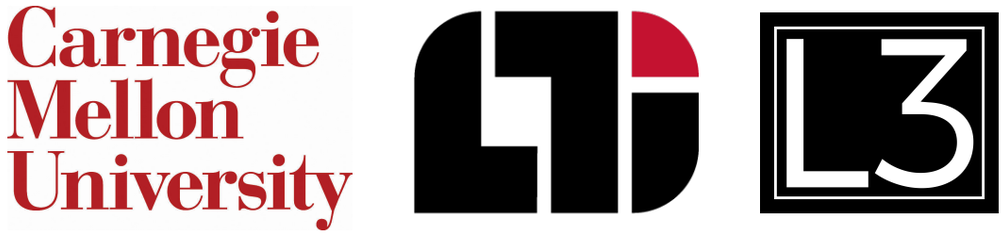
CS11-711 Advanced NLP

Introduction and Fundamentals

Daphne Ippolito and Fernando Diaz



https://cmu-anlp.org

Slides adapted from Sean Welleck and Graham Neubig with additional content borrowed from Alane Suhr and Sewon Kim

Today's Schedule

1. What is NLP?
2. (Computer) Learning in NLP
3. Semester Roadmap
4. Logistics

1. What is NLP?

What is Natural Language Processing (NLP)?

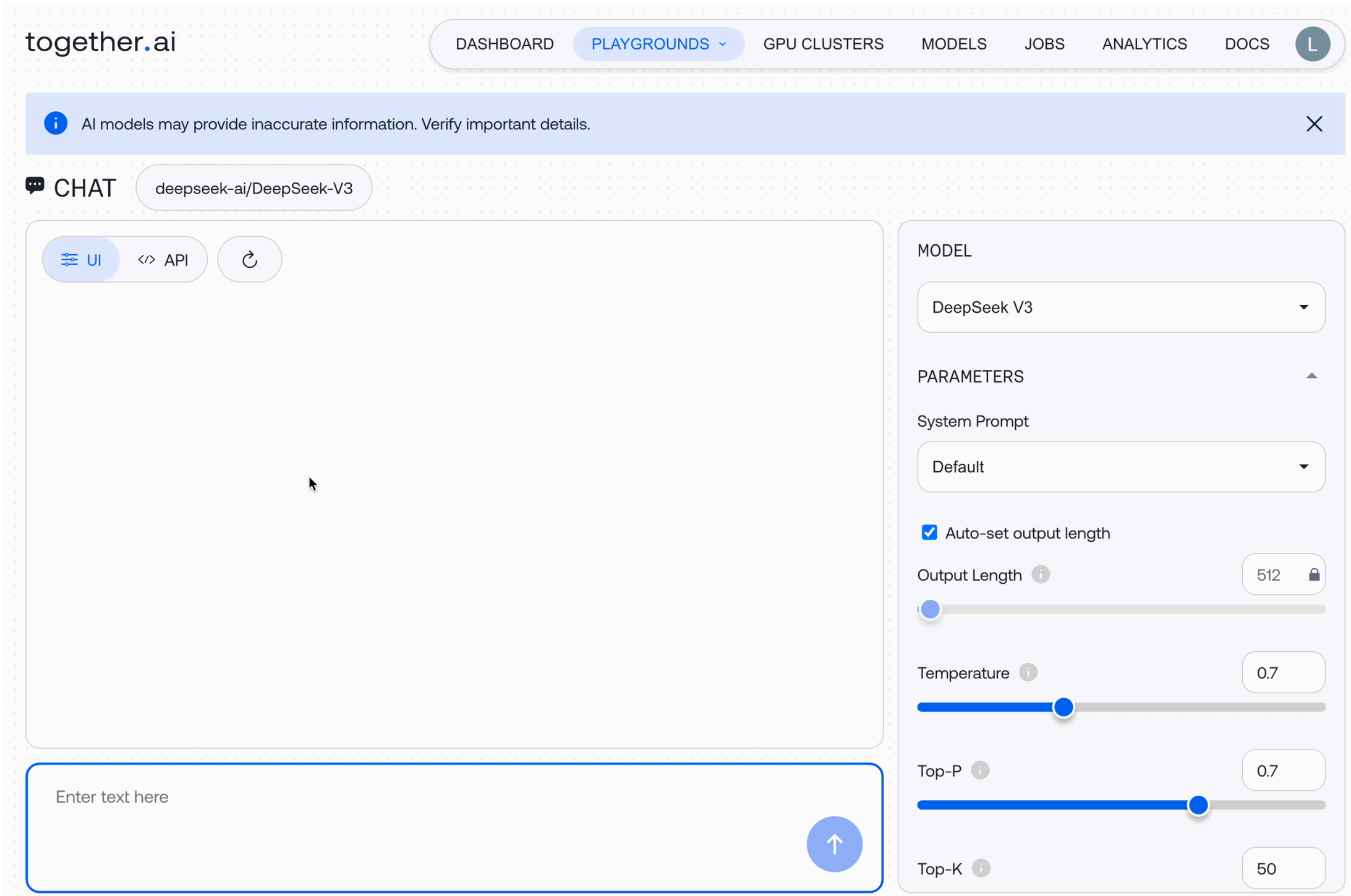
- Computer systems that can process, understand, generate, and interact with natural language in a genuinely meaningful way. Some key aspects:

  ➡ **Learn useful representations**: capture meaning from language in a structured way that is useful for downstream tasks (e.g., embeddings used to classify a document)

  ➡ **Generate language**: create language for tasks like dialogue, translation, or question answering.

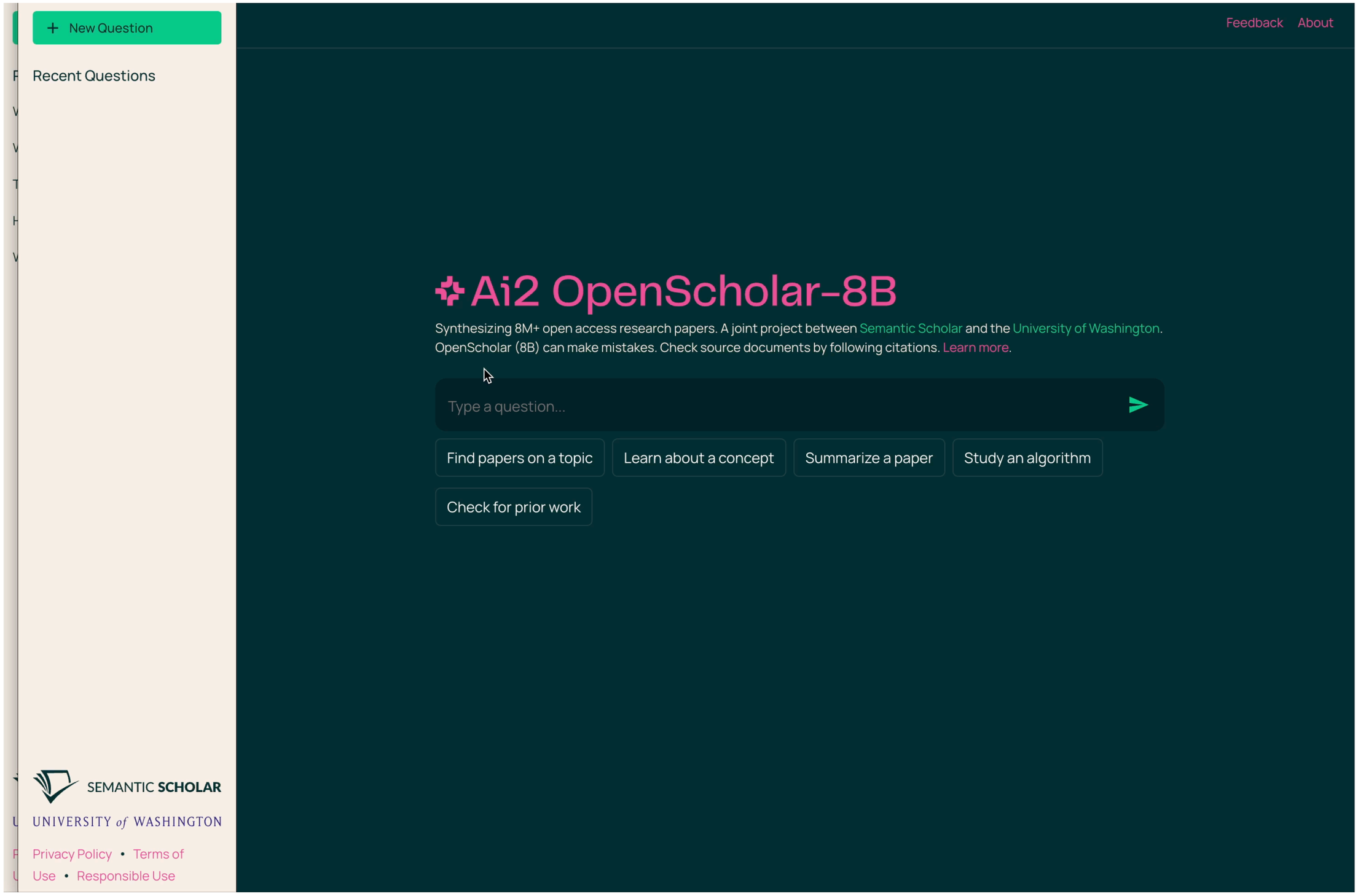
  ➡ **Bridge language and action**: Use language to perform tasks, solve problems, and interact with environments.

Today's NLP



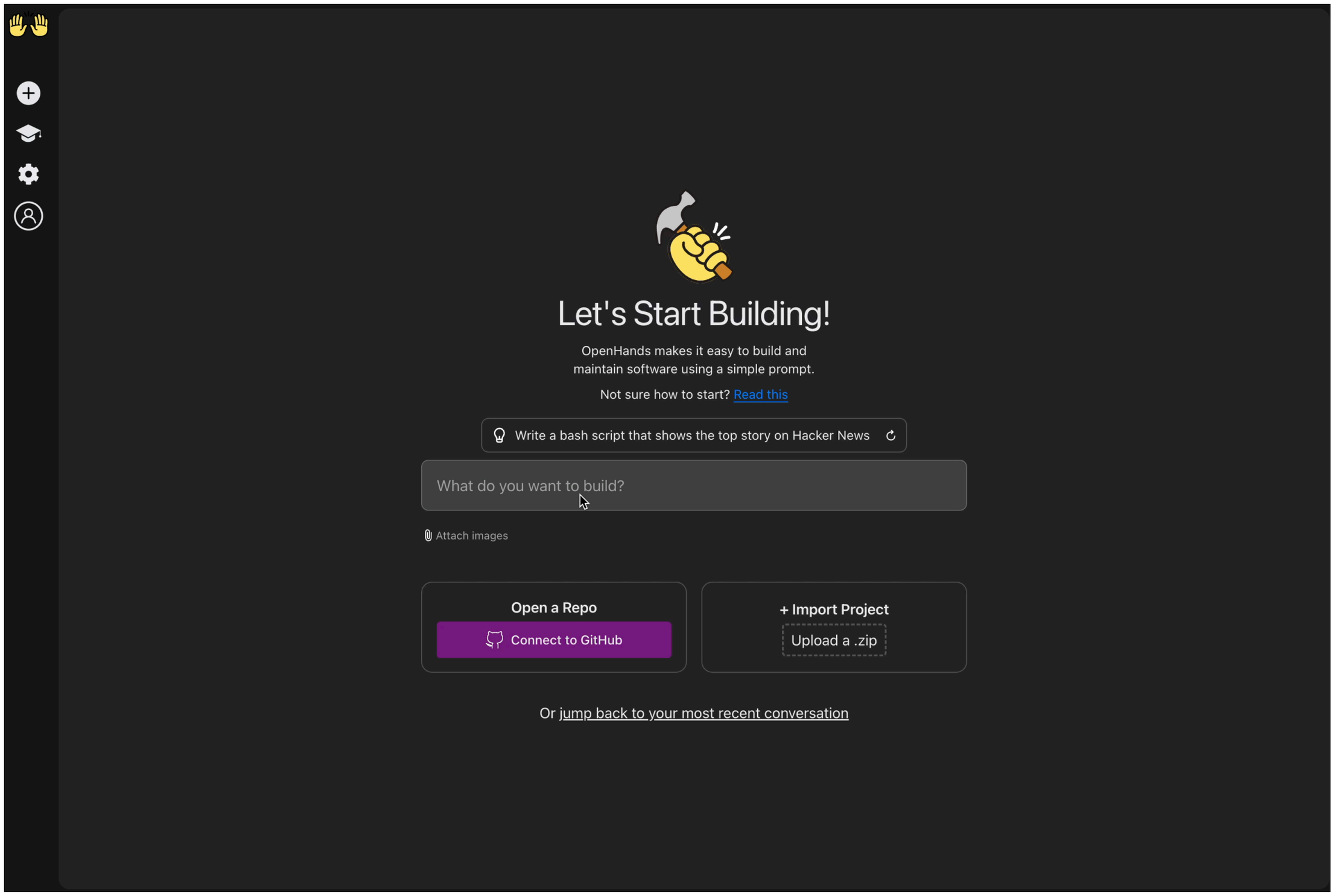
DeepSeek-V3 on Together.ai, Generated Jan 8, 2025

Today's NLP



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

Today's NLP

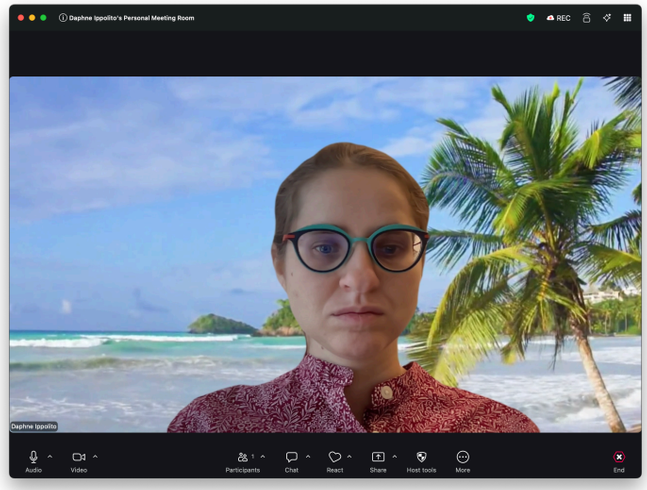


https://github.com/All-Hands-AI/OpenHands , generated Jan 8 2025

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

Why can human language be difficult for computers to understand?

- Ambiguity
  - “Can you tell me a little bit about your background?”
-



Why can human language be difficult for computers to understand?

- Ambiguity
  - “Can you tell me a little bit about your background?”
- Dialects and accents
  - “I’m annoyed about my flat.”

Why can human language be difficult for computers to understand?

- Ambiguity
  - “Can you tell me a little bit about your background?”
- Dialects and accents
  - “I’m annoyed about my flat.”
- Language is evolving
  - “6-7”, “harness”

Why can human language be difficult for computers to understand?

- Ambiguity
  - “Can you tell me a little bit about your background?”
- Dialects and accents
  - “I’m annoyed about my flat.”
- Language is evolving
  - “6-7”, “harness”
- Pragmatics
  - Gemini: “That’s such a wonderful question!”

Why can human language be difficult for computers to understand?

- Ambiguity
  - “Can you tell me a little bit about your background?”
- Dialects and accents
  - “I’m annoyed about my flat.”
- Language is evolving
  - “6-7”, “harness”
- Pragmatics
  - Gemini: “That’s such a wonderful question!”
- Humor, sarcasm, irony

Why can human language be difficult for computers to understand?

- Ambiguity
  - “Can you tell me a little bit about your background?”
- Dialects and accents
  - “I’m annoyed about my flat.”
- Language is evolving
  - “6-7”, “harness”
- Pragmatics
  - Gemini: “That’s such a wonderful question!”
- Humor, sarcasm, irony
- Longterm dependencies

In this class, we ask:

- What goes into building the state-of-the-art NLP systems that **work well at some tasks**?

- Where and why do current state-of-the-art NLP systems still **struggle**?

- How can we **make appropriate improvements** and **achieve whatever we want to do with NLP**?

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"
```

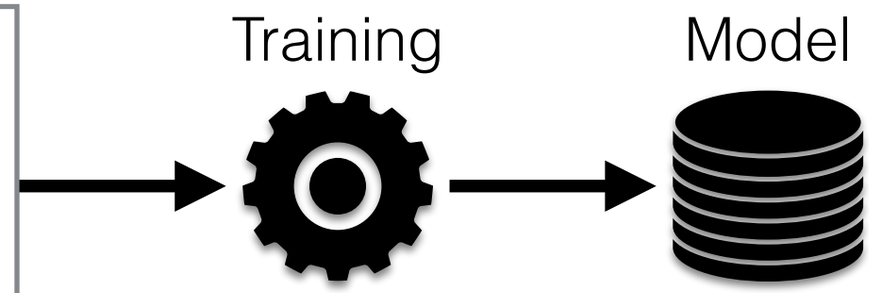
A Few Methods for Creating NLP Systems

- **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 labeled 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



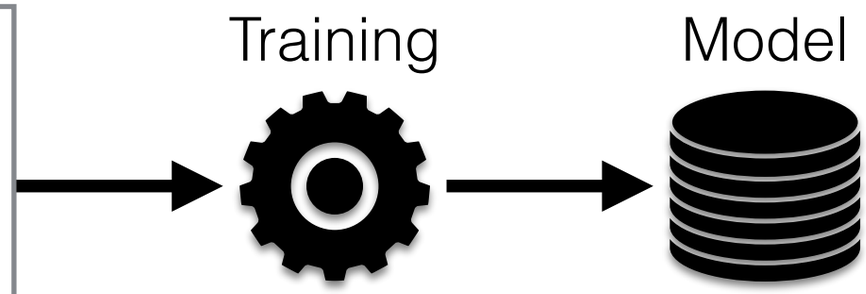
A Few Methods for Creating NLP Systems

- **Rules:** Manual creation of rules

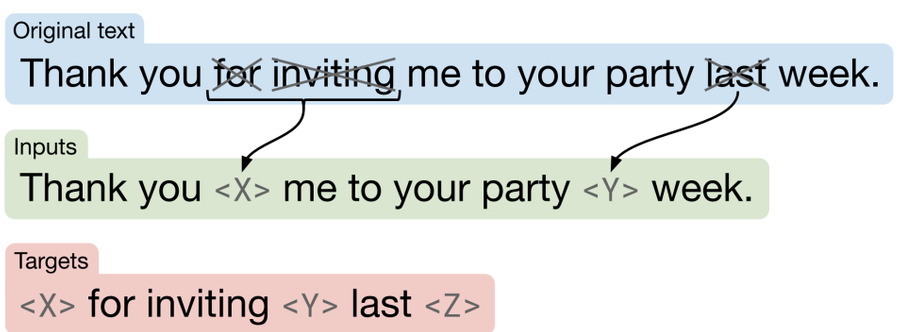
```
def classify(x: str) -> str:
    sports_keywords = ["baseball", "soccer", "football", "tennis"]
    if any(keyword in x for keyword in sports_keywords):
        return "sports"
    else:
        return "other"
```

- **Supervised learning:** Machine learning from labeled 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



- **Unsupervised learning**: Machine learning from unstructured data



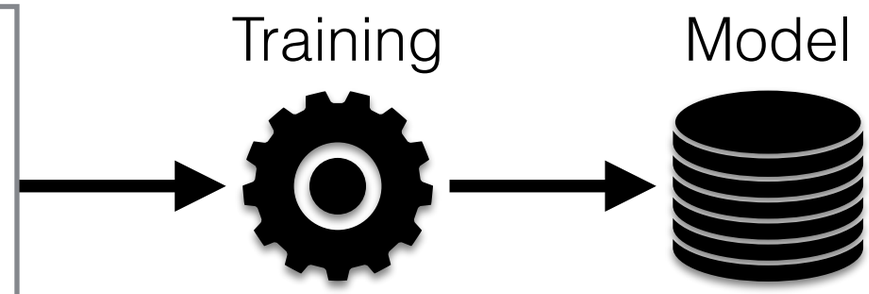
A Few Methods for Creating NLP Systems

- **Rules:** Manual creation of rules

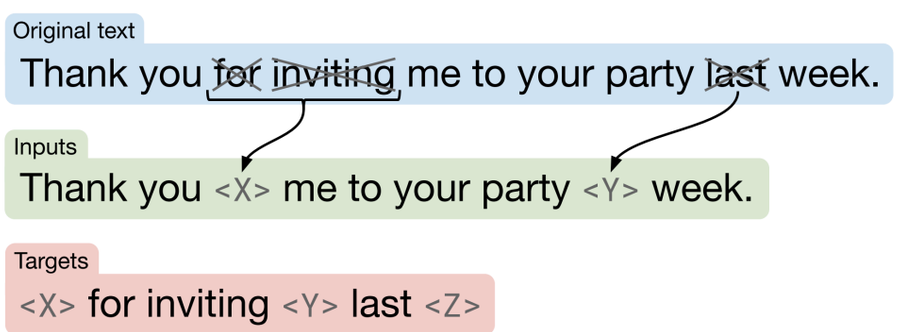
```
def classify(x: str) -> str:
    sports_keywords = ["baseball", "soccer", "football", "tennis"]
    if any(keyword in x for keyword in sports_keywords):
        return "sports"
    else:
        return "other"
```

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

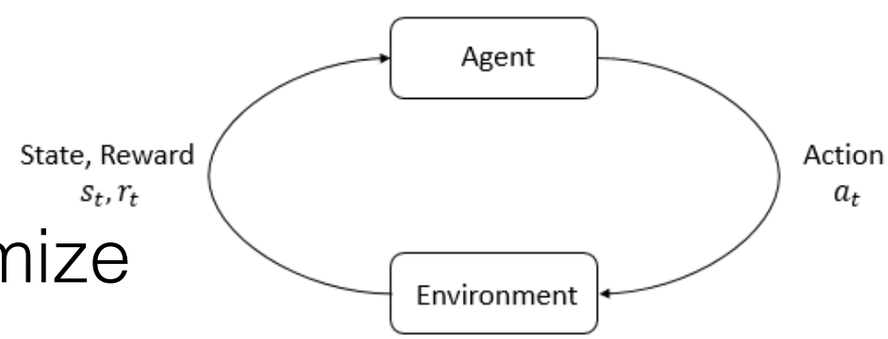
I love to play baseball.	sports
The stock price is going up.	other
He got a hat-trick yesterday.	sports
He is wearing tennis shoes.	other



- **Unsupervised learning**: Machine learning from unstructured data



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

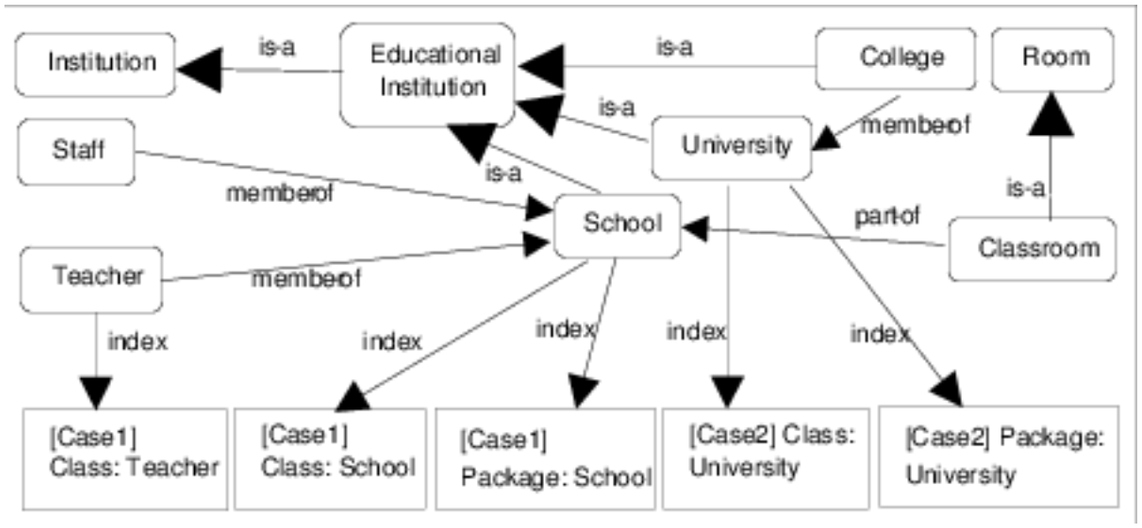


In all forms of NLP, data is key.

- Either data is needed to **develop/train** the system, or it is needed to **assess** how well the system is working.
- Before neural networks, computational linguists spent years collecting the data needed to build effective NLP systems.

In all forms of NLP, data is key.

- Either data is needed to **develop/train** the system, or it is needed to **assess** how well the system is working.
- Before neural networks, computational linguists spent years collecting the data needed to build effective NLP systems.



Wordnet

Battle-tested/NNP*/JJ industrial/JJ managers/NNS here/RB always/RB buck/VB*/VBP up/IN*/RP nervous/JJ newcomers/NNS with/IN the/DT tale/NN of/IN the/DT first/JJ of/IN their/PP$ countrymen/NNS to/TO visit/VB Mexico/NNP ,/, a/DT boatload/NN of/IN samurai/NNS*/FW warriors/NNS blown/VBN ashore/RB 375/CD years/NNS ago/RB ./.

Penn TreeBank

ID	English	German
1	I have a pen.	Ich habe einen Stift.
2	Can you come here?	Kannst du hierher kommen?
3	The sun is shining.	Die Sonne scheint.
4	She is a teacher.	Sie ist eine Lehrerin.
5	We went to the park.	Wir sind in den Park gegangen.
6	The cat is sleeping.	Die Katze schläft.
7	He likes to play soccer.	Er spielt gerne Fußball.
8	My favorite color is blue.	Meine Lieblingsfarbe ist blau.
9	She is reading a book.	Sie liest ein Buch.

WMT

Example of a classification 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)

I hate this movie → positive, neutral, negative

I love this movie → positive, neutral, negative

I saw this movie → positive, neutral, negative

Goal: design a classifier

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

Three general ingredients of a classifier

- **Modeling/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. You as a huge can do this; an optimizer can too.

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

One option for parameterizing the classifier

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, $\mathbf{w}$)

Options for the feature vector

Rule-based:

```python
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

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

-

Learning feature weights

- Rule-based system:
  - “Good” words are given weight 1, bad words weight weight -1

Inference

Decide which class x belongs to using the scoring function:

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

Code Walkthrough

https://github.com/cmu-l3/anlp-spring2026-code/blob/main/01_intro/rule_based_classifier.ipynb

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

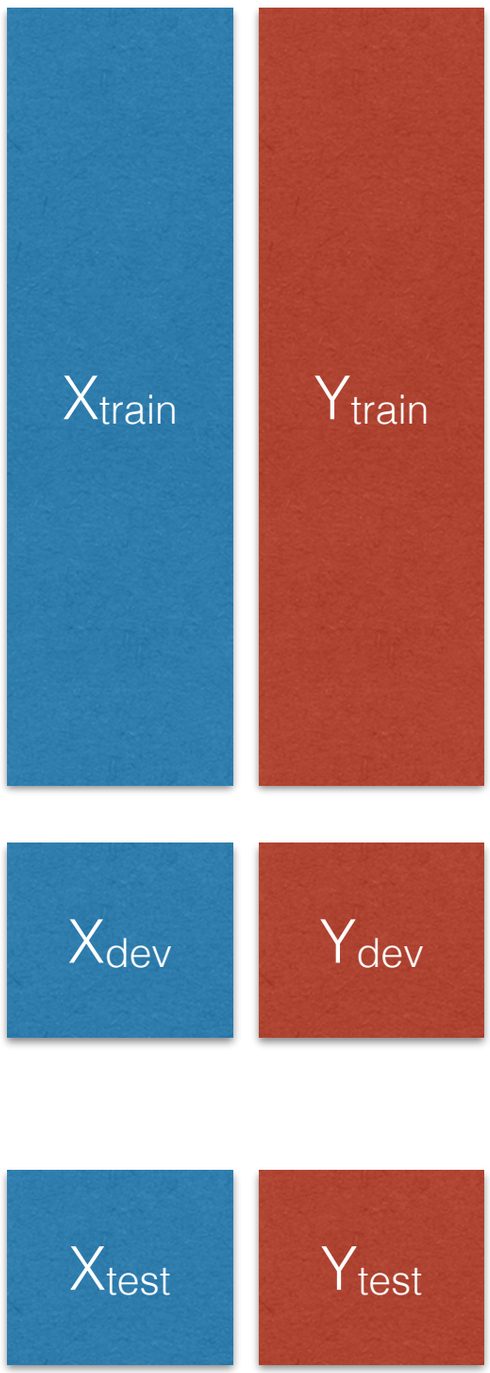
Solution?: Learn Japanese and re-do all the work?

2. Learning in NLP

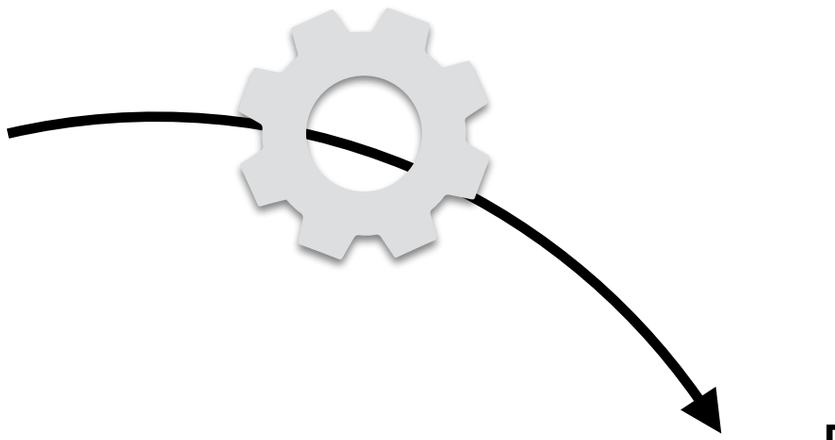
Learning the Scoring Function

Learning the scoring function

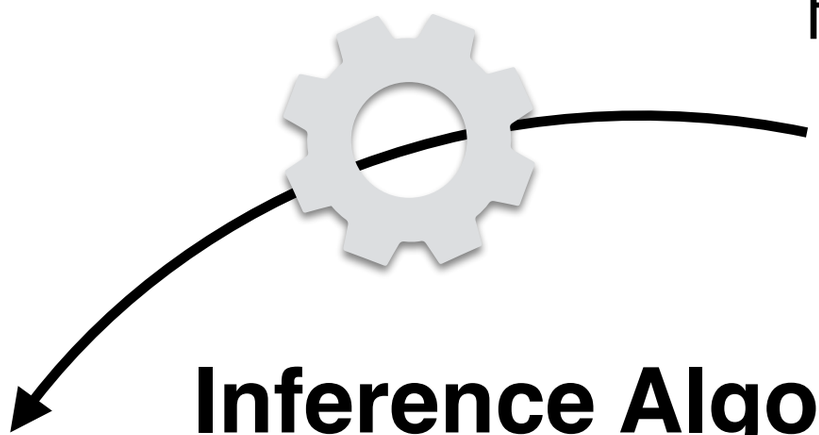
Supervision



Learning Algorithm



Learned
Feature Extractor
Weights



Inference Algorithm

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

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

*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:

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

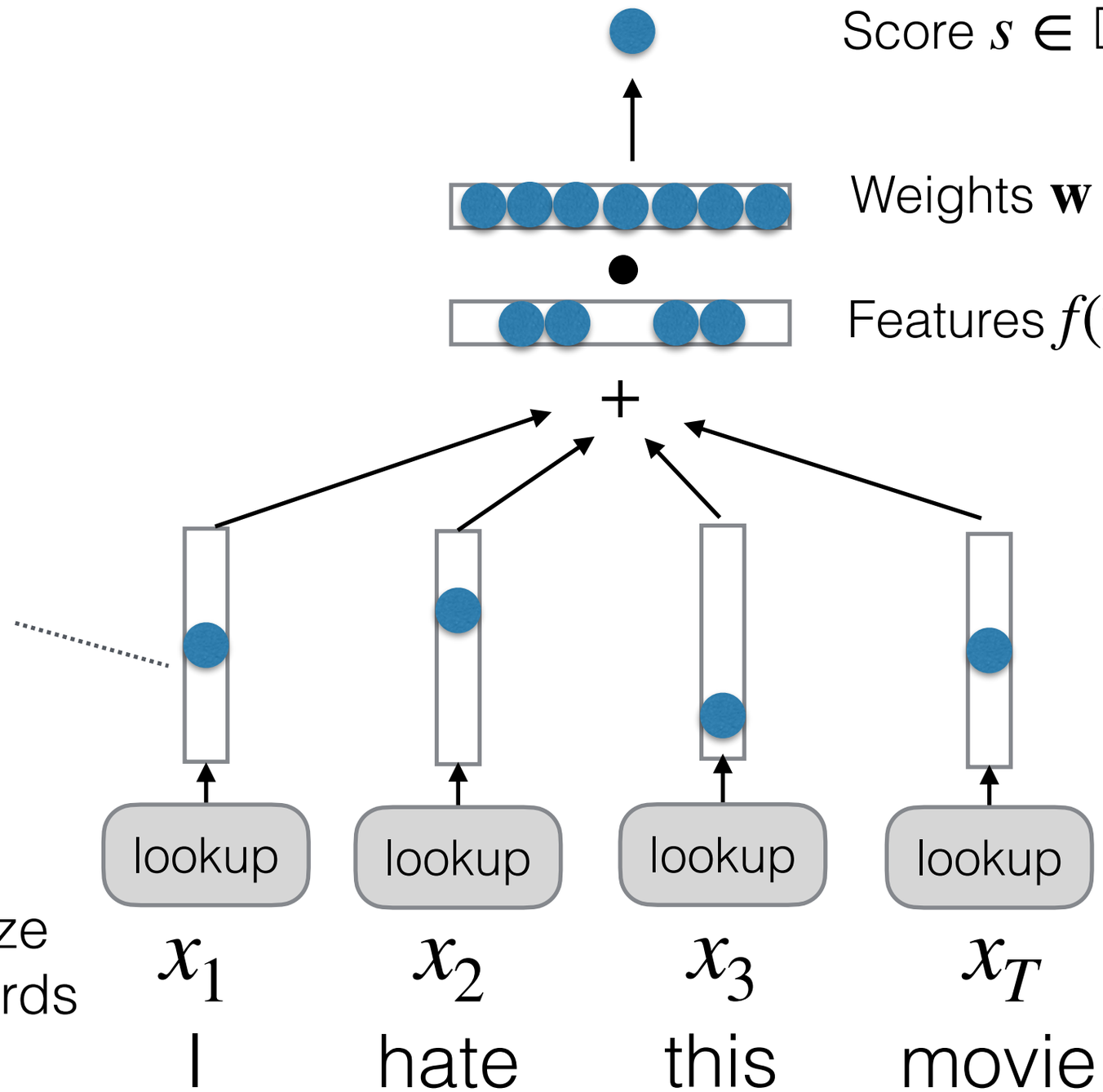
1. **Parameterization**: the form and parameters of the function (e.g., neural net architecture and its weights).
2. **Learning**: how we adjust the parameters using supervision (e.g., using input-output examples, a reward function).
3. **Inference**: how we make decisions after learning.

Example Parameterization: Bag of Words (BoW)

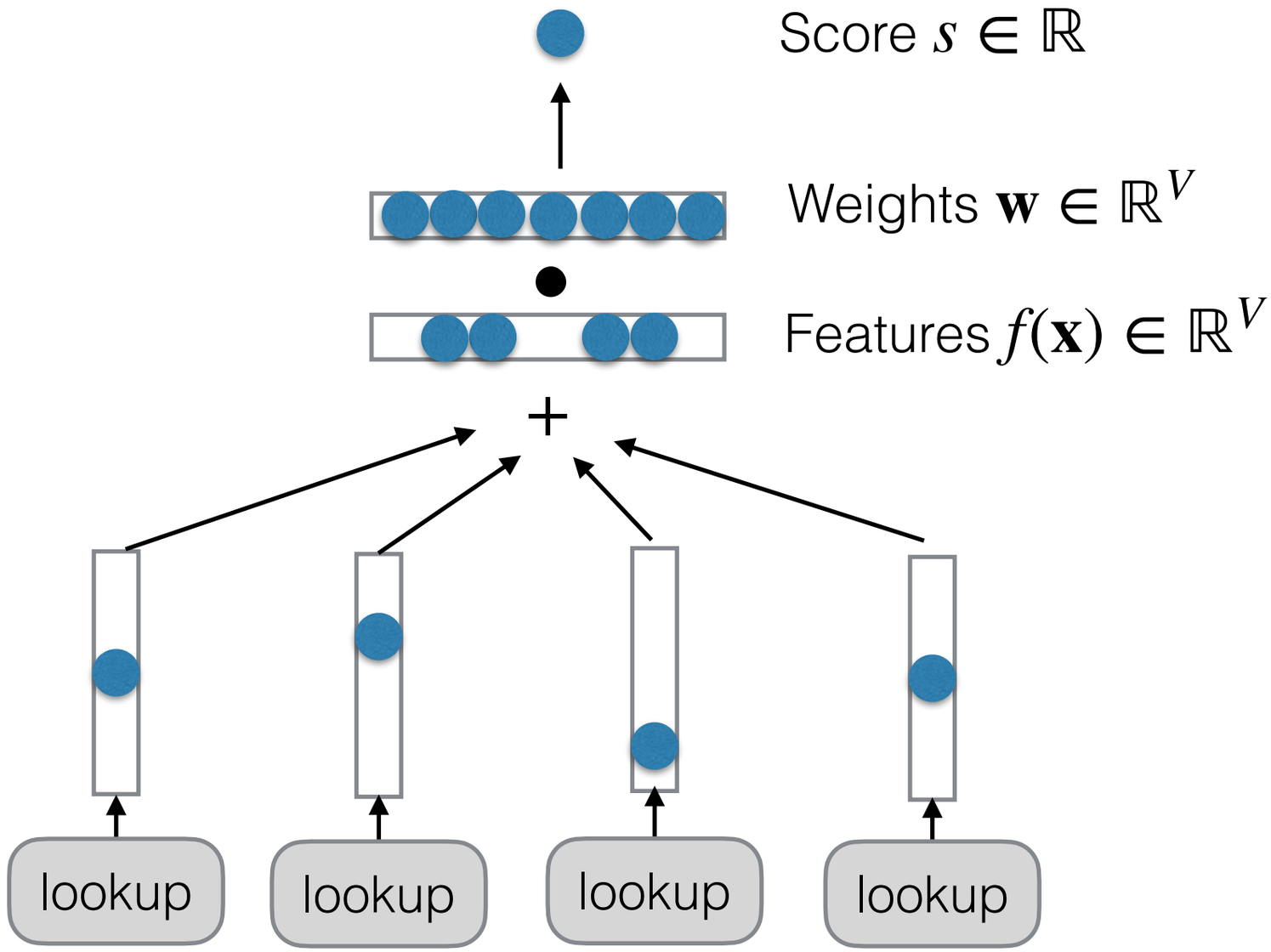
$$\begin{bmatrix} 0 & a \\ 0 & aardvark \\ \vdots & \vdots \\ 1 & I \\ \vdots & \vdots \\ 0 & zebra \end{bmatrix}$$

$$x_t \in \{0,1\}^V$$

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]

Binary

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

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

Multi-class

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

	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:

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

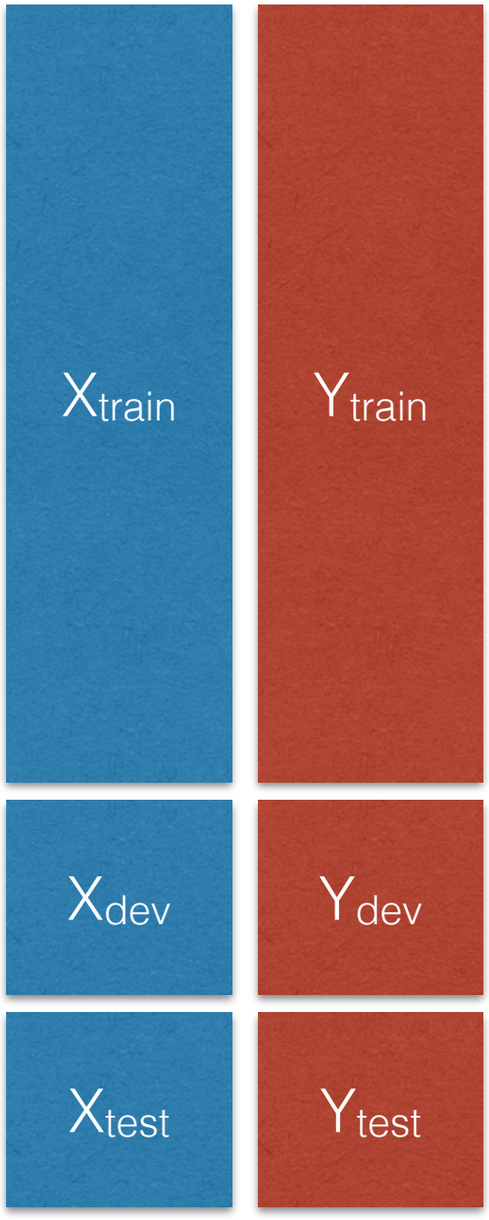
$$= \text{argmax}_{y \in \{-1,1\}} y s_\theta(x)$$

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

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”

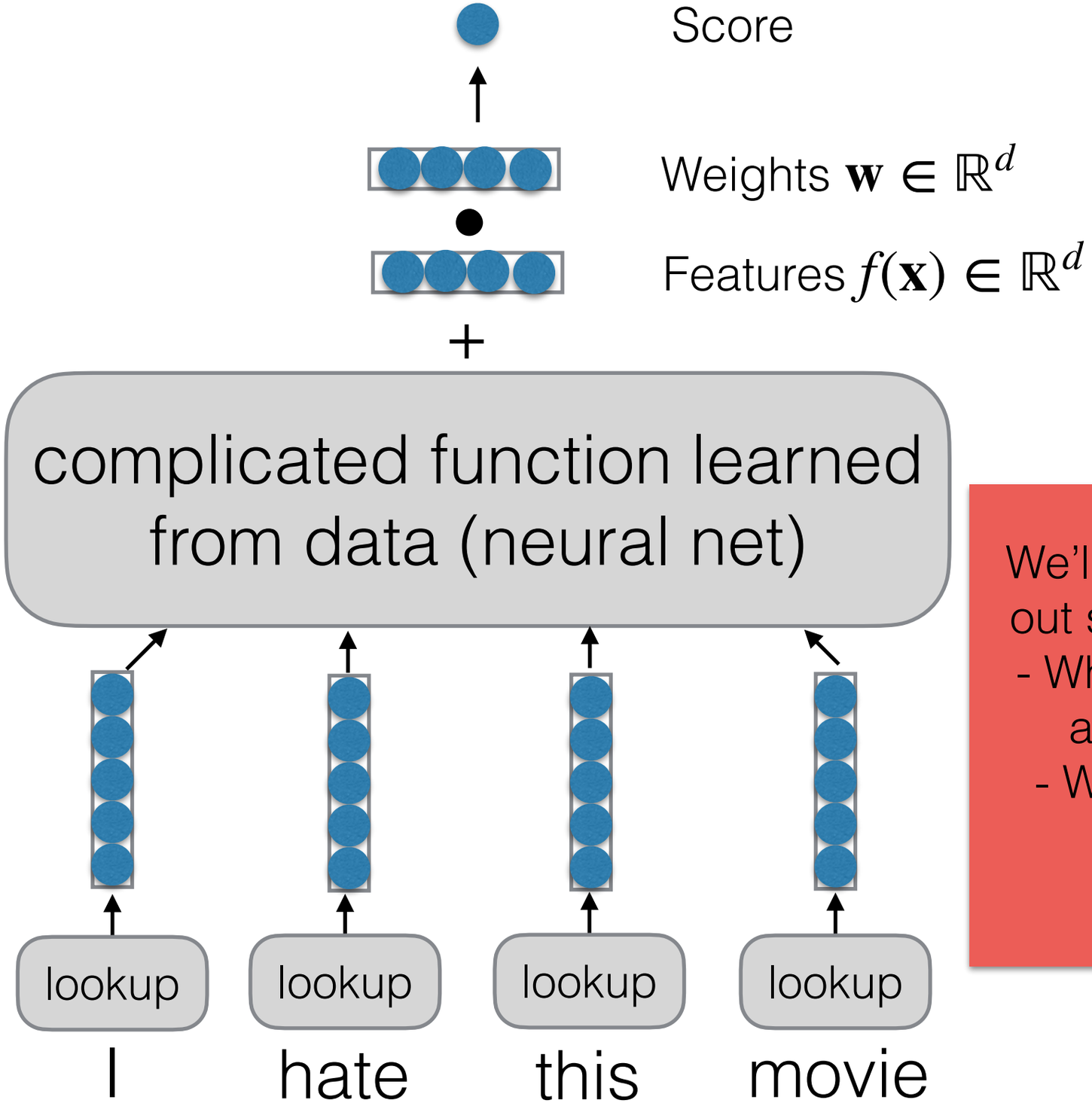
```
feature_weights = {}
for x, y in data:
    # Make a prediction
    features = extract_features(x)
    predicted_y = run_classifier(features)
    # Update the weights if the prediction is wrong
    if predicted_y != y:
        for feature in features:
            feature_weights[feature] = (
    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

https://github.com/cmu-l3/anlp-spring2026-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 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 distributions $p(y \mid x)$
  - 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.):

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

  - `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 modeling, learning, and inference decisions needed to achieve this

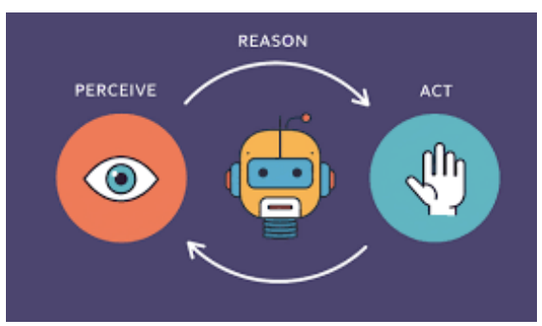
From generation to actions

- 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





3. Semester Roadmap

Topics

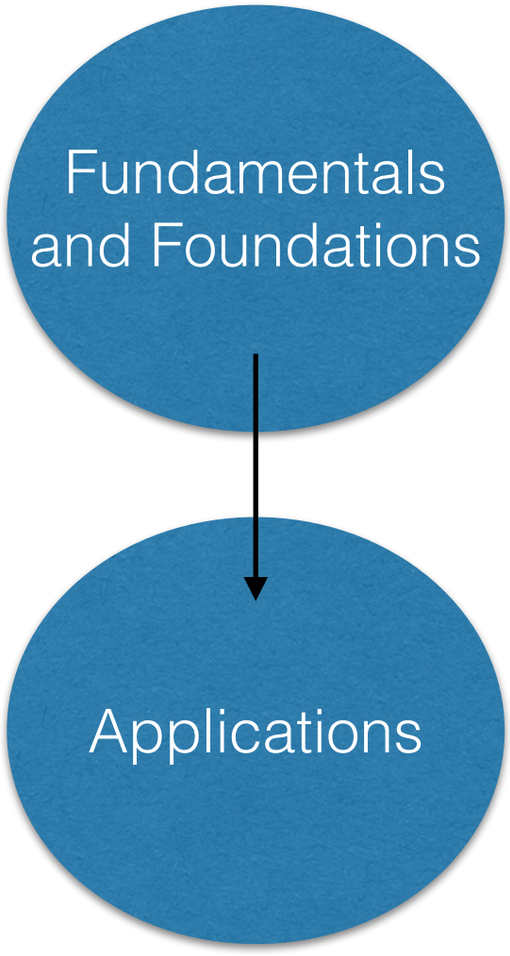
1. **NLP fundamentals**
2. **Methods for today's NLP**
3. **Implementing usable NLP systems**
4. **Evaluating and understanding your models**
5. **Research skills**

https://cmu-anlp.org/schedule/

Comparison to other courses

Advanced NLP introduces you to the fundamental tools and concepts from around NLP. With respect to LLMs:

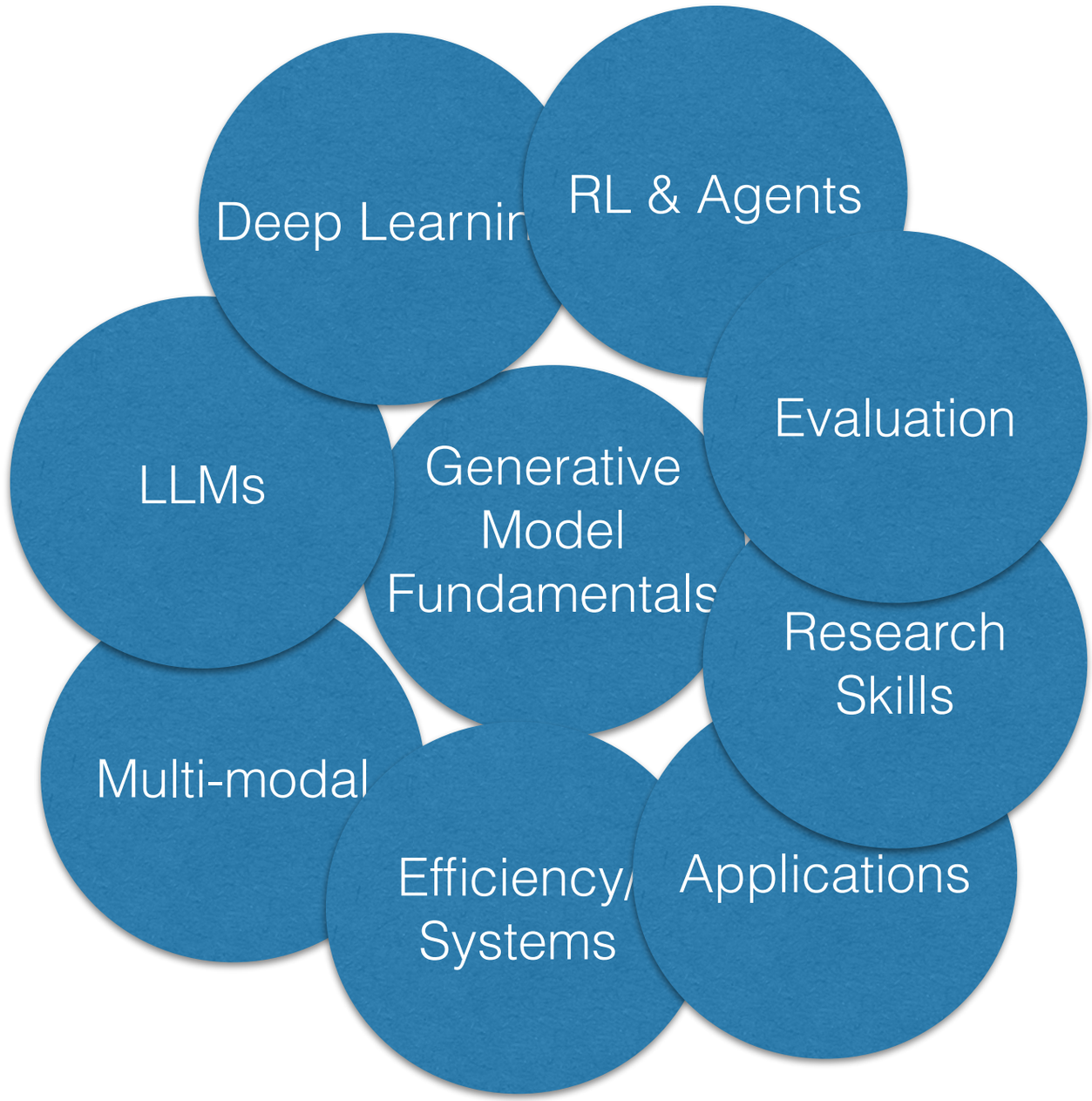
- **This course (11-711)**
  - Foundations and fundamentals of cutting edge NLP (including LLMs)
- **Large Language Model Applications (11-766)**:
  - Applications of LLMs



Comparison to other courses

Advanced NLP introduces you to a variety of fundamental tools and concepts from around cutting edge NLP. To go in further depth:

- **Advanced Deep Learning (10-707)**
  - Focus on fundamental building blocks of deep learning
- **Large Language Model Applications (11-776)**:
  - Focus on applications of LLMs
- **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
- **AI Agents (11-768)**
  - Focus on agents that perceive, reason, plan, and act in complex environments

Survey

We will use sli.do for in-class polling. These will form your participation grade.

When logging into sli.do, make sure to use your **andrewID** email address.

4. Logistics

Instructor Team

- Instructors:
  - Fernando Diaz: OH after class on Tuesdays **by appointment**
  - Daphne Ippolito: OH before class on Thursdays
- TAs:
  - Aashiq Muhamed, Abdul Waheed, André Duarte, Keyu He, Pranjal Aggarwal, Rosa Vitello, Sid Joshi
  - TA office hours will start in week 2.

Class Format

- **Before class:** Do main reading
- **During class:**
  - *Lecture:* We will go through the material together.
    - *Ask questions*: Raise your hands or ask on Sldi
    - *Participation*: Answer polls and quizzes on Slido.
    - *Code/Data Walkthrough:* The instructor will sometimes walk through some demonstration code, data, etc.

Grading

- 10%: Assignment 1
- 10%: Assignment 2
- 10%: Assignment 3
- 20%: Assignment 4 (Final Project)
- Midterm 20%
- Final exam 20%
- Participation 5%
- Peer feedback 5%

⚠️ This class has a midterm and a final exam. ⚠️

Midterm is in-class.

Final exam will be scheduled by the registrar in the exam period.

December 2026

Jump to Month

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
29	30	1	2	3	4 All Day Last Day of Classes All Day Semester & Mini-2 Voucher Deadline	5
6	7 Dec 7 - 8 Final Exams	8 Dec 7 - 8 Final Exams	9 All Day Reading Day	10 Dec 10 - 11 Final Exams	11 Dec 10 - 11 Final Exams	12 All Day Reading Day
13 All Day Final Exams All Day Semester & Mini-2 Faculty Course Evaluations Close	14 All Day Make Up Final Exams	15	16 All Day Final Grades Due at 4 p.m.	17	18	19

Assignments

- **Assignment 1 - Build-your-own LM:** *Individually* implement language model loading and training
- **Assignment 2 - End-to-end NLP System Building:** *Individually* perform data creation, modeling, and evaluation for a specified task
- **Project:** Teams of 2-3 students
  - **Assignment 3 - Survey and re-implementation:** Survey related work, 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.

Late Policy

- 5% penalty per 24 period late.
  - 25 minutes late = 5% penalty
  - 27 hours late = 10% penalty
- You may ask for a *non-medical* excused late day or days at least 168 hours (7 days) before the deadline.
- Valid excuses:
  - personal health issues
  - religious commitments
  - other extenuating life circumstances.
- For *medical* issues, take care of yourself first and email us second.
- **See syllabus for full policy.**

Exams

- Exams will be in-person, written, and closed-book.
- They will be non-cumulative—midterm exam assesses material from the first half of the semester and final exam from the second half.
- Fair game:
  - Questions from the lecture material
  - Questions about in-class discussions
  - Questions about the homeworks

Participation

- You will receive full credit if you participate in at least **80%** of the interactive polls/quizzes/questions on sli.do.
- Make sure to enter your AndrewID email address into sli.do to receive credit.

Peer feedback

- Give written feedback to other groups on their projects.
- These will be graded generously—all reasonable efforts will receive full credit.

Recordings and Attendance

- We will do our best to send out recordings of the lectures.
- ***Attendance***: we expect you to attend courses and participate in discussions/interactive elements during class.
  - We do not allow registering for the course when you have a schedule conflict.
  - You ***absolutely must*** attend:
    - Exams
    - Poster Sessions
- Note: 5% of the course grade is **in-class participation**.

Please check the course webpage for specific dates!

Waitlist

- We have a long waitlist.
- As a curtesy to those on the waitlist, if you think you might drop the class, do so sooner rather than later.
- Mailing us your qualifications does not improve your chances of getting off the waitlist.

Should I take this course?

- I’m certain that you’re excited about the course content! It’s extremely relevant and important content, and you’ll learn a lot.
- **Please be sure that you will be able to satisfy the logistics associated with the attendance (lectures, exams, project presentation) and other aspects of the course.**

Resources

- **Website and syllabus:** https://cmu-anlp.org
- **Code:** https://github.com/cmu-l3/anlp-spring2026-code
- **Piazza**: https://piazza.com/class/mt887u3ed8s1ou/

Thank you