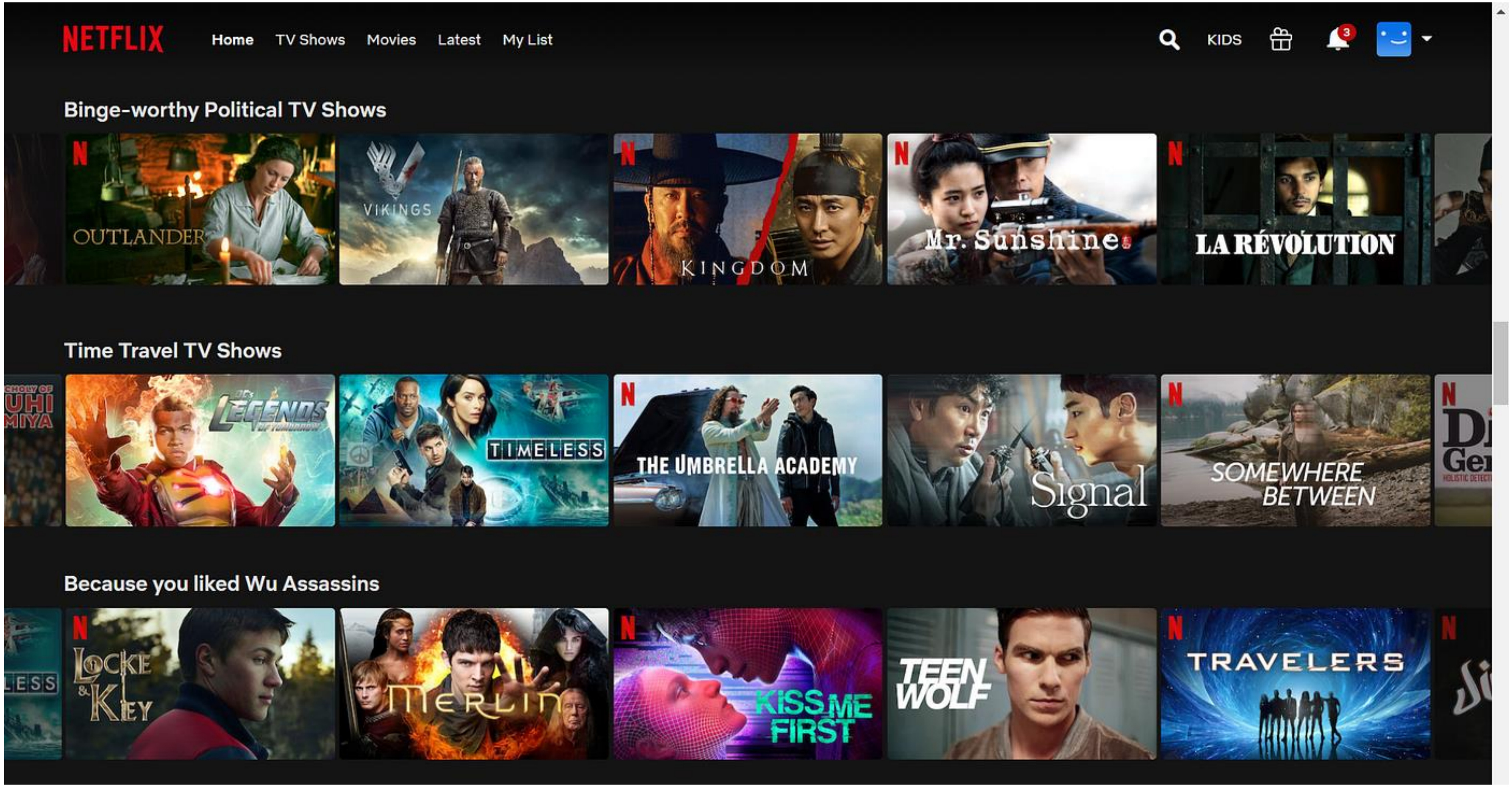
Recommender systems

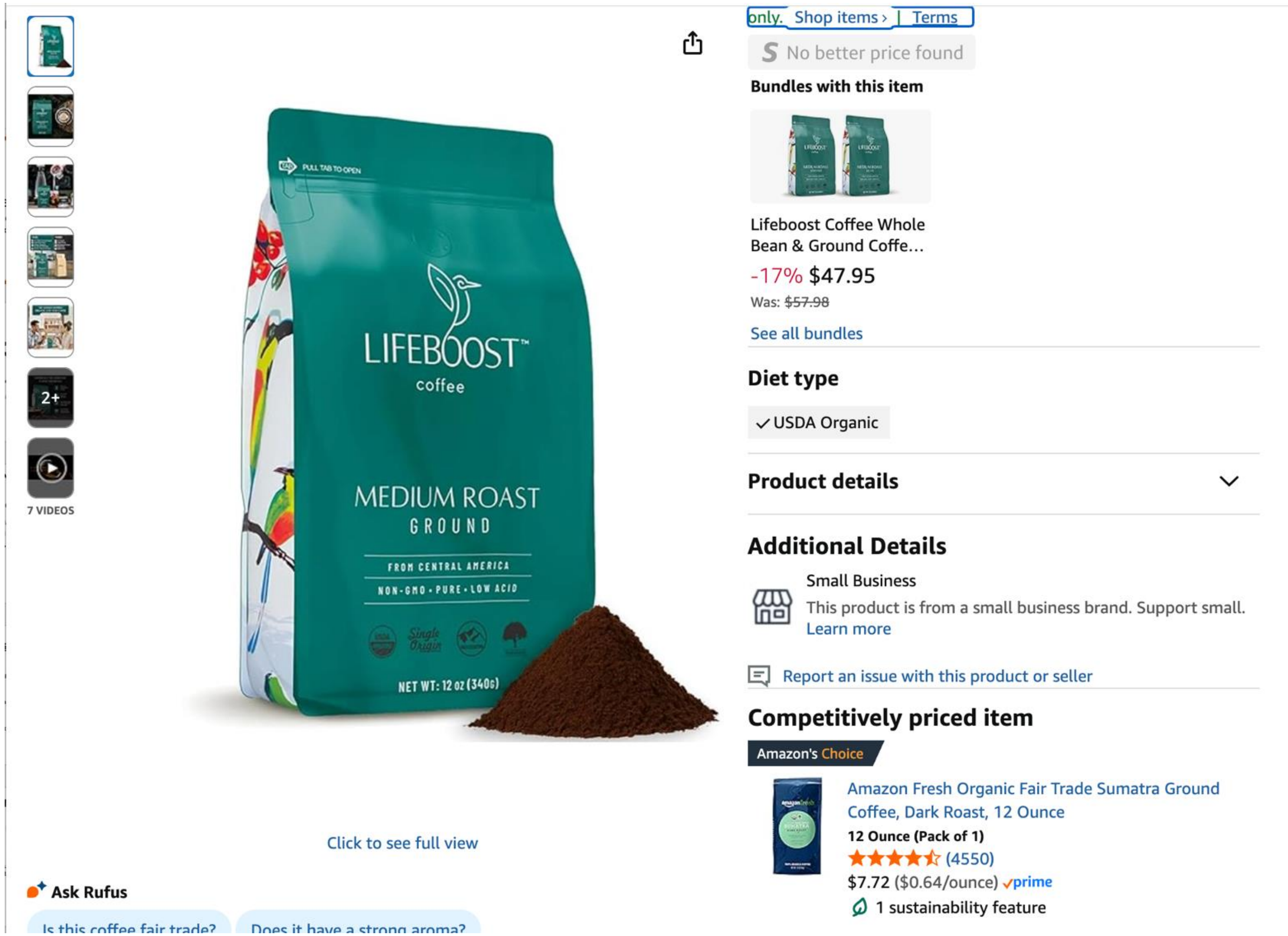
MKT 566

Instructor: Davide Proserpio

Recommendations are everywhere



Recommendations are everywhere



Why are recs important for marketing?

- **Engagement**: More relevant suggestions = more time spent on platform.
- **Conversion**: Better targeting = higher sales.
- **Customer lifetime value**: Stronger loyalty when users feel understood.
- **Trade-offs**: Over-personalization can create “filter bubbles.”

Clustering vs. recommendations

Aspect	Clustering	Recommendation Systems
Goal	Group similar items or people into clusters	Predict what a specific user will like or interact with
Output	Segment labels (e.g., “high spenders,” “price-sensitive”)	Ranked list of personalized suggestions
Approach	Finds structure in data **without labels** (unsupervised learning)	Uses user-item interactions, ratings, or behavior to make predictions (can be supervised or unsupervised)
Personalization	Same cluster members treated similarly	Individualized for each user
Typical Use in Marketing	Customer segmentation for targeting strategies	Product/content recommendations for each customer

Clustering vs. recommendations

- **Clustering** → "Organizing your customers into a few big buckets based on similarity"
- **Recommendations** → "Telling *this* customer what they're most likely to want next"

How can we implement recommendations

Tons of options based on simple data mining or more complex machine learning algorithms

How can we implement recommendations

Tons of options based on simple data mining or more complex machine learning algorithms

Aspect	Data mining	Machine learning
Primary goal	Discover patterns, segments, anomalies, associations	Learn a model to **predict** or **decide** on unseen cases
Typical output	Rules, clusters, summaries, dashboards, hypotheses	A trained model (e.g., classifier, regressor, recommender)
Orientation	Descriptive/explanatory ("what's in there?")	Predictive/optimization ("what will happen?")
Examples	Association rules (A→B), clustering segments, outlier detection	Churn prediction, demand forecasting, recommendations, Natural Language Processing
Evaluation	Interestingness, support/confidence/lift, business interpretability	Accuracy/AUC/RMSE, calibration, loss, offline/online metrics

Data mining recommenders

Association rule mining

Helpful for finding “what goes with what”

- A data mining technique to **discover relationships between items** in large datasets
- Often used to find **patterns of co-occurrence** in transactions
- Classic example: Customers who buy **bread** often also buy **butter**
- **Marketing Applications**
  - **Market basket analysis** → Which products are often bought together?
  - **Cross-selling** → “Frequently bought together” recommendations
  - **Store layout** → Place associated products near each other
  - **Promotion bundling** → Offer discounts on items often purchased together

Association rule mining

Transaction ID	Bread	Butter	Milk	Beer	Diapers
1	1	1	0	0	0
2	0	0	1	1	1
3	1	0	1	0	0
4	1	1	1	0	0
5	0	0	0	1	1

From here, the algorithm looks for **frequent itemsets** and then generates rules like:

- **Rule:** {Diapers} → {Beer}
  - **Support:** 2% of all transactions contain both
  - **Confidence:** 60% of diaper buyers also buy beer
  - **Lift:** 1.5 → diaper buyers are 50% more likely to buy beer than average

Association rule mining

Given the rule: A → B:

- **Support**: % of transactions containing both A and B

$$\text{support}(A, B) = \frac{\text{count}(A,B)}{\text{total transactions}}$$

- **Confidence**: % of transactions with A that also have B

$$\text{confidence}(A \to B) = \frac{\text{count}(A,B)}{\text{count}(A)}$$

- **Lift**: How much more likely B is bought with A vs at random

$$\text{lift}(A \to B) = \frac{\text{confidence}(A \to B)}{\text{support}(B)}$$

Example

- **Support**: % of transactions containing both A and B
  $\text{support}(A, B) = \frac{\text{count}(A,B)}{\text{total transactions}}$
- **Confidence**: % of transactions with A that also have B
  $\text{confidence}(A \rightarrow B) = \frac{\text{count}(A,B)}{\text{count}(A)}$
- **Lift**: How much more likely B is bought with A vs at random
  $\text{lift}(A \rightarrow B) = \frac{\text{confidence}(A \rightarrow B)}{\text{support}(B)}$

100 total transactions

- 2 transactions: diaper + beer
- 1 transaction: diaper only
- 38 transactions: beer only
- 59 transactions: neither

Let's compute support, confidence, and lift for diapers → beers

Example

- **Support**: % of transactions containing both A and B
  $$\text{support}(A, B) = \frac{\text{count}(A,B)}{\text{total transactions}}$$
- **Confidence**: % of transactions with A that also have B
  $$\text{confidence}(A \to B) = \frac{\text{count}(A,B)}{\text{count}(A)}$$
- **Lift**: How much more likely B is bought with A vs at random
  $$\text{lift}(A \to B) = \frac{\text{confidence}(A \to B)}{\text{support}(B)}$$

100 total transactions

- 2 transactions: diaper + beer
- 1 transaction: diaper only
- 38 transactions: beer only
- 59 transactions: neither

Let's compute support, confidence, and lift for diapers → beers

Support = 2 / 100 = 2%
Confidence = 2 / 3 = 66.7%
P(beer) = 40 / 100 = 40%
Lift = 0.667 / 0.40 = 1.67 ≈ 1.5

Machine Learning

Main approaches

- **Collaborative filtering**: “People like you also liked these.” (e.g., MBA/Marketing students like R → recommend Python).
- **Content-based filtering**: “What you liked in the past predicts what you’ll like in the future.” (e.g., you liked a sci-fi book → recommend another sci-fi book).
- **Hybrid models**: Most platforms mix both.

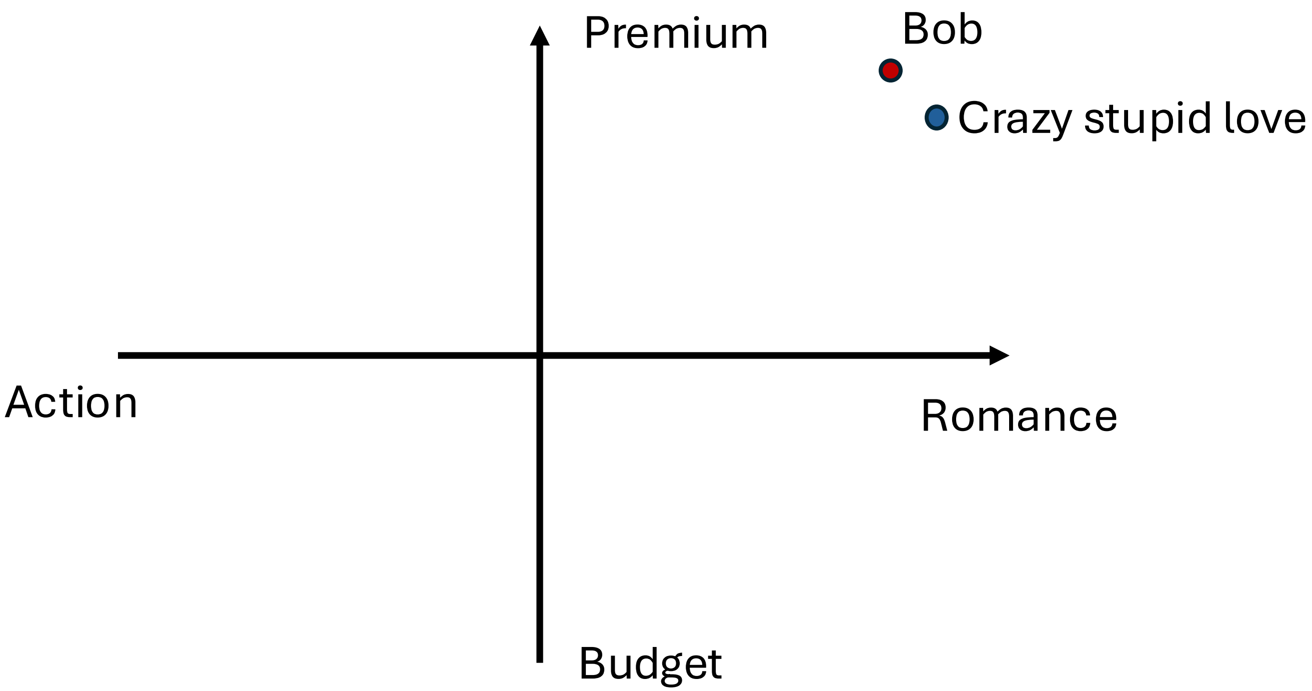
Core idea: Embeddings

- **Embeddings** are a vector representation of a user or item
- **Similarity** measures how close are two vectors
  - Dot product
  - Cosine similarity
- **Learned embeddings** capture **latent factors** (taste for genre/price/brand).
  - Nearby items/users share behavior even without identical histories
- Rec systems differ in how they learn and create these vectors
- Different models, **same idea**: get vectors for users/items and recommend using **nearest neighbors in embedding space**.

Mental picture (movies example)

Let's assume we have two-dimensional vectors (x, y) where:

- X: measure the continuum action ←→ romance
- Y: measure the continuum budget ←→premium



Collaborative filtering

- A **recommendation method** that predicts a user’s interests by **learning from the preferences of other users**
  - It requires **user-items interactions**
- Assumes that **similar users** will like similar things
- The “collaborative” part: the system utilizes the collective behavior of multiple users to make predictions

Collaborative filtering

User-Based Collaborative Filtering: Recommendations are made to a user based on what similar other users have liked.

User / Movie	**The Matrix**	**Titanic**	**Toy Story**	**The Godfather**	**Inception**
User 1	1	0	1	0	1
User 2	1	1	0	1	0
User 3	0	1	1	0	0
User 4	1	0	0	1	1
User 5	0	1	0	1	1

Main issue with collaborative filtering

“Cold start” problem: If I don’t have data about a user past choices, it is difficult to know what they will like

Content based recommenders

- Recommend items similar to a user's past choices

- Example: Movie recommender
  - A content-based recommendation system recommends movies to a user by considering the similarity of movies.
  - For example, we can recommend movies based on the movie description.

- Risk: **“filter bubble”** → recommendations are too similar to past choices so consumers do not try anything ”new” or “different”

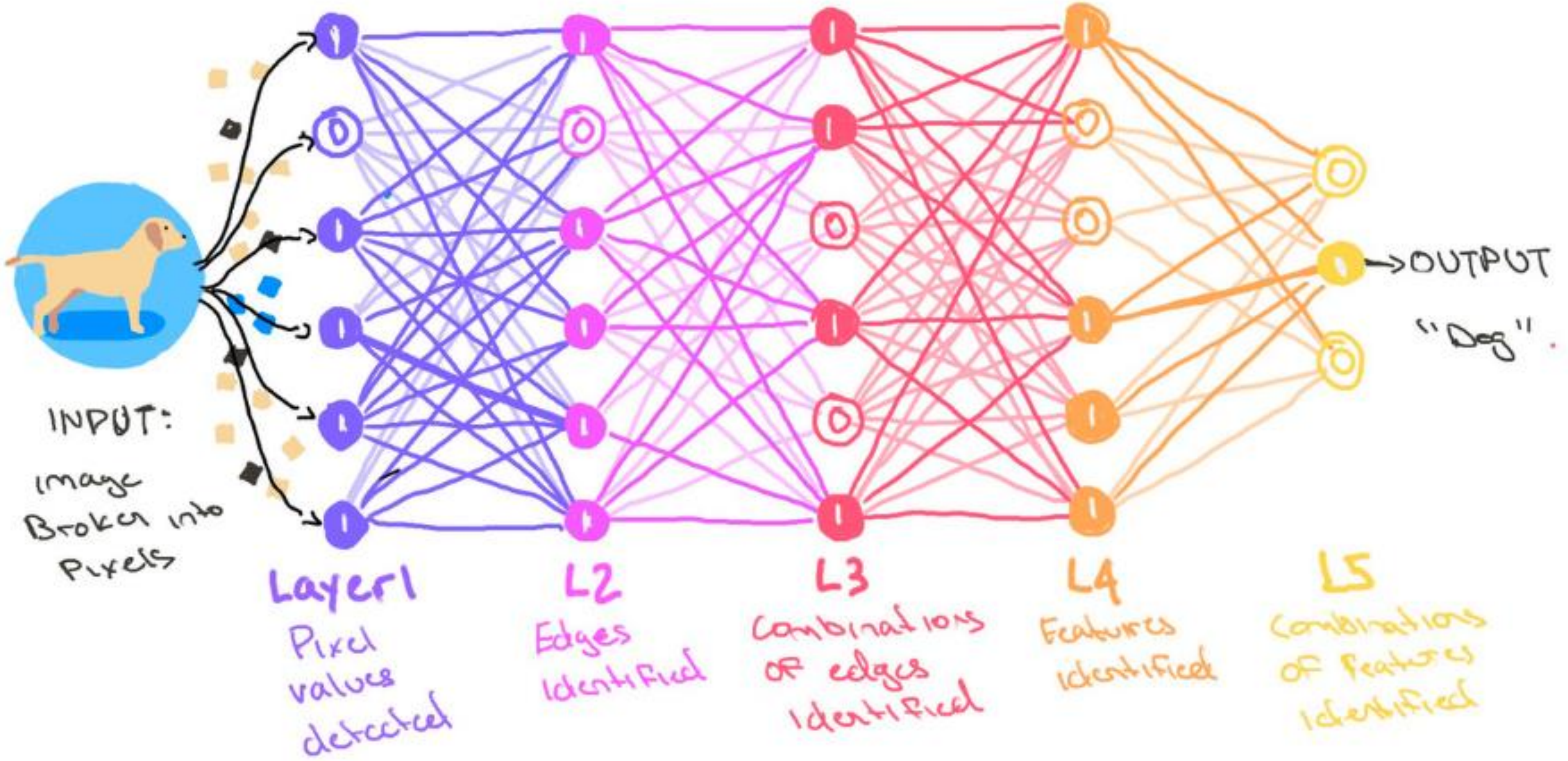
- Great to address the cold start problem since they don’t require too much past user behavior

Deep learning & Large Language Models

Deep learning & Large Language Models

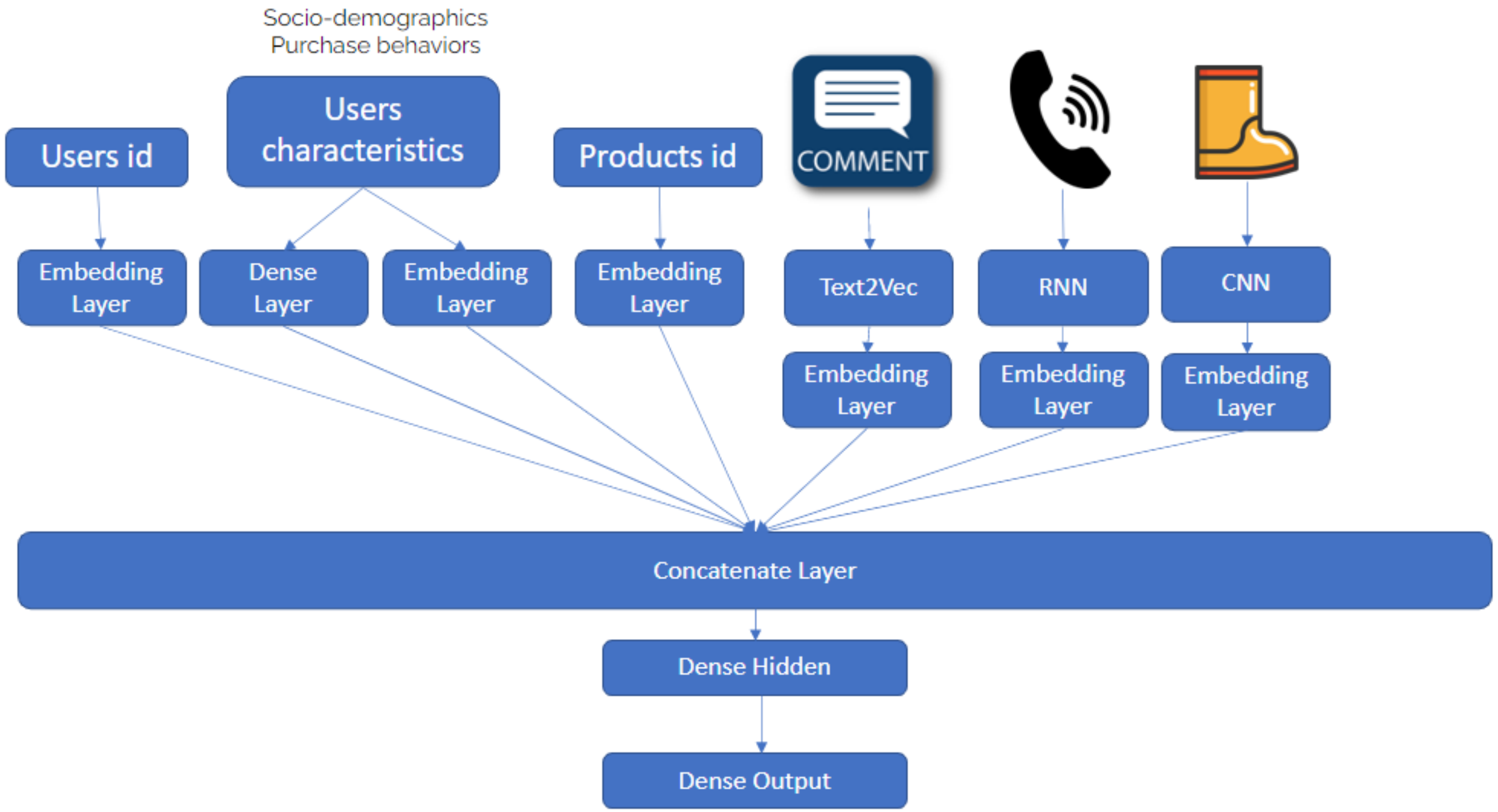
- **What is Deep Learning?**
  - A type of machine learning that uses **neural networks** with many layers ("deep").
  - Each layer learns to extract more **complex patterns** from data.
  - Works on **images, text, audio, clicks, videos** → almost any type of data.
- **What are LLMs?**
  - **Large Language Models** are deep learning models trained on vast amounts of text.
  - Can **understand, generate, and reason with language** (e.g., ChatGPT, Claude, Gemini).

Deep learning & Large Language Models



Deep learning & Large Language Models

They optimize/improve how we “embed” consumers/products because they rely on much more, and complex, data



Deep learning: content based

- Build “item” vectors from text/images/attributes of the items
  - E.g., movie vector: description + video + dialogues + cast + ratings + box office

- “Simple” approach for text data: Word2Vec, Doc2Vec, any LLM model these days:
  - E.g., Movie recommender: https://github.com/devalindey/Recommender-Systems-using-Word-Embeddings

Evaluation: Offline Metrics

- **Accuracy-based**
  - Precision@k → % of top-k recommendations that are **relevant**
  - Recall@k → % of relevant items captured in top-k
- **Coverage & Diversity**
  - How much of the catalog is recommended?
  - Are recommendations varied or too narrow?
- **Novelty**
  - Are users exposed to less popular / surprising items?

Evaluation: Offline Metrics

Precision@k:

- Precision@k $= \frac{\#\{\mathit{relevant\ items\ in\ top}\text{-}k\}}{k}$
- **(**Fraction of recommended items that are relevant.)

Recall@k:

- Recall@k $= \frac{\#\{\mathit{relevant\ items\ in\ top}\text{-}k\}}{\#\{\mathit{all\ relevant\ items}\}}$
- (Fraction of relevant items that are recommended.)

Evaluation: Online metrics

- **CTR (Click-Through Rate)** → Do users click on recommended items?
- **Conversion / Revenue Lift** → Do recs increase sales, bookings, streams?
- **Engagement / Retention** → Do users come back more often?

Beyond Metrics

- **Fairness / Bias** → Do recs treat products & users equitably?
- **Explainability / Transparency** → Can users understand “why” an item is recommended?
- **Long-Term Value** → Do recs build loyalty, not just short-term clicks?