CS11-711 Advanced NLP

(Reinforcement) Learning from Human Feedback

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/anlp2024/

Maximum Likelihood Training

- Maximum the likelihood of predicting the next word in the reference given the previous words

$$
\begin{aligned}
\ell(Y|X) &= -\log P(Y|X) \\
&= -\sum_t \log P(y_t|X, y_{<t})
\end{aligned}
$$

Problem 1: Some Mistakes are Worse than Others

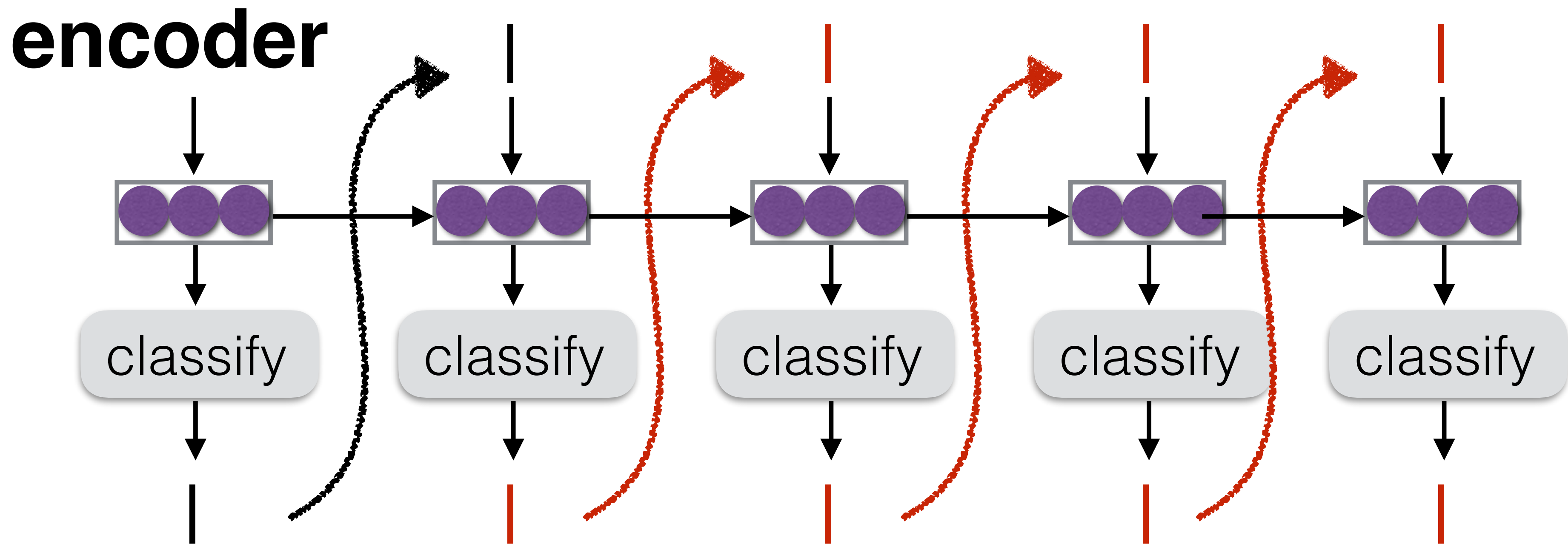
- In the end, we want good outputs
- Some mistaken predictions hurt more than others, so we'd like to penalize them appropriately
- e.g.:
  - Please send this package to Pittsburgh
  - Please send a package to Pittsburgh
  - Please send this package to Tokyo
  - ****ing send this package to Pittsburgh

Problem 2: The “Gold-standard” in MLE can be Bad

- Corpora are full of outputs that we wouldn’t want a language model reproducing!
- For instance:
  - Toxic comments in reddit
  - Disinformation
  - Translations from old machine translation systems

Problem 3: Exposure Bias

- MLE training doesn't consider the necessity for generation — relies on gold-standard context



- **Exposure bias:** The model is not exposed to mistakes during training, and cannot deal with them at test

Measuring how “Good” an Output Is

How to Measure output “Goodness"?

- Objective assessment
- Human subjective annotation
- Machine prediction of human preferences
- Use in another system

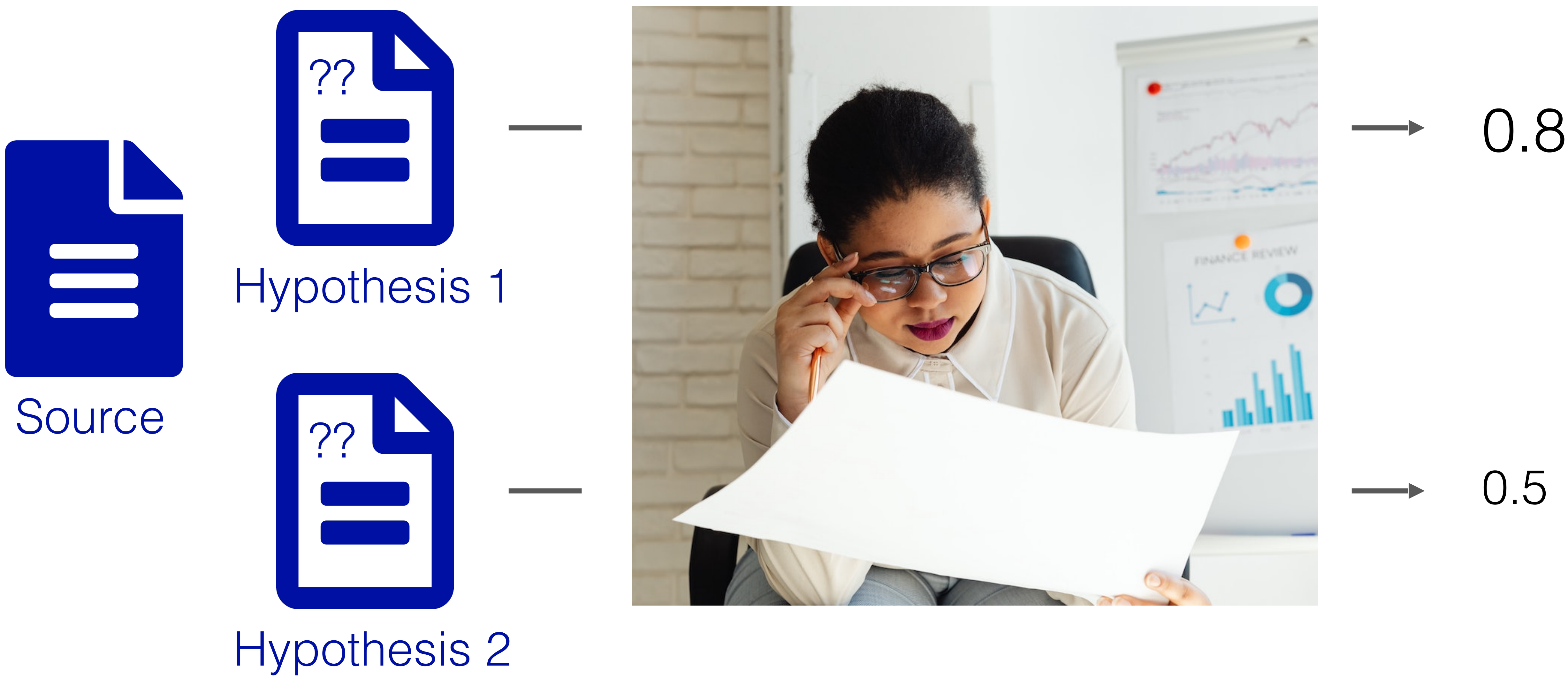
Objective Assessment

- Have an annotated “correct” answer and match against this
- e.g. in solving math problems, answering objective questions

Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

Source: GSM8K - Cobbe et al. 2021

Human Evaluation



Human Feedback: Direct Assessment

- Directly give a score

Please send this package to Tokyo → 2/10

- Often assign scores based on desirable traits
  - **Fluency:** how natural is the output
  - **Adequacy:** in translation, how well does the output reflect the input semantics?
  - **Factuality:** is the output factually entailed
  - **Coherence:** does the output fit coherently in a discourse?
  - etc. etc.

Human Feedback: Preference Ratings

- Preference rankings

Please send this package to Tokyo	worse
Please send a package to Pittsburgh	better

- + can be easier and more intuitive than direct assessment
- - can't tell if all systems are really good or really bad
- To rank multiple systems, can use ELO or TrueSkill rankings (Sakaguchi et al. 2014)

Human Feedback: Error Annotation

- Annotate individual errors within the outputs
- e.g. Multi-dimensional Quality Metrics (Freitag et al. 2021)

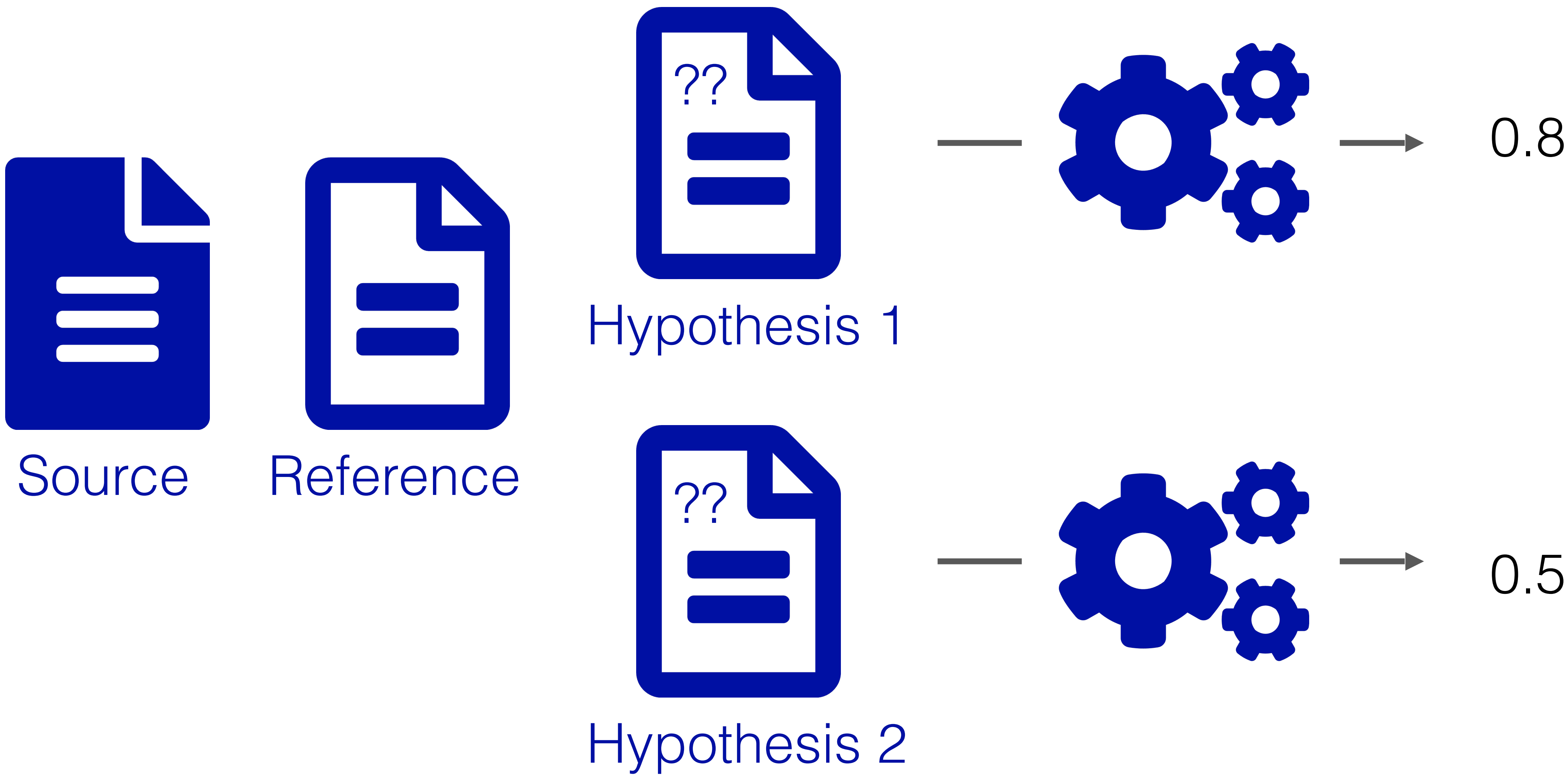
Can you send a package to Tokyo

minor/
linguistic
conventions

major/
accuracy

- + Gives more fine-grained feedback
- + Can be more consistent
- - Can be very time-consuming

An Alternative: Automatic Evaluation

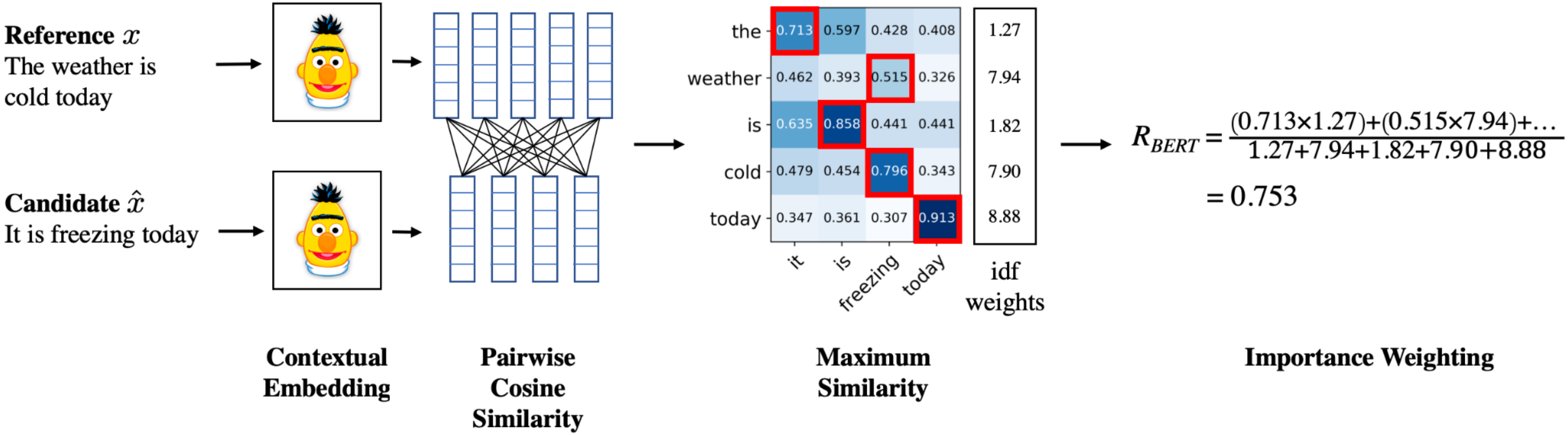


Machine Prediction of Human Preferences

- Predict human feedback automatically using a model
- Variously called
  - “automatic evaluation” e.g. in machine translation
  - “reward model” e.g. in chatbots
- Sometimes uses a “reference” output

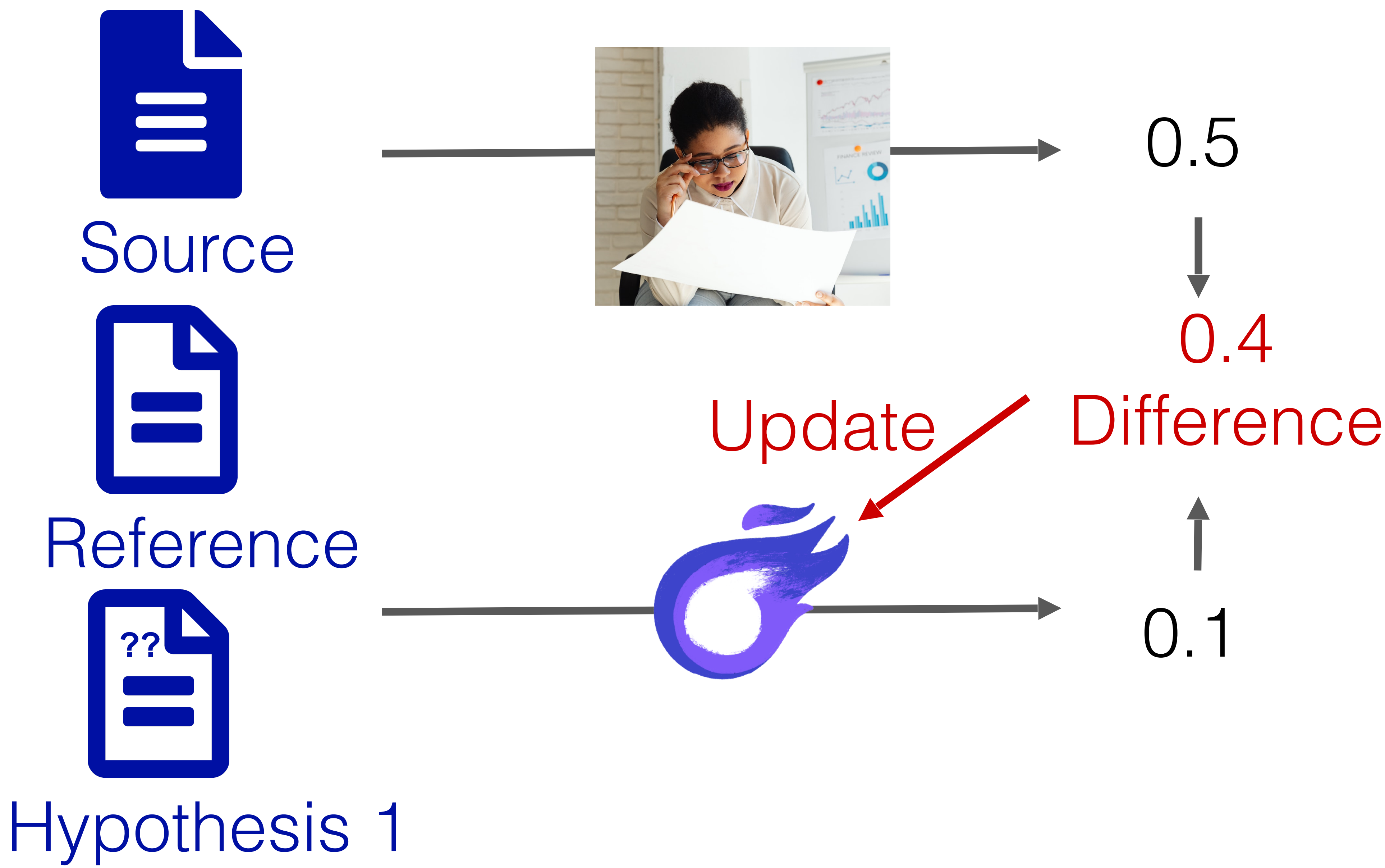
Embedding-based Evaluation

- Unsupervised calculation based on embedding similarity
- e.g. BERTScore (Zhang et al. 2019)



Regression-based Evaluation

- Supervised training of an embedding-based regressor
- e.g. COMET (Rei et al. 2020)



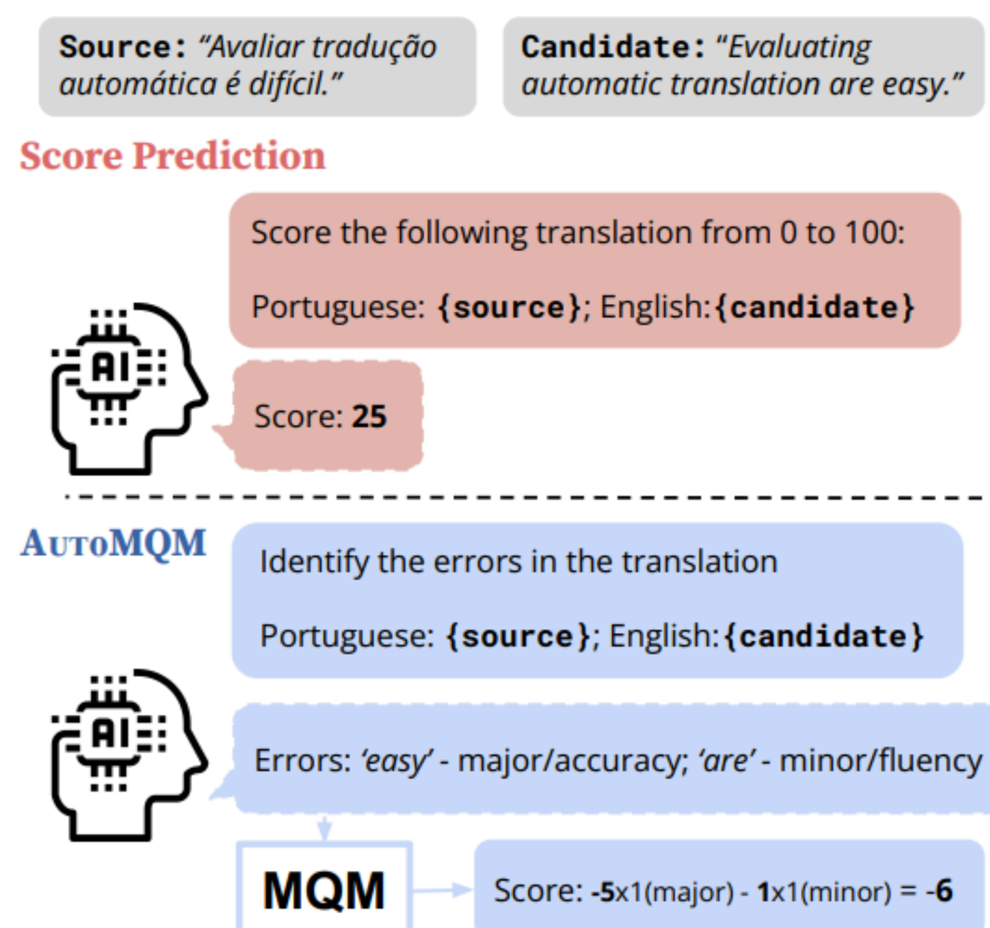
QA-based Evaluation

- Ask a language model how good the output is
- e.g. GEMBA (Kocmi and Federmann 2023)

```
Score the following translation from {source_lang} to {target_lang} with respect
to the human reference on a continuous scale from 0 to 100, where score of zero means
"no meaning preserved" and score of one hundred means "perfect meaning and grammar".

{source_lang} source: "{source_seg}"
{target_lang} human reference: {reference_seg}
{target_lang} translation: "{target_seg}"
Score:
```

- Can also ask about fine-grained mistakes - AutoMQM (Fernandes et al. 2023)



Meta-evaluation of Metrics

	Human	Automatic
	0.8	0.7
	0.5	0.1
	0.1	0.5
	0.6	0.4

Pearson = 0.23

Kendall = 0.33

Error = 0.28

- Use datasets like WMT shared tasks (Fabbri et al. 2020), SumEval (Freitag et al. 2023)

Use in a Downstream System

- **Intrinsic evaluation:** Evaluate the quality of the output itself
- **Extrinsic evaluation:** Evaluate output quality by its utility
- Example: evaluate LLM summaries through QA accuracy (Eyal et al. 2019)

Error and Risk

Error

- Generate an output

$$\hat{Y} = \underset{\tilde{Y}}{\operatorname{argmax}}\ P(\tilde{Y}|X)$$

- Calculate its "badness" (e.g. 1-eval score)

$$\mathrm{error}(Y, \hat{Y}) = 1 - \mathrm{eval}(Y, \hat{Y})$$

- We would like to minimize error

Problem: Argmax is Non-differentiable

- The argmax function makes discrete zero-one decisions
- The gradient of this function is zero almost everywhere, not-conducive to gradient-based training

Risk

- Risk is defined as the expected error

$$\mathrm{risk}(X, Y, \theta) = \sum_{\tilde{Y}} P(\tilde{Y} | X; \theta)\,\mathrm{error}(Y, \tilde{Y})$$

- This is includes the probability in the objective function!

- Differentiable, but the sum is intractable

- Minimum risk training minimizes risk, Shen et al. (2015) do so for NMT

Sampling for Tractability

- Create a small sample of sentences (5-50), and calculate risk over that

$$\text{risk}(X, Y, \theta) = \sum_{\tilde{Y} \in S} \frac{P(\tilde{Y} | X; \theta)}{Z} \text{error}(Y, \tilde{Y})$$

- Samples can be created using sampling or n-best search

- If sampling: be sure to deduplicate

Reinforcment Learning Basics: Policy Gradient

(Review of Karpathy 2016)

What is Reinforcement Learning?

- Learning where we have an
  - environment X
  - ability to make actions A
  - get a delayed reward R
- **Example of pong:** X is our observed image, A is up or down, and R is the win/loss at the end of the game

Why Reinforcement Learning in NLP?

- We may have a **typical reinforcement learning scenario**: e.g. a dialog where we can make responses and will get a reward at the end.
- We may have **latent variables** (e.g. chains of thought), where we decide the latent variable, then get a reward based on their configuration.
- We may have a **sequence-level evaluation metric** such that we cannot optimize without first generating a whole sentence.

Supervised MLE

- We are given the correct decisions

$$\ell_{\mathrm{super}}(Y, X) = -\log P(Y \mid X)$$

- In the context of reinforcement learning, this is also called "imitation learning," imitating a teacher (although imitation learning is more general)

Self Training

- Sample or argmax according to the current model

$$\hat{Y} \sim P(Y \mid X) \quad \text{or} \quad \hat{Y} = \text{argmax}_Y P(Y \mid X)$$

- Use this sample (or samples) to maximize likelihood

$$\ell_{\text{self}}(X) = -\log P(\hat{Y} \mid X)$$

- No correct answer needed! But is this a good idea?
- *One successful alternative:* co-training, only use sentences where multiple models agree (Blum and Mitchell 1998)
- *Another successful alternative:* noising the input, to match output (He et al. 2020)

Policy Gradient/REINFORCE

- Add a term that scales the loss by the reward

$$\ell_{\text{REINFORCE}}(X, \hat{Y}) = -R(Y, \hat{Y}) \log P(\hat{Y}|X)$$

- Outputs that get a bigger reward will get a higher weight
- Quiz: Under what conditions is this equal to MLE?

Credit Assignment for Rewards

- How do we know which action led to the reward?
- Best scenario, immediate reward:

a_1	a_2	a_3	a_4	a_5	a_6
0	+1	0	-0.5	+1	+1.5

- Worst scenario, only at end of roll-out:

a_1	a_2	a_3	a_4	a_5	a_6	
						+3

- Often assign decaying rewards for future events to take into account the time delay between action and reward

Stabilizing Reinforcement Learning

Problems w/ Reinforcement Learning

- Like other sampling-based methods, reinforcement learning is unstable
- It is particularly unstable when using bigger output spaces (e.g. words of a vocabulary)
- A number of strategies can be used to stabilize

Pre-training with MLE
(Ranzato et al. 2016)

- Start training with MLE, then switch over to RL
- Works only in the scenarios where we can run MLE (not latent variables or standard RL settings)

Regularization to an Existing Model

(e.g. Schulman et al. 2017)

- Have an existing model, and prevent it from moving too far away
- **Method one:** KL regularization

$$\ell_{regularized} = \underbrace{\frac{P(\hat{Y}|X;\theta)}{P(\hat{Y}|X;\theta_{old})} R(Y,\hat{Y})}_{\text{improve reward}} - \underbrace{\beta \mathrm{KL}\left[P(\cdot|X;\theta_{\mathrm{old}}), P(\cdot|X;\theta)\right]}_{\text{keep model similar}}$$

- **Method two:** proximal policy optimization (PPO)

$$\mathrm{rat}(Y,X) = \frac{P(Y|X;\theta)}{P(Y|X;\theta_{\mathrm{old}})}$$

$$\ell_{PPO} = \min(\mathrm{rat}(\hat{Y},X)R(\hat{Y}), \underbrace{\mathrm{clip}(\mathrm{rat}(\hat{Y},X), 1+\epsilon, 1-\epsilon)R(\hat{Y})}_{\text{don't reward large jumps}})$$

Adding a Baseline

- Basic idea: we have expectations about our reward for a particular sentence

	Reward	Baseline	B-R
"This is an easy sentence"	0.8	0.95	-0.15
"Buffalo Buffalo Buffalo"	0.3	0.1	0.2

- We can instead weight our likelihood by B-R to reflect when we did **better or worse than expected**

$$\ell_{\text{baseline}}(X) = -(R(\hat{Y}, Y) - B(\hat{Y})) \log P(\hat{Y} \mid X)$$

- (Be careful to not backprop through the baseline)

Calculating Baselines

- Choice of a baseline is arbitrary
- Option 1: predict final reward using linear from current state (e.g. Ranzato et al. 2016)
  - **Sentence-level:** one baseline per sentence
  - **Decoder state level:** one baseline per output action
- Option 2: use the mean of the rewards in the batch as the baseline (e.g. Dayan 1990)

Contrasting Pairwise Examples

(e.g. Rafailov et al. 2023)

- Can learn directly from pairwise (human) preferences, which provides more stability
- e.g. direct preference optimization (DPO)

$$\ell_{DPO} = \log \sigma \left(\underbrace{\beta \frac{P(Y_w|X;\theta)}{P(Y_w|X;\theta_{\text{old}})}}_{\text{better outputs}} - \underbrace{\beta \frac{P(Y_l|X;\theta)}{P(Y_l|X;\theta_{\text{old}})}}_{\text{worse outputs}} \right)$$

Increasing Batch Size

- Because each sample will be high variance, we can sample many different examples before performing update
- We can increase the number of examples (roll-outs) done before an update to stabilize
- We can also save previous roll-outs and re-use them when we update parameters (experience replay, Lin 1993)

Questions?