oda | Mathem

Neural methods in online groceries

Agenda

1. Embeddings in groceries
2. Neural recommendations and explore/exploit
3. Agents



0. Intro

About Lars, Fredrik and Oda

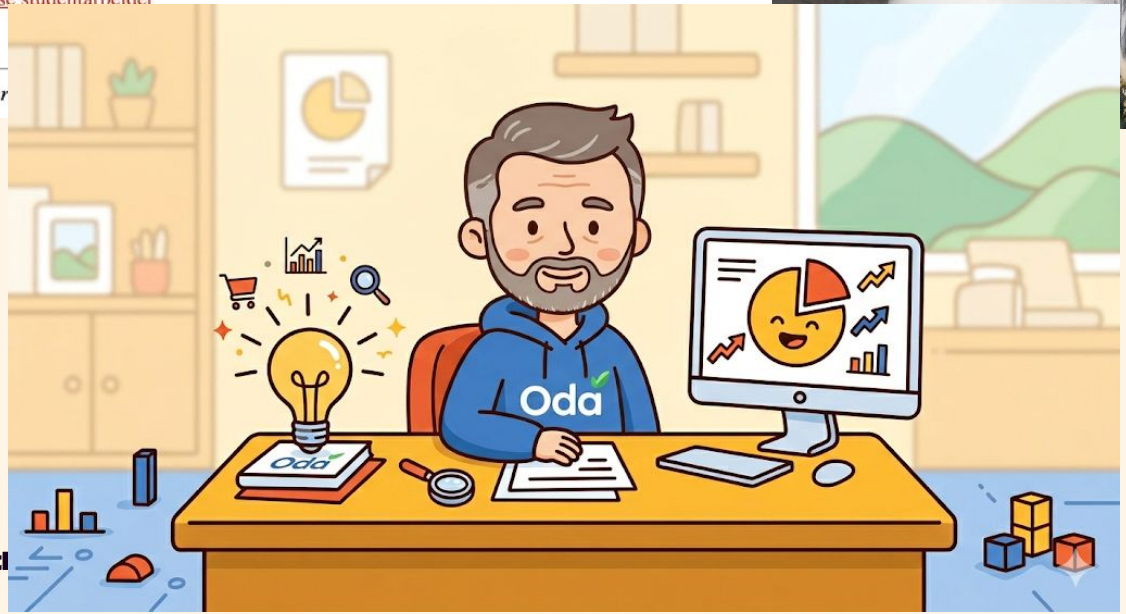
Lars Mushom

- Data Scientist at Oda for 4.5 years.
- Worked previously in the AI group at SINTEF.
- Master thesis on *Deep Generative Models* from NTNU.

Fredrik Jørgensen







History of Oda

- 2013: **Kolonial.no founded** by 10 people in Norway with a vision to make grocery shopping effortless

- 2014–2017: **Iterating on Logistics** – While competitors relied on picking groceries in existing stores, Oda began building its own fulfillment center. They realized early that the "secret sauce" wasn't just the shop, but the efficiency of the warehouse.

- 2018–2020: **World-Class Efficiency** – Oda achieved some of the highest picking productivity rates in the world, mostly due to process optimization, data, and algorithms “everywhere”.

- 2021-2023: Going **international** (Finland and Germany) and rebranding to **Oda**, fueled by a massive investment from SoftBank and Prosus. In 2023, shifting focus **back to Norway**, **and** exiting Finland and Germany, due to a changing economic climate and new investors

- Late 2023 - now: **Merged** with Swedish Mathem, and working our way towards becoming the first online grocery company to be **profitable**

Customer Journey



Hear about Oda/Mathem

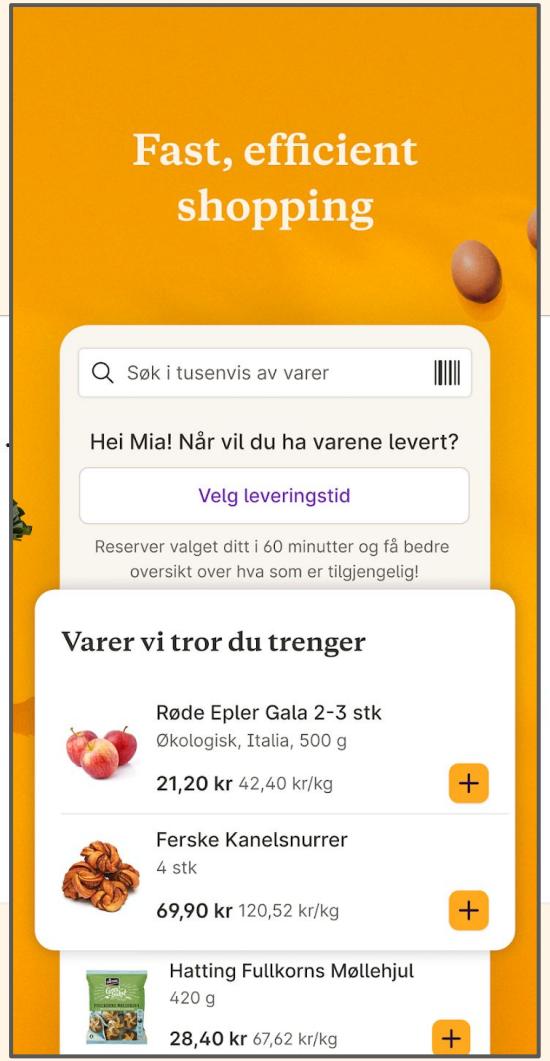


Visit / Sign up

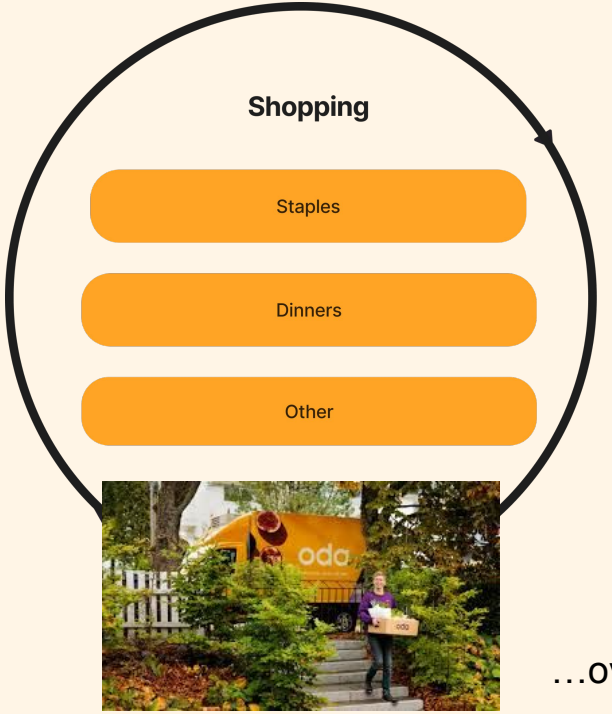


Place order

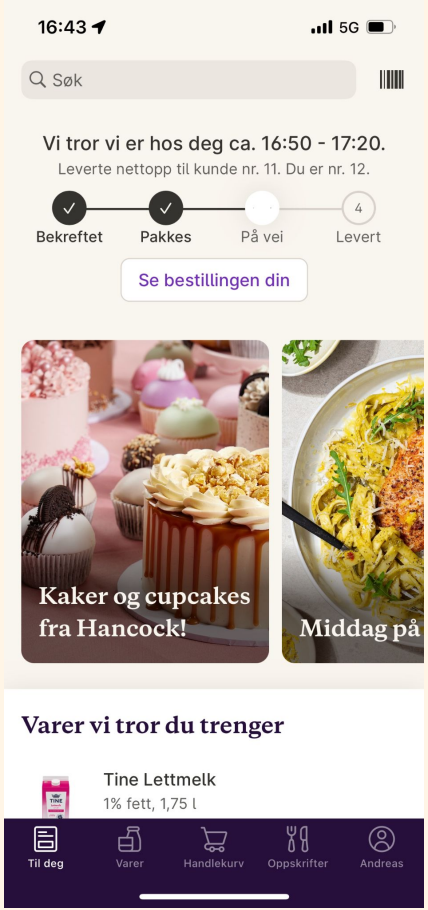




Shopping at Oda



…over and over



2. Embeddings

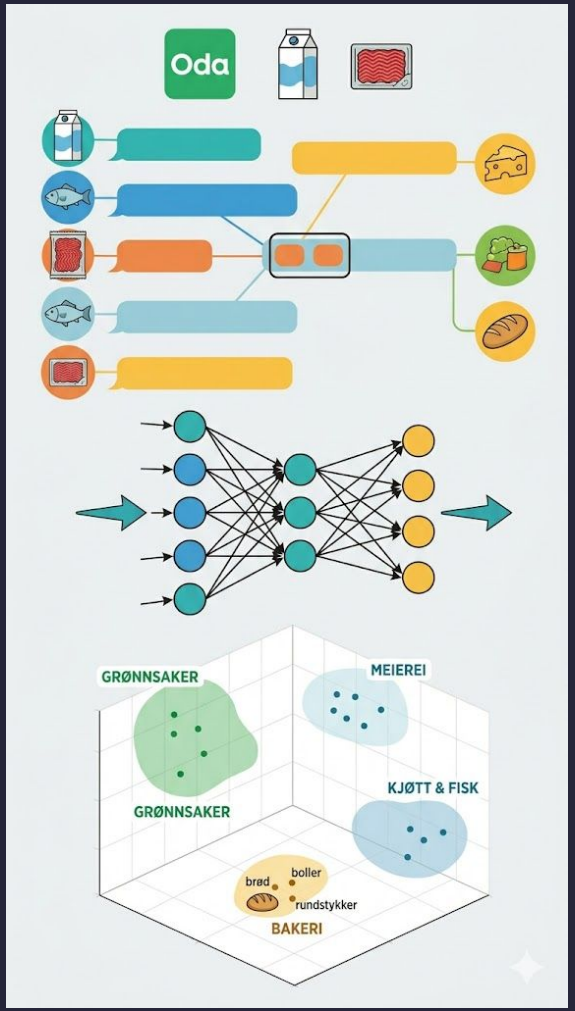
How products are words, and also
how words are words too

User problem: “I want to find alternative products”

1. **Embeddings**
2. Recommendations
3. Agents

How products are words

A grammar of groceries



oda | Mathem

Word2Vec – Learning Representations from Context

- **The Distributional Hypothesis**
  "You shall know a word by the company it keeps" (Firth, 1957)
  Core Idea: Words appearing in similar contexts should have similar vector representations.
  Goal: Map every word `w` in a vocabulary `V` to a dense, low-dimensional vector $\mathbb{R}^d$ $(d \ll |V|)$.

- **Two Main Architectures**
  Example:
  CBOW: Predict the center word
  Skip-gram: Predict surrounding words

  I had lasagna for dinner
  I had _____ for dinner
  _ __ lasagna ___ _____

- **The Training Trick**
  We treat word embedding as a “fake task”. We don't actually care about predicting the next word, we only care about the weights of the hidden layer.

Input:	One-hot encoded vector of the word.
Hidden Layer:	A linear projection (this is your embedding lookup table).
Output:	Softmax layer calculating probabilities for context words.

Word2Vec @ Oda

Sentences = Orders

Words = Products

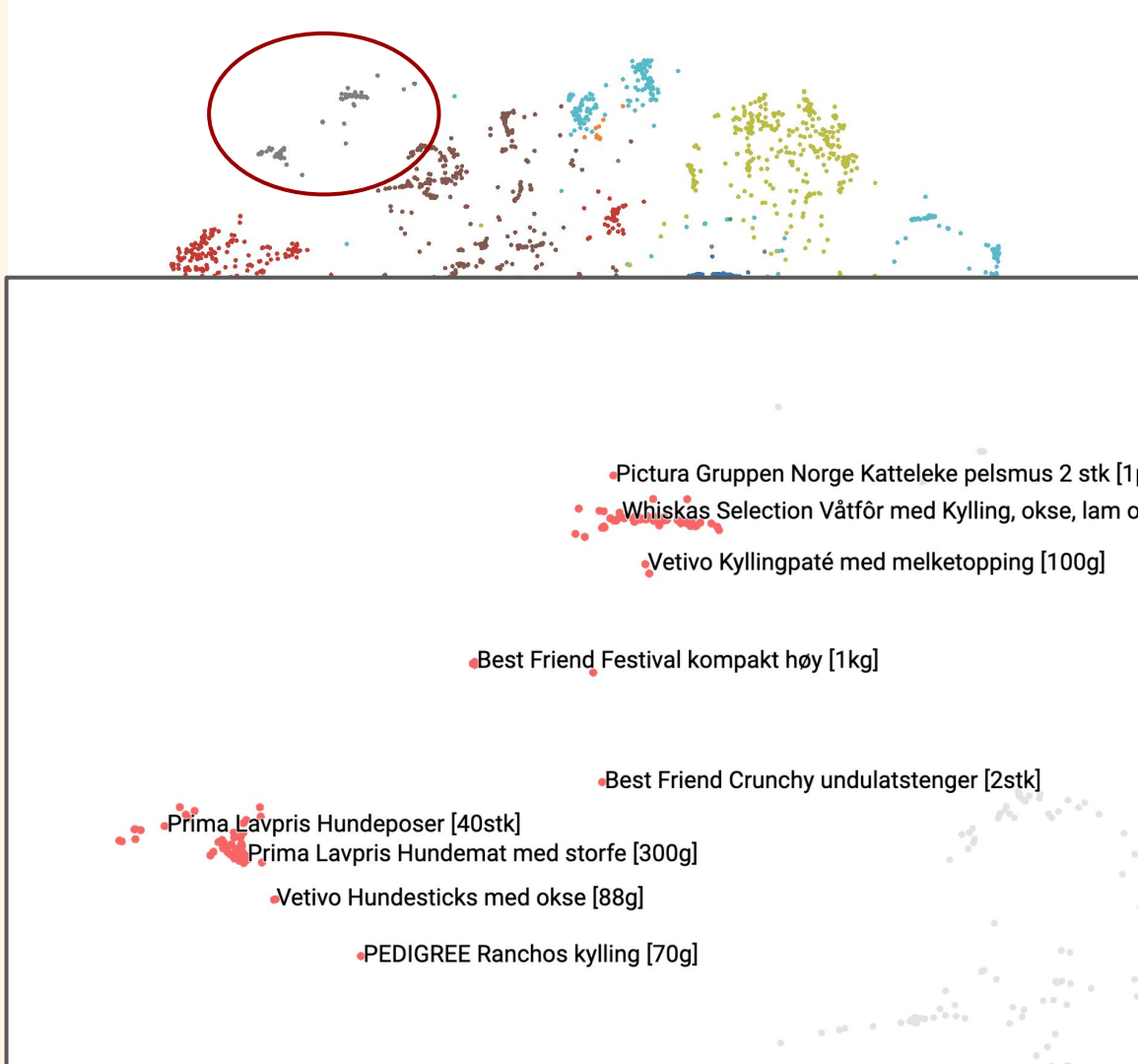
Hypothesis: There is a “grammar” of groceries



Visualisation

- All Norwegian products in a 2D space (from 128D)
- Tensorboard Projector
- T-SNE
- Colored by their Oda category



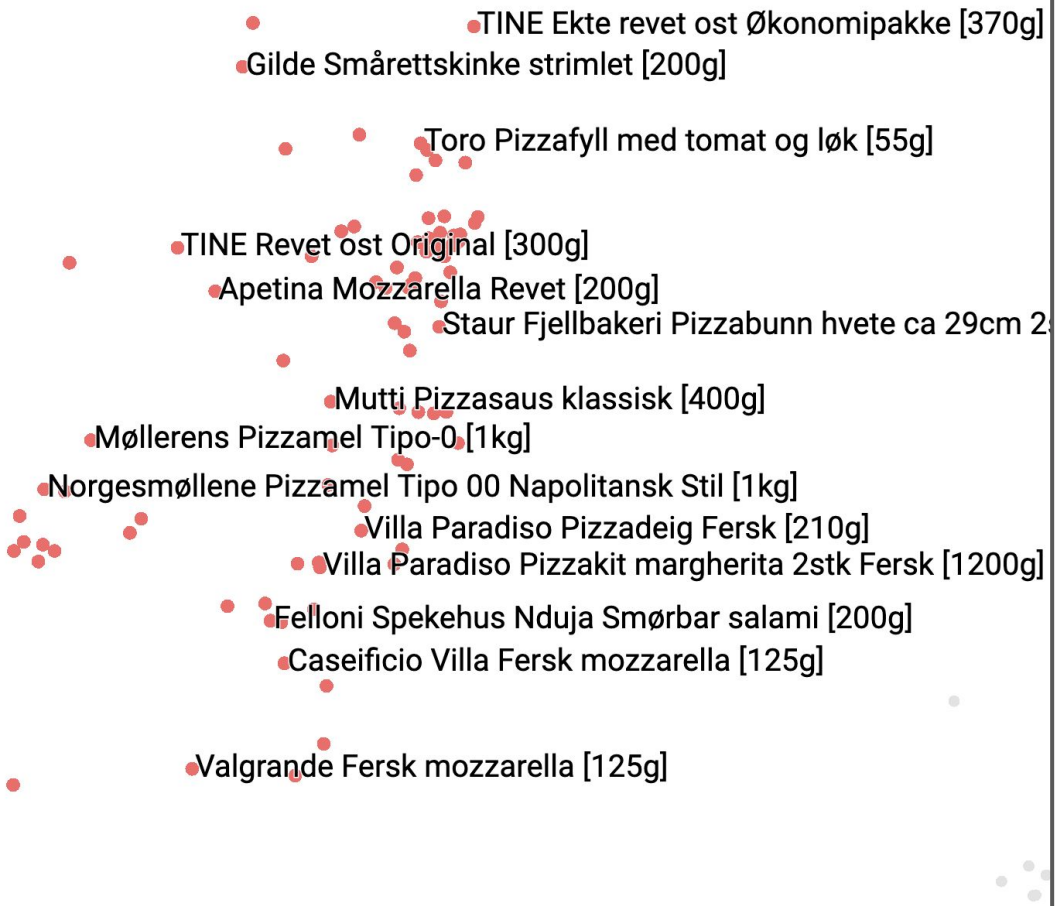


oda | Mathem

Visualisation

- All Norwegian products in a 2D space (from 128D)
- Tensorboard Projector
- T-SNE
- Colored by their Oda category





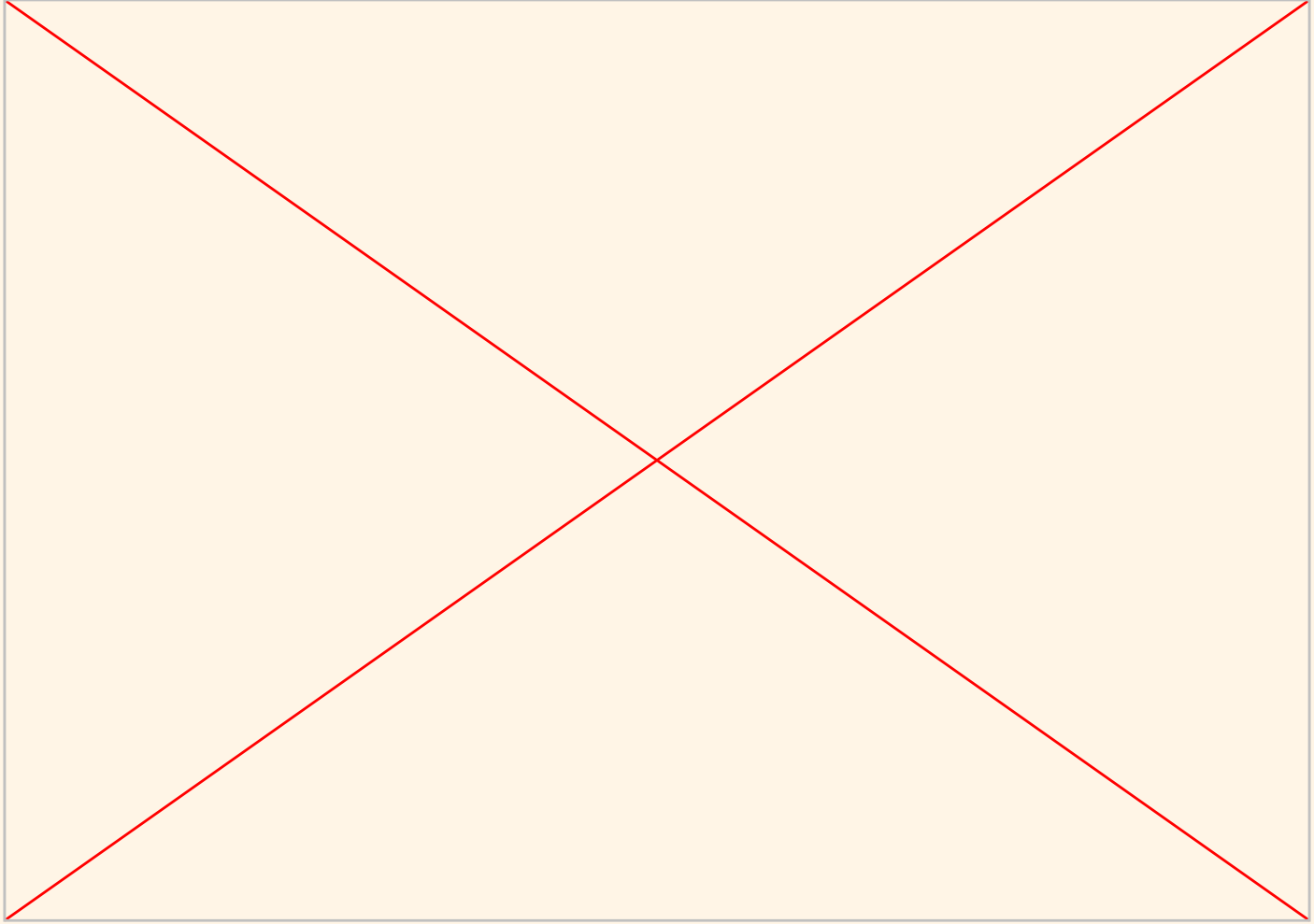
Sparse data, new products and “cold start”

Just as with infrequent words, the infrequent products will also have poor embedding quality

Too be fair, we often don’t know if these are caused by sparsity, or just undesirable/taboo purchasing patterns...



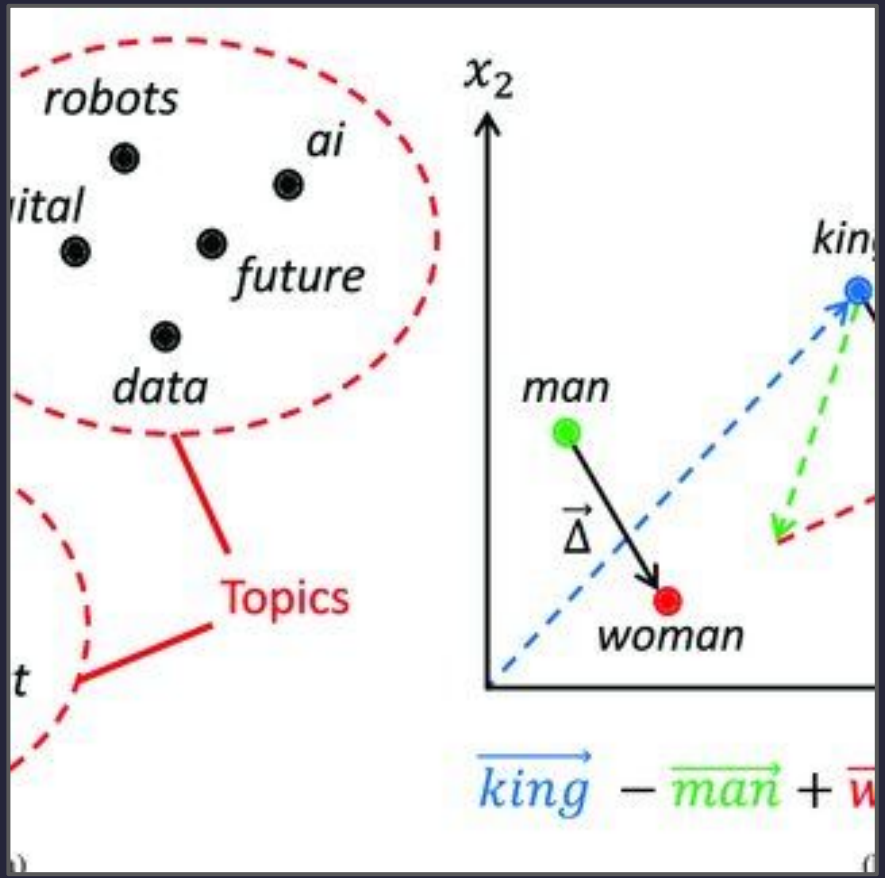
So, what it does the embeddings look like in the app?



How words are words too

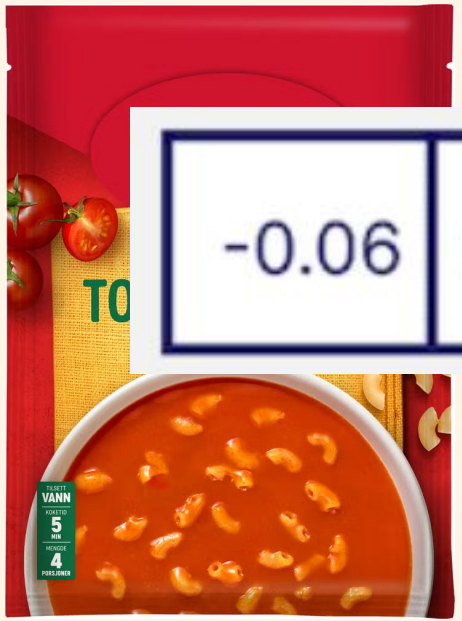
Mixed embeddings

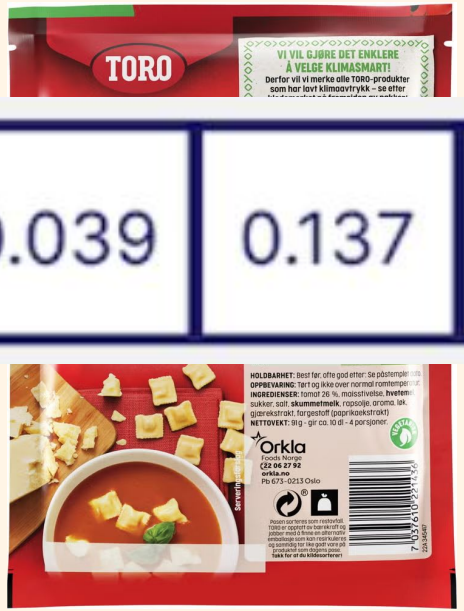
User problem: “I see weird products”



Embedding the products from text

Hjem / Alle produkter / Middager og tilbehør / Middagsbaser / Posesupper





-0.06	0.039	0.137	...	0.045

- Toro
- Tørrmat
- Tomat
- Skummetmelk
- Løk
- Paprikaekstrakt
- middag
- middag
- base
- t
- Hytte mat
- Middagsrett
- Posesuppemix
- Kokesuppe

Combining the products linearly

Gensim / behavior



LLM / text



gives us two similarity scores

Combining the products linearly

Linear mix:

$\alpha \times \text{gensim_similarity} + (1 - \alpha) \times \text{text_similarity}$

Interpreting α:

- Larger α → more behavioral (gensim)
- Smaller α → more text (LLM)

We constantly tune this based on user feedback

2. Recommendations

Recommendations, personalisation,
transformers and towers

1. Embeddings
2. **Recommendations**
3. Agents

The Exploit vs. Explore tradeoff

- What restaurant to go to? What movie do you want to watch? Or what job do jo want to have? **How to make a decision under uncertainty***?*

- **Exploit**: choosing the best option based on the information that you have.
- **Explore**: trying a new option that might lead to better reward in the future.

- Assumption is that we can measure the **reward.**
- The goal is to maximize the long-term benefit, or the **cumulative reward.**

- **This is a fundamental tradeoff in recommendations systems.**

The Exploit vs. Explore tradeoff

At Oda/Mathem:

- **Problem**: what product or recipe do we show to each user at what time?
- **Objective**: maximize number of products/recipes added to your cart.
- **Constraint**: limited space on a phone screen.

What happens if we only exploit?

- Filter bubble.
- Over time this will also corrupt the training data.



The Exploit vs. Explore tradeoff

We can address this tradeoff in both **product design** and **model development.**

We'll look at three different models that is used in different part of the shop:

1. **Next-order prediction:** exploit-heavy model that predicts what you are most likely to shop in your next order.
2. **Two towers:** explore-heavy model for general preference and what separates you from other users.
3. **Online bandit models:** balances explore/exploit based on real-time events while you shop.



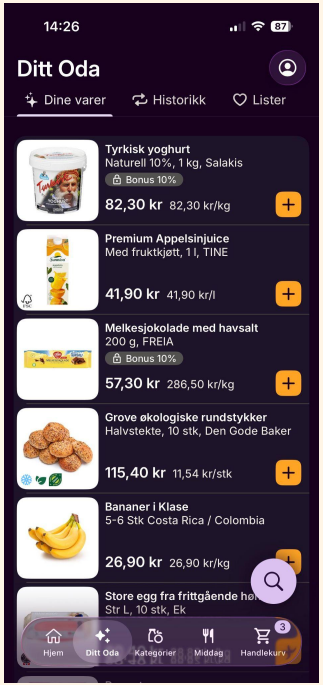
Next-order prediction

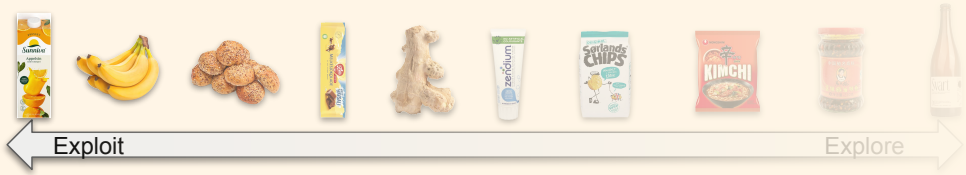
Next-order prediction

The goal is to predict all your usual items that you need now.

Presented in a “infinite scroll” where you can quickly fill up most of your cart.

Pure exploit, not as good at finding interesting products you have not bought before.





oda | Mathem

The NLP parallel: Next-word prediction

NLP concept	Recommendation equivalent
Word	Product ID
Sentence	Single order (cart)
Document	Order history (5 carts)
Vocabulary	Product catalog (~10,000 items)
Next word prediction	Next order prediction

Modeling the problem

Frame this as a classification problem

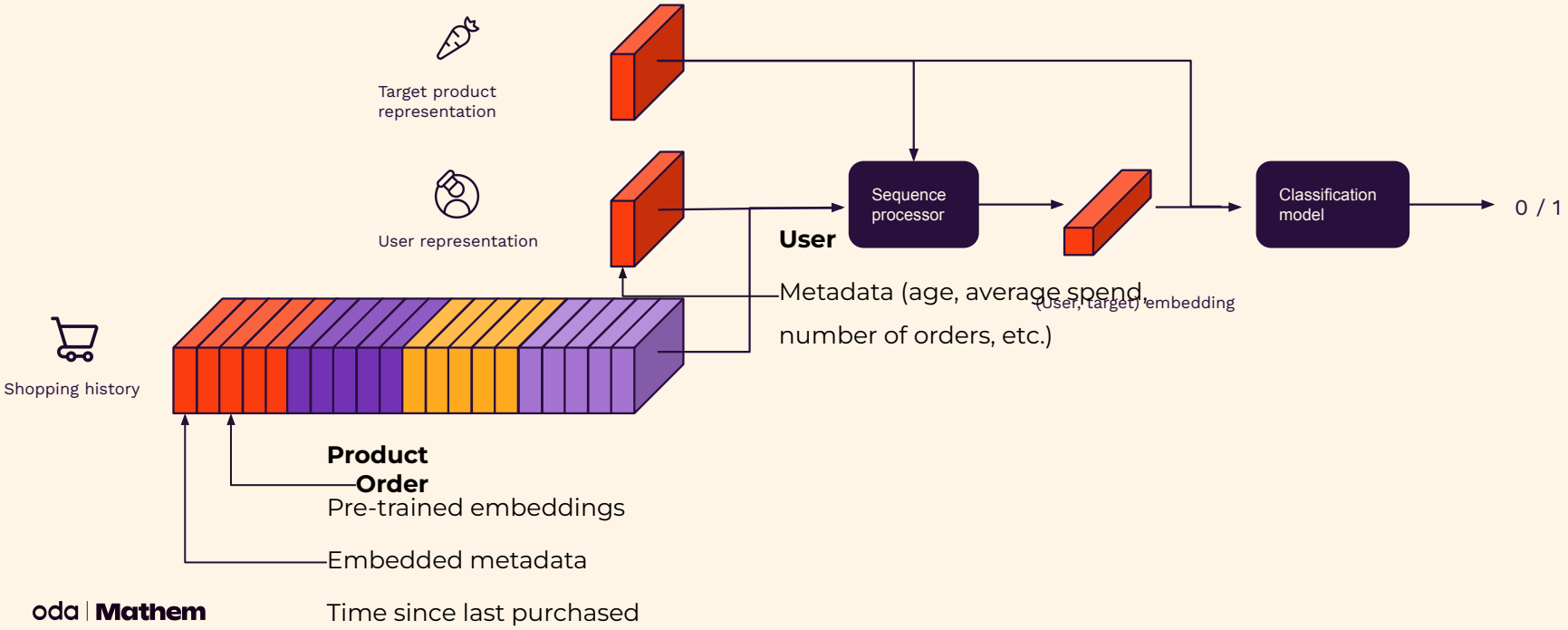
For each user *(user, product)* pair, will this product be in the users next order?

Sequence matters

Repeat purchases, frequencies, recency, dependencies between products etc.

We are using **attention** to model the sequence patterns.

Architecture

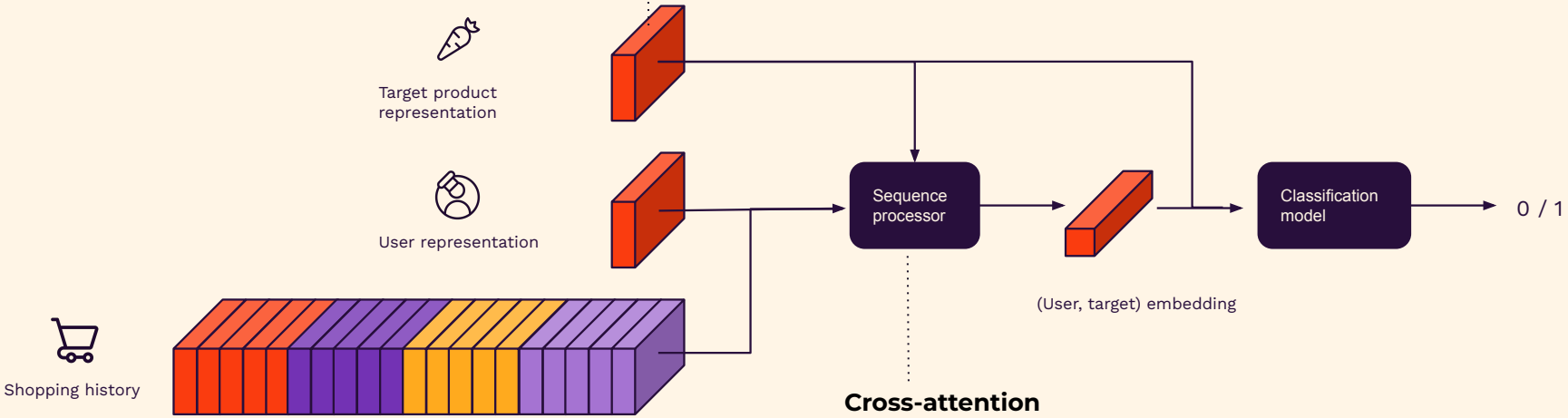


oda | Mathem

Architecture

Trainable target embedding

The embedding can adapt to the training objective



Cross-attention

“Given that the candidate is pasta sauce, which of my past purchases are relevant?"

Training

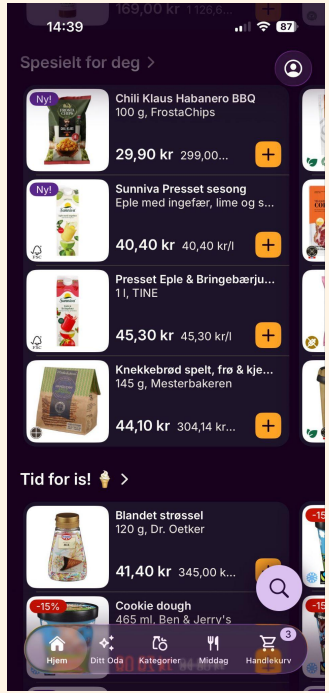
- We sample batches of positive and negative target products that we train the classification model on.
- Random sampling of negatives would be too easy for the model and lead to **slow training.**
- We accelerate training by sampling **hard negatives**
    - During training, run the current model on a subset of the products.
    - Select negatives from products the model thinks the user would buy, but didn’t.

Two towers

Discovery

The goal is to find interesting products for the user that they have not bought before .

Collaborative problem: *users that are similar to you also bought this*.



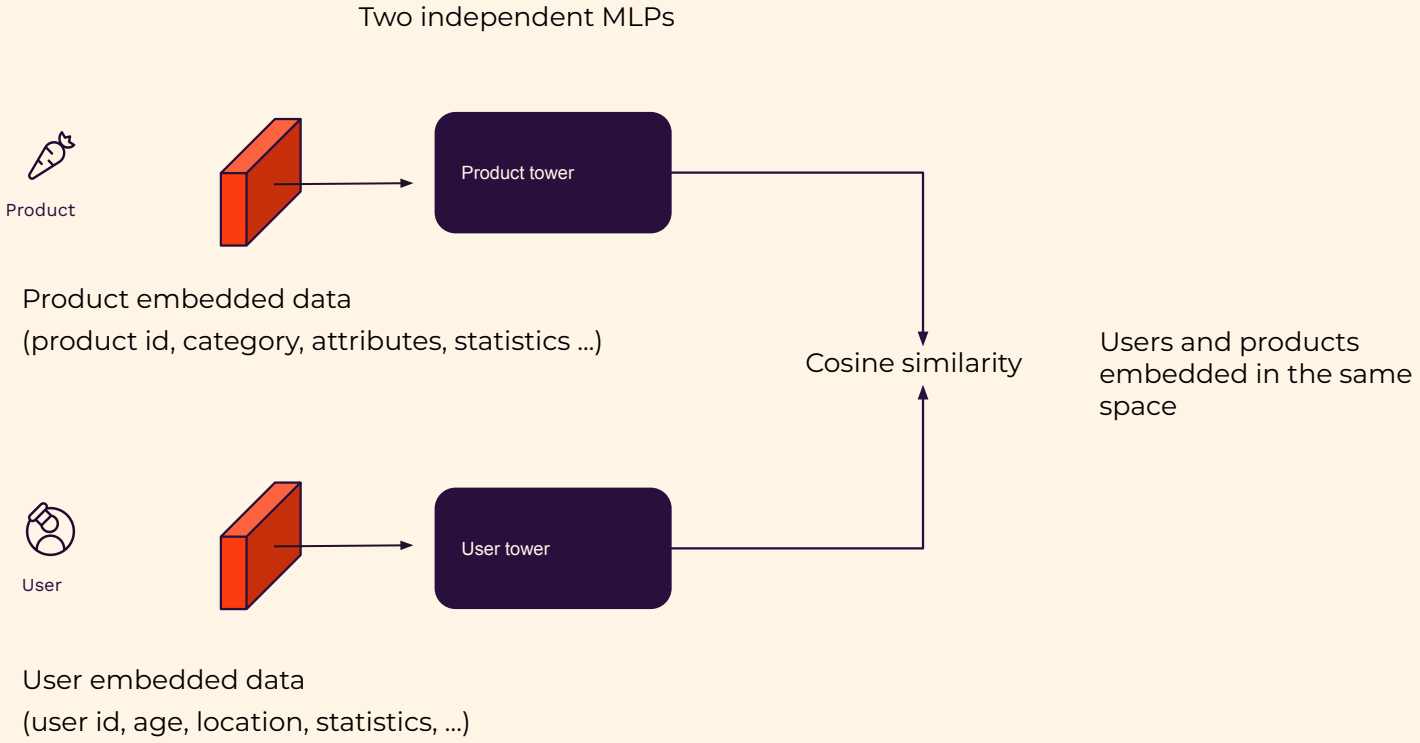


oda | Mathem

The NLP parallel: CLIP

NLP concept	Recommendation equivalent
Image encoder	User encoder
Text encoder	Product encoder
Contrastive loss	Contrastive loss
Temperature scaling	Temperature scaling
Zero-shot classification	Top-K recommendation retrieval

Architecture



Training

For a batch of B (user, product) purchase pairs:

1. Encode all B users and products
2. Compute the similarity matrix
3. Scale by 1/temperature
4. For each row (user), run softmax cross-entropy: we want high scores on diagonal, low on the off-diagonal

Training Data Matrix (One-Hot Pattern)

	Item A	Item B	Item C	Item D	Item E
User 1	✅ pos	— neg	— neg	— neg	— neg
User 2	— neg	✅ pos	— neg	— neg	— neg
User 3	— neg	— neg	✅ pos	— neg	— neg
User 4	— neg	— neg	— neg	✅ pos	— neg
User 5	— neg	— neg	— neg	— neg	✅ pos

Transfer learning: Recipes

Step 1: Train product model

Train UserTower + ItemTower jointly with InfoNCE on user–product purchase pairs. Both towers converge to a shared 128-d space.

Step 2: Freeze the user tower

Once product training is done, lock all UserTower weights. user_tower.trainable = False. The user encoder will never change again.

Step 3: Train the recipe tower

Create a new RecipeTower (same architecture as ItemTower). Train it on user–recipe interaction pairs. The UserTower produces the same user embeddings as before. Only the RecipeTower updates.



Offline vs. online serving

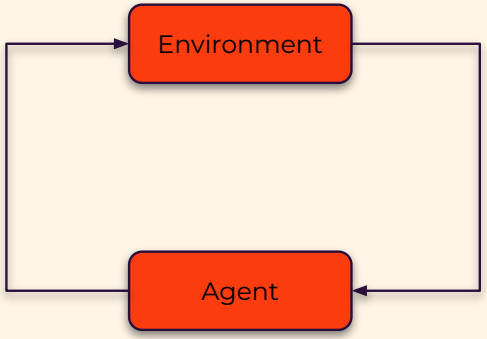
- We want low latency when we serve recommendations in the shop.
- Both the next-order and two-tower model run as nightly predictions jobs.
    - At serving we only do a lookup - no neural network inference needed.
    - For two towers, since the product and user tower are independent, we can pre-compute the embeddings, and then only do the cosine similarity at serving.

- In online serving we run the model inference as the user is in the shop.
    - Can use the current cart as a feature.
    - Can use in-session data as views and clicks.
    - Recommendation is live, can change as you browse and adapt to your session.
    - Can balance explore/exploit in-session.

Online learning: Bandits

Bandit problem

The agent makes a decision and acts on the environment.



The environment creates a reward that goes back to the agent

Problem protocol: Stochastic bandits

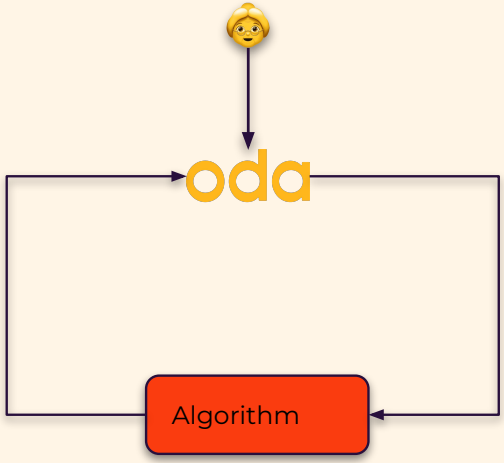
Parameters: K arms, T rounds (both known); reward distribution $\mathcal{D}_a$ for each arm a (unknown).
In each round $t \in [T]$:

1. Algorithm picks some arm a_t.
2. Reward $r_t \in [0, 1]$ is sampled independently from distribution $\mathcal{D}_a$, $a = a_t$.
3. Algorithm collects reward r_t, and observes nothing else.

oda

Bandit problem

The algorithm chooses which products to display to the user



The user browses the store. The events generated feeds into the algorithm.



oda

Model

Objective: maximize your earnings at the slot machines 💰

Different, and unknown, chance of winning at each machine.
What's the best strategy for choosing arms?



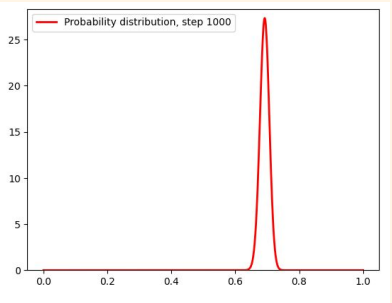
❌✅❌❌❌✅❌❌❌❌



✅✅❌❌❌✅❌❌❌✅



❌✅❌✅✅✅❌✅✅✅



✅ = cash payout
❌ = no prize

Multi-armed bandit: sampling strategy to optimally balance exploration and exploitation.

Thompson sampling: At every step, sample from the distribution for every arm and pick the best one.

$$p_i \sim B(\alpha_i, \beta_i)$$

oda

Model

Objective: maximize number of products added

Different, and unknown, chance of adding a product.
What's the best strategy for deciding which product to show?



❌✅❌❌❌✅❌❌❌❌



✅✅❌❌❌✅❌❌❌✅



❌✅❌✅✅✅❌✅✅✅

✅ = add to cart, given a view
❌ = view, but no add to cart

Multi-armed bandit: sampling strategy to optimally balance exploration and exploitation.

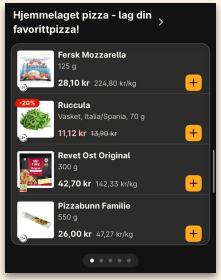
Thompson sampling: At every step, sample from the distribution for every product and pick the best one.

oda

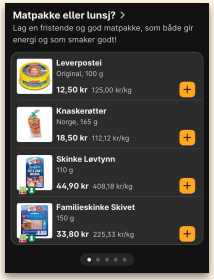
Model

Objective: maximize number of products added

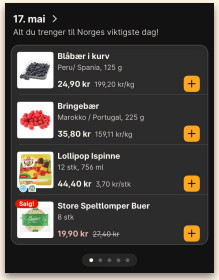
Different, and unknown, chance of adding from a product block.
What's the best strategy for deciding which block to show?



❌✅❌❌❌✅❌❌❌❌



✅✅❌❌❌✅❌❌❌✅



❌✅❌✅✅✅❌✅✅✅

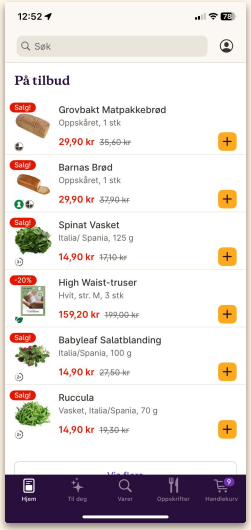
✅ = **add to cart**, given a view
❌ = **view**, but no add to cart

Multi-armed bandit: sampling strategy to optimally balance exploration and exploitation.

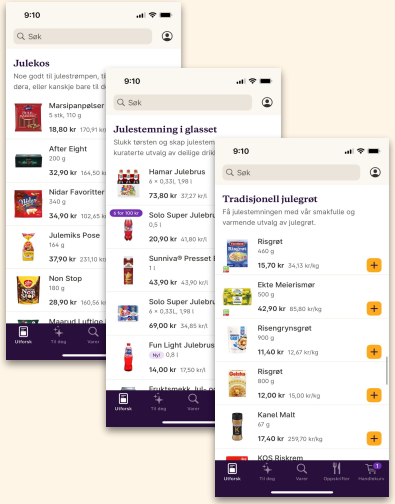
Thompson sampling: At every step, sample from the distribution for every product and pick the best one.

oda

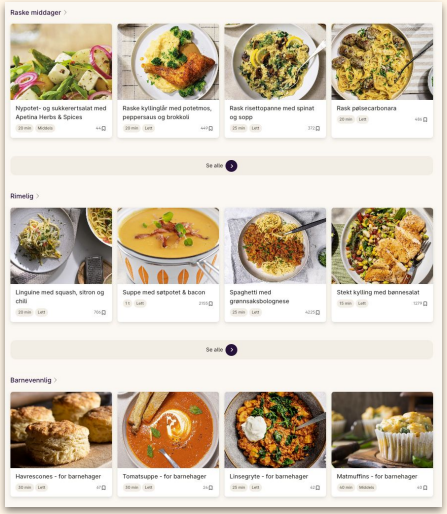
Applications



Ranking of produts



Ranking of blocks

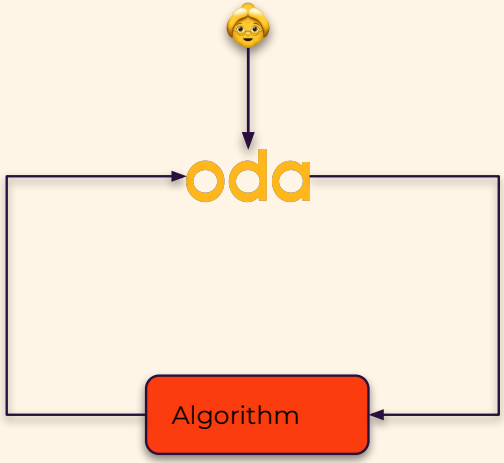


Ranking of recipes and sections

oda

Real-time feedback loop

The algorithm chooses which products to display to the user



The user browses the store. The events generated feeds into the algorithm.

We update the posterior model in real time based on what users see and interact with.

oda

Contextual bandits

We want to use the bandit framework, but let the predictions adapt to a specific **context**.

Context could be:

- Popularity of the product
- Scores from offline personalization model
- Time of day
- In-session data like views
- The current cart



Collect of all of the context into a feature vector **x.**

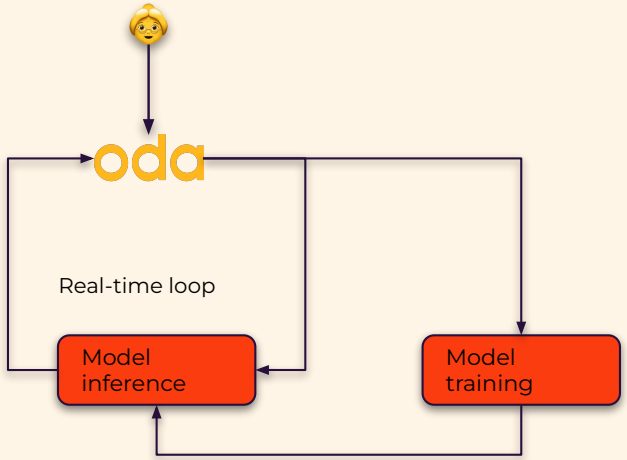
We then need to estimate a **posterior function**

$$p_i = f(x_i; \tilde{\theta}) \quad \text{where} \quad p(\theta) = \mathcal{N}(\mu, \Sigma)$$

Contextual bandit

Algorithm 1 Generic contextual bandit algorithm

for $t = 1, \ldots$ **do**
  Observe context x_t, compute $p_t := \texttt{explore}(x_t)$;
  Choose action $a_t \sim p_t$, observe loss $\ell_t(a_t)$;
  $\texttt{learn}(x_t, a_t, \ell_t(a_t), p_t)$;
end for



Online training in hourly increments

Model

The model needs

- Fast inference
- Online training updates
- Uncertainty estimates in the weights

We use a **bayesian logistic regression.**

Algorithm 3 Regularized logistic regression with batch updates

Require: Regularization parameter $\lambda > 0$.

$m_i = 0,\ q_i = \lambda$. {Each weight w_i has an independent prior $\mathcal{N}(m_i, q_i^{-1})$}

for $t = 1, \ldots, T$ **do**

Get a new batch of training data $(\mathbf{x}_j, y_j),\ \ j = 1, \ldots, n$.

Find $\mathbf{w}$ as the minimizer of: $\frac{1}{2}\sum_{i=1}^{d} q_i(w_i - m_i)^2 + \sum_{j=1}^{n} \log(1 + \exp(-y_j \mathbf{w}^\top \mathbf{x}_j))$.

$m_i = w_i$

$q_i = q_i + \sum_{j=1}^{n} x_{ij}^2 p_j(1 - p_j),\ \ p_j = (1 + \exp(-\mathbf{w}^\top \mathbf{x}_j))^{-1}$ {Laplace approximation}

end for

Model - Extention

The logistic model has limited expression - no interaction between features.

It is harder to use a neural network directly, as we would need uncertainty estimates on the weights.

We are currently testing using a neural network as a feature extractor followed by a bayesian logistic regression as the final layer.

The counterfactual problem

Standard ML: we assume train and test data come from the same distribution.

Here the training data is generated by the current production model. Products ranked high by the model will get more views and be overrepresented in the training data for the next model update.

Product A: 10 000 views, 800 added to cart (8% CTR)

Product B: 500 views, 60 added to cart (12% CTR)

Inverse Propensity Scoring (IPS): if an item was shown with low probability but converted, that conversion is more informative than a conversion for an item shown constantly. Weight each training example by the inverse of the probability it was shown.

Summary

Next-order prediction: exploit-heavy model that predicts what you are most likely to shop in your next order.

Online bandit models: balances explore/exploit based on real-time events while you shop.

Two towers: explore-heavy model for general preference and what separates you from other users.



3. Agents

Failures and ongoing experimentation

1. Embeddings
2. Recommendations
3. **Agents**

Part 1: Failure

Personal Shopping Assistant

User problem: “I need help while shopping”



User problem

Who can I ask for help?

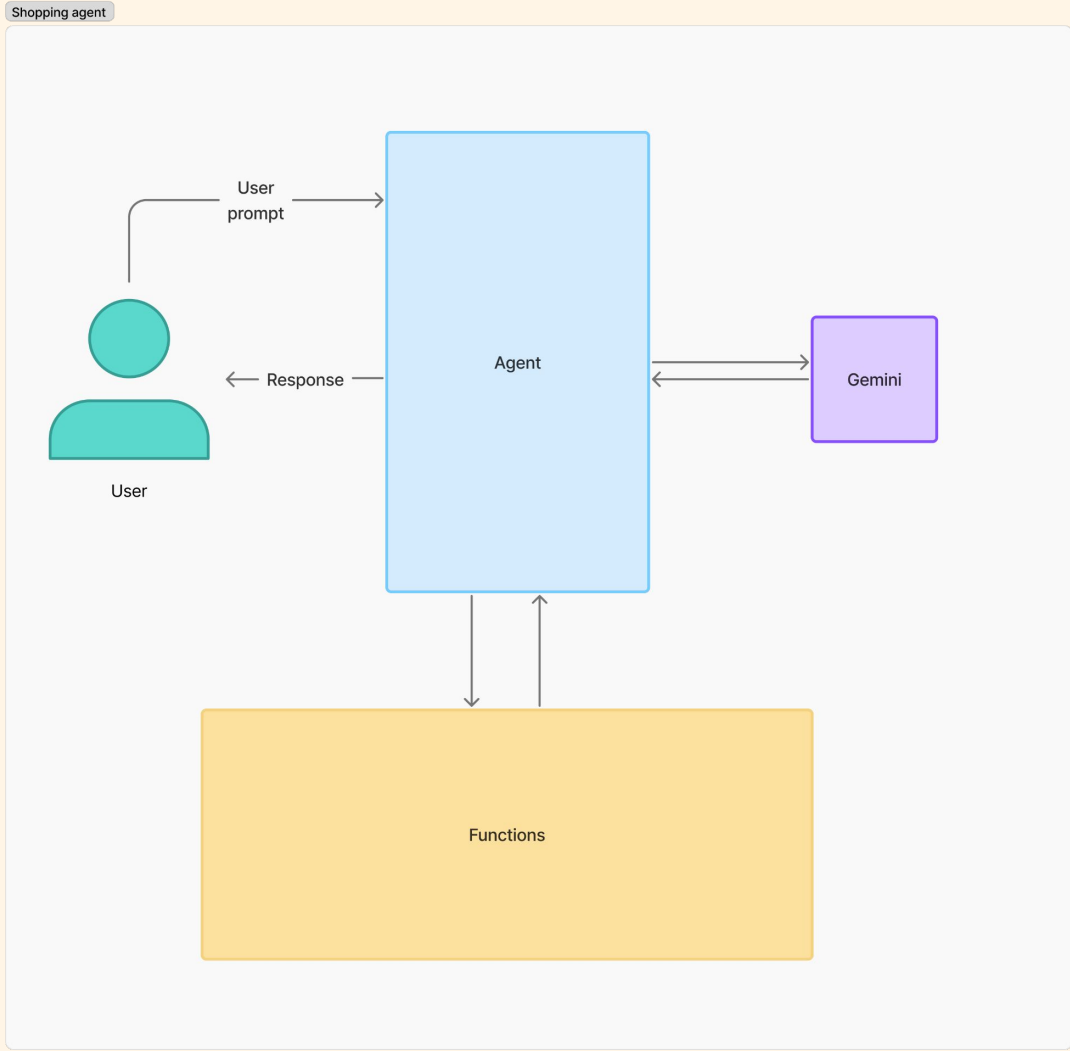
A ReAct agent



How to set up a ReAct agent

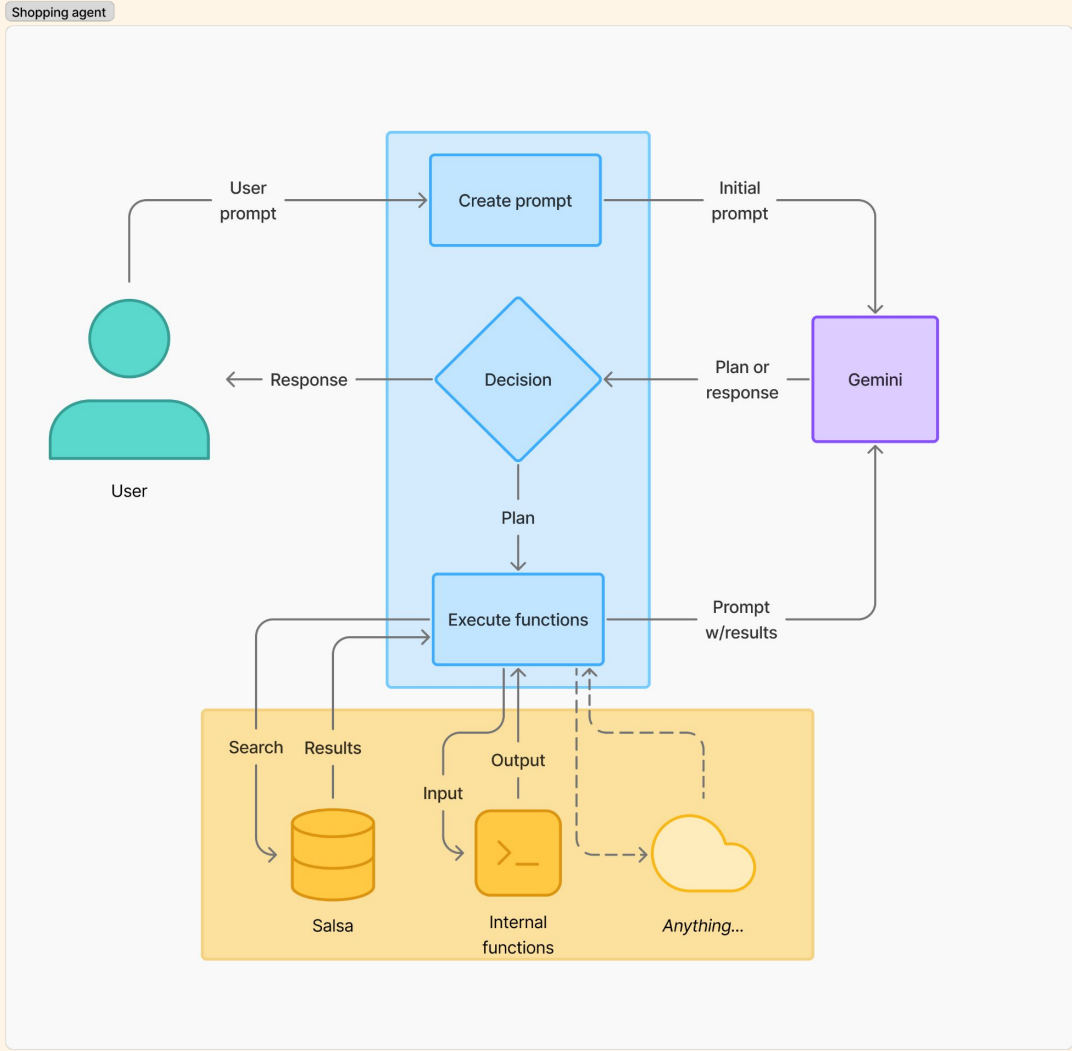
ReAct = Reasoning + Acting

ReAct agent



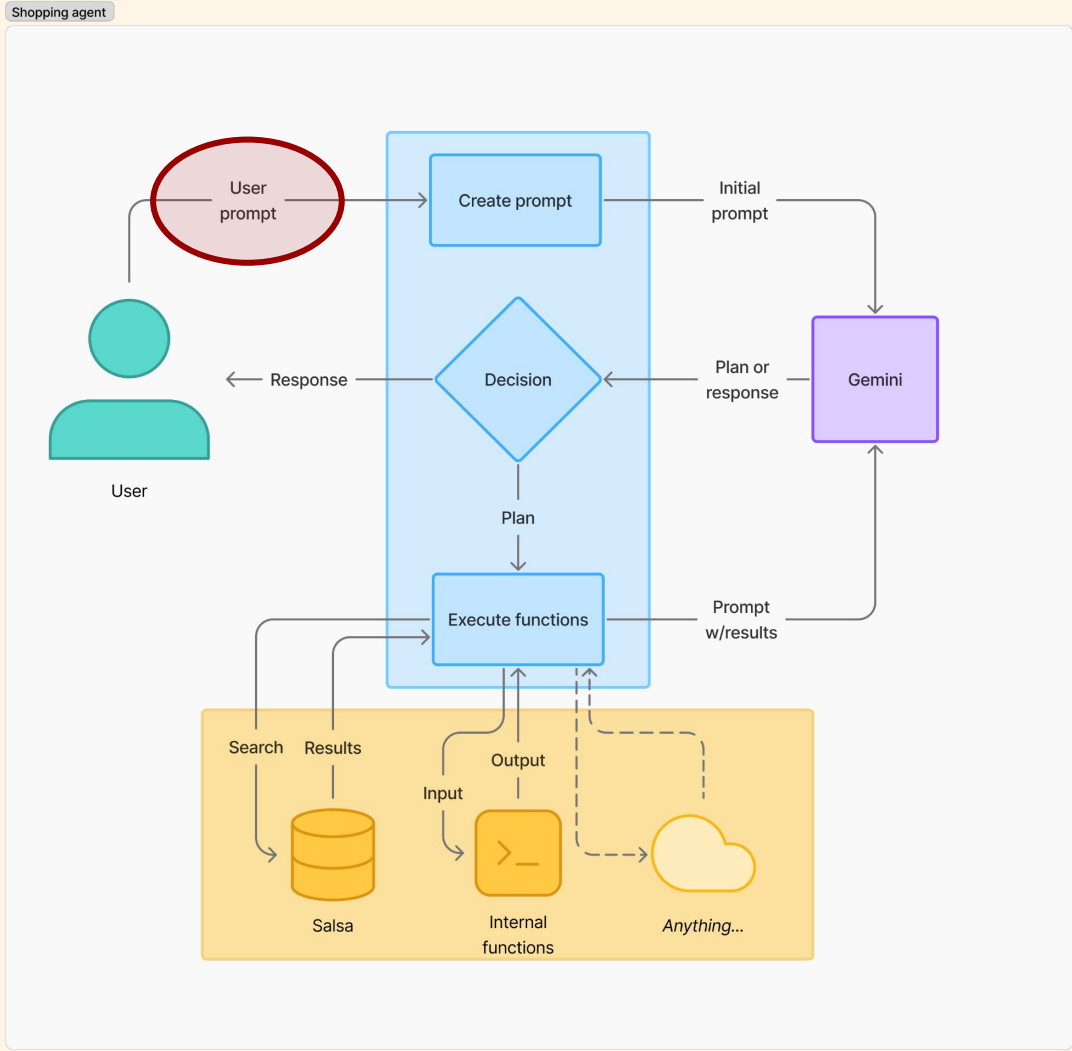
oda | Mathem

ReAct agent



oda | Mathem

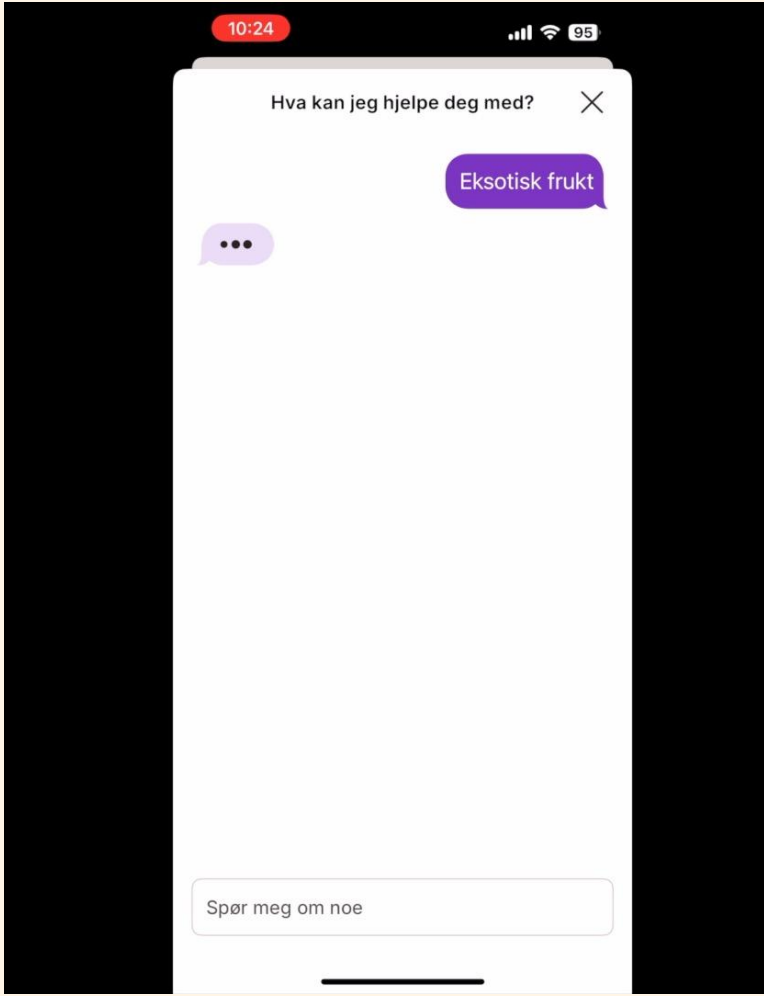
ReAct agent



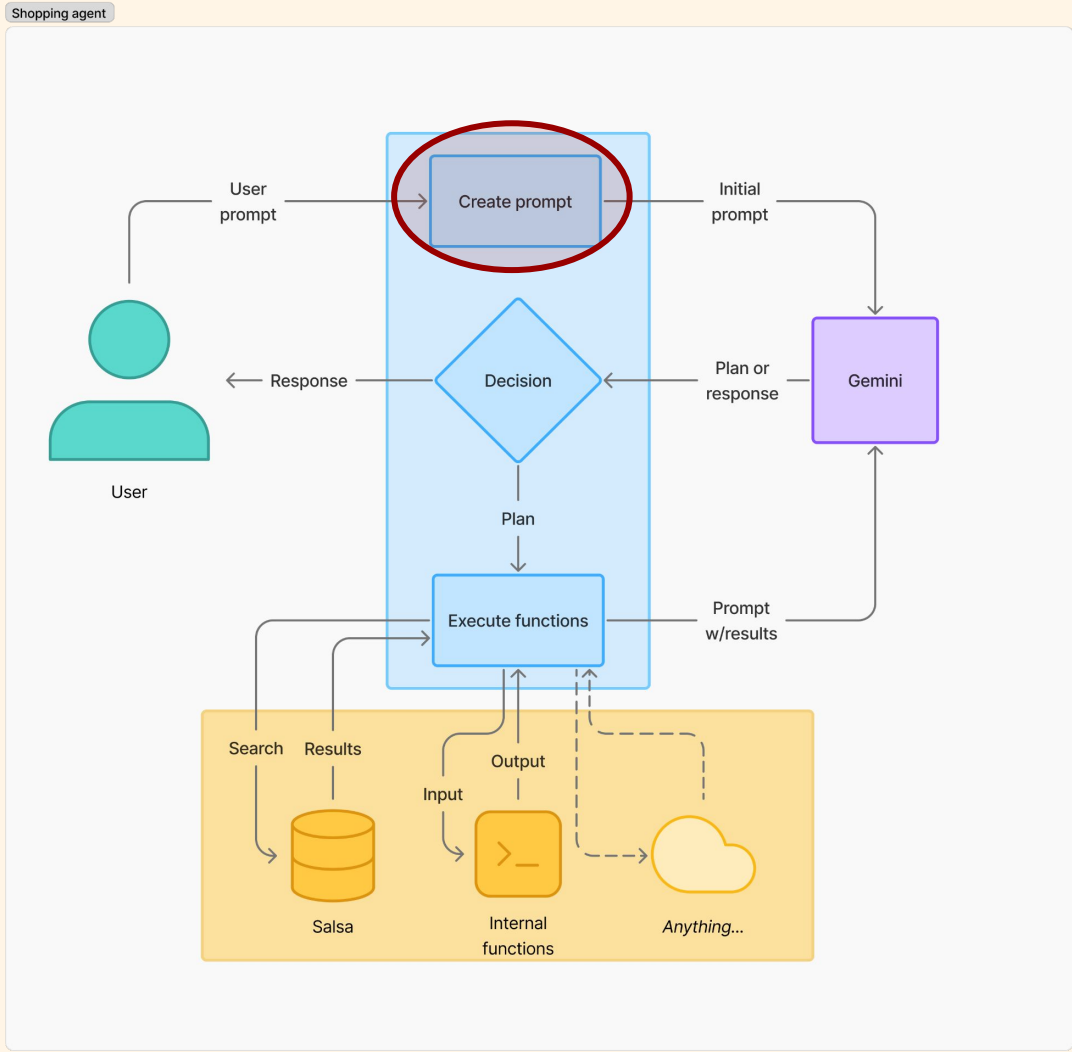
oda | Mathem

Prompt

- User prompt (“exotic fruit”)



ReAct agent



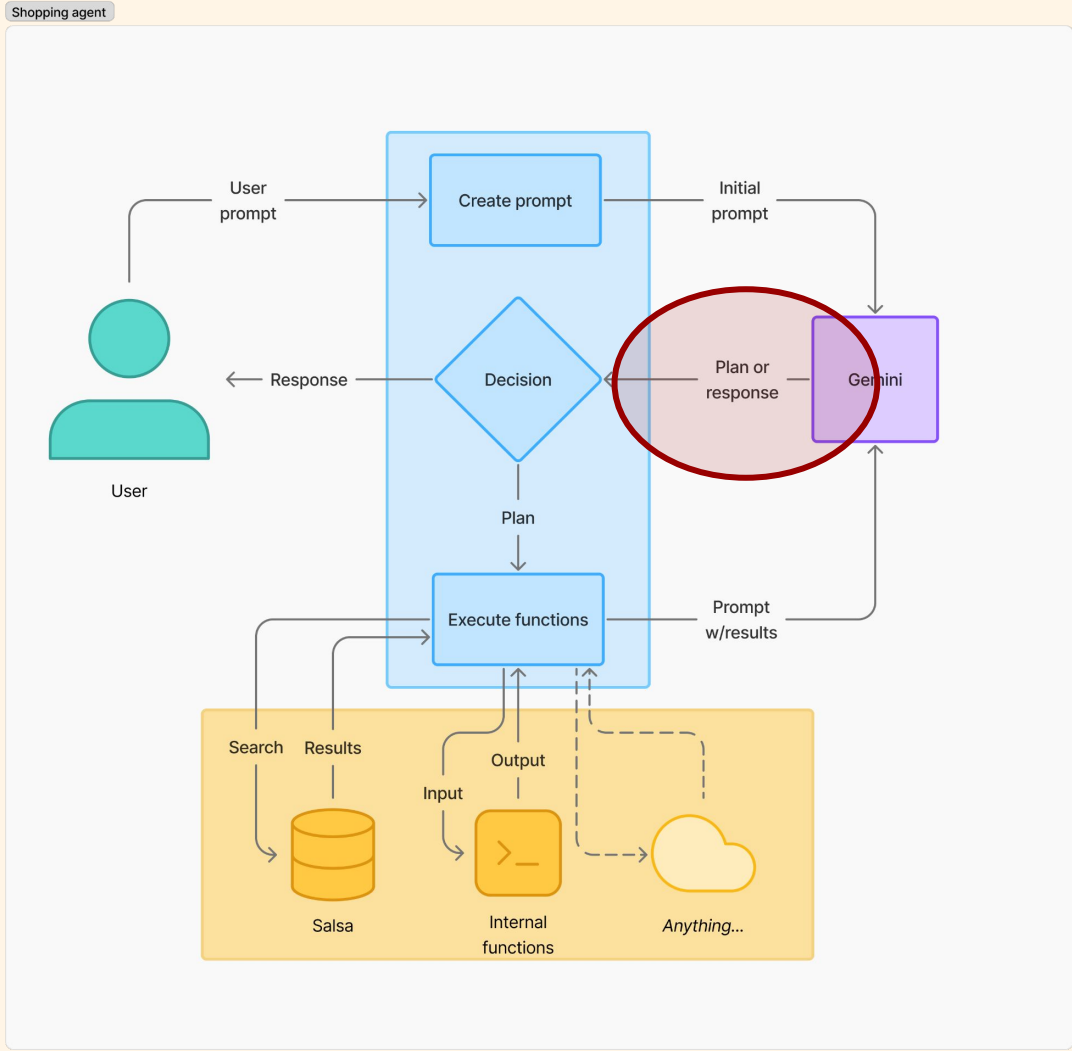
oda | Mathem

Prompt

- User prompt (“exotic fruit”)
- General part of prompt: (“You are a shop assistant, bla bla…”)

```
SYSTEM_PROMPT = """You are AIDA, an AI shopping assistant for the Norwegian online grocery store Oda.
Your goal is to help users find products, give recommendations, make informed shopping decisions
and make the shopping experience inspiring and efficient.
```

ReAct agent



oda | Mathem

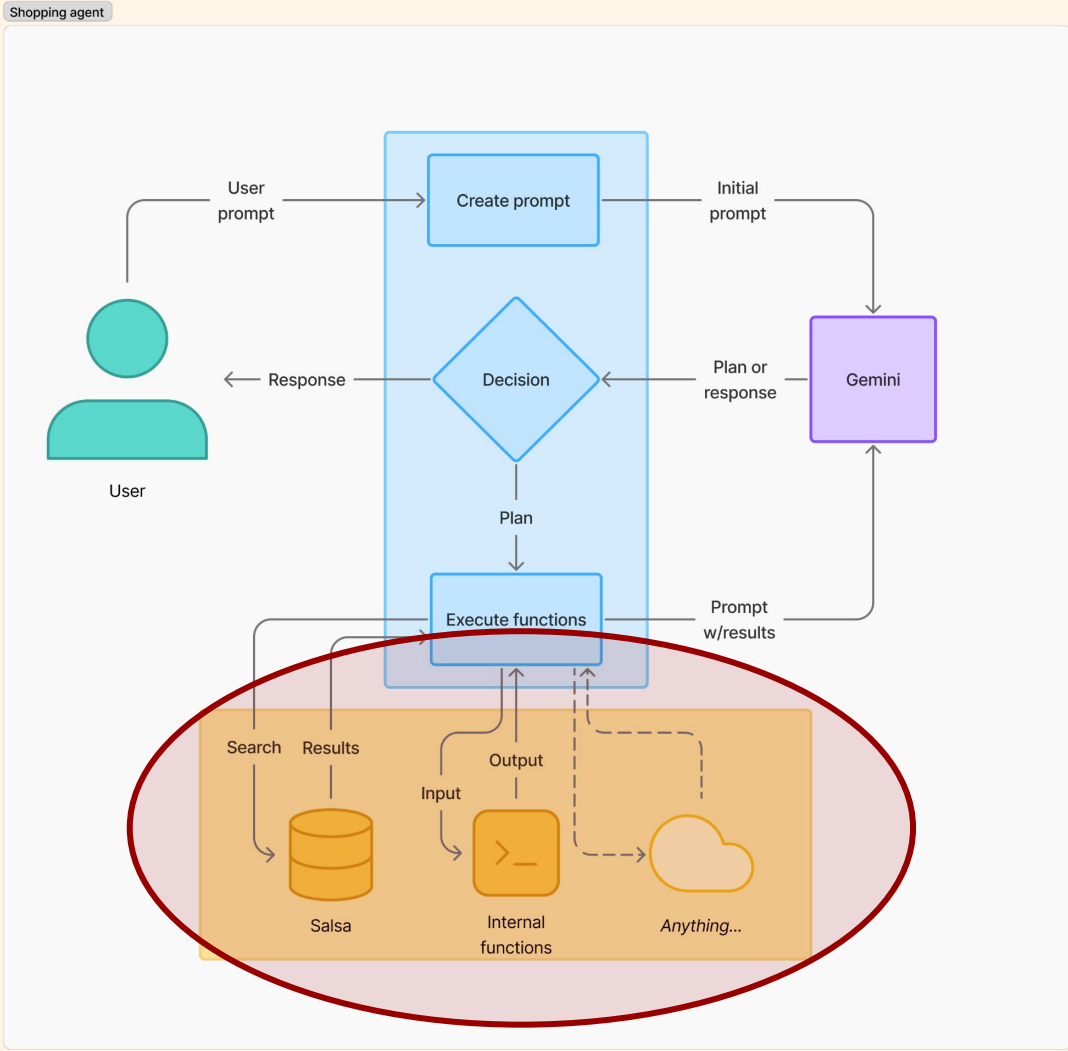
Prompt

- User prompt (“exotic fruit”)
- General part of prompt: (“You are a shop assistant, bla bla…”)
- Agentic part of prompt: (“Make a plan, Stan”)

```
Your function output must either be generate_plan or generate_response - never both.

In a plan, You have access to these functions for gathering information:
1. search_products: Find products with a text query
2. semantic_recipe_search: Find recipes using natural language queries
3. get_recommendations: Fetch recommended products from recommender models (likely_to_buy, bonus
4. get_recipe_details: Return a recipe's basic info and its products (requires recipe_id)
5. get_discounts: Fetch products on discounts
```

ReAct agent



oda | Mathem

Prompt

- User prompt (“exotic fruit”)
- General part of prompt: (“You are a shop assistant, bla bla…”)
- Agentic part of prompt: (“Make a plan, Stan”)
- Function definitions (“How to search?)

Prompt

- User prompt (“exotic fruit”)
- General part of prompt: (“You are a shop assistant, bla bla…”)
- Agentic part of the prompt: (“Make a plan, Stan”)
- Function definitions (“How to search?)

```python
class AgentTools:

    async def search_products(self, query: str, top_k: int = 20) -> List[Tuple[str, Optional[Dict]]]:
        """
        Search for specific products in the store using Norwegian text queries.
        Works best with precise, single-product searches.
        Break down complex user queries into smaller, more specific product-specific queries.

        This is useful when you need to:
        - Find a specific product by its exact name (e.g. "Tine Lettmelk")
        - Search for a single, well-defined product type (e.g. "havremelk")
        - Look up products by specific brand names (e.g. "First Price")

        Args:
            query (str): The search query in Norwegian. Most effective with specific product names or types.
                         Examples: "Tine Lettmelk", "havremelk", "First Price brød"
            top_k (int, optional): Maximum number of products to return. Defaults to 20. For broad queries,
                prefer 20-40 and let the agent select the most relevant for the response.
```

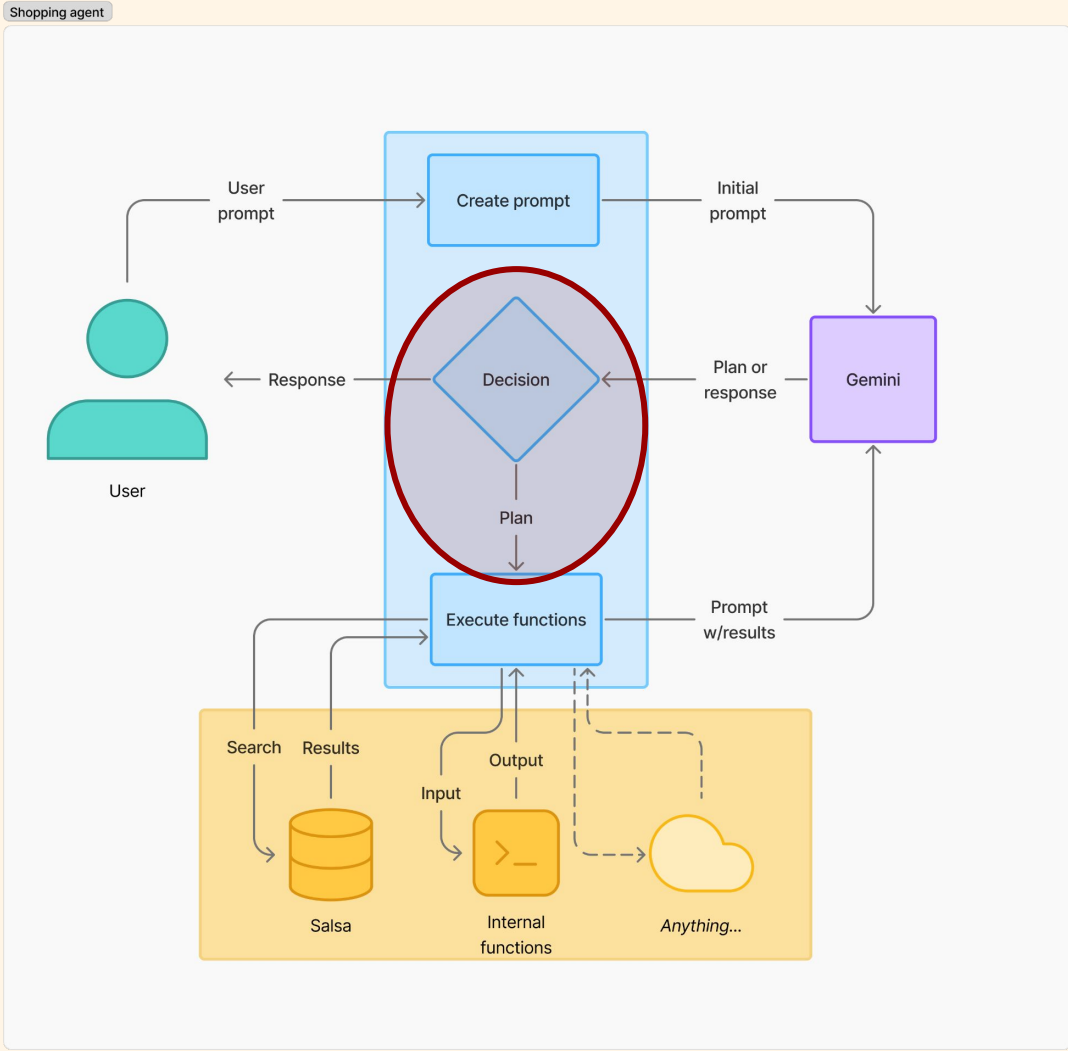
Prompt

- User prompt (“exotic fruit”)
- General system prompt: (“You are a shop assistant, bla bla…”)
- Agentic part of the prompt: (Plan, call, respond)
- Function definitions (“How to search?)

+ other small components

- Tone of voice
- Behavioral tips
- Clarifications on goals
- Desired response
- etc.

LLM response 1: A plan



oda | Mathem

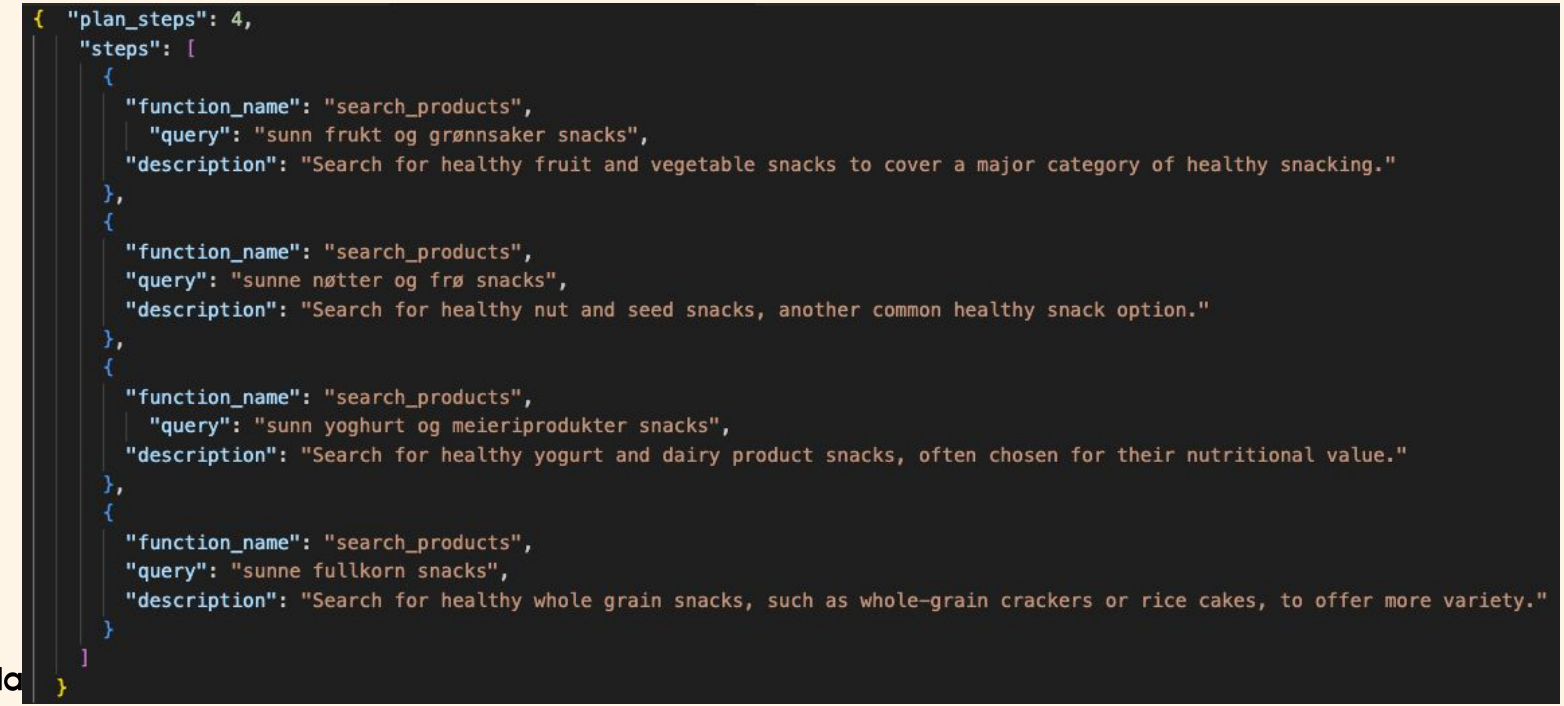
A plan

Gemini returns a list of functions with arguments to call.


```
  "message_type": "assistant_thinking",
  "plan_steps": 1,
  "json_data": {
    "steps": [
      {
        "function_name": "search_products",
        "args": {
          "query": "søt eksotisk frukt",
          "top_k": 20
        },
        "depends_on": [],
        "description": "Search for sweet exotic fruits, as the user indicated a dislike for sour fruits, refining their previous request for exotic fru
its."
      }
    ],
    "plan_rationale": "The user previously asked for 'eksotisk frukt' (exotic fruit) and now has refined their preference by stating 'Jeg liker ikke su
re frukter' (I don't like sour fruits). Therefore, the plan is to search for 'søt eksotisk frukt' (sweet exotic fruit) to align with the user's updated
preference. A top_k of 20 is chosen to provide a good selection of sweet exotic fruits."
  }
```

A plan

Sometimes, the plan consists of several steps



```
{  "plan_steps": 4,
    "steps": [
      {
        "function_name": "search_products",
          "query": "sunn frukt og grønnsaker snacks",
        "description": "Search for healthy fruit and vegetable snacks to cover a major category of healthy snacking."
      },
      {
        "function_name": "search_products",
        "query": "sunne nøtter og frø snacks",
        "description": "Search for healthy nut and seed snacks, another common healthy snack option."
      },
      {
        "function_name": "search_products",
          "query": "sunn yoghurt og meieriprodukter snacks",
        "description": "Search for healthy yogurt and dairy product snacks, often chosen for their nutritional value."
      },
      {
        "function_name": "search_products",
        "query": "sunne fullkorn snacks",
        "description": "Search for healthy whole grain snacks, such as whole-grain crackers or rice cakes, to offer more variety."
      }
    ]
}
```

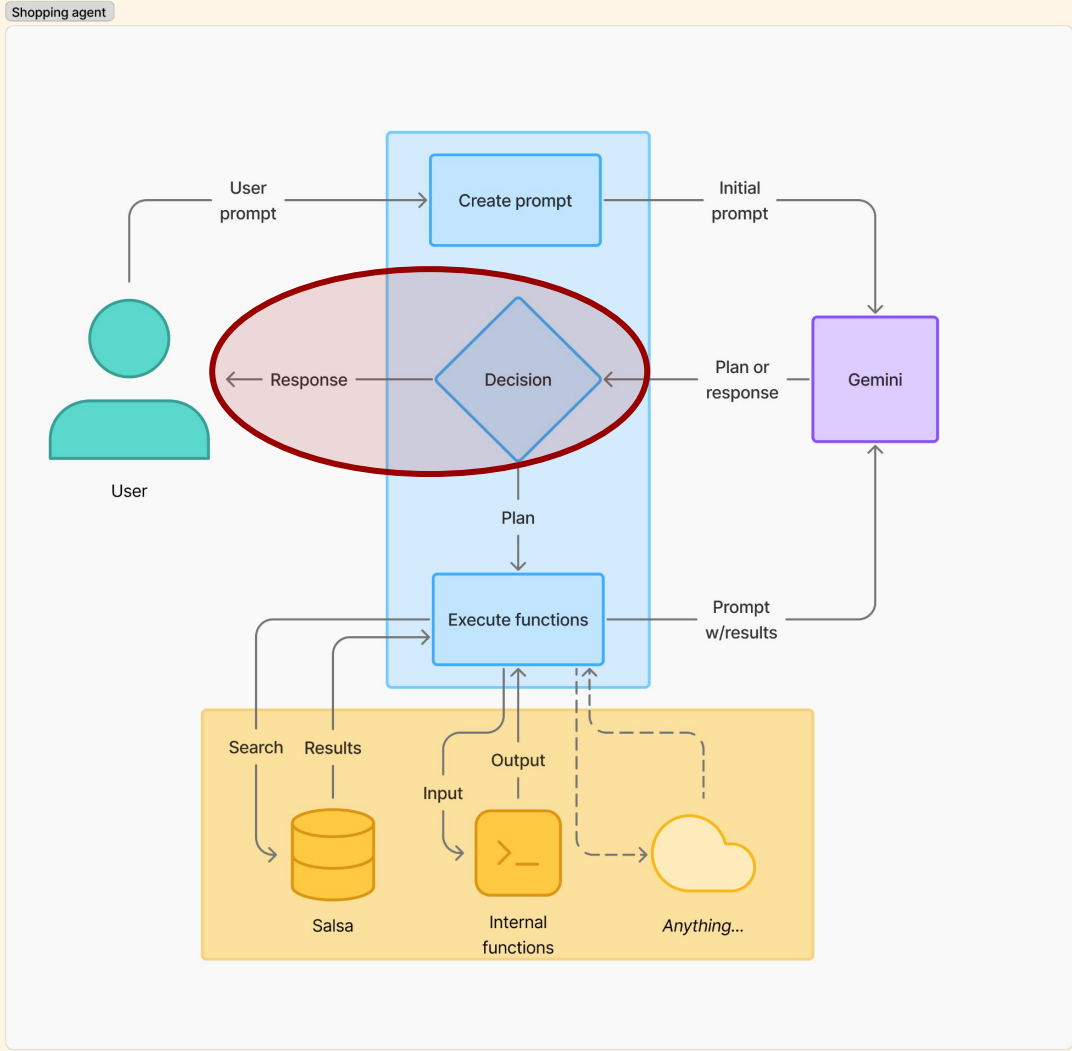
oda

A plan

`plan_rationale:`

The user is asking for 'Sunne snacks' (Healthy snacks).This is a broad query, so the plan decomposes it into multiple targeted searches across different common categories of healthy snacks: fruits and vegetables, nuts and seeds, yogurt and dairy products, and whole grain snacks. This approach ensures a comprehensive retrieval of relevant healthy snack options, and for each search, a top_k of 20 products is requested to provide a good variety to the user

A response



oda | Mathem

A response


```
{
  "timestamp": "2025-09-17T07:22:01.882690Z",
  "event": "Generated final response",
  "json_data": [
    { "type": "message", "content": "Her er noen forslag til sunne snacks fra ulike kategorier:" },

    { "type": "message", "content": "Frukt og grønnsaker er alltid et godt valg:" },
    { "type": "product_list", "content": ["..."] },

    { "type": "message", "content": "Du kan også prøve sunne yoghurt- og meieriprodukter:" },
    { "type": "product_list", "content": ["..."] },

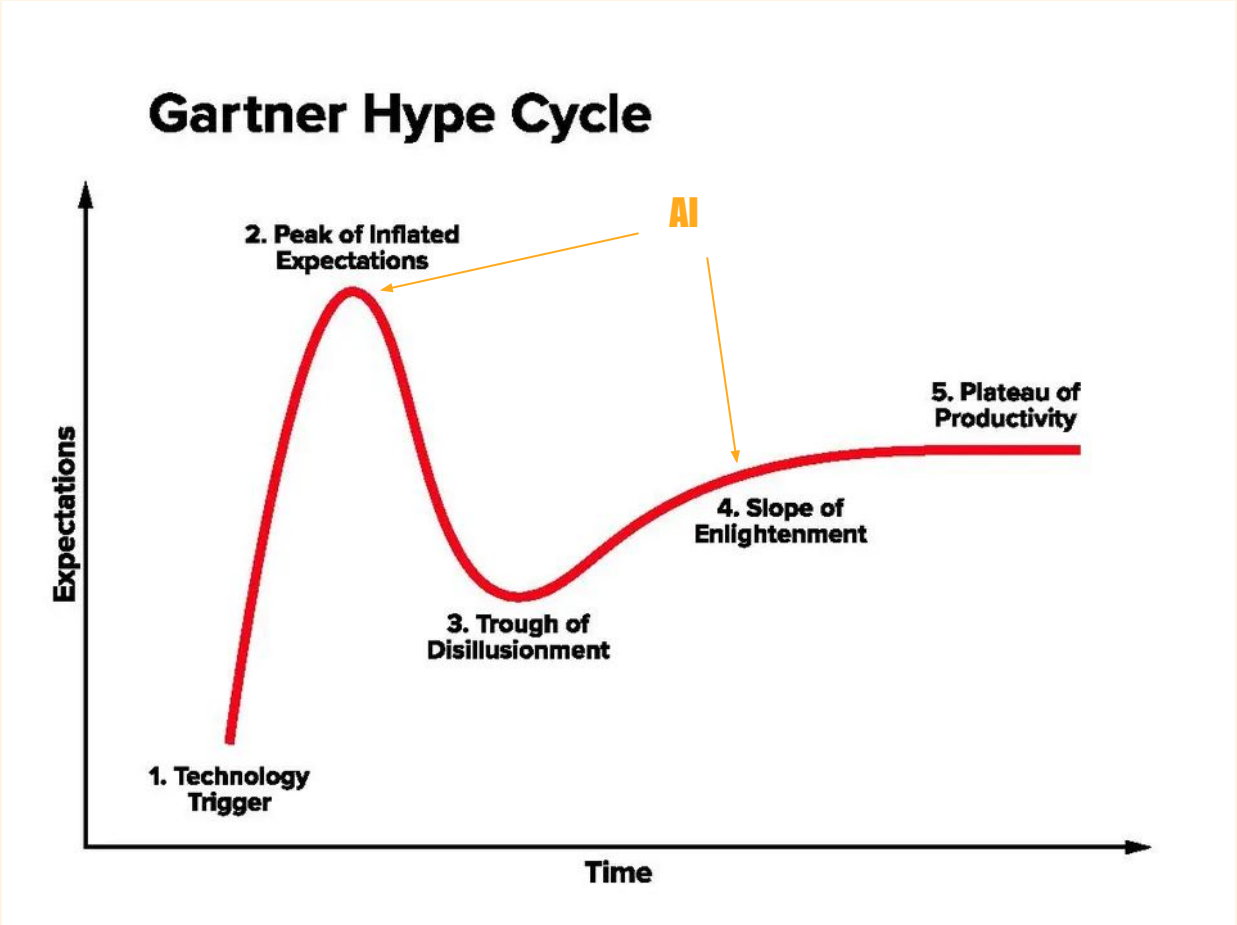
    { "type": "message", "content": "Nøtter og tørket frukt kan også være gode snacksalternativer:" },
    { "type": "product_list", "content": ["..."] },

    { "type": "suggested_prompts",
      "content": [
        "Vis meg sunne lunsjalternativer",
        "Hvilke vegetariske snacks finnes det?",
        "Gi meg oppskrifter på sunne snacks"
        ]
      }
  ]
}
```

Final product, launched with interal video 🎉

(Skip if time constraint)




Gartner Hype Cycle
AI
2. Peak of Inflated Expectations
5. Plateau of Productivity
4. Slope of Enlightenment
3. Trough of Disillusionment
1. Technology Trigger
Expectations
Time

Unsuccessful experiment

A/B test

- Split all users randomly in two groups
- Compare the performance of two different versions (A and B) of something
- Determine which version is more effective based on data



Products added: Shop Assistant	0 *0 / 107.2K*	0.007 *747 / 107K*	*No data*

Experimentation is a method to learn

...and we learned something
we already knew...



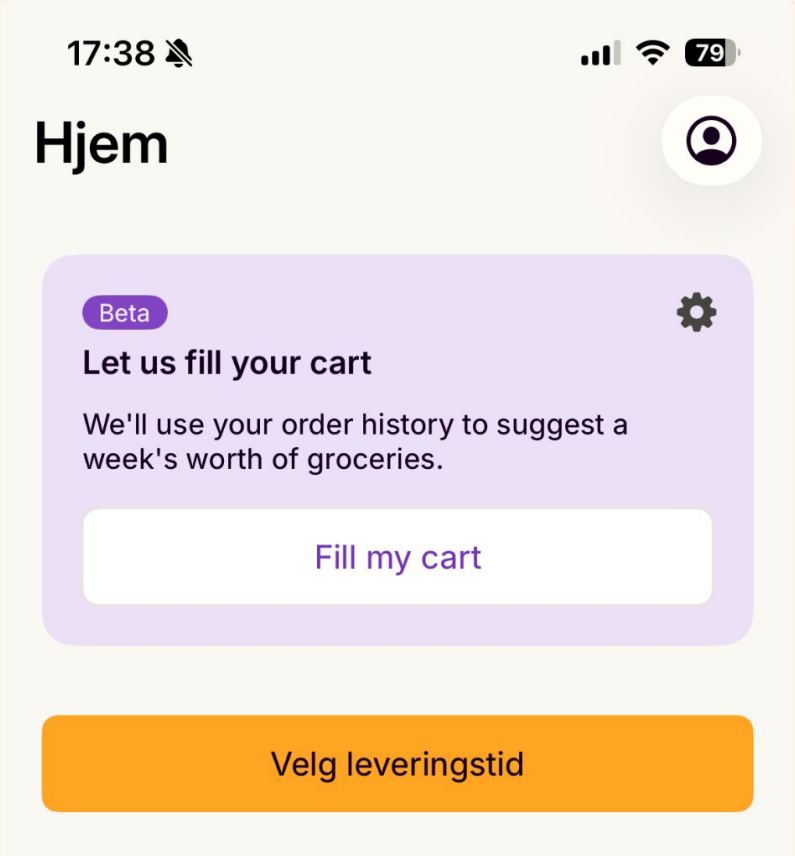
Part 2: Ongoing work

Fill my cart



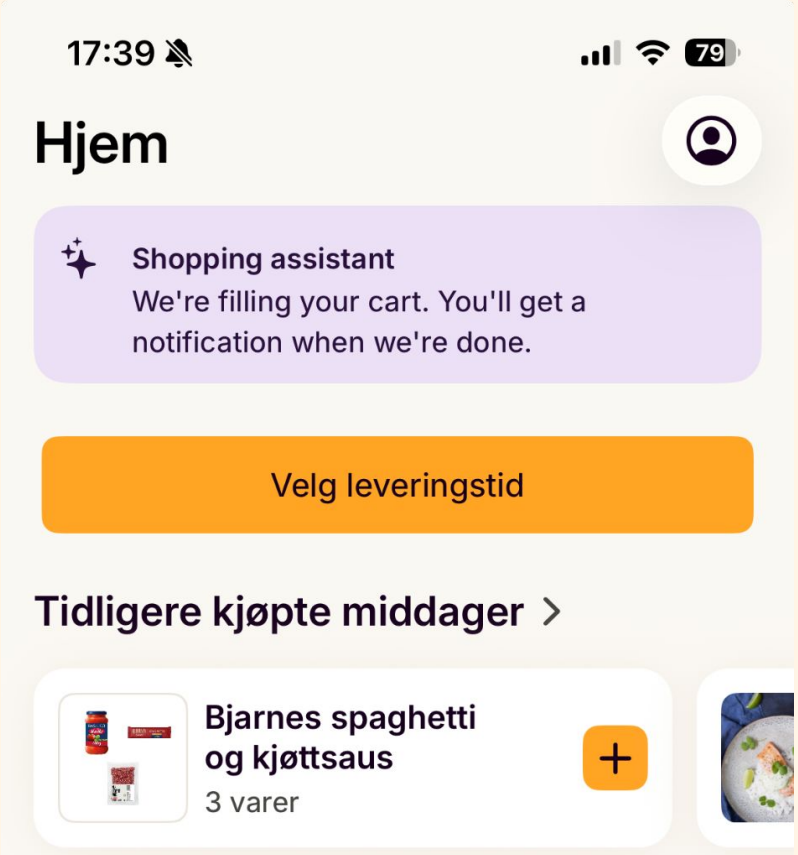
Fill my cart

No chat, just one click

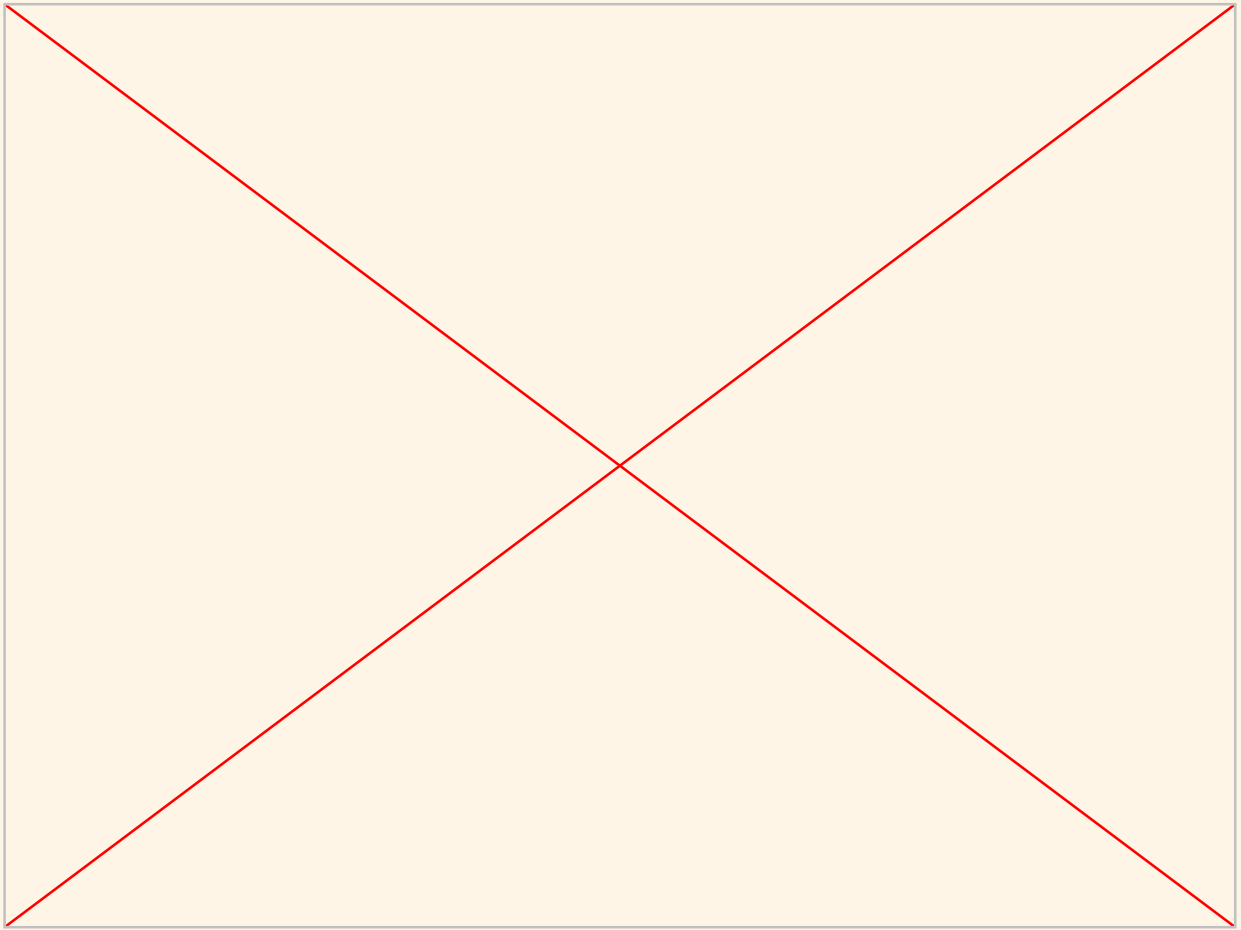


Fill my cart

High latency, let iut work
in the beackground



Fill my cart



Appendix

Answering the real question

1. Embeddings
2. Recommendations
3. Agents

Appendix

Summary

1. *Embeddings*

   Word2vec applied on products ✅
   LLMs and **text embeddings** ✅

2. *Recommendations*

   Neural networks ✅
   Transformer architecture ✅

3. *Agents*

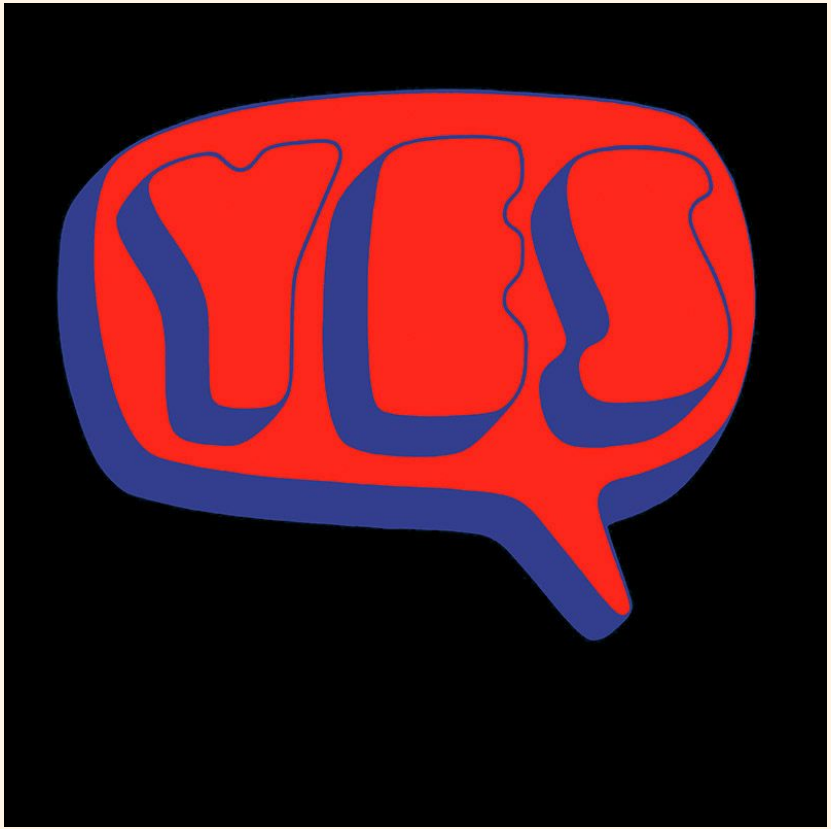
   Applying **AI** on real world problems ✅
   Understanding the limitations and possibilities of **LLMs** ✅

1. Embeddings
2. Recommendations
3. Agents

Appendix

The real question

User problem: “Will I get a job?”



Thank you!

Disposisjon

1. Intro (Fredrik) - 5 min
   a. Om Fredrik
   b. Om Lars
   c. Om Oda
   d. Om produktet - tegne et kart

Interlude: Embeddings (Fredrik)

2. Home-grown Embeddings: Related products (Fredrik)
   a. Word2Vec
   b. Dildo gate and text embeddings
   c. Utforsk-knappen
3. Pre-trained embeddings: Search (Fredrik)
   a. Embeddings in search
   b. What text to embed
   c. Also used in related products (now)

Interlude: Explore/exploit tradeoff (Lars)

4. Transformer (exploit): Likely to buy (Lars)
5. Two towers (explore) (Lars)
   - Algorithm
   - Visualisation
   - Use cases
   - Transfer learning to recipes

Interlude - Online training (vs batch) (Lars)

6. Bandits (Lars)
   - Global multi-armed bandits
   - Contextual (two towers)
   - Log reg vs MLP
     (for interaction between features)

Interlude - Agentic setup (Lars)

7. Agents
   - Failed shopping assistant (Fredrik)
   - Current approach - Pre-fill cart (Lars)
8. Future of groceries
   - More ppl use Oda?

     Is Oda an MCP or an app?

