CS11-711 Advanced NLP

Introduction and Fundamentals

Sean Welleck

Carnegie Mellon University

https://cmu-l3.github.io/anlp-fall2025/

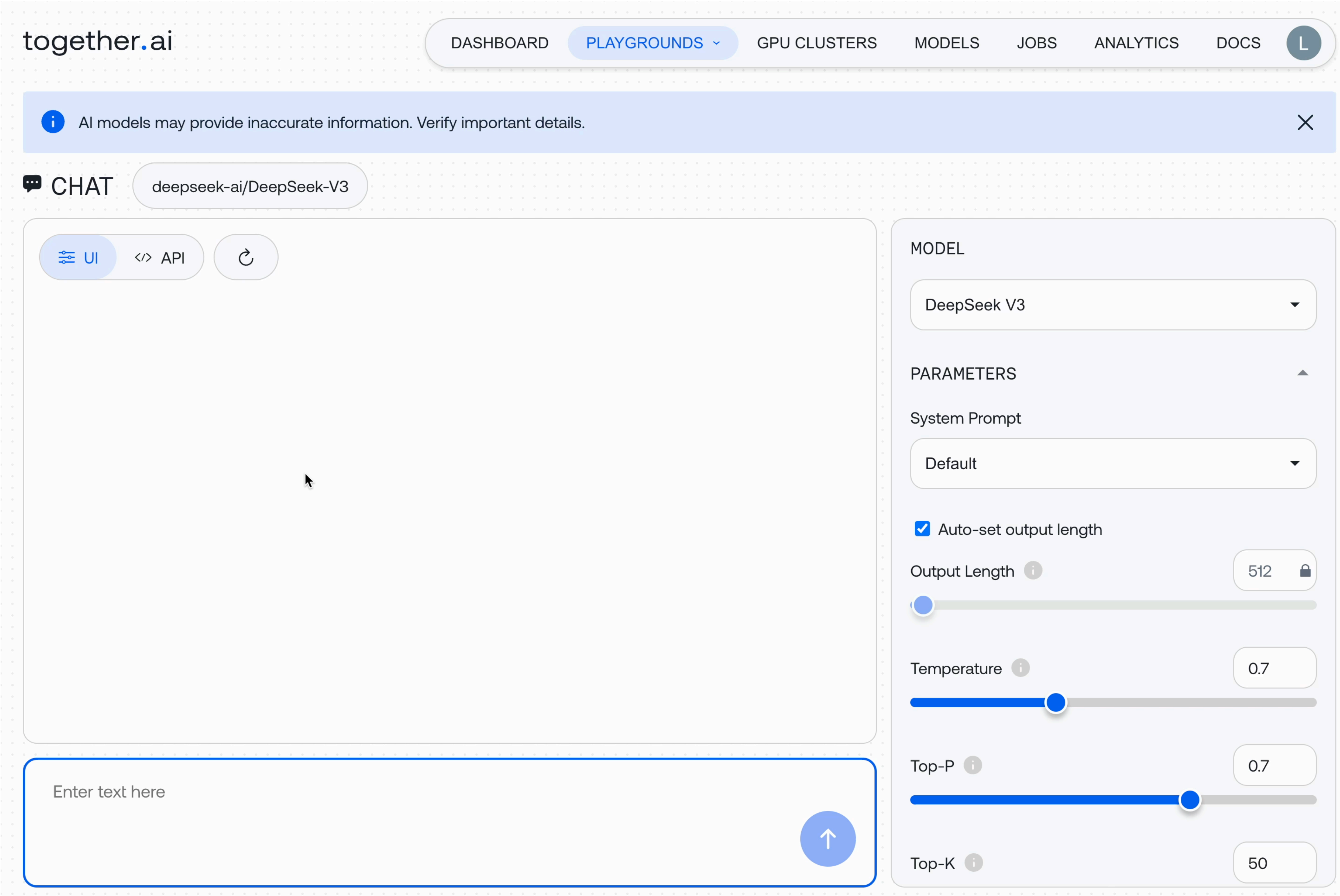
https://github.com/cmu-l3/anlp-fall2025-code

Slides adapted from Graham Neubig's Fall 2024 course

What is Natural Language Processing (NLP)?

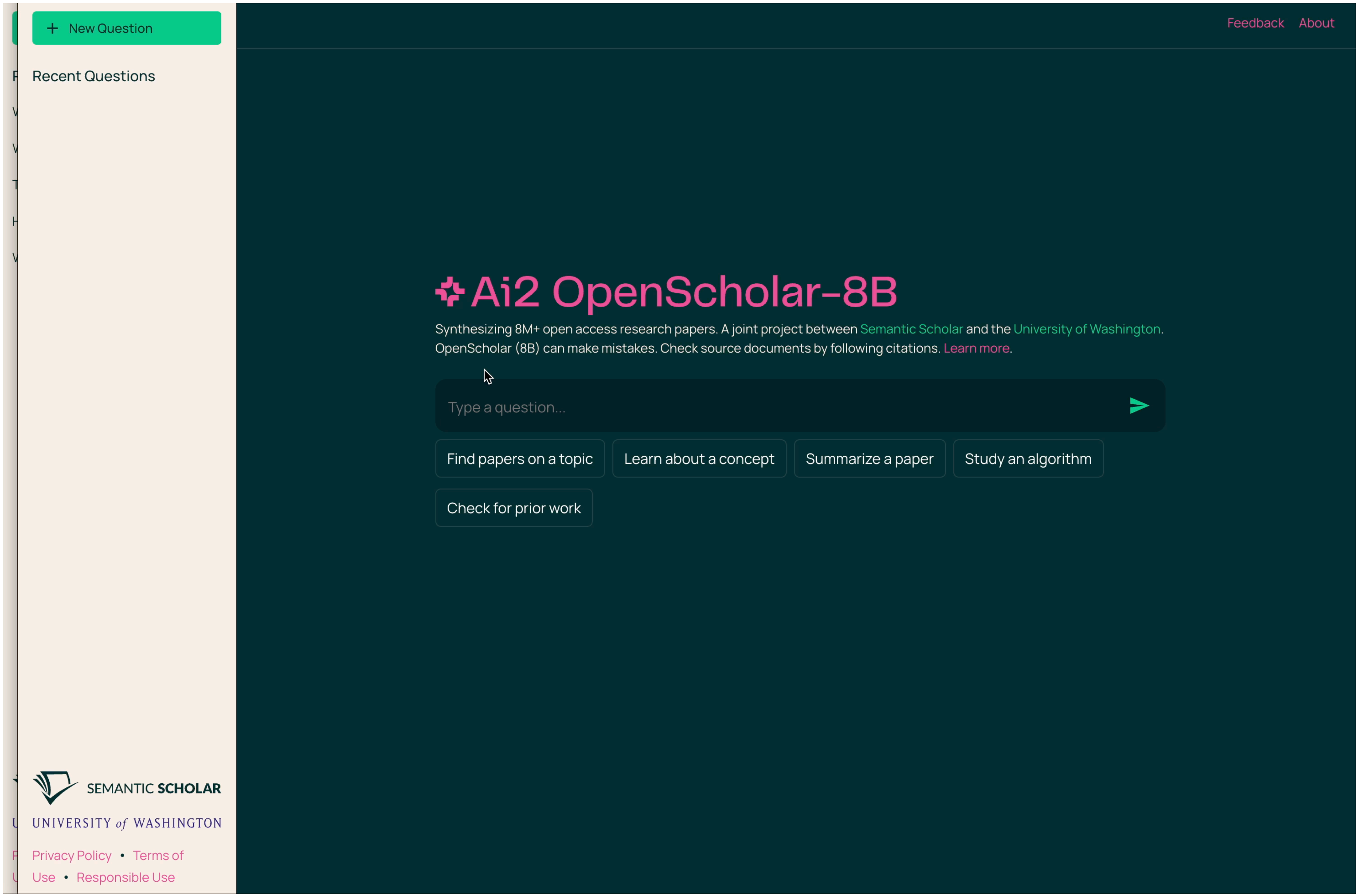
- Technology that enables computers to process, generate, and interact with language (e.g., text). Some key aspects:
    - **Learn useful representations**: capture meaning in a structured way that can be used for downstream tasks (e.g., embeddings used to classify a document)
    - **Generate language**: create language (e.g., text, code) for tasks like dialogue, translation, or question answering.
    - **Bridge language and action**: Use language to perform tasks, solve problems, interact with environments (e.g., a code IDE)

Today's NLP



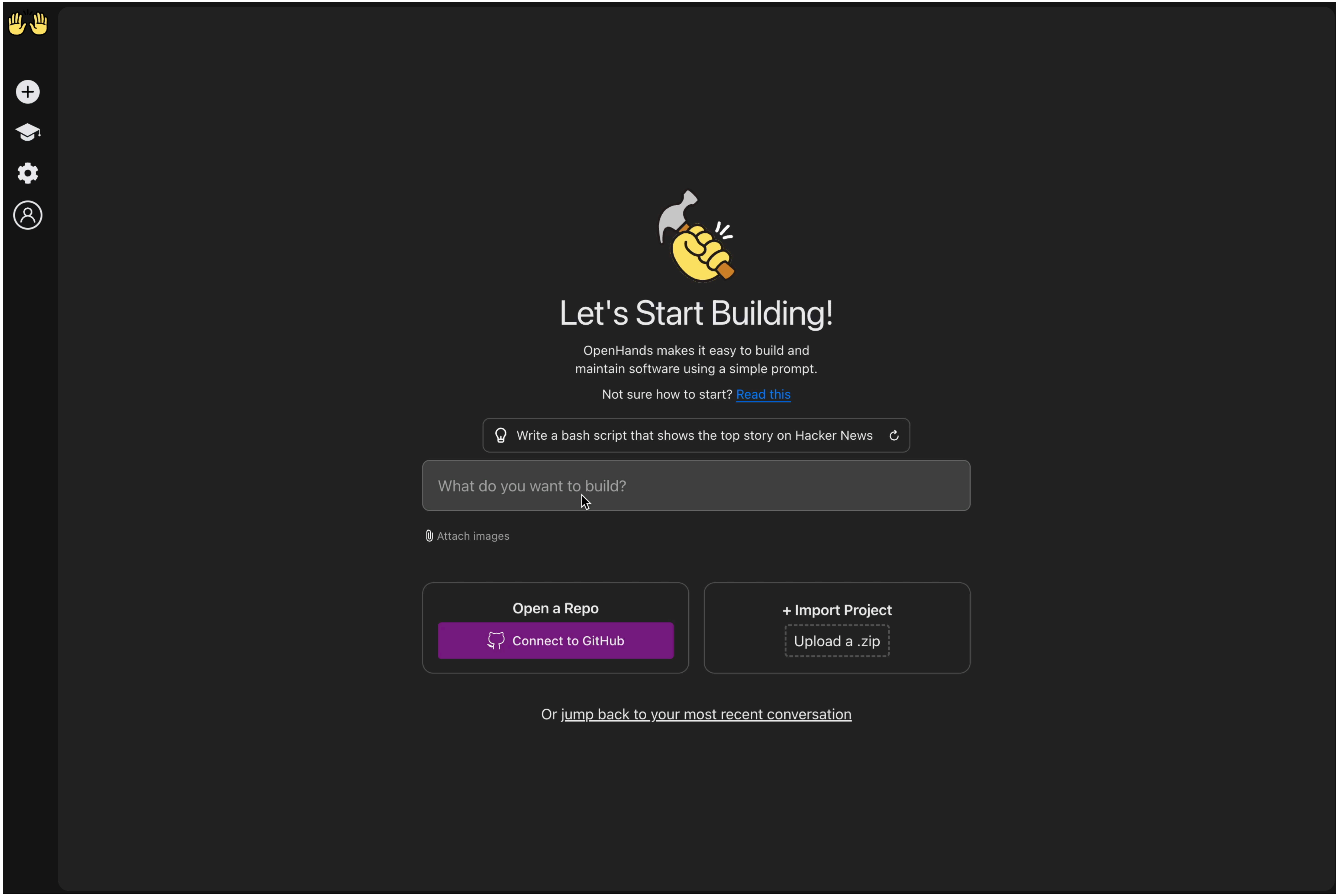
DeepSeek-V3 on Together.ai, Generated Jan 8, 2025

Today's NLP



https://openscholar.allen.ai/, Generated Jan 8, 2025

Today's NLP



https://github.com/All-Hands-AI/OpenHands , generated Jan 8 2025

In this class, you’ll learn the fundamental concepts and practical techniques underlying systems like these

Tasks Performed by NLP Systems

- Many tasks involve an input $x \in \mathcal{X}$ and an output $y \in \mathcal{Y}$, where x and/or y might involve language.

Input x	Output y	Task
Text	Label	Text Classification
Text	Text in Other Language	Translation
Image	Text	Image Captioning
Search query	List of documents	Retrieval
State of an environment	Action	Decision-Making Agent Tasks

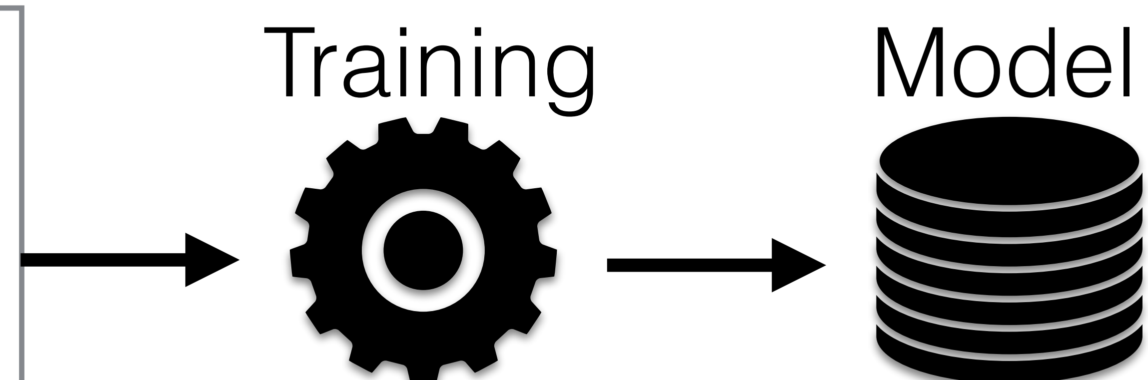
A Few Methods for Creating NLP Systems

- **Rules:** Manual creation of rules

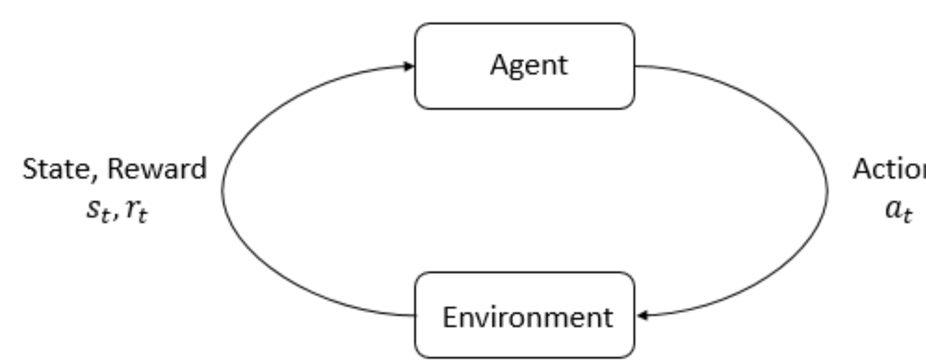
```
def classify(x: str) -> str:
    sports_keywords = ["baseball", "soccer", "football", “tennis"]
    if any(keyword in x for keyword in sports_keywords):
        return "sports"
    else:
        return "other"
```

- **Supervised learning:** Machine learning from data

I love to play baseball.	sports
The stock price is going up.	other
He got a hat-trick yesterday.	sports
He is wearing tennis shoes.	other



- **Reinforcement Learning**: Learning to maximize reward by interacting with an environment



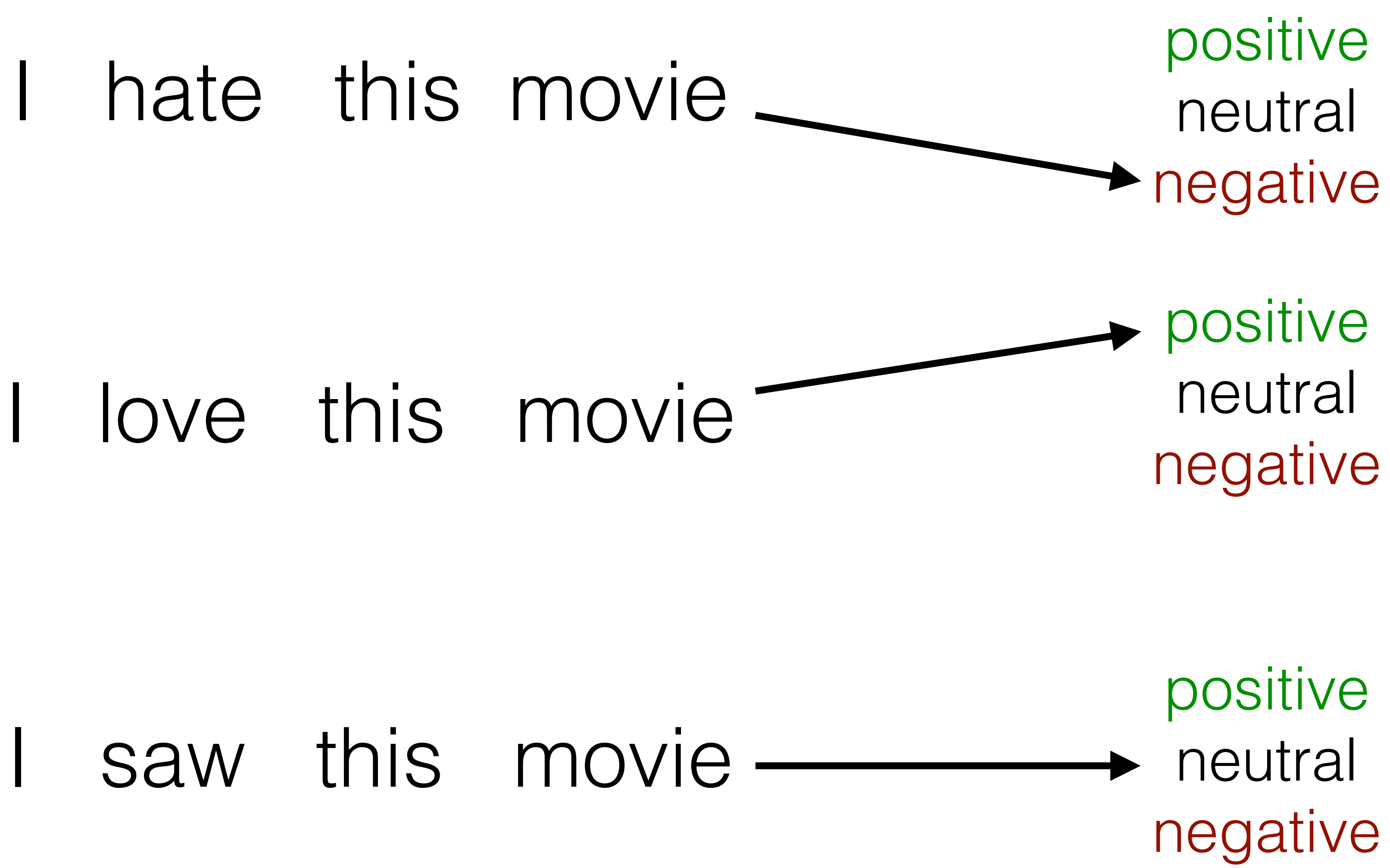
Data Requirements for System Building

- **Rules/prompting based on intuition:**
  No data needed, but also no performance guarantees
- **Rules/prompting based on spot-checks:**
  A small amount of data with input x only
- **Supervised learning:**
  Additional training set. More is often better
- **Reinforcement learning**:
  An environment (inputs, states/actions/transitions, reward)

A Rule-Based NLP System

Example: classification

- Given a review on a reviewing web site (*x*), decide whether its label (*y*) is positive (1), negative (-1) or neutral (0)



Goal: design a classifier

- $g : \mathscr{X} \rightarrow \mathscr{Y}$
  - $x \in \mathscr{X}$: input sentence
  - $y \in \mathscr{Y}$: {-1,0,1}
- We are given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$

General pattern: features and score

Extract a *feature vector* $f(x)$, and compute a score:

- $s_\theta(x) = \mathbf{w}^\top f(x)$
    - $\mathbf{w} \in \mathbb{R}^{h \times 1}$
    - $f(x) \in \mathbb{R}^{h \times 1}$
- θ are parameters (here, $\mathbf{w}$)

Making a decision

Decide which class x belongs to using the scoring function:

$$g(x) = \begin{cases} 1 & s(x) > 0 \\ 0 & s(x) = 0 \\ -1 & s(x) < 0 \end{cases}$$

Three general ingredients

- **Parameterization**: choose how the scoring function is computed and which parameters (e.g., numbers or rules) need to be set.

- **Learning**: setting the parameters based on data.

- **Inference**: make a decision given a scoring function.

Example

Parameterization:

```
def extract_features(x: str) -> dict[str, float]:
    features = {}
    x_split = x.split(' ')

    # Count the number of "good words" and "bad words" in the text
    good_words = ['love', 'good', 'nice', 'great', 'enjoy', 'enjoyed']
    bad_words = ['hate', 'bad', 'terrible', 'disappointing', 'sad', 'lost', 'angry']
    for x_word in x_split:
        if x_word in good_words:
            features['good_word_count'] = features.get('good_word_count', 0) + 1
        if x_word in bad_words:
            features['bad_word_count'] = features.get('bad_word_count', 0) + 1

    # The "bias" value is always one, to allow us to assign a "default" score to the text
    features['bias'] = 1

    return features
```

```
score = 0
for feat_name, feat_value in extract_features(x).items():
    score = score + feat_value * feature_weights.get(feat_name, 0)
```

“Learning”:

```
feature_weights = {'good_word_count': 1.0, 'bad_word_count': -1.0, 'bias': 0.5}
```

Inference:

```
if score > 0:
    return 1
elif score < 0:
    return -1
else:
    return 0
```

https://github.com/cmu-l3/anlp-fall2025-code/blob/main/01_intro/rule_based_classifier.ipynb

Code Walkthrough

https://github.com/cmu-l3/anlp-fall2025-code/blob/main/01_intro/rule_based_classifier.ipynb

- See code for all major steps:
  1. Computing features
  2. Scoring
  3. Inference
  4. Accuracy calculation
  5. Error analysis

Some Difficult Cases

Low-frequency Words

The action switches between past and present , but the material link is too **tenuous** to anchor the emotional connections that **purport** to span a 125-year divide .

negative

Here 's yet another studio horror franchise **mucking** up its storyline with **glitches** casual fans could correct in their sleep .

negative

Solution?: Keep working until we get all of them? Incorporate external resources such as sentiment dictionaries?

Conjugation

An operatic , sprawling picture that 's **entertainingly** acted , **magnificently** shot and gripping enough to sustain most of its 170-minute length .

positive

It 's basically an **overlong** episode of Tales from the Crypt .

negative

Solution?: Use the root form and part-of-speech of word?

Negation

This one is not nearly as dreadful as expected .

positive

Serving Sara does n't serve up a whole lot of laughs .

negative

Solution?: If a negation modifies a word, disregard it?

Metaphor, Analogy

Puts a human face on a land most Westerners are unfamiliar with.

positive

Green might want to hang onto that ski mask , as robbery may be the only way to pay for his next project .

negative

Has all the depth of a wading pool .

negative

Solution?: ???

Other Languages

見事に視聴者の心を掴む作品でした。

positive

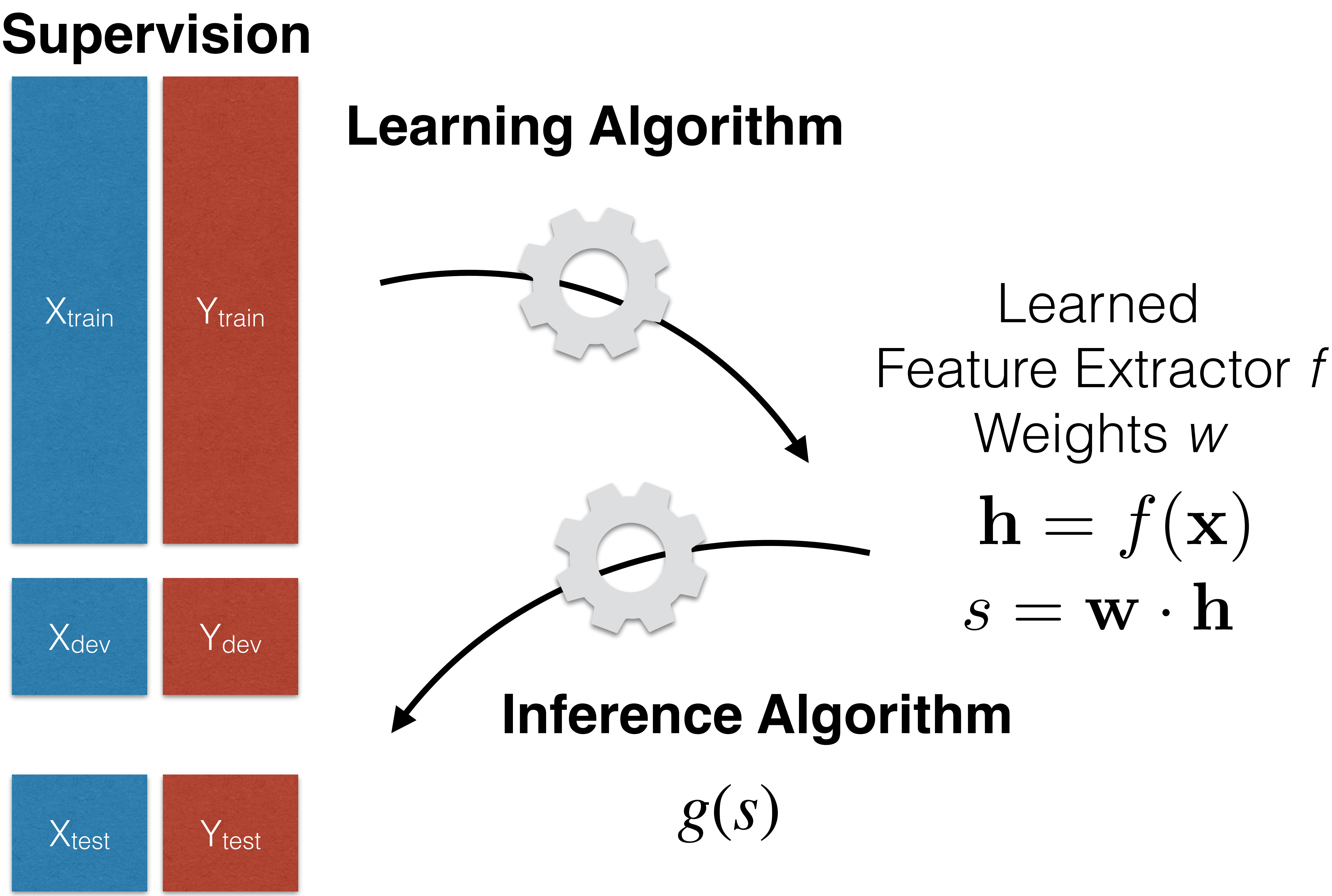
モンハンの名前がついてるからとりあえずモンハン要素をちょこちょこ入れればいいだろ感が凄い。

negative

Solution?: Learn Japanese and re-do all the work?

Learning the Scoring Function

Learning the scoring function



A more general recipe

- Goal: Learn a scoring function ("energy function") that says how compatible output y is for input x:

$$s_\theta(x, y) \in \mathbb{R}$$

- Higher score: more compatible.
  Lower score: less compatible.

- Binary classifier: $y \in \{-1, 1\}$
  - $s_\theta(x) = \mathbf{w}^\top f(x)$
  - $s_\theta(x, y) = y \cdot s_\theta(x)$

- Multi-class: $y \in \{0, 1, \ldots, K\}$
  - $s_\theta(x) = \mathbf{W}^\top f(x)$
    - $\mathbf{W} \in \mathbb{R}^{h \times K}$
    - $f(x) \in \mathbb{R}^{h \times 1}$
  - $s_\theta(x, y) = s_\theta(x)[y]$

*The (negative) score is also referred to as an "energy" $E(x, y) = -s(x, y)$
See e.g., [LeCun 2006, Cho 2025]

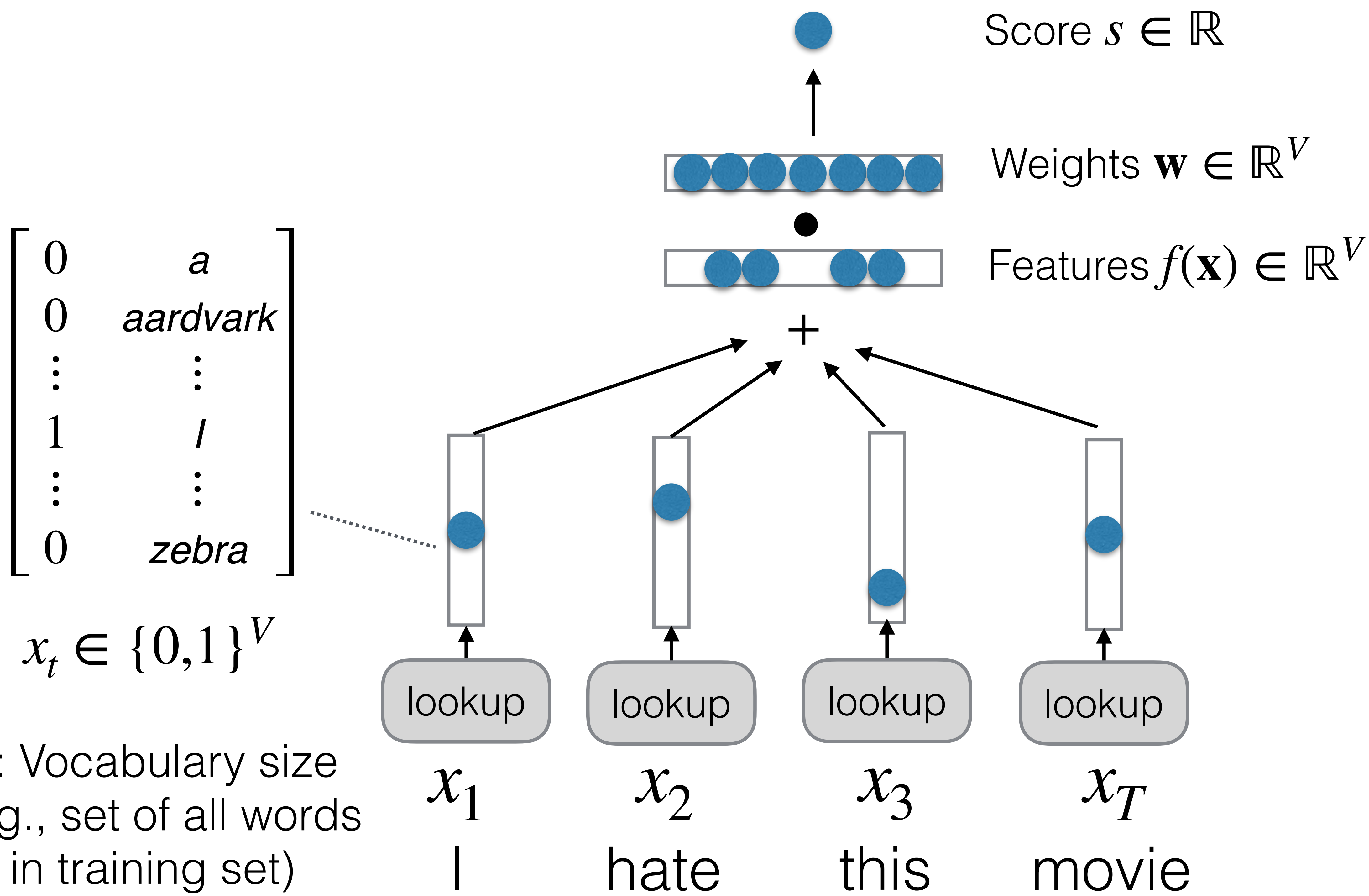
Three general ingredients

- Goal: Learn a scoring function (“energy function”) that says how compatible output y is for input x:

$$s_\theta(x, y) \in \mathbb{R}$$

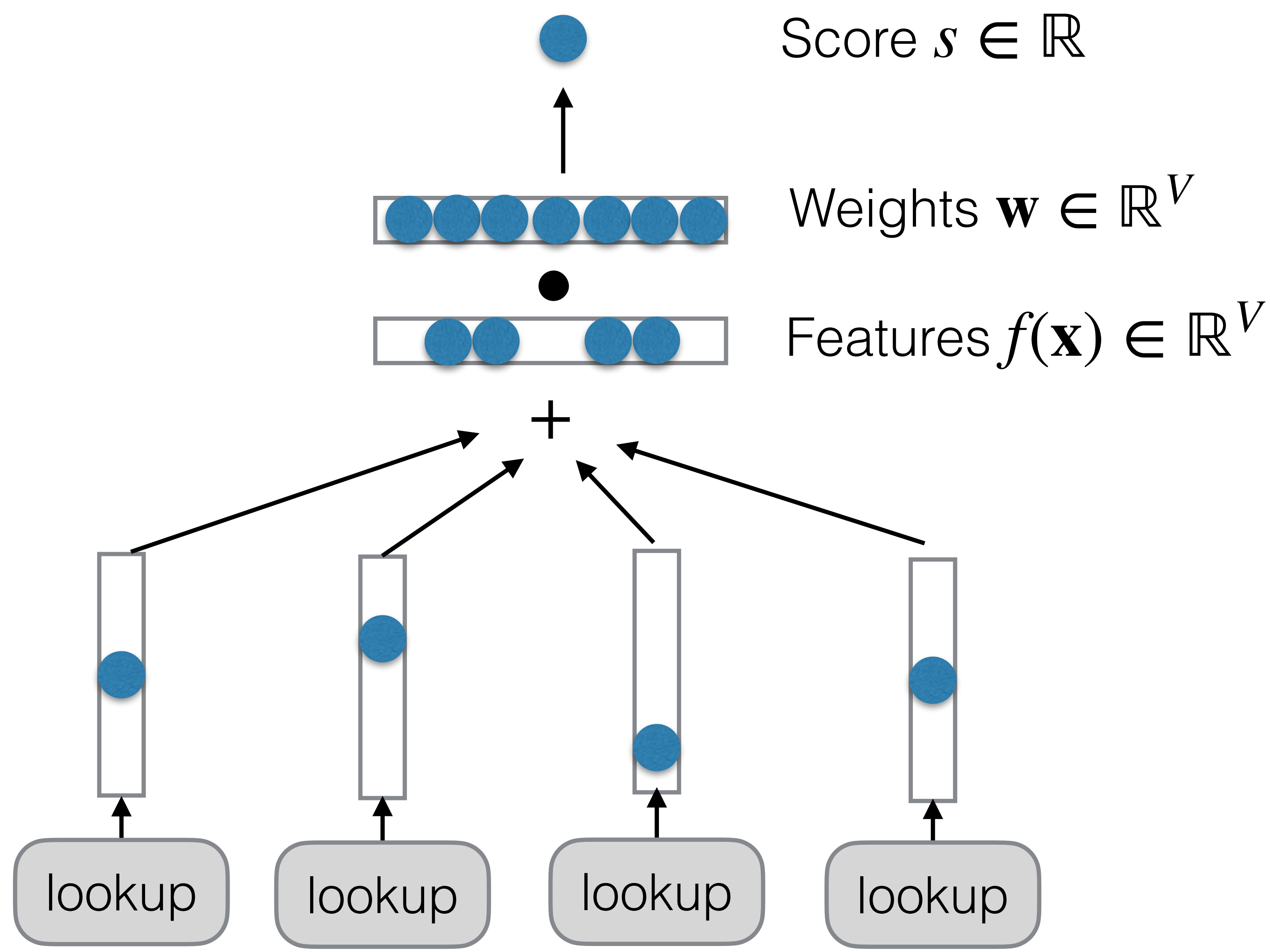
1. **Parameterization**: the form and parameters of the function (e.g., neural net architecture and its weights).
2. **Learning**: how we adjust the parameters using supervision (e.g., using input-output examples, a reward function).
3. **Inference**: how we make decisions after learning.

Example Parameterization: Bag of Words (BoW)



V: Vocabulary size (e.g., set of all words in training set)

Example Parameterization: Bag of Words (BoW)



Features *f* are based on word identity, weights *w* learned

Which problems mentioned before would this solve?

What do the parameters represent?

- **Binary classification:** Each word has a single scalar, positive indicating "positive" and negative indicating "negative"
- **Multi-class classification:** Each word has its own 5 elements corresponding to e.g. [very pos, pos, neutral, neg, very neg]

<u>Binary</u>

$\mathbf{w} \in \mathbb{R}^V$

love	2.4
hate	-3.5
nice	1.2
no	-0.2
dog	-0.3
…	…

<u>Multi-class</u>

$\mathbf{W} \in \mathbb{R}^{V \times K}$
$K = 5$

	v. positive	positive	neutral	negative	v. negative
love	2.4	1.5	-0.5	-0.8	-1.4
hate	-3.5	-2.0	-1.0	0.4	3.2
nice	1.2	2.1	0.4	-0.1	-0.2
no	-0.2	0.3	-0.1	0.4	0.5
dog	-0.1	0.3	0.6	0.2	-0.2
…			…		

Example inference

- Example for a binary classifier:

$$\hat{y} = \text{argmax}_y s_\theta(x, y)$$

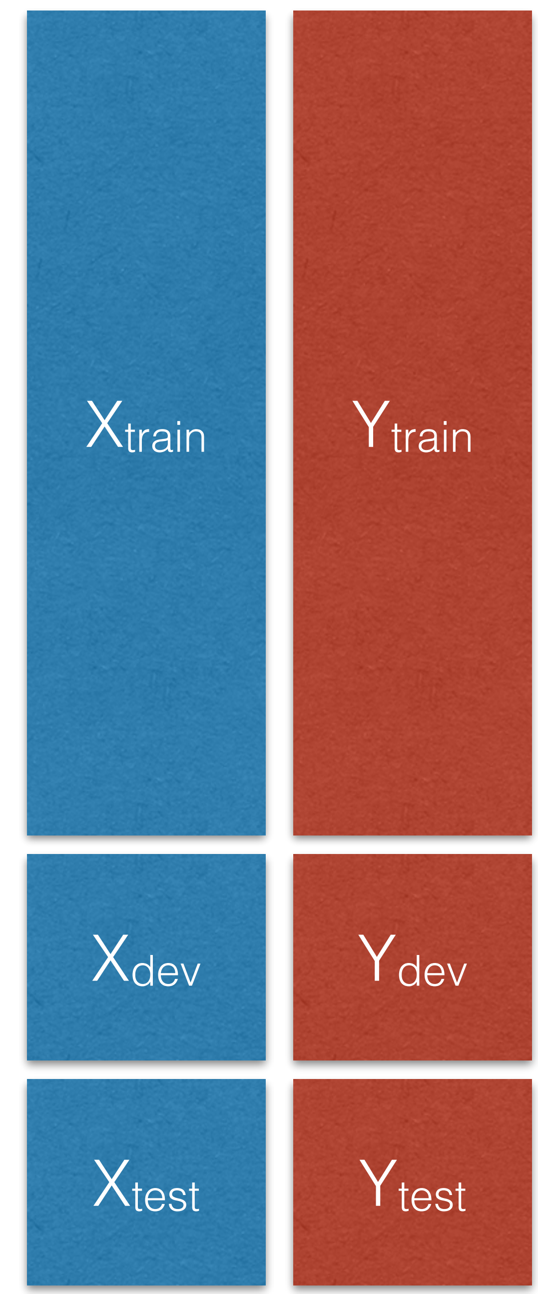
$$= \text{argmax}_{y \in \{-1,1\}} y s_\theta(x)$$

$$= \text{sign}(s_\theta(x))$$

E.g., the output scalar from the bag-of-words model on the previous slide

Example learning

- Given (x, y) examples split into $D_{train}, D_{dev}, D_{test}$
- Define a loss function:
- $$\mathscr{L}(\theta, D) = \sum_{(x,y) \in D} L(x, y, \theta)$$
- Run an algorithm that solves:
- $$\min_{\theta} \mathscr{L}(\theta, D_{train})$$



Example learning

- Use an algorithm called “structured perceptron”

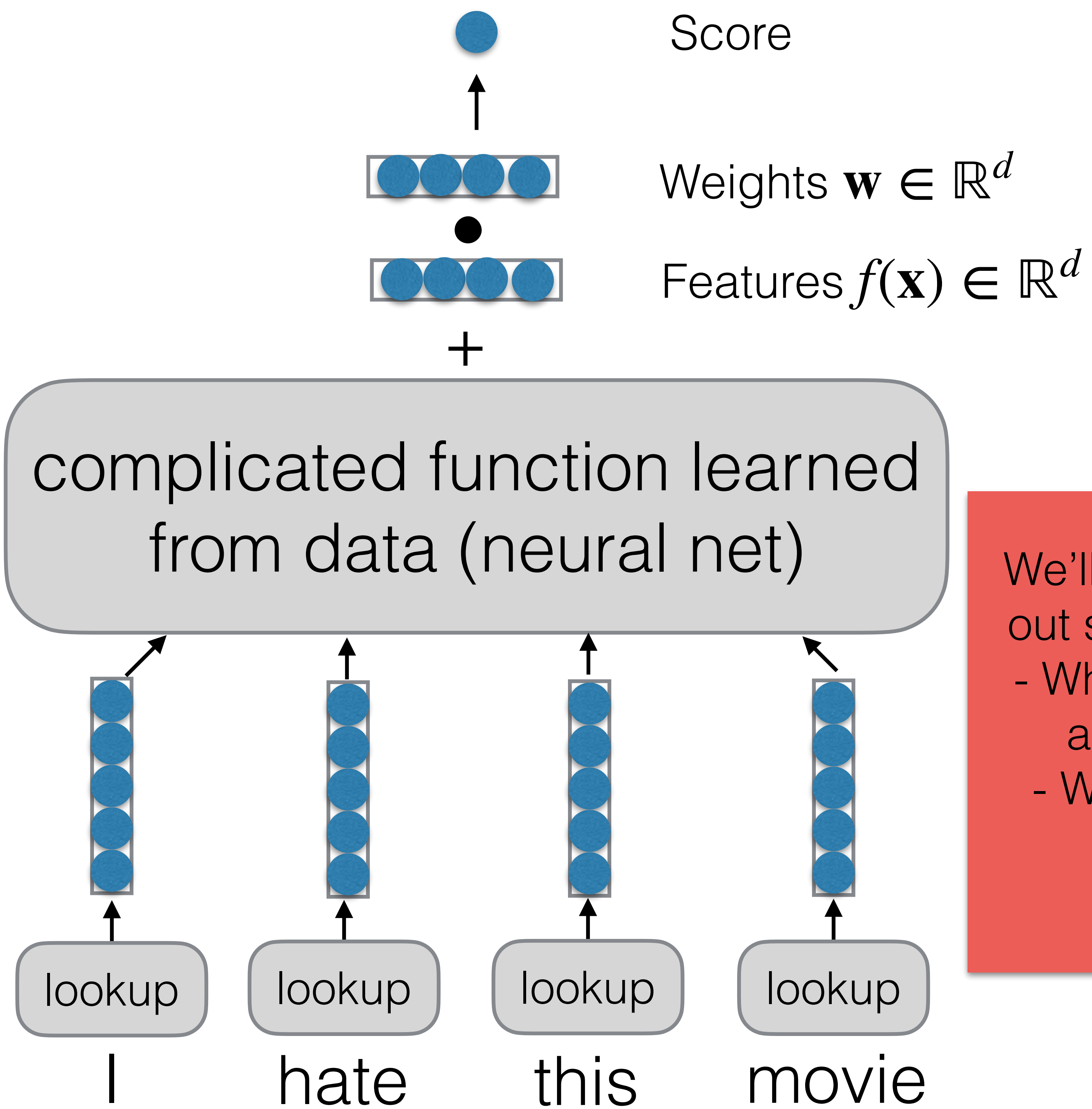
```
feature_weights = {}
for x, y in data:
    # Make a prediction
    features = extract_features(x)
    predicted_y = run_classifier(features)
    # Update the weights if the prediction is wrong
    if predicted_y != y:
        for feature in features:
            feature_weights[feature] = (
                feature_weights.get(feature, 0) +
                y * features[feature]
            )
```

https://github.com/cmu-l3/anlp-fall2025-code/blob/main/01_intro/trained_bow_classifier.ipynb

What’s Missing?

- Handling of *conjugated or compound words*
    - I **love** this move -> I **loved** this movie
- Handling of *word similarity*
    - I **love** this move -> I **adore** this movie
- Handling of *sentence structure*
    - It has an interesting story, **but** is boring overall
- …

A Better Parameterization: Neural Networks



We'll need to figure out several details:
- Which neural net architecture?
- Which learning algorithm?

…

From classification to general tasks

A General Recipe

- Build a parameterized scoring function ("energy function") that says how compatible output y is for input x:

$$s_\theta(x, y) \in \mathbb{R}$$

- **Parameterization**: choose form of s_θ and the parameters to set
- **Learn** the parameters using supervision (e.g., labels, rewards)
- **Inference:** select an output (e.g., maximization, sampling)

$$\hat{y} = g(s, x)$$

A General Recipe

- Classification: assign high scores to correct classes, low scores to incorrect classes.
- Ranking: given a query x, assign scores to documents $y_1, y_2, \ldots$ so that they're in the correct order
- Probabilistic modeling: assign scores so that we have a distribution $p(y | x)$
    - Example:
        - x: English sentence, y: Japanese sentence
        - x: Conversation history, y: response
        - …

From scores to probabilities

- Given a scoring function, we can build a probabilistic model:

$$p_\theta(y \mid x) = \frac{\exp\left(s_\theta(x, y)\right)}{\sum_{y'} \exp\left(s_\theta(x, y')\right)}$$

- For instance:
  - `I hate this movie -> [negative = 0.98, neutral = 0.01, positive = 0.01]`

- With a probabilistic model we can do inference by **sampling**:

$$\hat{y} \sim p_\theta(y \mid x)$$

From classification to generation

- Now suppose the output space is any sequence (of text, images, etc.):

$$p_\theta(y\,|\,x) = \frac{\exp\left(s_\theta(x, y)\right)}{\sum_{y'} \exp\left(s_\theta(x, y')\right)}$$

  - `I hate this movie -> because it isn't creative.`

- We can generate text, images, or make decisions by sampling.
  - Example: large language models
- We'll cover the parameterization, learning, and inference for achieving this!

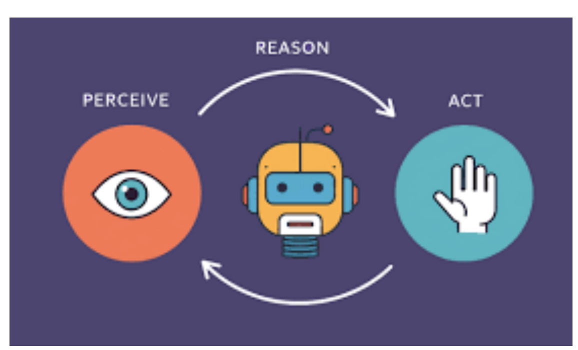
From generation to actions

- We can use such a model to form a "*policy*" that is used to decide which action a to take in state s:

$$\pi(a \mid s) \iff p_\theta(y = a \mid x = s)$$

  - `S: {Movie streaming website}`
    `The user said: "I hate this movie"`
  - `A: [CLICK]pause button`

- Example: AI agents



Roadmap

Goal: build good learning-based systems for any NLP task

- **Parameterization**:
  - Neural network architectures
  - Autoregressive, diffusion
  - Images, retrieval, tools
- **Learning**
  - Unstructured data
  - Paired data
  - Environment with reward function
- **Inference**
  - Optimization and sampling
  - Multi-sample strategies
  - Efficient strategies

Broadly: fundamentals -> advanced

Fundamentals of cutting-edge NLP:

- Lectures 1-17

Advanced topics in cutting-edge NLP:

- Lectures 18-27

Topic 1: Fundamentals

- Fundamentals
  - General framework: Lecture 1
  - Deep learning and learned representations: Lecture 2
  - Language modeling: Lecture 3

# 2 08/28/2025	Lecture	**Fundamentals** Fundamentals: Learned Representations	Main readings: ◦ Natural Language Understanding with Distributed Representation (Ch. 2, Ch. 3) (Cho 2015) ▸ Additional references

Note: Lecture X means class meeting slot #X on the schedule

Topic 2: Neural Network Architectures for NLP

Fundamentals:

- Recurrent neural networks: lecture 4
- Attention and transformers: lecture 5

Advanced:

- Long sequence models: lecture 24
- Mixture of experts: lecture 25

Topic 3: Learning and Inference for NLP

Fundamentals:

- Pre-training: lecture 6
- In-context learning: lecture 7
- Fine-tuning and distillation: lecture 8
- Decoding algorithms: lecture 9

Advanced:

- Advanced inference strategies: lecture 26

Topic 4: Generative Models for NLP

Fundamentals:

- Autoregressive models: lecture 2
- Retrieval and RAG: lecture 10
- Multimodal models: lecture 11, 12

Advanced:

- Diffusion: lecture 27

Topic 5: Evaluation and research skills

Fundamentals:

- Evaluation techniques: lecture 13
- Experimental design & research skills: lecture 14

Topic 6: Reinforcement Learning and Agents in NLP

Fundamentals / advanced:

- RL fundamentals: lecture 17
- RL applications in NLP: lecture 18
- Agents: lecture 19

Topic 7: Scaling, Efficiency, and Deployment

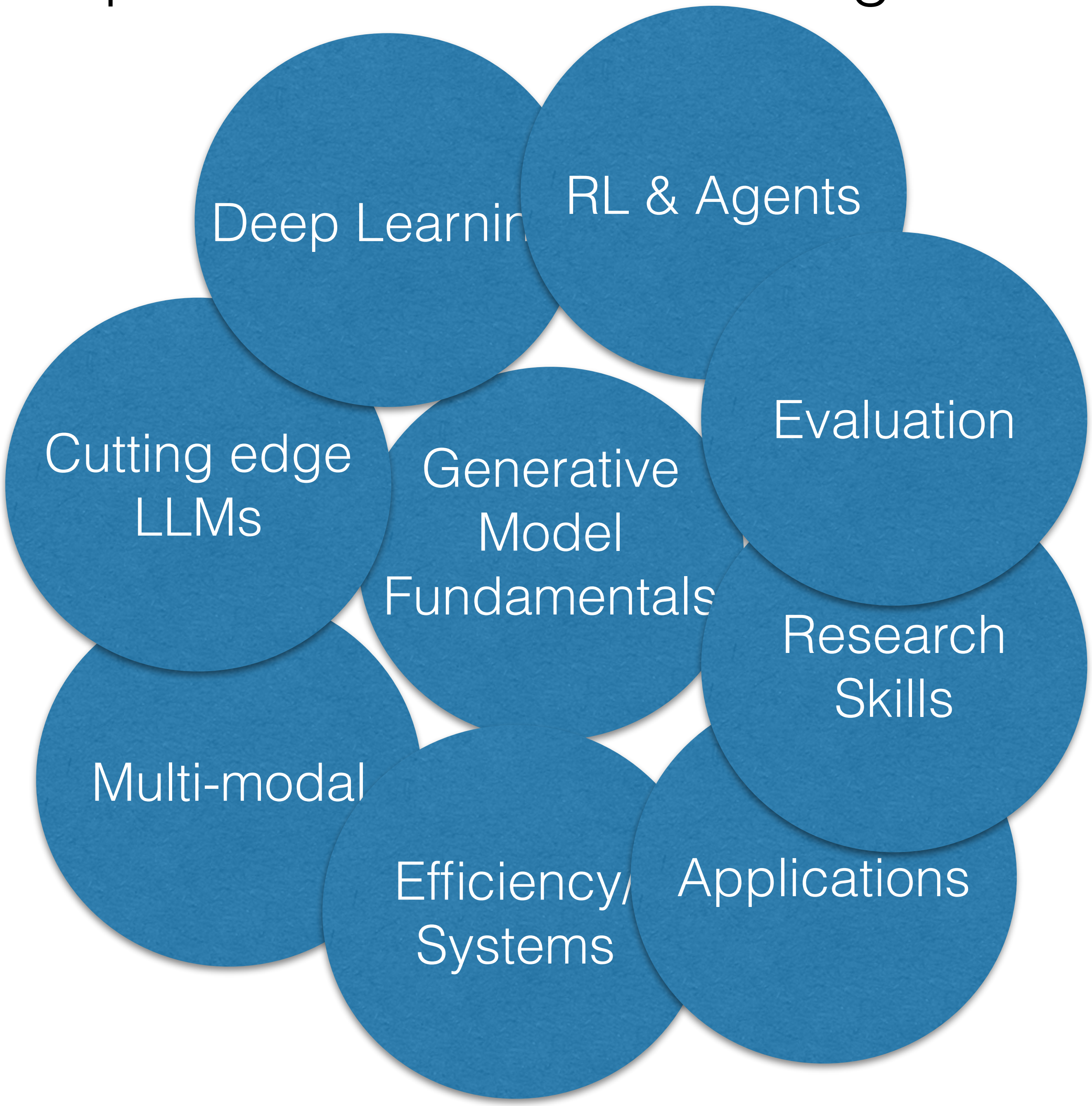
Advanced:

- Quantization: lecture 22
- Parallel and distributed training: lecture 23
- Advanced inference strategies: lecture 26

Comparison to other courses

Advanced NLP introduces you to the fundamental tools and concepts from around NLP. To go in further depth:

- **Advanced Deep Learning (10-707)**
  - Focus on fundamental building blocks of deep learning
- **Large Language Models (11-667)**:
  - Focus on large-scale autoregressive language models of text
- **Multimodal Machine Learning (11-777)**
  - Focus on non-text
- **Systems (11-868, 15-642)**
  - Focus on systems, scaling, efficiency



- **Reinforcement learning (10-703)**
  - Focus on reinforcement learning
- **Code generation (11-891)**
  - Focus on applications related to code
- **Inference for LMs (11-664)**
  - Focus on language model inference

Class Format/Structure

Class Content

- Learn in detail about **building NLP systems from a research perspective**
- Learn basic and advanced topics in **machine learning approaches** to NLP and language models
- See several case studies of **NLP applications** and learn how to identify unique problems for each
- Learn how to debug **when and where NLP systems fail**, and build improvements based on this

Class Format

- **Before class:** Do recommended reading
- **During class:**
    - *Lecture/Discussion:* Go through material and discuss
    - *Code/Data Walkthrough:* The instructor will sometimes walk through some demonstration code, data, or model predictions
- **After class:** Do quiz about class or reading material

Assignments

- **Assignment 1 - Build-your-own LLaMa:** *Individually* implement LLaMa model loading and training
- **Assignment 2 - NLP Task from Scratch:** *In a team,* perform data creation, modeling, and evaluation for a specified task
- Project
  - **Assignment 3 - Survey and re-implementation:** Survey literature, re-implement and reproduce results from a recently published NLP paper
  - **Assignment 4 - Final project:** Perform a unique project that either (1) improves on state-of-the-art, or (2) applies NLP models to a unique task. Present a poster and write a report.
- For assignments 1-3, we give a total of 5 late days. Feel free to use these for unexpected circumstances or delays.

Quizzes

- Released by the end of the day of a lecture (11:59pm).
- Due at the end of the following day (11:59pm).
- Example:
  - For Thursday's lecture (8/28), the quiz will be released by 11:59pm on 8/28 and due by 11:59pm on 8/29.
- We will drop your three lowest quiz grades.
  - Feel free to use these for unexpected circumstances.

Recordings and Attendance

- We will do our best to send a Zoom recording by the end of the day of the lecture.
- ***Attendance***: we expect you to attend courses and participate in discussions during class.
    - We do not allow registering for the course when you have a schedule conflict.
    - We will not make exceptions for quizzes if there are Zoom connection issues, recording issues/delays, etc.
    - You ***absolutely must*** attend:
        - Project Hours (10/30/2025)
        - Project Poster Sessions (12/02/2025 and 12/04/2025)

Waitlist

- We have a long waitlist; thank you for the excitement!
- **Policy**: out of fairness, we can’t prioritize individual cases.

Should I take this course?

- I’m certain that you’re excited about the course content!
- Please be sure that you will be able to satisfy the logistics associated with the quizzes, attendance, and other aspects of the course.

Teaching Team and Resources

- **Instructor:** Sean Welleck
- **TAs:**
  - Chen Wu (Head TA)
  - Joel Mire, Dareen Alharthi, Neel Bhandari, Akshita Gupta, Ashish Marisetty, Manan Sharma, Sanidhya Vijayvargiya
  - **Office hours**: see course website. They will begin on 9/2.
- **Website:** https://cmu-l3.github.io/anlp-fall2025/
- **Code:** https://github.com/cmu-l3/anlp-fall2025-code
- **Piazza:** https://piazza.com/cmu/fall2025/11711

Thank you