CSCE 689: Special Topics in Trustworthy NLP

Lecture 2: Natural Language Processing Basics (1)

Kuan-Hao Huang
khhuang@tamu.edu

(Some slides adapted from Chris Manning, Dan Jurafsky, Danqi Chen, and Karthik Narasimhan)

Course Information

- Office Hour: Wednesday 1pm – 2pm @ PETR 219
- Email: khhuang@tamu.edu Please use “[CSCE 689] Subject …”
- More Information: https://khhuang.me/CSCE689-F24/

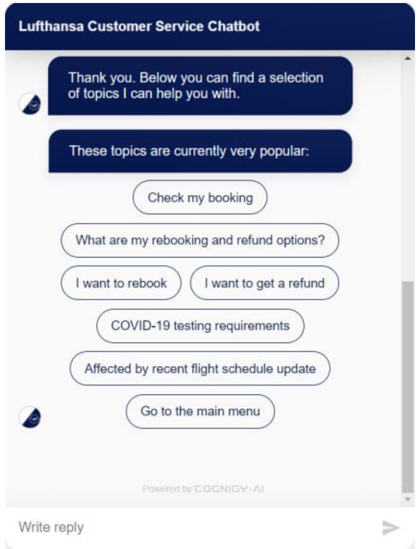
Additional Opportunity for Trustworthy AI Research

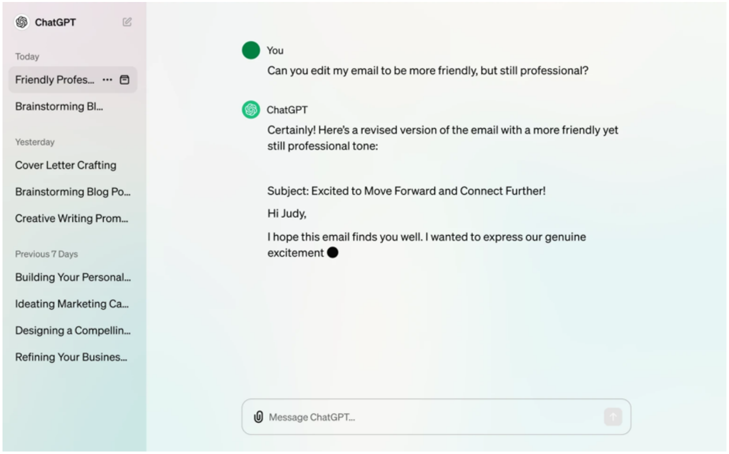
- Amazon Trusted AI Challenge
- https://www.amazon.science/trusted-ai-challenge
- An option for team project
- Timeline
  - Submit a proposal by 9/1/2024
  - Only 10 teams will be selected
  - Notification of selection 9/16/2024
  - Competition from November 2024 to June 2025

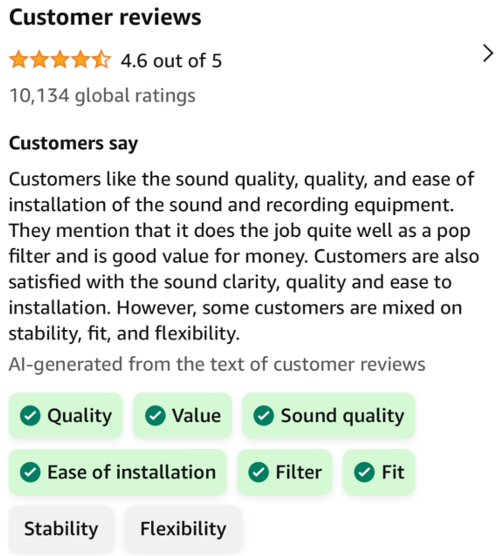
Lecture Plan

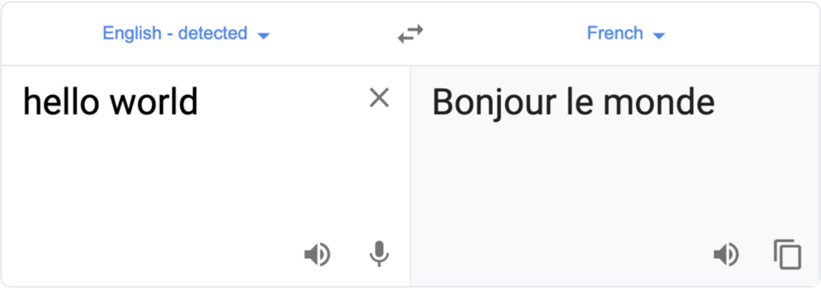
- Natural Language Processing Basics
- Common NLP Tasks
  - Classification
  - Generation
- Training Pipelines
  - Feature Extractor
  - Model Parameters
  - Optimization
- Word Embeddings

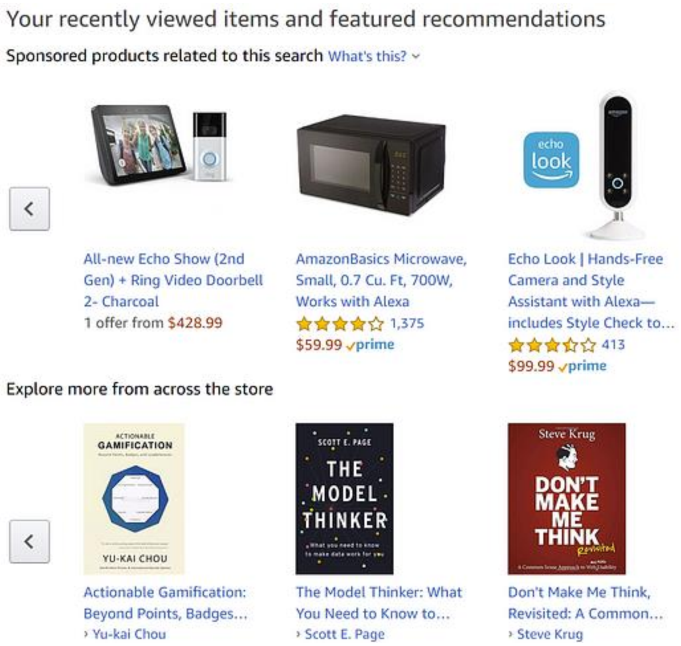
NLP Applications











How to formulate those problems?

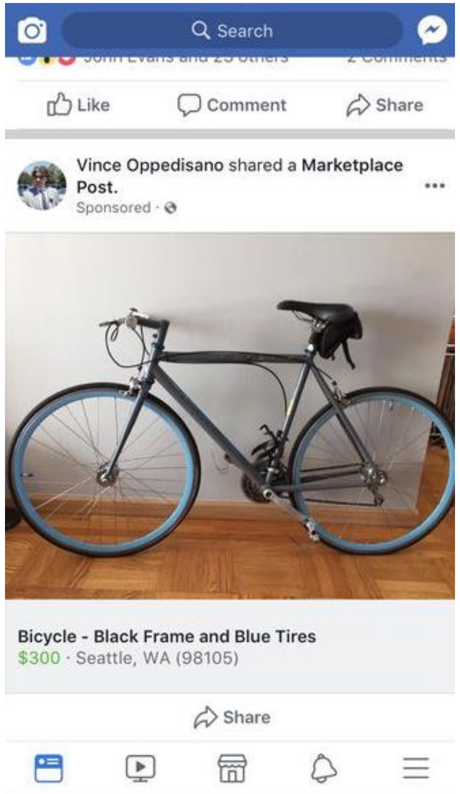
Formulation

- Build an NLP model to learn the association between input x and output y
- Input x: a sequence of symbols
  - What’s the temperature now?
  - I like this restaurant.
- Output y: label
  - Category
  - Structure
  - Text
  - …

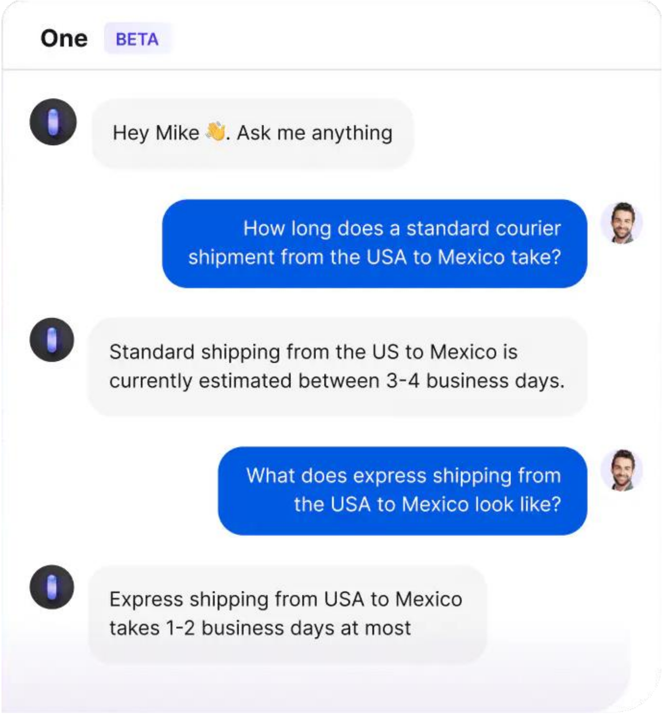
Text Classification

- Input x → Output y (category)

$y \in \{pos, neg\}$



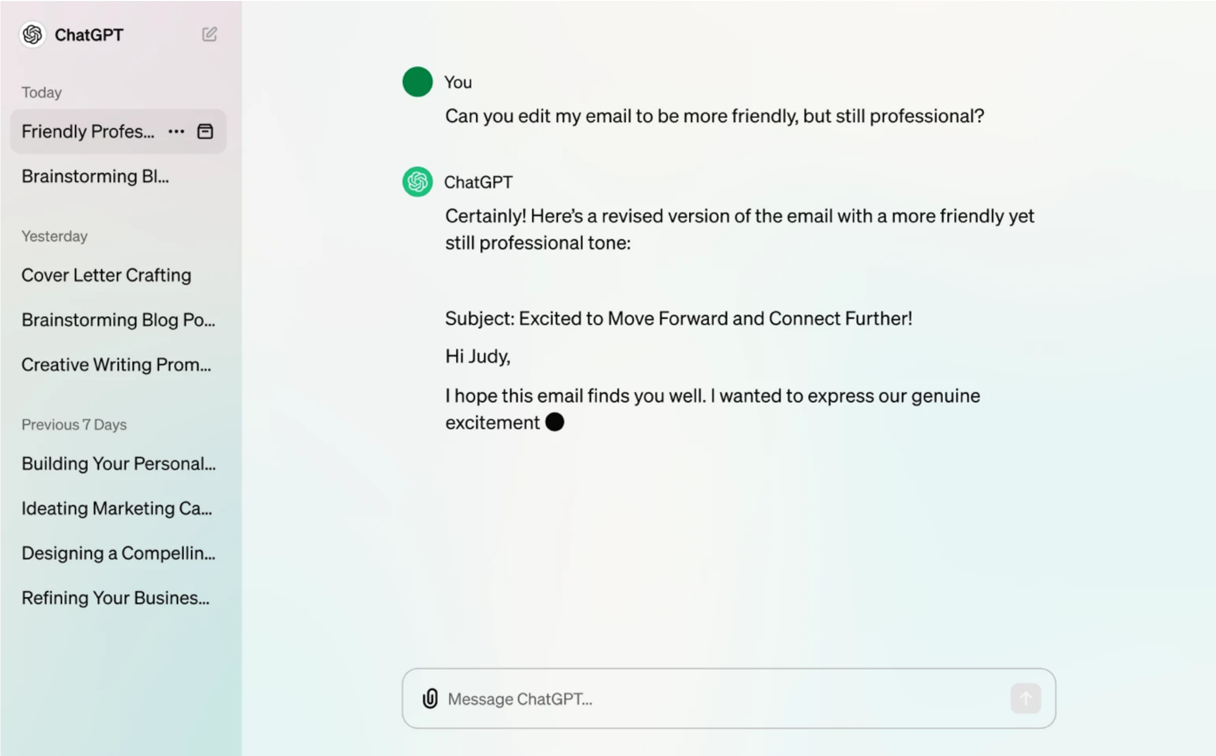
$y \in \{normal, fraud\}$



$y \in \{engineer, business, marketing, IT\ service\}$

Text Classification

- Input x → Output y (category)



$$y \in \{normal, math, code, \dots\}$$

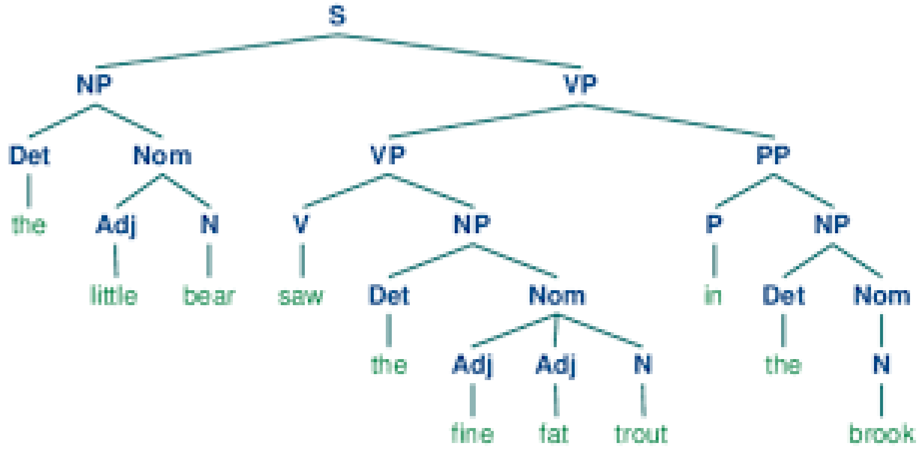
$$y \in \{using\ tool, not\ using\ tool\}$$

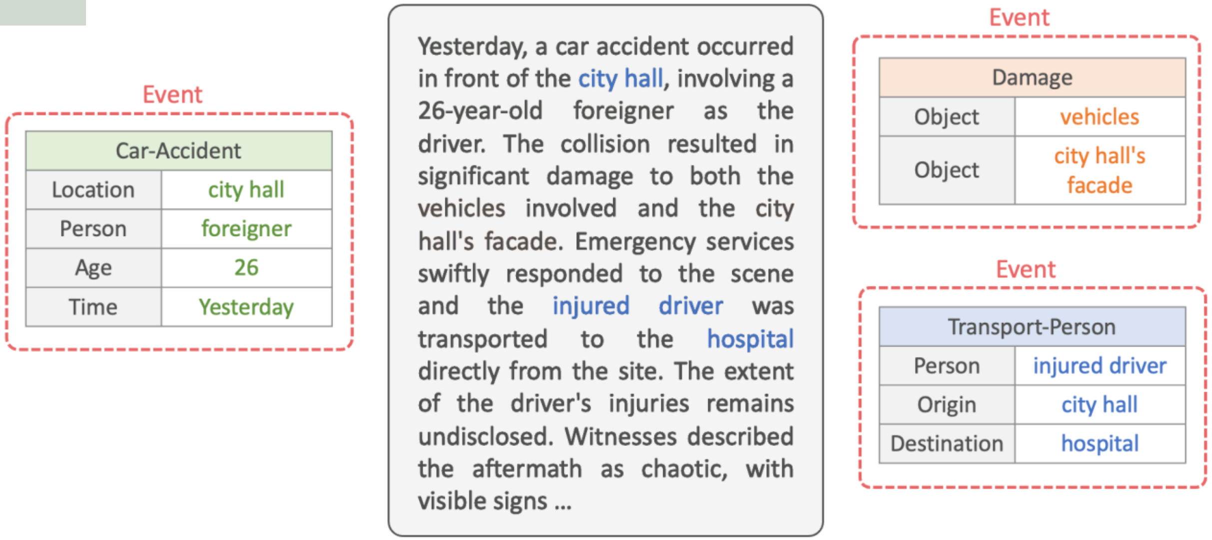
$$y \in \{ethical\ issue, noethical\ issue\}$$

…

Structured Classification

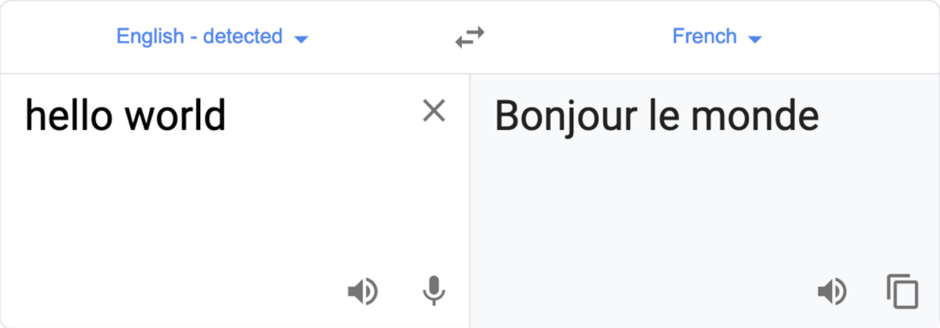
- Input x → Output y (structure)
  - Multiple labels with dependency

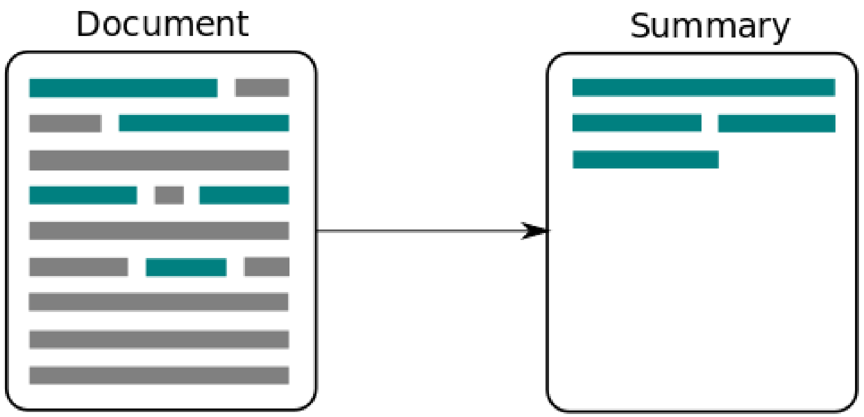




Generation

- Input x → Output y (text)
  - Also called sequence-to-sequence tasks





The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

Classification vs. Generation

- There is no clear boundary between classification and generation
- Generation = Structured Token Classification

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through

$y \in \{all\ possible\ words\}$ $\quad$ $y \in \{all\ possible\ words\}$

Classification vs. Generation

- There is no clear boundary between classification and generation
- Classification problems can be solved by generation

What's the sentiment of the following text: I very like this restaurant.

⬇

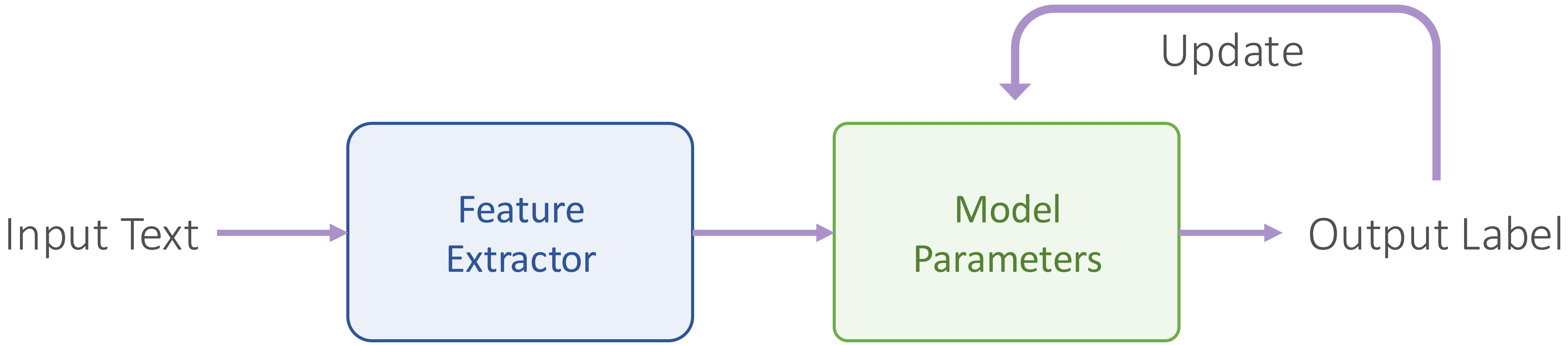
The sentiment is positive.

Lecture Plan

- Natural Language Processing Basics
- Common NLP Tasks
  - Classification
  - Generation
- Training Pipelines
  - Feature Extractor
  - Model Parameters
  - Optimization
- Word Embeddings

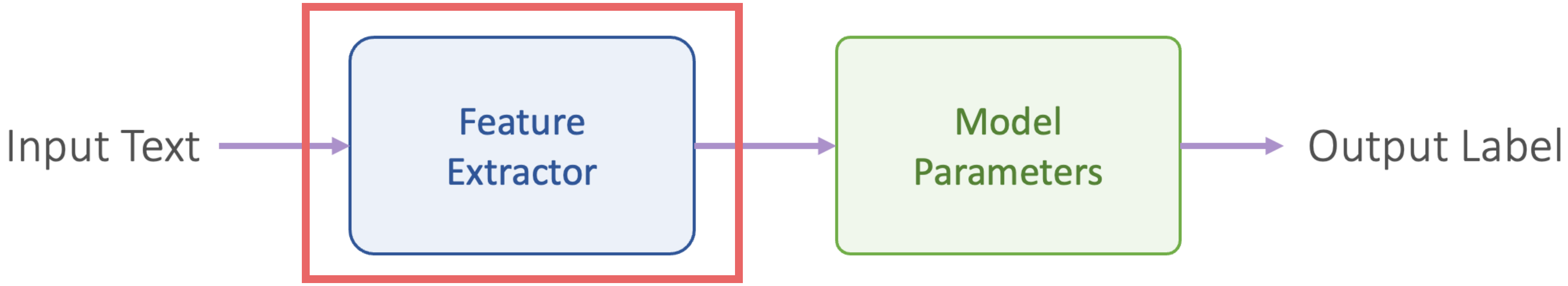
How to Learn an NLP model?

- Machine learning method: supervised learning
  - Training examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
  - Learn model $F: \mathcal{X} \rightarrow \mathcal{Y}$



Let's start with a simple solution and gradually improve it!

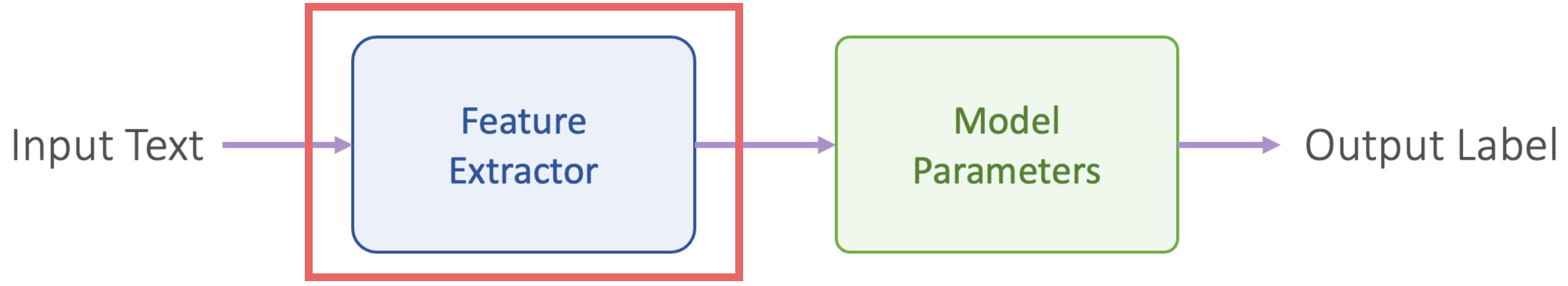
Feature Extractor



- Convert a text to a meaningful vector that captures essential characteristics of the text
  - Traditional method: human-crafted values
  - Cutting-edge method: word embeddings (we will talk about it later!)

Input Text → A Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$

Feature Extractor

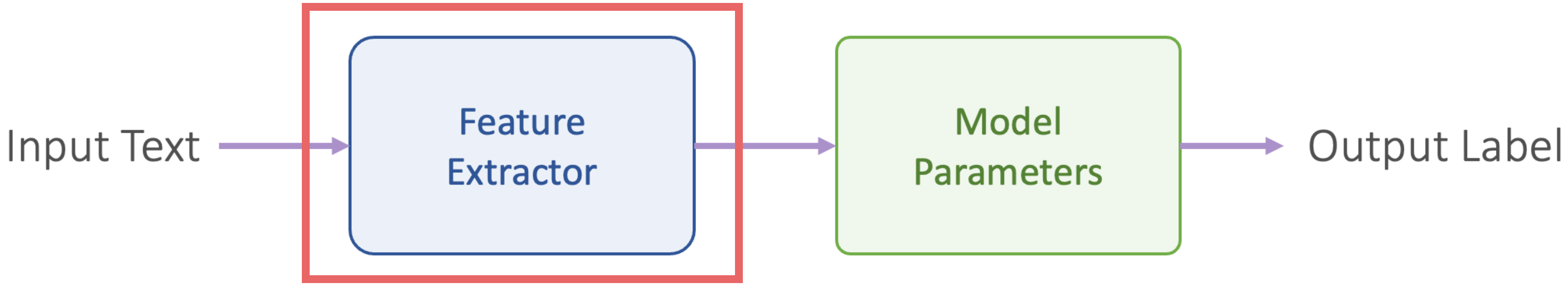


Bag of words (BoW)

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

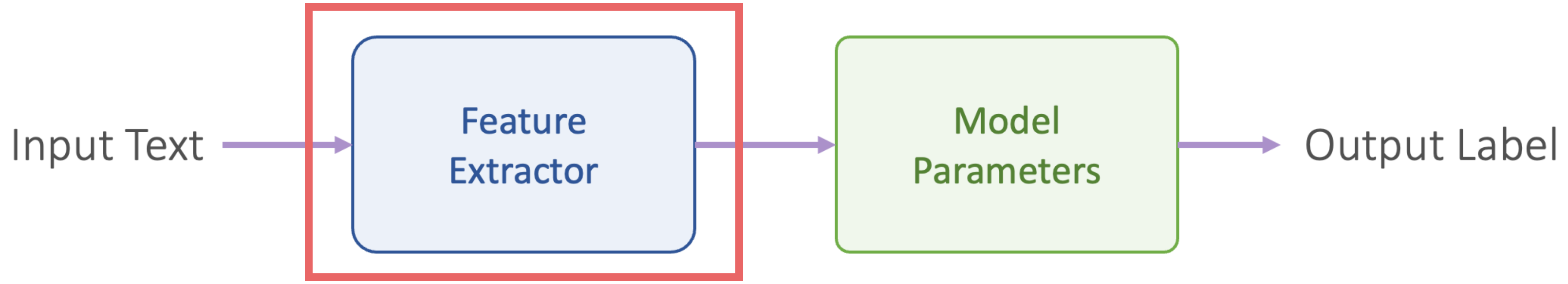
Feature Extractor



Rank	Category	Feature	Rank	Category	Feature
1	Subject	Number of capitalized words	1	Subject	Min of the compression ratio for the bz2 compressor
2	Subject	Sum of all the character lengths of words	2	Subject	Min of the compression ratio for the zlib compressor
3	Subject	Number of words containing letters and numbers	3	Subject	Min of character diversity of each word
4	Subject	Max of ratio of digit characters to all characters of each word	4	Subject	Min of the compression ratio for the lzw compressor
5	Header	Hour of day when email was sent	5	Subject	Max of the character lengths of words
		(a)			(b)
		Spam URLs Features			
1	URL	The number of all URLs in an email	1	Header	Day of week when email was sent
2	URL	The number of unique URLs in an email	2	Payload	Number of characters
3	Payload	Number of words containing letters and numbers	3	Payload	Sum of all the character lengths of words
4	Payload	Min of the compression ratio for the bz2 compressor	4	Header	Minute of hour when email was sent
5	Payload	Number of words containing only letters	5	Header	Hour of day when email was sent

Var	Definition
x_1	$\text{count}(\text{positive lexicon}) \in \text{doc})$
x_2	$\text{count}(\text{negative lexicon}) \in \text{doc})$
x_3	$\begin{cases} 1 & \text{if ``no''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	$\text{count}(\text{1st and 2nd pronouns} \in \text{doc})$
x_5	$\begin{cases} 1 & \text{if ``!''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	$\log(\text{word count of doc})$

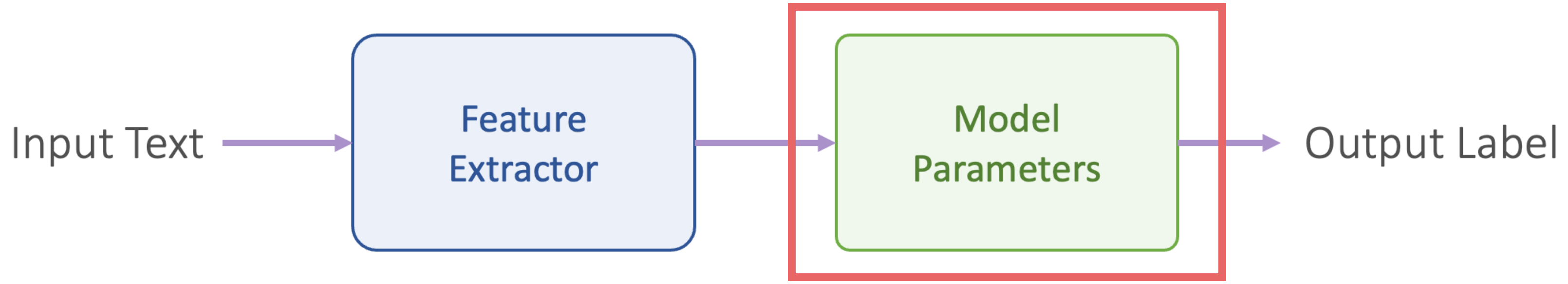
Feature Extractor



- Convert a text to a meaningful vector that captures essential characteristics of the text
  - Traditional method: human-crafted values
  - Cutting-edge method: word embeddings (we will talk about it later!)

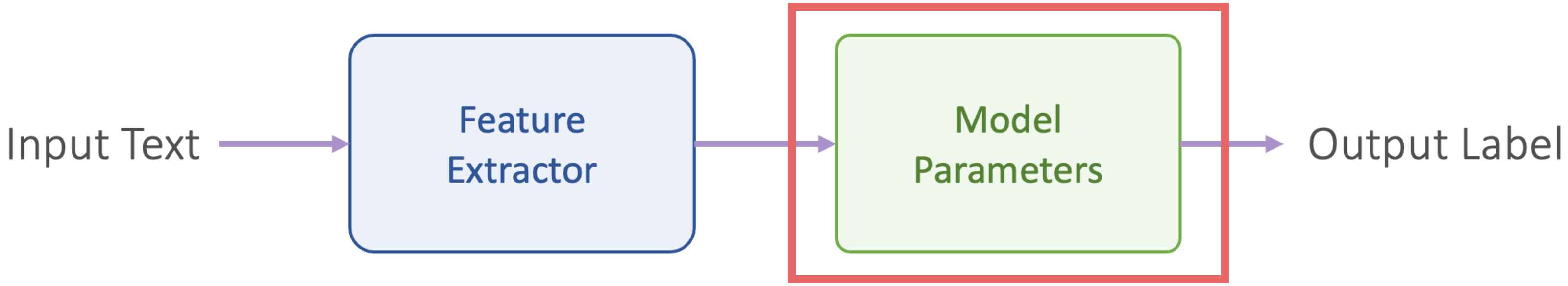
Input Text → A Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n]$

Model Parameters



- Convert a feature vector $\mathbf{x}$ to an output label
  - Traditional methods: Naive Bayes, Logistic Regression
  - Deep learning methods: CNN, RNN, LSTM, Transformers (we will talk about them later!)

Model Parameters - Logistic Regression



- Logistic Regression for binary classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n]$ Label $y = 0\ or\ 1$

Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \ldots, w_n]$ Bias b

Learnable Model Parameters

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$P(y = 1 | \mathbf{x}) = \sigma(z)$$

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Sigmoid Function

Model Parameters - Logistic Regression

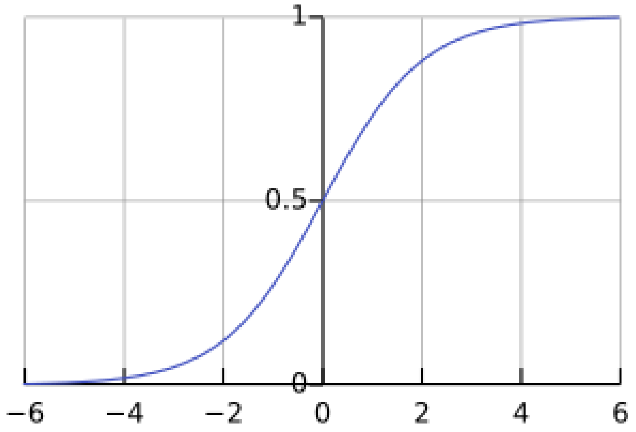
Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \ldots, w_n]$

Bias b

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

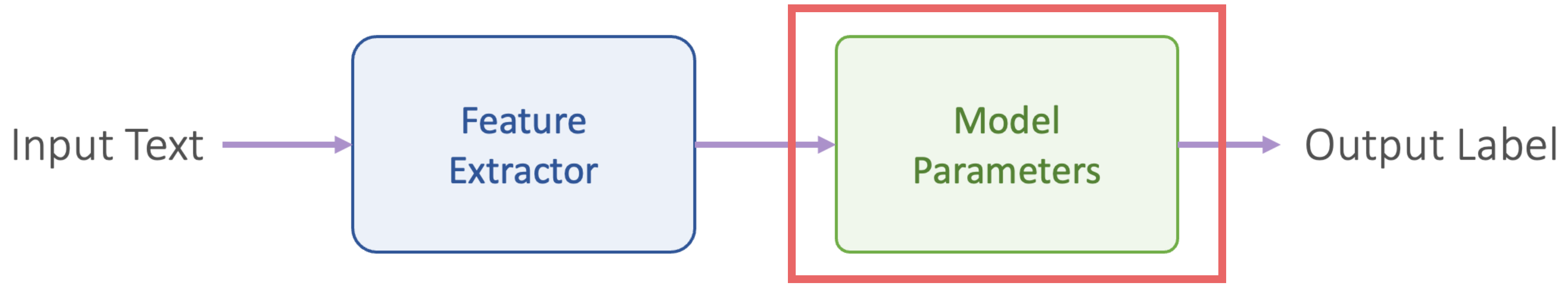
Sigmoid Function



$$\tilde{y} = P(y = 1 | \mathbf{x}) = \sigma(z) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}$$

$$\text{Output Label} = \begin{cases} 1 & \text{If } \tilde{y} > 0.5 \\ 0 & \text{If } \tilde{y} \leq 0.5 \end{cases}$$

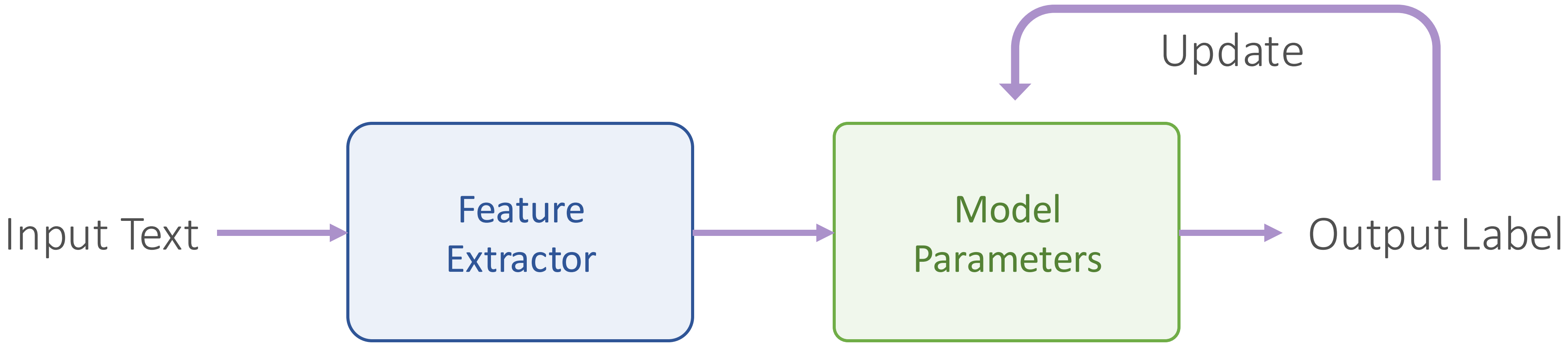
Model Parameters - Logistic Regression



- Convert a feature vector $\mathbf{x}$ to an output label
  - Traditional methods: Naive Bayes, Logistic Regression
  - Deep learning methods: CNN, RNN, LSTM, Transformers (we will talk about them later!)

How to Learn an NLP model?

- Machine learning method: supervised learning
  - Training examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
  - Learn model $F: \mathcal{X} \rightarrow \mathcal{Y}$



How to learn the model parameters?

Loss Function

- We need an indicator to know how well the output label is
- One training example (x, y)
- Output label is decided by $\tilde{y} = P(y = 1 |\, \mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$

Cross Entropy Loss

$$\mathcal{L}_{single} = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

$y = 1$ and $\tilde{y} = 0.9$ $\quad \mathcal{L}_{single} = -[1 \cdot \log 0.9 + 0 \cdot \log(0.1)] = -\log 0.9 \approx 0.105$

$y = 1$ and $\tilde{y} = 0.1$ $\quad \mathcal{L}_{single} = -[1 \cdot \log 0.1 + 0 \cdot \log(0.9)] = -\log 0.1 \approx 2.302$

$y = 0$ and $\tilde{y} = 0.9$ $\quad \mathcal{L}_{single} = -[0 \cdot \log 0.9 + 1 \cdot \log(0.1)] = -\log 0.1 \approx 2.302$

$y = 0$ and $\tilde{y} = 0.1$ $\quad \mathcal{L}_{single} = -[0 \cdot \log 0.1 + 1 \cdot \log(0.9)] = -\log 0.9 \approx 0.105$

The lower the loss is, the more accurate the output label is

Loss Function

- Training examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Output labels is decided by $\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_m$

Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_i [y_i \log \tilde{y}_i + (1 - y_i)\log(1 - \tilde{y}_i)]$$

Find model parameters such that the loss is minimized!

$$\theta = [\mathbf{w}; b]$$

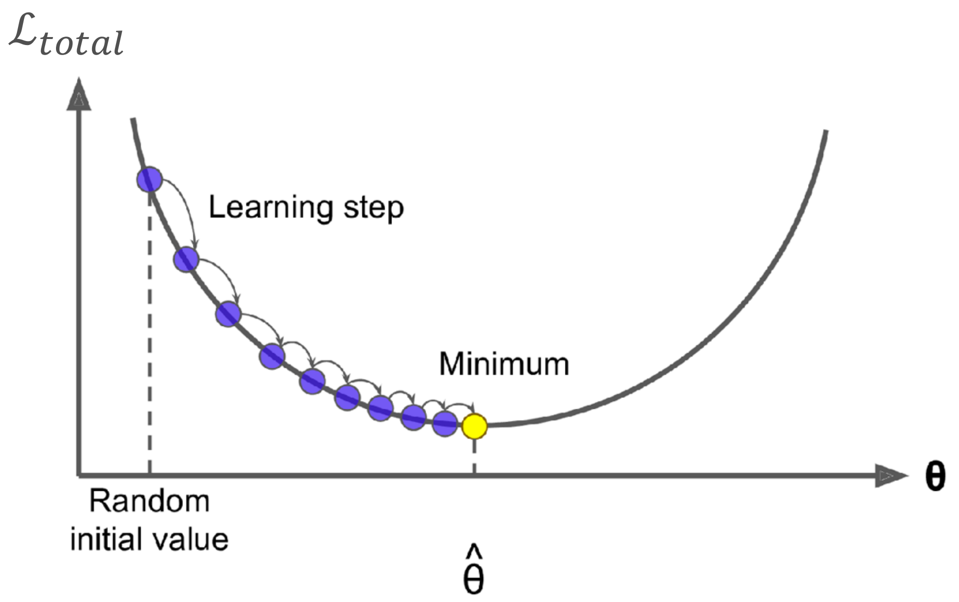
$$\hat{\theta} = \arg\min_\theta \mathcal{L}_{total}$$

Stochastic Gradient Descent

- Randomly initialize parameters $\theta = [\mathbf{w}; b]$
- Iteratively do the following
  - Compute $\mathcal{L}_{total}$
  - Update parameters $\theta \leftarrow \theta - \eta \, \nabla_\theta \mathcal{L}_{total}$

Learning step

Gradient

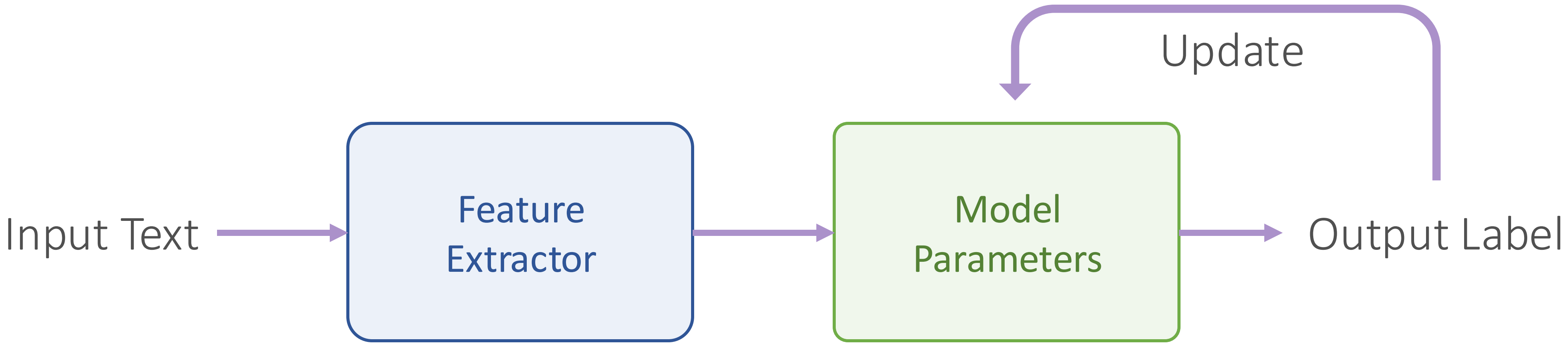


$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)\mathbf{x}_i$$

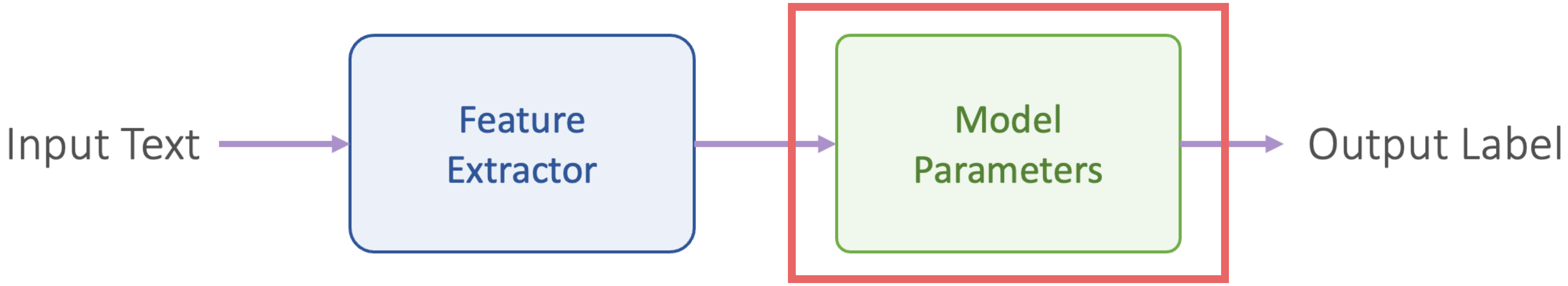
$$\frac{\partial \mathcal{L}_{total}}{\partial b} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)$$

How to Learn an NLP model?

- Machine learning method: supervised learning
  - Training examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
  - Learn model $F: \mathcal{X} \rightarrow \mathcal{Y}$



From Binary Classification to Multiclass Classification



- Logistic Regression for binary classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$ Label $y = 0\ or\ 1$

Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \dots, w_n]$ Bias b

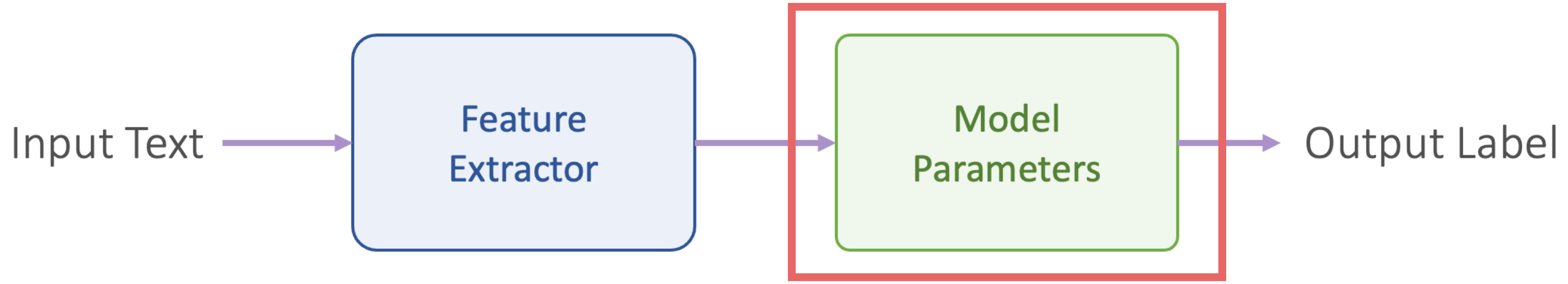
Learnable Model Parameters

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$P(y = 1 | \mathbf{x}) = \sigma(z) \qquad \sigma(t) = \frac{1}{1 + e^{-t}}$$

Sigmoid Function

From Binary Classification to Multiclass Classification



- Logistic Regression for multiclass classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]$ Label $y = 0,1, \dots, C-1$

Weight Vectors $\mathbf{w}_c = [w_{c,1}, w_{c,2}, w_{c,3}, \dots, w_{c,n}]$ Bias b_c

Learnable Model Parameters

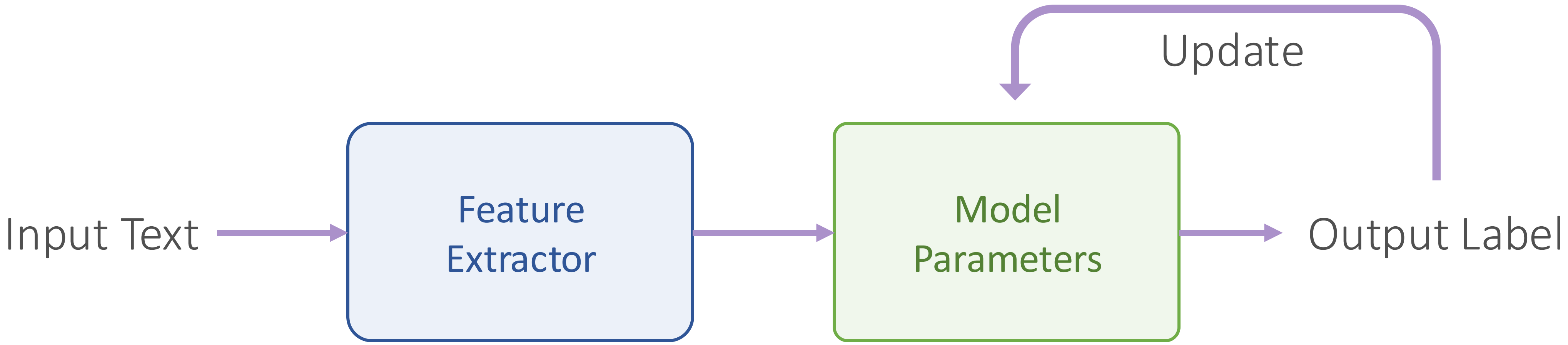
$$z_c = \mathbf{w_c} \cdot \mathbf{x} + b_c$$

$$P(y = c | \mathbf{x}) = \mathrm{softmax}(z_c) \qquad \mathrm{softmax}(t) = \frac{e^{z_c}}{\sum_c e^{z_c}}$$

Softmax Function

How to Learn an NLP model?

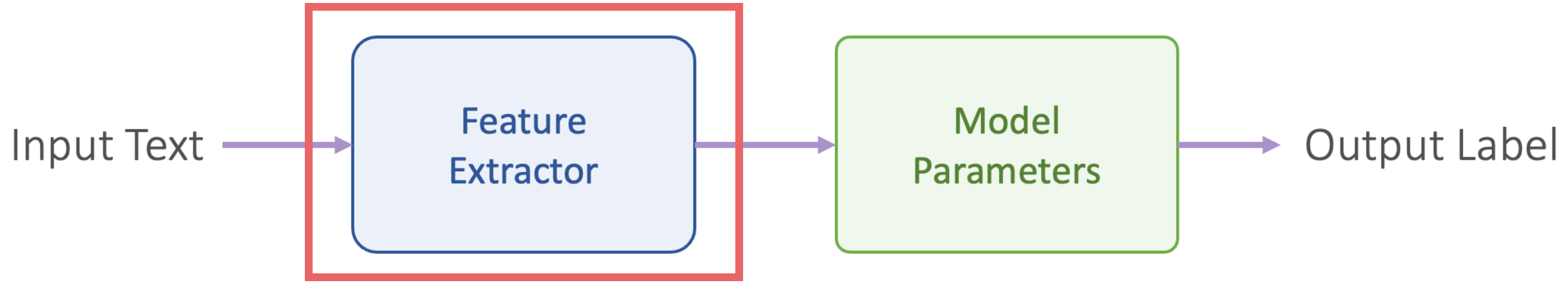
- Machine learning method: supervised learning
  - Training examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
  - Learn model $F: \mathcal{X} \rightarrow \mathcal{Y}$



Lecture Plan

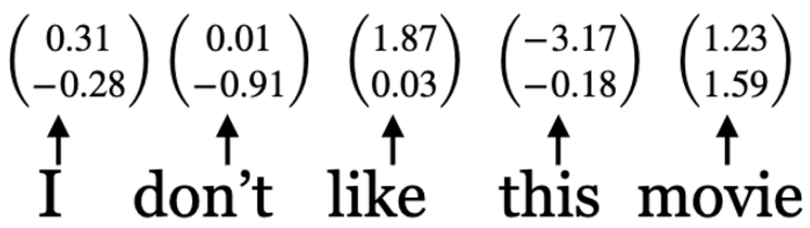
- Natural Language Processing Basics
- Common NLP Tasks
    - Classification
    - Generation
- Training Pipelines
    - Feature Extractor
    - Model Parameters
    - Optimization
- Word Embeddings

Feature Extractor



- Convert a text to meaningful vectors that captures essential characteristics of the text
  - Traditional method: human-crafted values
  - Cutting-edge method: word embeddings (we are talk about it now)

Input Text → A Sequence of Word Vectors



How to Represent Words?

In traditional NLP, we regard words as discrete symbols:

good, great, bad — a localist representation

One 1, the rest 0s

Words can be represented by one-hot vectors:

good = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]

great = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]

bad = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]

good bad great

Vector dimension = number of words in vocabulary (e.g., 500,000+)

Any disadvantages?

Problem with Words as Discrete Symbols

Example: in web search, if a user searches for "good restaurant", we would like to match documents containing "great restaurant"

But

good = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
great = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]

These two vectors are orthogonal

There is no way to encode similarity of words in these vectors!

Any solutions?

Previous Solution: Synonyms, Antonyms, and Hypernyms

Consider external resources like WordNet, a thesaurus containing lists of Synonyms, antonyms, and hypernyms

```
from nltk.corpus import wordnet as wn
poses = { 'n’ : 'noun', 'v’ : 'verb', 's’ : 'adj (s)', 'a’ : 'adj', 'r’ : 'adv'}
for synset in wn.synsets(“bad”):
    print("{}: {}".format(poses[synset.pos()],
             ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: bad, badness
adj: bad
adj (s): bad, big
adj (s): bad, tough
adj (s): bad, spoiled, spoilt
adj: regretful, sorry, bad
adj (s): bad, uncollectible
…
adj (s): bad, risky, high-risk, speculative
adj (s): bad, unfit, unsound
adj (s): bad, forged
adj (s): bad, defective
adv: badly, bad
```

Previous Solution: Synonyms, Antonyms, and Hypernyms

Consider external resources like WordNet, a thesaurus containing lists of Synonyms, antonyms, and hypernyms

$$
\begin{array}{ll cccccccccccccccc}
 & & & \text{welfare} & & & & & & & & & \text{sorry} & & & & & \\
\text{good} & = & [0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0] \\
\text{great} & = & [0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0] \\
\text{bad} & = & [0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0] \\
 & & & & & \text{good} & & & & \text{bad} & & & & & & \text{great} & &
\end{array}
$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||}$$

Similarity(good, great) > Similarity(good, bad)

Any disadvantages?

Problems with Resources Like WordNet

- A useful resource but missing nuance
  - e.g., "sorry" is listed as a synonym for "bad"
  - This is only correct in some contexts
- Subjective
- Missing new meanings of words
  - COVID-19, Doodle, etc.
  - Difficult to keep up-to-date
- Requires human labor to create and adapt

Representing Words by Their Contexts

Distributional hypothesis: words that occur in similar contexts tend to have similar meanings

J.R.Firth 1957

- "You shall know a word by the company it keeps"
- One of the most successful ideas of modern statistical NLP!

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent banking

Distributional Hypothesis

C1: A bottle of ___ is on the table.

C2: Everybody likes ___.

C3: Don’t have ___ before you drive.

C4: I bought ___ yesterday.

	C1	C2	C3	C4
juice	1	1	0	1
loud	0	0	0	0
motor-oil	1	0	0	1
chips	0	1	0	1
choices	0	1	0	0
wine	1	1	1	1

Words that occur in similar contexts tend to have similar meanings

Words as Vectors

- A model to represent words focusing on similarity
  - Each word is a vector
  - Similar words are "nearby in space"
- A first solution: we can just use context vectors to represent the meaning of words!
  - Collect a bunch of texts (corpora)
  - Compute word-word co-occurrence matrix

Word Vector

High cosine similarity!

	shark	computer	data	eat	result	sugar
apple	0	0	0	8	0	2
bread	0	0	0	9	0	1
digital	0	6	5	0	2	0
information	0	4	10	0	2	0

Words as Vectors

Problem: using raw frequency counts is not always very good…

- Solution: let's weight the counts!
- PPMI = Positive Pointwise Mutual Information

$$\mathrm{PPMI}(w,c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0\right)$$

	data	eat	result	sugar
apple	7	807	1	124
bread	2	991	0	233
digital	5648	17	2677	0
information	10230	52	2038	10

	data	eat	result	sugar
apple	0	2.47	0	3.30
bread	0	1.79	0	5.51
digital	0.17	0	0.29	0
information	0.09	0	0.25	0

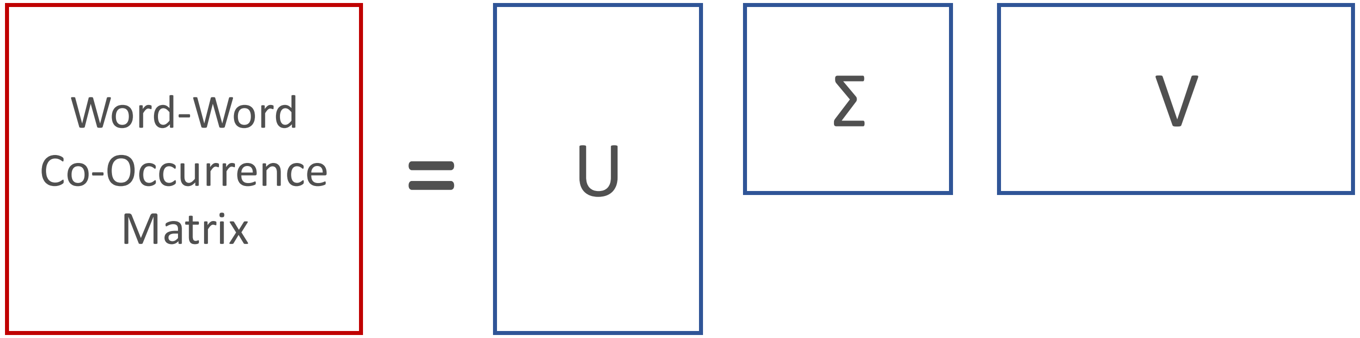
Sparse Vectors vs. Dense Vectors

- Still, the vectors we get from word-word co-occurrence matrix are sparse (most are 0's) and long (vocabulary size)
- Alternative: we want to represent words as short (50-300 dimensional) and dense (real-valued) vectors
  - The focus of this lecture
  - The basis of all the modern NLP systems

$$v_{apple} = \begin{pmatrix} -0.224 \\ 0.479 \\ 0.871 \\ -0.231 \\ 0.101 \end{pmatrix} \quad v_{digital} = \begin{pmatrix} 0.257 \\ 0.587 \\ -0.972 \\ -0.456 \\ -0.002 \end{pmatrix}$$

Why Dense Vectors?

- Short vectors are easier to use as features in ML systems
- Dense vectors may generalize better than storing explicit counts
- Different methods for getting dense vectors
  - Matrix decomposition from word-word co-occurrence matrix
  - Word2Vec and its variant: “learn” the vectors!



Next Lecture

- Natural Language Processing Basics
- Word Embeddings
  - Word2Vec
- Tokenization