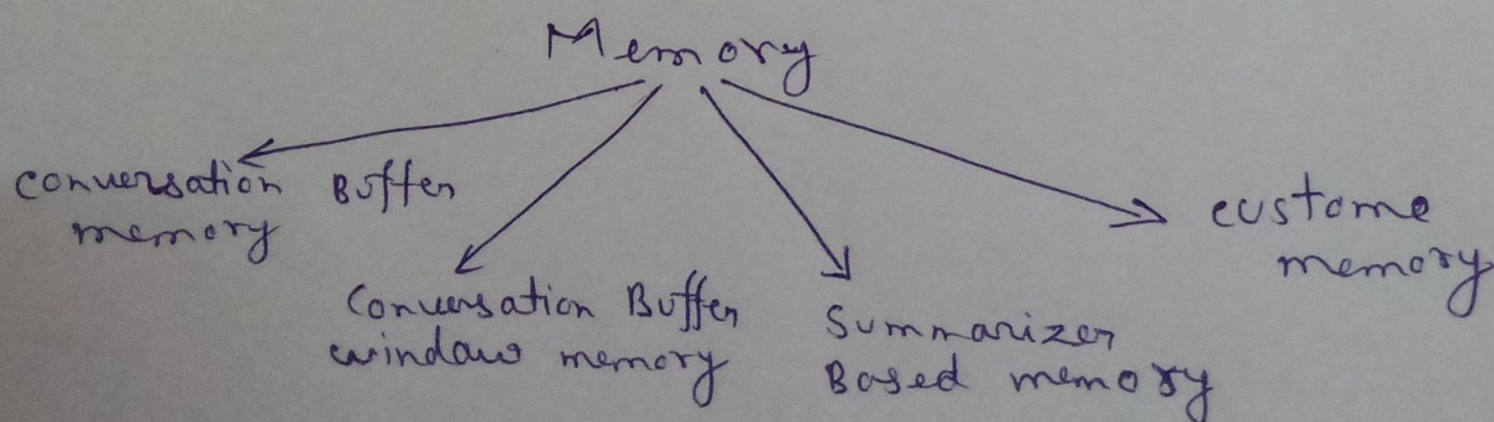
example of parallel chain.

```
                          → [LLM1]
                         /         \
   [input]  ────────────→ [LLM2] ──── • ────→ [output]
                         \         /       combined report
  give report of 26/11    → [LLM3]
```

4. Indexes:

Indexes connect the application to the external knowledge such as PDFs, website or database.

```
                    Indexes
          _____/____|_____
         ↙     ↙      ↘         ↘
   Doc        Text    Vector    Retrievers
   Loader     Spitter  Store
```

5. Memory:

- each request are independent to each other, current request not have knowledge of previous one if we use the LLM's API.
- LLM API calls are stateless

```
                    Memory
          _____/____|_____
         ↙     ↙      ↘          ↘
 conversation Buffer            custome
 memory                          memory
      Conversation Buffer   Summarizer
      window memory         Based memory
```

6. Agents:

AI agents have reasoning capabilities and tools.