Architectural Patterns for AI Recommendation Systems

Software Systems Architecture
Team 32

José Francisco Veiga
Marco Vilas Boas
Pedro Januário
Pedro Lima
Pedro Marcelino

Embeddings

What are they?

They ‘embed’ entities into (relatively) **low-dimensional vectors**, **learned** via **neural networks**.

How can we use them in recommender systems?

Embed both the **users** and the **items** (movies, songs, …) and them **combine them** with an **operation** like dot product or neural network.

Advantages

Scales very well – Captures latent (hidden) features – Works well within Deep Learning frameworks

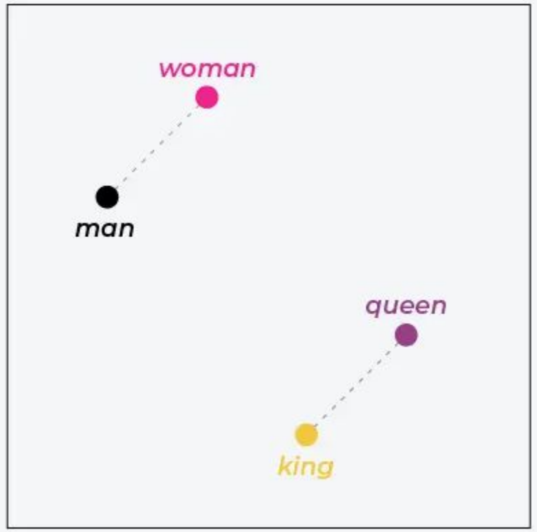
Challenges

Cold starts – Requires lots of data to get good representations – Needs periodic retraining

word	living being	feline	human	gender	royalty	verb	plural
man	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman	0.7	0.3	0.8	-0.7	0.1	-0.5	-0.4
king	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

word

Word embedding



Visualization of word embedding

Adaptive Learning Pattern

How does it work?

Models directly evolve from real-time feedback or changing environments

Core components

User Feedback Loop: Real-time or delayed responses (clicks, ratings, dwell time) continuously update the system.

Model Adaptation: Online learning or periodic retraining using new data.

Exploration vs. Exploitation: Balance known user preferences vs. discovering new ones.

Advantages

Personalization improves over time

Robust to changing user behavior

Challenges

Can overfitting to noise or Short-Term Trends and can also reinforce existing preferences ("rich get richer").

Use Case: Streaming Platform Recommendations

1. Initial Recommendation (Cold Start):

Bootstraps with user metadata or global popularity

2. Adaptive Feedback Integration:

Tracks user interactions

Incrementally updates user profile or item embeddings

Implement techniques to recommend diverse content to avoid creating bubbles

3. Real-Time Model Update Techniques:

Deep Learning for continual learning

Reinforcement Learning for sequential decision making

Outcome:

Recommendations should reflect current user interests

Rapid adaptation to trends, moods, and niche preferences while avoiding overfitting

Better long-term engagement and retention

Role Prompting Pattern

How does it work?

In this pattern, an LLM is instructed to take on a specific **role** or **persona**. Without this guidance, the model's style or assumptions may not match your needs.

How can we use them in recommender systems?

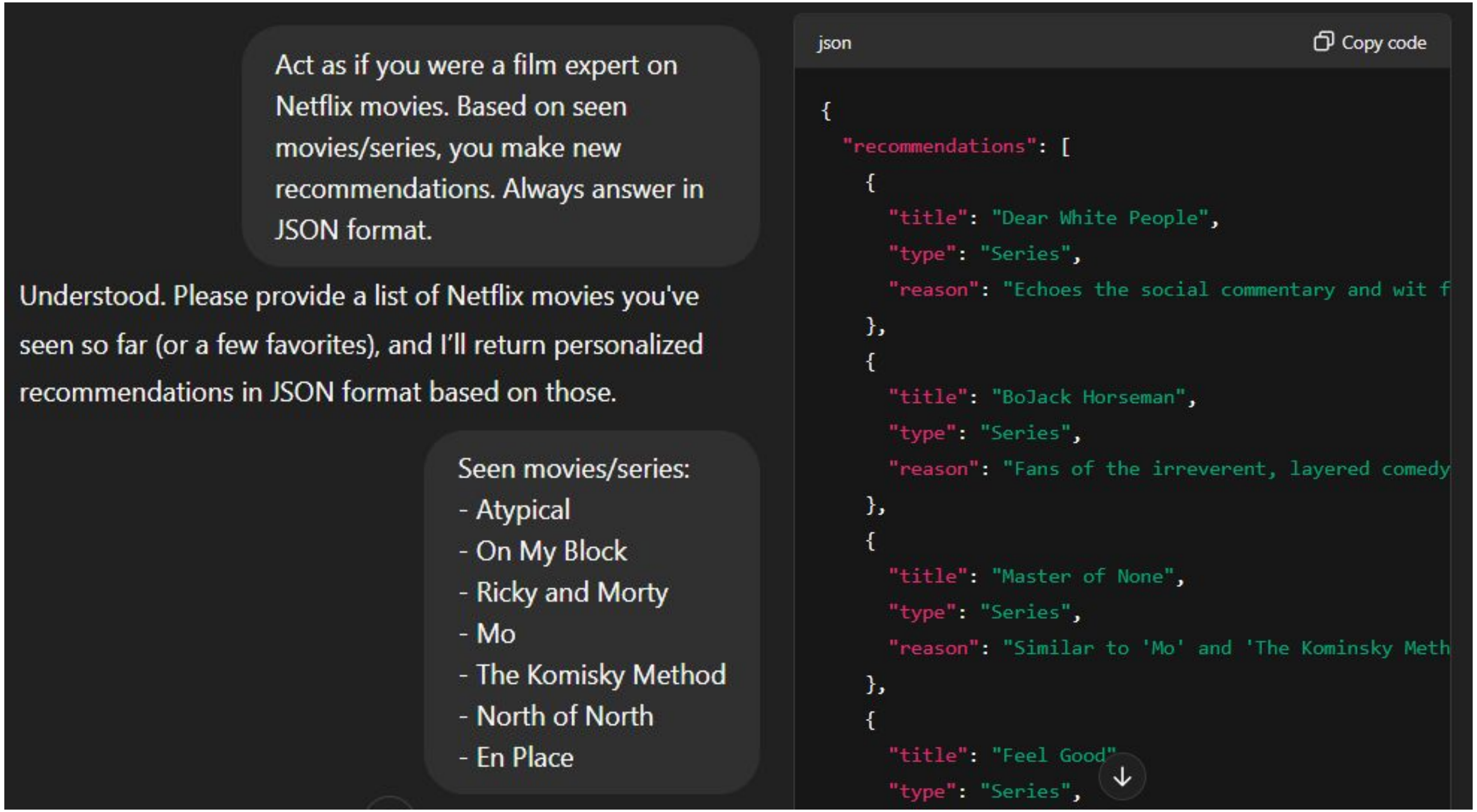
Tell the LLM to assume the role of an expert on the field (movies, songs, ...) and, given some data, ask to suggest instances in the same style. It is also useful to specify the format of the answers.

Advantages

Helps to contextualize and format responses.

Challenges

This is prone to hallucinations. So it is very important to verify if the instances are real.



Act as if you were a film expert on Netflix movies. Based on seen movies/series, you make new recommendations. Always answer in JSON format.

Understood. Please provide a list of Netflix movies you've seen so far (or a few favorites), and I'll return personalized recommendations in JSON format based on those.

Seen movies/series:
- Atypical
- On My Block
- Ricky and Morty
- Mo
- The Komisky Method
- North of North
- En Place

```json
{
  "recommendations": [
    {
      "title": "Dear White People",
      "type": "Series",
      "reason": "Echoes the social commentary and wit f
    },
    {
      "title": "BoJack Horseman",
      "type": "Series",
      "reason": "Fans of the irreverent, layered comedy
    },
    {
      "title": "Master of None",
      "type": "Series",
      "reason": "Similar to 'Mo' and 'The Kominsky Meth
    },
    {
      "title": "Feel Good",
      "type": "Series",
```

Training — Scheduled Batch, Manual ad-hoc, Real Time

What is their purpose?

Defines how a recommendation system's model is trained, i.e. how often and by which means new data are fed to them.

Scheduled Batch mode

Model trained at a regular predetermined interval, relying on a constant data pipeline.

Manual ad-hoc mode

Model trained manually initially and when needed (e.g. significant new data available, performance loss).

Real Time updates

Model trained in real-time according to fresh data, thus relying heavily on the User Interaction.

Tool Use Pattern

What is it?

Agents invoke **external tools** or services to solve subtasks (calculators, search engines, APIs, code interpreters)

How can we use them in recommender systems?

Agents decide **when** and **how** to invoke a tool programmatically.

Advantages

Expand capabilities beyond pretraining | Improve accuracy for specific tasks (math, code)

Disadvantages

Complex | Error handling and fallback needed | Performance & latency issues

References

Rahul Suresh. 2025. Beyond the Gang of Four: Practical Design Patterns for Modern AI Systems. [accessed 29 May 2025]. https://www.infoq.com/articles/practical-design-patterns-modern-ai-systems/.

Ravneet. Jul 2024. Design Patterns for Deploying Recommendation Systems. [accessed 29 May 2025] https://medium.com/design-bootcamp/design-patterns-for-deploying-recommendation-systems-8bb9f2dd5937

Thank you!

Software Systems Architecture

May 2025

Team 32

José Francisco Veiga

Marco Vilas Boas

Pedro Januário

Pedro Lima

Pedro Marcelino