CS 577: NATURAL LANGUAGE PROCESSING

Abulhair Saparov

Lecture 13: Reinforcement Learning

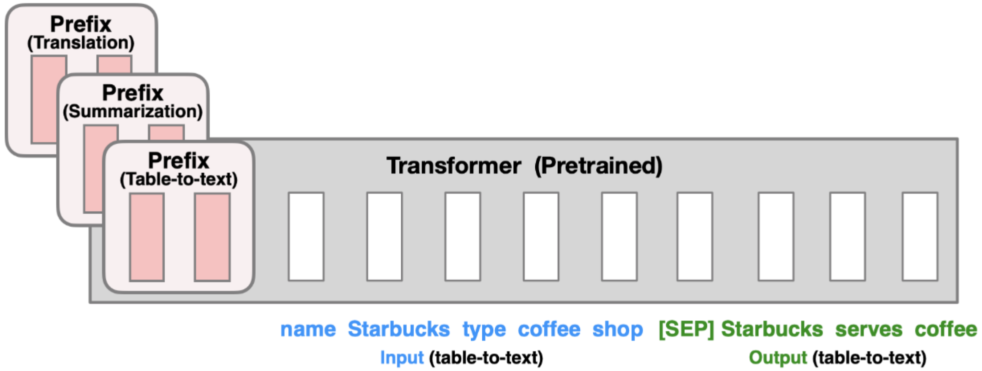
[some slide content from: Graham Neubig, 11-711 Lecture Slides, 2024]

WRAPPING UP PEFT

- At the end of last lecture, we discussed parameter-efficient fine-tuning methods (PEFT).
  - In PEFT, generally, we freeze most of the parameters of the model and only compute gradients for a small set of parameters during fine-tuning.
  - E.g., LoRA
- There are many other PEFT methods, and research into new methods is ongoing.

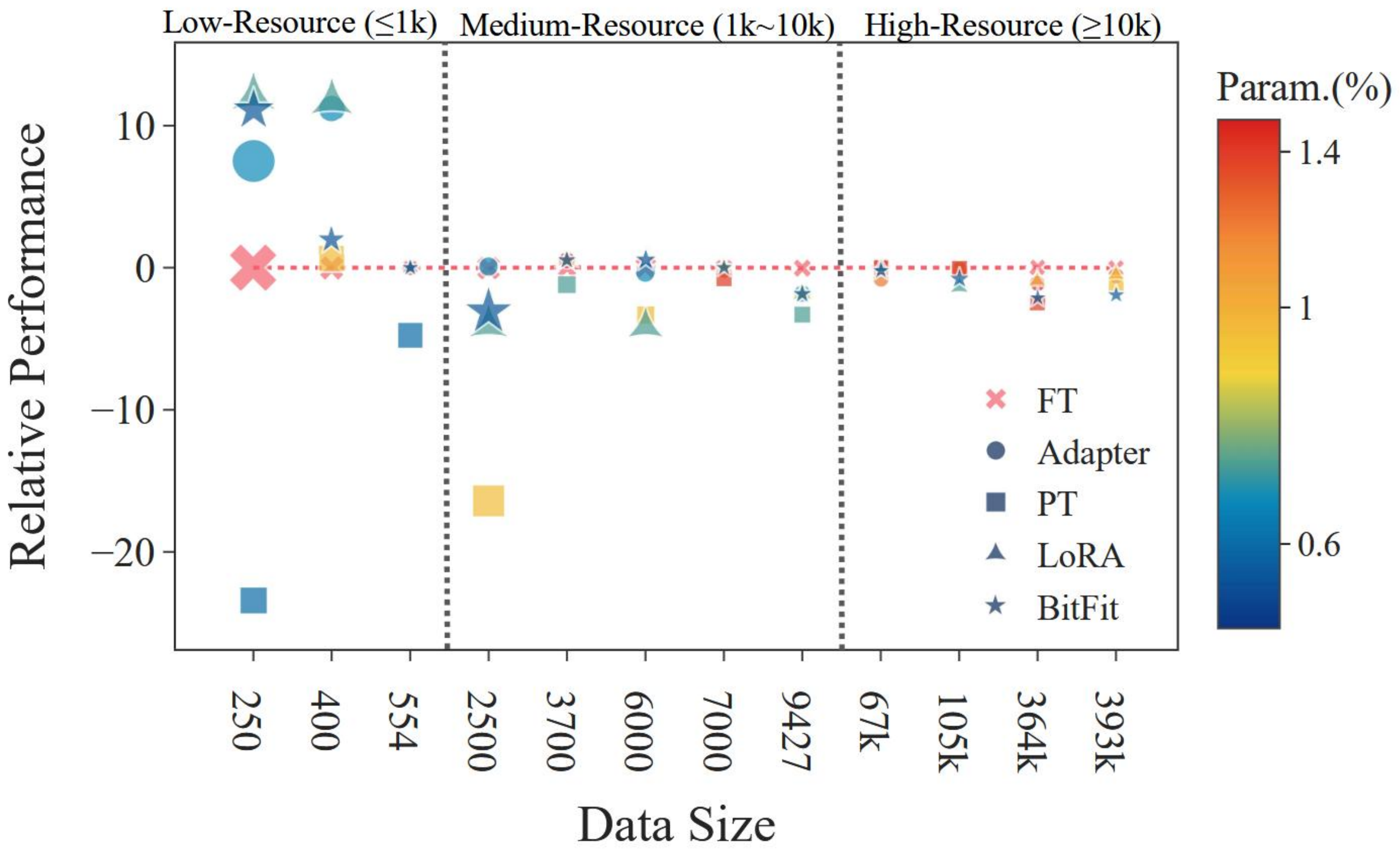
PREFIX TUNING

- Adapter methods and LoRA are similar in that they freeze the original model's parameters
  - And instead added a small number of trainable parameters.
- Prefix tuning is another PEFT method where new tokens are added to the beginning of the input prompt.
  - Unlike regular text tokens, these added tokens are continuous.
  - Their embeddings are the only trainable parameters in the model.



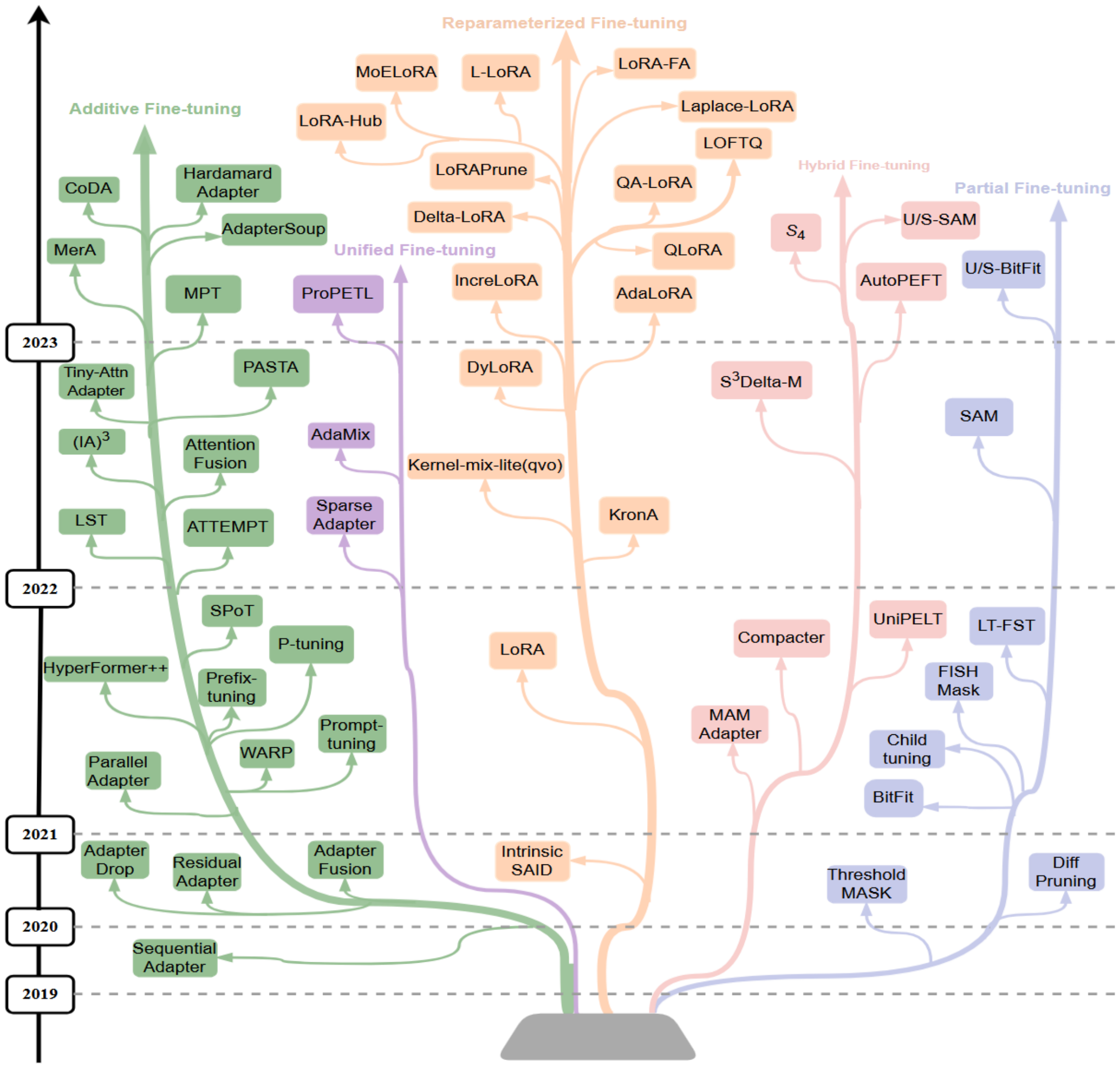
[Li and Liang, 2021]

PEFT



[Chen et al., 2022]

PEFT



[Xu et al., 2023]

EVALUATING MODEL RESPONSES

SO FAR: MINIMIZE CROSS-ENTROPY LOSS

- All machine learning techniques we have discussed thus far involved minimizing loss functions (usually cross-entropy).
- Recall minimizing cross-entropy loss is equivalent to maximizing likelihood.
- This approach is only teaching the model how to predict the correct output, for a given input.
  - For this reason, this kind of training is often called imitation learning.

SOME MISTAKES ARE BETTER THAN OTHERS

- NLP models often make mistakes.
- Some mistakes are better than others:

  Input: “`What is a substitute for baking soda in a cake recipe?`”

  Mistaken output 1: “`whipped cream`”

  Mistaken output 2: “`salt`”

  Mistaken output 3: “`bleach`”

  Mistaken output 4: “`How should I know, you #@&%!?`”
- Maximizing likelihood on only correct data does not teach the model which mistake is better.

ADAPTING TO MODEL-GENERATED INPUTS

- Training NLP models only on inputs from a gold dataset
  - E.g., Only containing human-generated inputs.
- If we are generating outputs autoregressively, where each output token is appended to the input in the next step,
- The model's performance may deteriorate,
- Especially if the distribution of the model's generated text is very different from the distribution of the training text.
- Called exposure bias.
- Example: the model makes a mistake when generating one token.
  - The model will be more likely to make mistakes on all subsequent tokens.

UNDESIRABLE CONTENT IN TRAINING DATA

- Also, the training dataset contains text that we don’t want the model to generate!
  - Misinformation
  - Comments from social networks
  - Conspiracy theories
  - Baises/stereotypes
  - Outputs from older/lower-quality NLP models
- If we simply maximize the likelihood, the resulting model will just learn to produce the same undesirable content that appears in the training data.

ALTERNATIVES TO MAXIMUM LIKELIHOOD TRAINING

- These shortcomings are a consequence of the fact that, for each example in the training set, there is only one “correct” output,
- And all other outputs are equally bad.
  - “All or nothing”
- Are there other ways we can measure the quality of an output that provides more information:
  - Which incorrect outputs are better than others?

ALTERNATIVES TO MAXIMUM LIKELIHOOD TRAINING

- Possible alternatives:
  - Maximize task-specific objective correctness score
  - Maximize human evaluation score
  - Train another machine learning model to produce a correctness score
    - Then maximize the model's predicted correctness score

OBJECTIVE CORRECTNESS SCORE

- There are some tasks where correctness scores are more easily defined.
- Example: math word problem solving

```
Mary starts with 8 apples. She buys 7 from the grocery store and
gives 12 to her friend, Jesse. How many apples does Mary have left?
```

Correct output: 3

OBJECTIVE CORRECTNESS SCORE

- There are some tasks where correctness scores are more easily defined.
- Example: math word problem solving
- One idea for correctness metric:

$$\exp\{(\mathit{predicted\_number} - \mathit{correct\_number})^2\}$$

```
Mary starts with 8 apples. She buys 7 from the grocery store and
gives 12 to her friend, Jesse. How many apples does Mary have left?
```

Correct output: 3

OBJECTIVE CORRECTNESS SCORE

- This approach does not easily generalize to other tasks.
- How would you define a correctness score for the question-answering task from earlier:

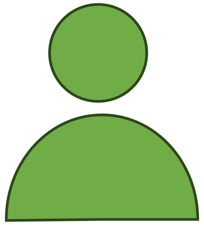
  Input: “`What is a substitute for baking soda in a cake recipe?`”

- This is especially difficult for subjective tasks, such as text generation.
  - E.g., generating fictional stories.
  - Paraphrasing.
  - etc...

HUMAN EVALUATION

- We can ask humans to score different outputs from NLP models.
- But this is very expensive.
  - We can't ask humans to label all possible model outputs.
  - We can only ask them to label some outputs.
- What kind of scale should we use?

Input: "`What is a substitute for baking soda in a cake recipe?`"
Output: "`baking powder`"



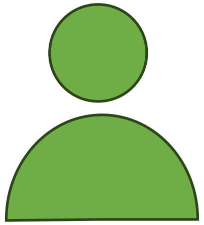
Human annotator

10 / 10

HUMAN EVALUATION

- We can ask humans to score different outputs from NLP models.
- But this is very expensive.
  - We can't ask humans to label all possible model outputs.
  - We can only ask them to label some outputs.
- What kind of scale should we use?

Input: “`What is a substitute for baking soda in a cake recipe?`”
Output: “`whipped cream`”



Human annotator

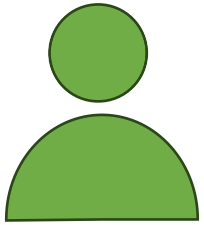
6 / 10

HUMAN EVALUATION

- We can ask humans to score different outputs from NLP models.
- But this is very expensive.
  - We can't ask humans to label all possible model outputs.
  - We can only ask them to label some outputs.
- What kind of scale should we use?

Input: `“What is a substitute for baking soda in a cake recipe?”`
Output: `“salt”`

→



Human annotator

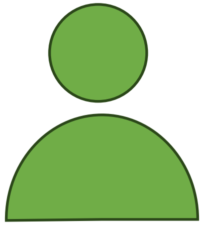
→

2 / 10

HUMAN EVALUATION

- We can ask humans to score different outputs from NLP models.
- But this is very expensive.
  - We can't ask humans to label all possible model outputs.
  - We can only ask them to label some outputs.
- What kind of scale should we use?

Input: "What is a substitute for baking soda in a cake recipe?"
Output: "bleach"

→  → 0 / 10

Human annotator

HUMAN EVALUATION

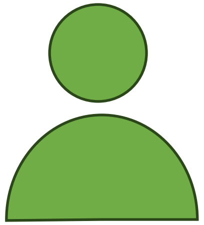
- We can ask humans to score different outputs from NLP models.
- Human evaluators are often asked to score multiple aspects of the output:
  - Fluency: How natural is the output?
  - Adequacy: In translation, does the output capture the meaning/semantics of the input?
  - Factuality: Is the output factual? Does it follow logically from the input?
  - Coherence: Does the output fit coherently in the discourse?
  - etc...

PREFERENCE RANKING

- Instead of asking humans to give a score for each output, we can ask them to give a ranking.

Input: “What is a substitute for baking soda in a cake recipe?”
Output 1: “salt”
Output 2: “whipped cream”
Output 3: “bleach”

→



Human annotator

→

“whipped cream” > “salt” > “bleach”

- This is easier to do for annotators.
- But annotators can’t specify the “degree” of output quality.
  - How much worse is “bleach” than “salt”?

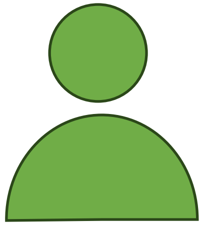
PREFERENCE RANKING

- Instead of asking humans to give a score for each output, we can ask them to give a ranking.
- How do you convert rankings into a numerical score for each output?
  - Elo
    - Used in chess
    - Only supports binary comparisons
  - TrueSkill (Sakaguchi et al., 2014)
    - Designed for Xbox Live and online gaming
    - Supports n-way comparisons
  - Train a model.
    - We will discuss this approach later in this lecture.

HUMAN EVALUATION: ERROR ANNOTATION

- Humans annotators can provide more fine-grained feedback:
  - Annotate specific errors in the output.

Input: “`Translate: Envía el paquete a Londres.`”
Output: “`Send a package to Tokyo.`”



Human annotator

Annotated output:
“`Send a package to Tokyo.`”

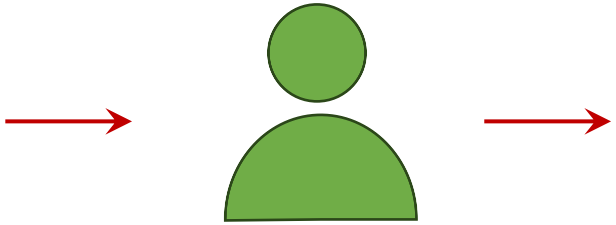
“`a`”: minor error/linguistic convention
“`Tokyo`”: major error

- This approach is used in machine translation.
  - Multidimensional quality metrics (Frietag et al., 2021).
- But this is a lot of work for annotators.
  - Difficult to scale to large numbers of examples.

AUTOMATED EVALUATION

- Can we try automating the evaluation process?

Input: “`Translate: Envía el paquete a Londres.`”
Output: “`Send a package to Tokyo.`”



Human annotator

3 / 10

AUTOMATED EVALUATION

- Can we try automating the evaluation process?
- Potentially save a lot of human annotation time.
- Much easier to scale to many many examples.

Input: “`Translate: Envía el paquete a Londres.`”
Output: “`Send a package to Tokyo.`”

→ model → 3 / 10

AUTOMATED EVALUATION

- If we have a reference output (i.e., gold output), we can compute a similarity score between the predicted output and the reference output.
  - (we automatically have reference outputs if we have supervised labels)
- What text similarity metrics can we use?

Input: “`Translate: Envía el paquete a Londres.`”
Output: “`Send a package to Tokyo.`”

$\longrightarrow$ `similarity`(x, y) $\longrightarrow$ 6 / 10

$\uparrow$

Reference: “`Send the package to London.`”

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`

Reference: `Send the package to London.`

- First, consider all 1-grams that appear in the prediction: “Send”, “a”, “package”, “to”, “Tokyo”.
- For each 1-gram, count how many times it appears in the prediction as well as in the reference.
    - “Send” appears 1 time in the prediction, and 1 time in the reference.
    - “Tokyo” appears 1 time in the prediction, and 0 times in the reference.
    - etc...

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`
Reference: `Send the package to London.`

$$\frac{\sum_{s \,\in\, \texttt{1-grams of}\, x} \texttt{\# of times}\ s\ \texttt{appears in}\ y}{\sum_{s \,\in\, \texttt{1-grams of}\, x} \texttt{\# of times}\ s\ \texttt{appears in}\ x}$$

where x is the prediction and y is the reference.

- So in the above example, the numerator is 1 + 0 + 1 + 1 + 0 = 3.
- The denominator is 1 + 1 + 1 + 1 + 1 = 5.
- So the ratio is 3/5 = 0.6.

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`

Reference: `Send the package to London and to Paris.`

$$\frac{\sum_{s\ \in\ \texttt{1-grams of}\ x} \texttt{min\{\# of times}\ s\ \texttt{appears in}\ x\texttt{, \# of times}\ s\ \texttt{appears in}\ y\texttt{\}}}{\sum_{s\ \in\ \texttt{1-grams of}\ x} \texttt{\# of times}\ s\ \texttt{appears in}\ x}$$

where x is the prediction and y is the reference.

- Caveat: To properly handle the case where s appears in the reference more than in the prediction, we need to add a min to the numerator.
- Consider the slightly modified example above with the 1-gram "to."

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`
Reference: `Send the package to London.`

$$\frac{\sum_i \sum_{s\ \in\ \texttt{1-grams of}\ x_i} \texttt{min\{\# of times}\ s\ \texttt{appears in}\ x_i,\ \texttt{\# of times}\ s\ \texttt{appears in}\ y\texttt{\}}}{\sum_i \sum_{s\ \in\ \texttt{1-grams of}\ x_i} \texttt{\# of times}\ s\ \texttt{appears in}\ x_i}$$

where x_i is the i^{th} predicted sentence and y is the reference.

- BLEU was developed to work with multiple sentences in both the predicted output and the reference output.

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`

Reference: `Send the package to London.`

$$\frac{\sum_i \sum_{s\ \in\ \texttt{1-grams of}\ x_i} \texttt{min}\{\texttt{\# of times}\ s\ \texttt{appears in}\ x_i, \texttt{max}_j\{\texttt{\# of times}\ s\ \texttt{appears in}\ y_j\}\}}{\sum_i \sum_{s\ \in\ \texttt{1-grams of}\ x_i} \texttt{\# of times}\ s\ \texttt{appears in}\ x_i}$$

where x_i is the i^{th} predicted sentence and y_j is the j^{th} reference sentence.

- BLEU was developed to work with multiple sentences in both the predicted output and the reference output.

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`
Reference: `Send the package to London.`

$$\mathrm{S}_n(x, y) = \frac{\sum_i \sum_{s \,\in\, n\text{-grams of } x_i} \texttt{min\{\# of times } s \texttt{ appears in } x_i\texttt{, max}_j\texttt{\{\# of times } s \texttt{ appears in } y_j\texttt{\}\}}}{\sum_i \sum_{s \,\in\, n\text{-grams of } x_i} \texttt{\# of times } s \texttt{ appears in } x_i}$$

- In BLEU, we compute the above quantity for 1-grams, 2-grams, ..., n-grams, for multiple values of n.

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`
Reference: `Send the package to London.`

$$\mathrm{S}_n(x, y) = \frac{\sum_i \sum_{s \in n\text{-grams of } x_i} \texttt{min\{\# of times } s \texttt{ appears in } x_i\texttt{, max}_j\texttt{\{\# of times } s \texttt{ appears in } y_j\texttt{\}\}}}{\sum_i \sum_{s \in n\text{-grams of } x_i} \texttt{\# of times } s \texttt{ appears in } x_i}$$

- For the above example, we had computed $\mathrm{S}_1(x, y) = 0.6$.
- Let's compute $\mathrm{S}_2(x, y)$:
  - 2-grams of x are: "Send a", "a package", "package to", "to Tokyo".
  - Numerator is: 0 + 0 + 1 + 0 = 1
  - Denominator is: 1 + 1 + 1 + 1 = 4

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`

Reference: `Send the package to London.`

$$\mathrm{S}_n(x, y) = \frac{\sum_i \sum_{s \in n\text{-grams of } x_i} \min\{\#\ \text{of times } s \text{ appears in } x_i,\ \max_j\{\#\ \text{of times } s \text{ appears in } y_j\}\}}{\sum_i \sum_{s \in n\text{-grams of } x_i} \#\ \text{of times } s \text{ appears in } x_i}$$

- For the above example, we had computed $\mathrm{S}_1(x, y) = 0.6$, $\mathrm{S}_2(x, y) = 0.25$.

$$\mathrm{BLEU}(x, y) = (\text{brevity penalty}) \exp\{\textstyle\sum_n \frac{1}{n} \log \mathrm{S}_n(x, y)\}$$

- The full BLEU score is the geometric mean of $\mathrm{S}_n(x, y)$, multiplied by a “brevity penalty”.

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send the package`

Reference: `Send the package to London.`

- Why do we need a brevity penalty?
- Consider the above example.
  - All n-grams of the prediction appear in the reference.
  - Without the brevity penalty, the BLEU score would be 1 (perfect score).

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send the package`

Reference: `Send the package to London.`

```
brevity penalty = max{0, exp{1 - r/c}}
```

where c is the total number of words in the predicted sentences,

and r = $\sum_i$`|reference sentence with length closest to |x`$_i$`||.`

TEXT SIMILARITY METRICS

- One prototypical metric is BLEU (bilingual evaluation understudy; Papineni et al., 2002).

Prediction: `Send a package to Tokyo.`

Reference: `Send the package to London.`

$$\mathrm{BLEU}(x, y) = (\mathrm{brevity\ penalty})\exp\{\textstyle\sum_n \frac{1}{n} \log S_n(x, y)\} = 0.387$$

- Going back to our example:
  - Suppose we only consider 1-gram and 2-grams. (usually, n goes up to 4)
  - $S_1(x, y) = 0.6$, $S_2(x, y) = 0.25$
  - The geometric mean is 0.387.
  - `c = |"Send a package to Tokyo."| = 5`
  - `r = 5` (we only have 1 reference sentence)
  - `brevity penalty = max{0, exp{1 - r/c}} = max{0, exp{0}} = 1`

BLEU DISADVANTAGES

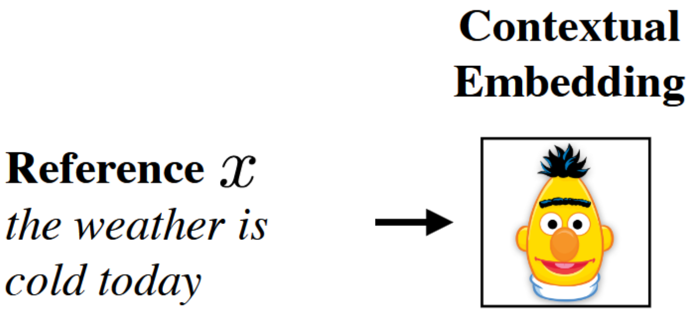
- BLEU does not consider meaning of sentences (i.e., semantics).
  - It only looks at subsequences of words.
- BLEU(“Do not send a package to Tokyo”, “Send a package to Tokyo”) = 0.544
- BLEU(“Please do send a package to Tokyo”, “Send a package to Tokyo”) = 0.544

- But it is ubiquitous in NLP,
- And is still being used (but maybe not as much as before).

LEARNED TEXT SIMILARITY METRICS

- Another approach: Compare the embeddings of the predicted sentence and the reference sentence.

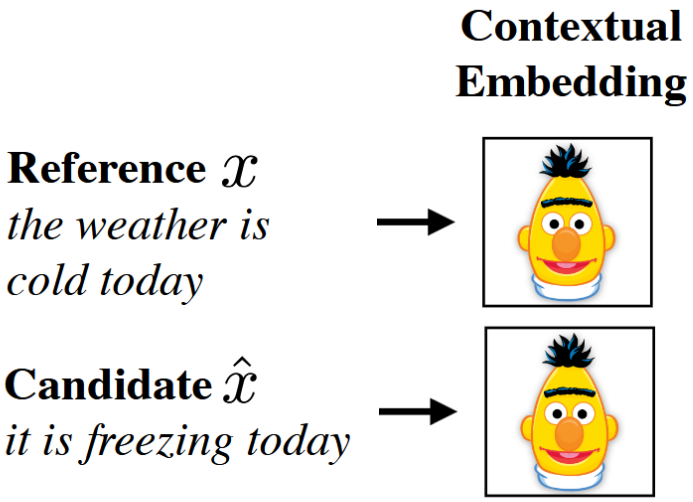
  Step 1: Compute embeddings for each token in the reference sentence.



[Zhang* and Kishore* et al., 2020]

LEARNED TEXT SIMILARITY METRICS

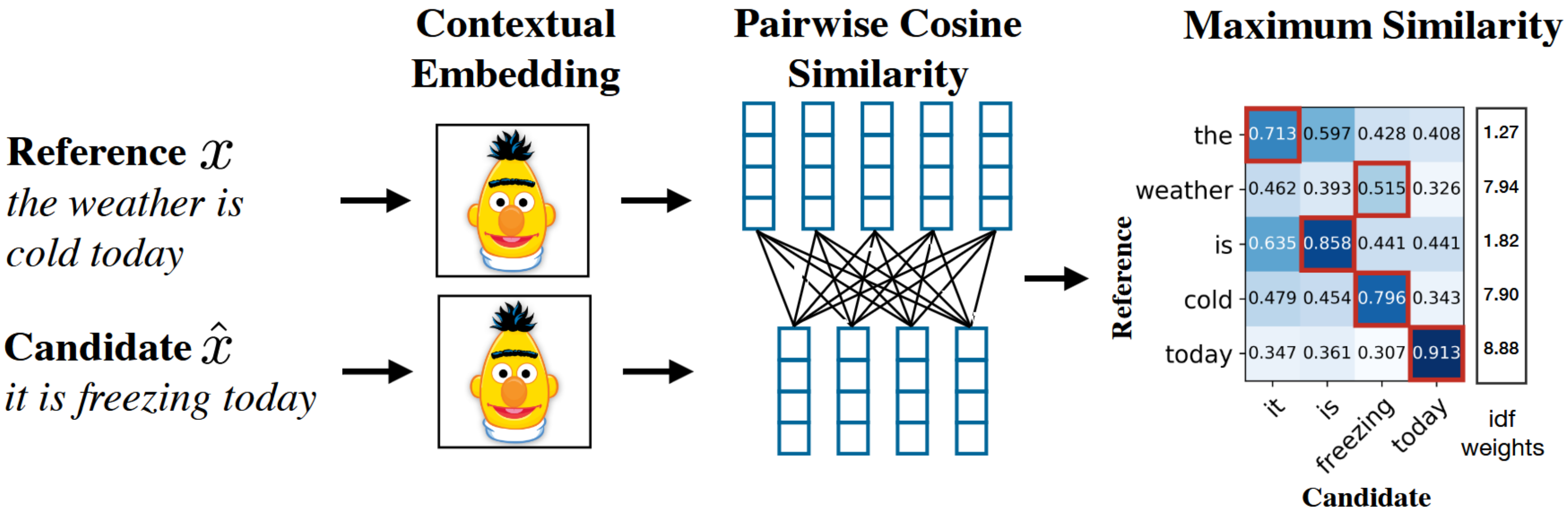
Step 2: Compute embeddings for each token in the predicted sentence.



[Zhang* and Kishore* et al., 2020]

LEARNED TEXT SIMILARITY METRICS

Step 3: For each embedding in the reference sentence, compute the maximum cosine similarity with embedding in the predicted sentence.

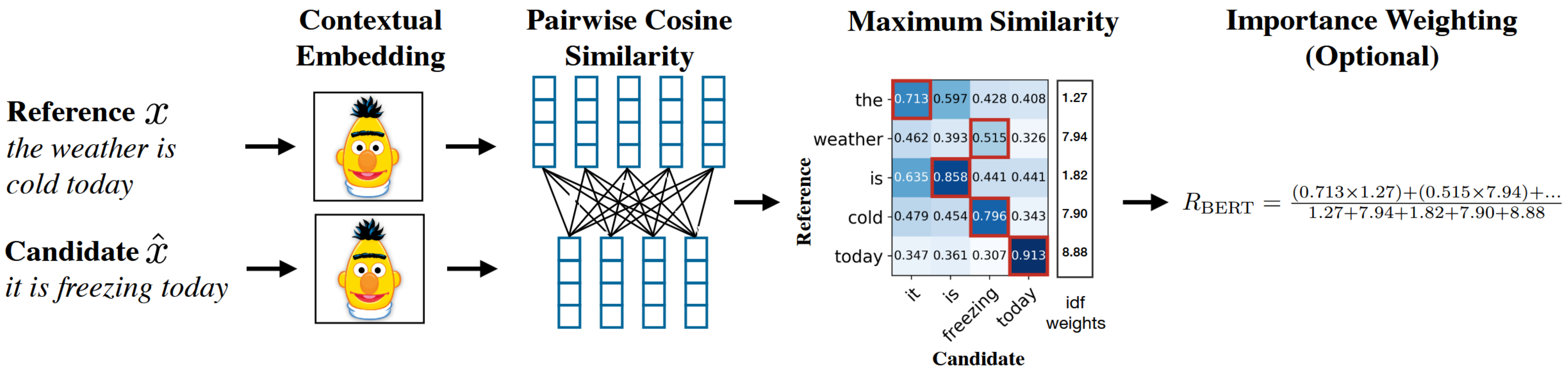


[Zhang* and Kishore* et al., 2020]

LEARNED TEXT SIMILARITY METRICS

Step 4: Compute the mean of these cosine similarities.

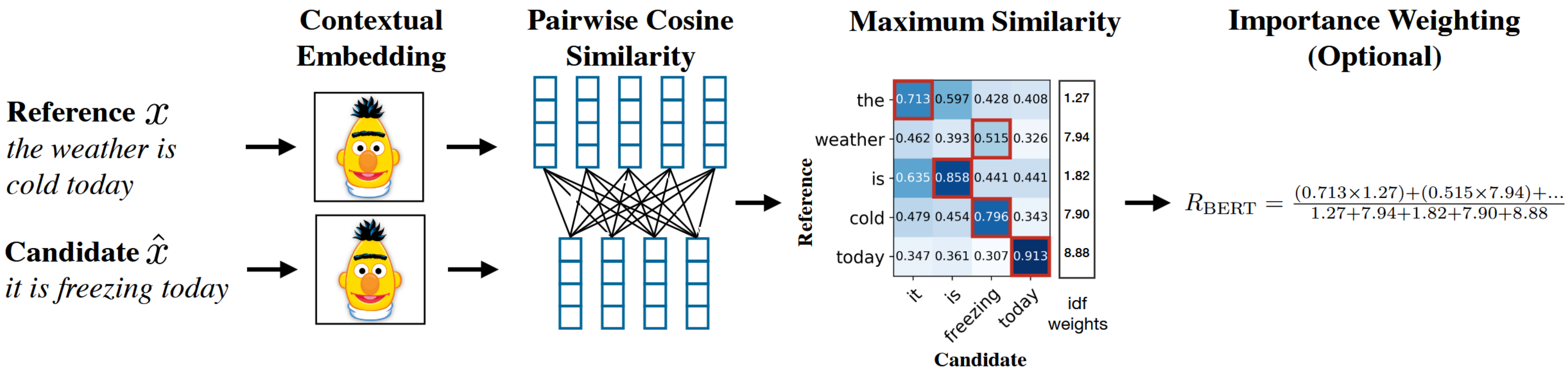
(optional) Compute a weighted average.



[Zhang* and Kishore* et al., 2020]

LEARNED TEXT SIMILARITY METRICS

- This approach is called BERTScore.



[Zhang* and Kishore* et al., 2020]

LEARNED TEXT SIMILARITY METRICS

- Another idea: Train BERT to directly output the similarity score.
- Add a linear layer to pretrained BERT.
- Fine-tune this linear layer on a large corpus, where each example is:
  - Reference sentence
  - Predicted sentence
  - Human rating of the sentence similarity
- Problem: Human annotated datasets are not very big.
- Training on this not-so-large dataset will result in overfitting.

LEARNED TEXT SIMILARITY METRICS

- Possible solution: Augment the training set using synthetic data (Sellam et al., 2020).
- How do we create new examples from existing examples?
- Synthetically generate perturbations of sentences that preserve their meaning.
  - Use machine translation to translate into a different language and then translate back.
  - Randomly mask some words and use a masked language model to fill in the blanks.
- Train the model on this augmented data set.
- This metric is called BLEURT.

AUTOMATED EVALUATION

- Text similarity metrics rely on having a reference sentence for every example.

Input: "Translate: Envía el paquete a Londres."
Output: "Send a package to Tokyo."

$\texttt{similarity}(x, y)$

6 / 10

Reference: "Send the package to London."

AUTOMATED EVALUATION

- Text similarity metrics rely on having a reference sentence for every example.
- If we aim to predict a score for a much larger set of examples,
- We need a method to evaluate outputs with less supervision.

Input: “`Translate: Envía el paquete a Londres.`”
Output: “`Send a package to Tokyo.`”

`similarity`(x, y)

???

Reference: ???

AUTOMATED EVALUATION

- Text similarity metrics rely on having a reference sentence for every example.
- If we aim to predict a score for a much larger set of examples,
- We need a method to evaluate outputs with less supervision.
- Idea: Train a model to predict a score for each example.

Input: "`Translate: Envía el paquete a Londres.`"
Output: "`Send a package to Tokyo.`"

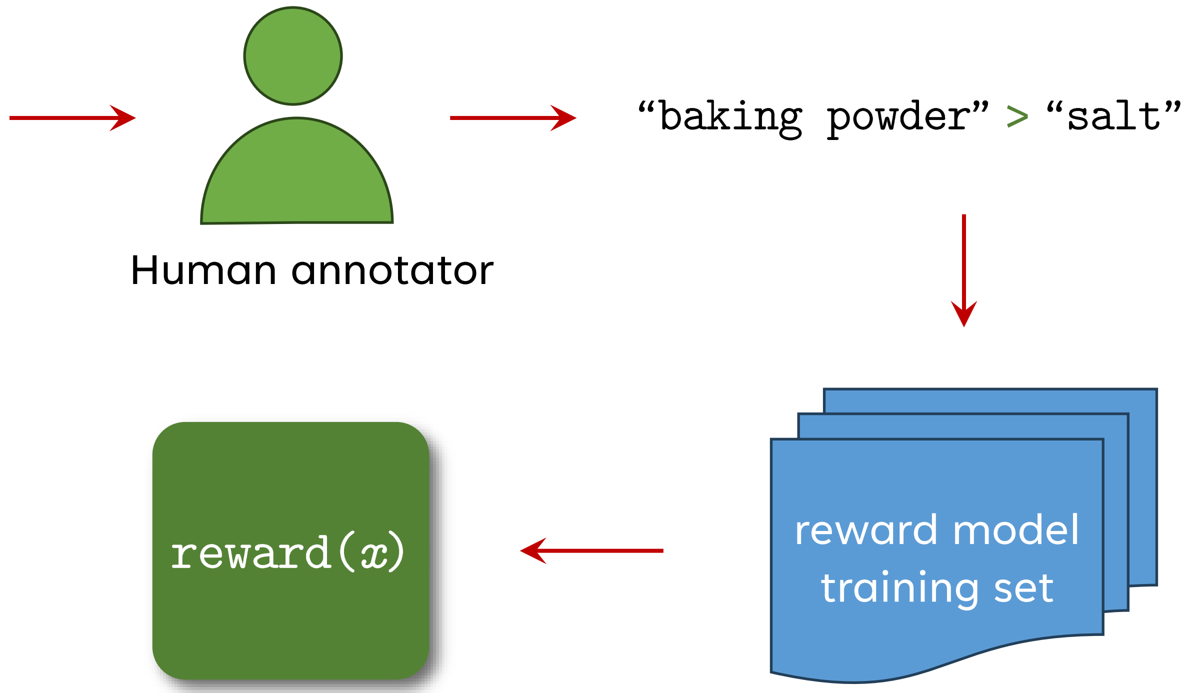
$\longrightarrow$ $\texttt{reward}(x)$ $\longrightarrow$ 6 / 10

- Note: If this model is used in reinforcement learning (which we will discuss later), it is called a reward model.

REWARD MODEL

- How do we train a reward model?
- Basic idea: Use human-provided preference annotations.

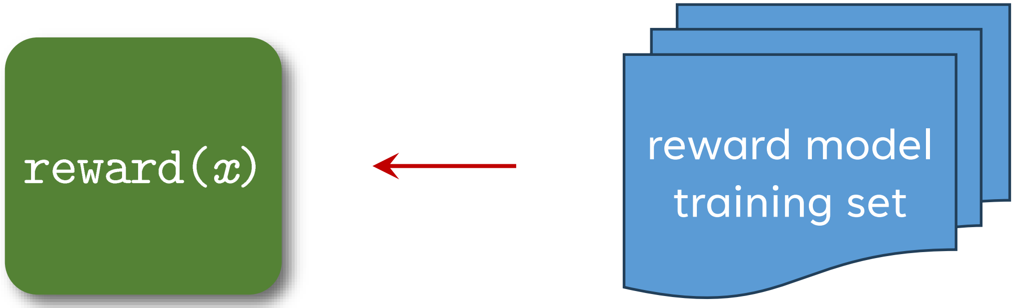
Input: “What is a substitute for baking soda in a cake recipe?”
Output 1: “salt”
Output 2: “baking powder”



REWARD MODEL

- How do we train a reward model?
- Basic idea: Use human-provided preference annotations.
- For each example in the reward training set, we have two outputs x_1 and x_2, where x_1 is preferred over x_2.
- We train a model r using the loss function:

$$L(w) = -\log \sigma(r_w(x_1) - r_w(x_2))$$



REWARD MODEL

- How do we train a reward model?
- Basic idea: Use human-provided preference annotations.
- For each example in the reward training set, we have two outputs x_1 and x_2, where x_1 is preferred over x_2.
- We train a model r using the loss function:

$$L(w) = -\log \sigma(r_w(x_1) - r_w(x_2))$$

- If $r_w(x_1) > r_w(x_2)$, then the output of the sigmoid will be closer to 1, the logarithm of which will be close to 0, and so the overall loss will be close to 0.
- If $r_w(x_1) < r_w(x_2)$, then the output of the sigmoid will be closer to 0, the logarithm of which will be negative, and so the overall loss will be positive.

REINFORCEMENT LEARNING

- With a trained reward model, we can compute the score of any output.
- Now that we have output scores, how do we use them to train an NLP model?
  - (as opposed to using supervised maximum likelihood training)
- Reinforcement learning (RL) is a machine learning technique that is often used in settings without supervision.
  - Where not all examples have a “correct” label.
- In RL, we have an agent that takes actions in an environment over time.
  - The agent receives reward from the environment depending on their actions.
  - The goal is to teach the agent to maximize reward.

REINFORCEMENT LEARNING

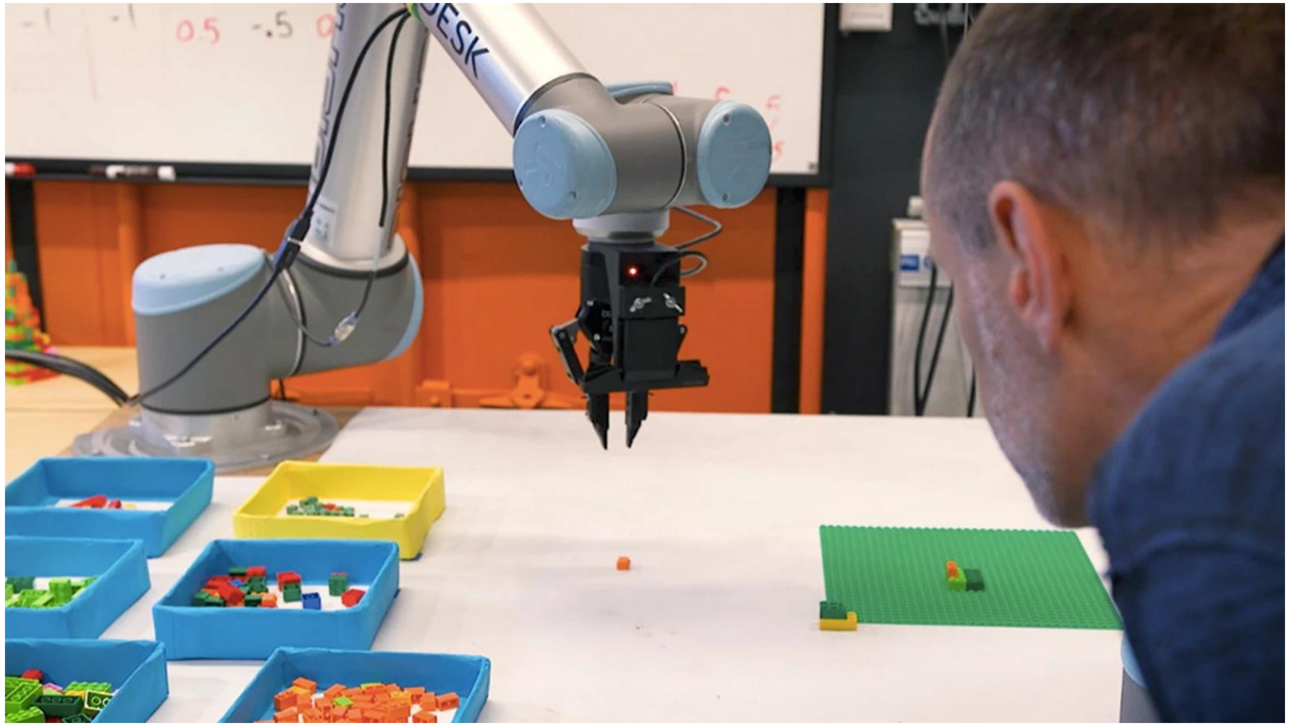
- The RL setting is highly flexible.
- Teaching agents to play games:
  - Environment: Current Tetris board
  - Actions: Rotate current tile, move left, move right
  - Reward: Game score

REINFORCEMENT LEARNING

- The RL setting is highly flexible.
- Teaching agents to play games:
  - Environment: Minecraft world state
  - Actions: Walk forward, mine, equip item, jump, etc.
  - Reward: Crafting items, staying alive, etc.

REINFORCEMENT LEARNING

- The RL setting is highly flexible.
- Teaching robots to perform tasks:
  - Environment: Physical world around robot.
  - Actions: Move arm left, right, forward, open hand, close hand.
  - Reward: Complete task.



REINFORCEMENT LEARNING

- The RL setting is highly flexible.
- Teaching NLP models to produce better outputs.
  - Environment: Input text
  - Actions: All possible outputs of the NLP model
  - Reward: Provided by a trained reward model
- At each “episode”, the environment provides the NLP model with a random input.
  - In language modeling, this a random prompt.
  - At each timestep, the language model predicts one token of the output.

REINFORCEMENT LEARNING

- How do we train an agent to maximize reward?
- There are many algorithms; RL is a very deep field.
- Next lecture, we will discuss some of these algorithms, including those that are commonly used to train language models.
  - Reinforcement learning from human feedback (RLHF)

QUESTIONS?