AI Applications Lecture 8

Evaluation of Probabilistic Language Models

SUZUKI, Atsushi
Jing WANG

Outline

Introduction

1.1 Review of the Previous Lecture

In the previous lectures, we learned about the **natural language sequence generation pipeline**, including neural networks and tokenization.

1.1 Review of the Previous Lecture

In the previous lectures, we learned about the **natural language sequence generation pipeline**, including neural networks and tokenization.

We also provided a rigorous formulation of:

- **probabilistic language models**
- **token generators** based on **sampling**, **greedy search**, and **beam search**.

1.1 Review of the Previous Lecture

In the previous lectures, we learned about the **natural language sequence generation pipeline**, including neural networks and tokenization.

We also provided a rigorous formulation of:

- **probabilistic language models**
- **token generators** based on **sampling**, **greedy search**, and **beam search**.

In this lecture, we will focus on **evaluation**, addressing how to **automatically and quantitatively** evaluate both probabilistic language models and natural language inputs/outputs.

1.2 Learning Outcomes

Through this lecture, students should be able to:

- Explain the **non-triviality** of evaluation in natural language processing compared to evaluation in classical supervised machine learning.

1.2 Learning Outcomes

Through this lecture, students should be able to:

- Explain the **non-triviality** of evaluation in natural language processing compared to evaluation in classical supervised machine learning.
- Distinguish between the **evaluation of natural language string input/output** and the **evaluation of probabilistic language models**.

1.2 Learning Outcomes

Through this lecture, students should be able to:

- Explain the **non-triviality** of evaluation in natural language processing compared to evaluation in classical supervised machine learning.
- Distinguish between the **evaluation of natural language string input/output** and the **evaluation of probabilistic language models**.
- Evaluate natural language string input/output using probabilistic language models with **appropriate metrics**.

Preliminaries: Mathematical Notations

2. Preliminaries: Mathematical Notations

Set:

- Sets: $\mathcal{A}$
- Membership: $x \in \mathcal{A}$
- Empty set: $\{\}$
- Roster notation: $\{a, b, c\}$
- Set-builder: $\{x \in \mathcal{A} | P(x)\}$
- Cardinality: $|\mathcal{A}|$
- Real numbers: $\mathbb{R}, \mathbb{R}_{>0}, \mathbb{R}_{\geq 0}$
- Integers: $\mathbb{Z}, \mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0}$
- Integer range: $[1, k]_{\mathbb{Z}} := \{1, \ldots, k\}$

Function:

- $f : \mathcal{X} \to \mathcal{Y}$: f maps from $\mathcal{X}$ to $\mathcal{Y}$.
- $y = f(x)$: The output of f for input x.

Definition:

- $(\mathrm{LHS}) := (\mathrm{RHS})$: Left side is defined by the right side.

Sequence: Denoted by $\boldsymbol{a} = (a_1, a_2, \dots)$.

- A function $\boldsymbol{a} : [1, n]_{\mathbb{Z}} \to \mathcal{A}$.
- Length is denoted by $|\boldsymbol{a}|$.

Vector: Denoted by $\boldsymbol{v}$.

- A column of numbers, $\boldsymbol{v} \in \mathbb{R}^n$.
- i-th element is v_i.

Matrix: Denoted by $\boldsymbol{A}$.

- $m \times n$ matrix: $\boldsymbol{A} \in \mathbb{R}^{m,n}$.
- (i, j)-th element is $a_{i,j}$.
- Transpose: $\boldsymbol{A}^\top$.

Tensor: Denoted by $\underline{\boldsymbol{A}}$.

- Simply a multi-dimensional array.
- Vector $\to$ 1st-order, Matrix $\to$ 2nd-order.

The Importance of Automatic Evaluation

3. The Importance of Automatic Evaluation

When evaluating computational methods, it is practically useful to employ **automatic and quantitative** evaluation whenever possible.

3. The Importance of Automatic Evaluation

When evaluating computational methods, it is practically useful to employ **automatic and quantitative** evaluation whenever possible.

Automatic evaluation does not require human resources and also contributes to ensuring the **reproducibility** of experiments.

Motivation for Introduction.

In supervised learning, since the input and correct output are explicitly given, fixing an abstract framework that provides an **average evaluation** by comparing model predictions with the correct answers allows for a unified description of evaluation metrics for individual tasks.

3.1 Formulation of Automatic Evaluation in Classical Supervised Learning

Motivation for Introduction.

In supervised learning, since the input and correct output are explicitly given, fixing an abstract framework that provides an **average evaluation** by comparing model predictions with the correct answers allows for a unified description of evaluation metrics for individual tasks.

Definition (Framework for Classical Evaluation)

Given an input space $\mathcal{X}$, an output space $\mathcal{Y}$, a trained map $f : \mathcal{X} \to \mathcal{Y}$, and an evaluation function $E : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$. For a test dataset $\{(x_i, y_i)\}_{i=1}^{N}$, the evaluation value is defined as

$$\mathrm{Eval}(f) := \frac{1}{N} \sum_{i=1}^{N} E\big(f(x_i),\, y_i\big). \tag{1}$$

Here, $N \in \mathbb{Z}_{>0}$ is the number of test points, and E can be a **loss** (smaller is

Remark

Typical examples include the **squared Euclidean distance** $E(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$ when $\mathcal{Y} = \mathbb{R}^d$, and the **0-1 loss** $E(\hat{y}, y) = \mathbf{1}[\hat{y} \neq y]$ when $\mathcal{Y}$ is a finite set.

Example (Calculation Example of Squared Error and 0-1 Loss)

Regression: If $\hat{\boldsymbol{y}} = (2, 0)^\top,\ \boldsymbol{y} = (1, 1)^\top$, then

$$\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2 = (2-1)^2 + (0-1)^2 = 1 + 1 = 2. \tag{2}$$

Classification: If $\hat{y} = \mathrm{cat},\ y = \mathrm{dog}$, then $E(\hat{y}, y) = 1$.

Exercise (Exercise on Classical Evaluation)

(1) Find the squared distance between $\hat{\boldsymbol{y}} = (3, -1)^\top$ and $\boldsymbol{y} = (1, 2)^\top$.

Exercise (Exercise on Classical Evaluation)

(1) Find the squared distance between $\hat{\boldsymbol{y}} = (3, -1)^\top$ and $\boldsymbol{y} = (1, 2)^\top$. (2) Find the 0-1 distance for $\hat{y} = \mathrm{A}, y = \mathrm{B}$.

Answer

(1) Step-by-step calculation of squared distance:

- First, find the difference vector: $\hat{\boldsymbol{y}} - \boldsymbol{y} = (3 - 1, -1 - 2)^\top = (2, -3)^\top$.

Answer

(1) Step-by-step calculation of squared distance:

- First, find the difference vector: $\hat{\boldsymbol{y}} - \boldsymbol{y} = (3-1, -1-2)^\top = (2, -3)^\top$.
- Then, calculate the squared norm:

$$\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2 = (2)^2 + (-3)^2 = 4 + 9 = 13.$$

Answer

(1) Step-by-step calculation of squared distance:

- First, find the difference vector: $\hat{\boldsymbol{y}} - \boldsymbol{y} = (3-1, -1-2)^\top = (2, -3)^\top$.
- Then, calculate the squared norm:

$$\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2 = (2)^2 + (-3)^2 = 4 + 9 = 13.$$

(2) Step-by-step calculation of 0-1 distance:

- Since the prediction $\hat{y}$ and the true label y do not match, the indicator function is true: $\mathbf{1}[\hat{y} \neq y] = 1$.

3.2 Why Automatic Evaluation of Natural Language Output is Difficult

In the space of natural language strings, **semantic equivalence** is essential, and there can be **infinitely many correct answers**.

3.2 Why Automatic Evaluation of Natural Language Output is Difficult

In the space of natural language strings, **semantic equivalence** is essential, and there can be **infinitely many correct answers**.

Therefore:

- (i) it is impractical to prepare all possible correct answers on the test side.

3.2 Why Automatic Evaluation of Natural Language Output is Difficult

In the space of natural language strings, **semantic equivalence** is essential, and there can be **infinitely many correct answers**.

Therefore:

- (i) it is impractical to prepare all possible correct answers on the test side.
- (ii) **automatically determining the semantic equivalence** between input strings is not easy.

3.2 Why Automatic Evaluation of Natural Language Output is Difficult

In the space of natural language strings, **semantic equivalence** is essential, and there can be **infinitely many correct answers**.

Therefore:

- (i) it is impractical to prepare all possible correct answers on the test side.
- (ii) **automatically determining the semantic equivalence** between input strings is not easy.

For this reason, various **evaluation metrics** have been proposed.

Two Major Families of Evaluation

4. Two Major Families: Language Model Evaluation vs. String Output Evaluation

Evaluation methods can be broadly divided into the following two categories:

- **Evaluation of Probabilistic Language Models**:
  - Examples include **perplexity** and the **most likely option** in multiple-choice tasks (MMLU, HellaSwag, etc.).
  - The advantage is that it avoids the difficulties of string generation.
  - The disadvantage is that it does not measure the string output performance itself.

4. Two Major Families: Language Model Evaluation vs. String Output Evaluation

Evaluation methods can be broadly divided into the following two categories:

- **Evaluation of Probabilistic Language Models**:
  - Examples include **perplexity** and the **most likely option** in multiple-choice tasks (MMLU, HellaSwag, etc.).
  - The advantage is that it avoids the difficulties of string generation.
  - The disadvantage is that it does not measure the string output performance itself.
- **Evaluation of Natural Language String Input/Output**:
  - Examples include **Exact Match/F1**, **BLEU**, **ROUGE-L**, **BERTScore**, and **Accuracy** for math QA.
  - The advantage is that it measures the output performance directly.
  - The disadvantage is that the calculation method differs for each task and may have language dependencies.

Evaluation of Probabilistic Language Models

Input/Output Format. Given a trained language model P and a tokenized evaluation sequence $\boldsymbol{t} = (t_1, \ldots, t_n)$.

Example (Sample from WikiText-2 [6] (English text))

Sample text:

Rifenburg lived 37 of his years in Buffalo . His wife , the former Jane Morris , was the head of the Buffalo Jills cheerleaders when they met . Rifenburg , who was survived by three sons , (Douglas

Motivation for Introduction.

In free-form generation, there is no **single correct sequence**, making it difficult to define a simple accuracy.

5.1 Perplexity

Motivation for Introduction.

In free-form generation, there is no **single correct sequence**, making it difficult to define a simple accuracy.

Therefore, we introduce **perplexity**, which evaluates how much **likelihood** the model assigns to the observed sequence, using a length-normalized average negative log-likelihood.

Definition (Rigorous Definition of Cross-Entropy and Perplexity)

Let the **vocabulary** set be $\mathcal{V}$, and a **token sequence** be $\boldsymbol{t} = (t_1, \ldots, t_n)$. The **language model** P provides a probability distribution over $\mathcal{V}$ for the next token, conditioned on previous tokens:

$$P(\cdot \mid t_{<i}) \text{ with } t_{<i} := (t_1, \ldots, t_{i-1}) \tag{3}$$

Let the base of the logarithm be e. Then, the **average negative log-likelihood** (token-level cross-entropy) is defined as

$$H(\boldsymbol{t}; P) := -\frac{1}{n} \sum_{i=1}^{n} \log P\big(t_i \mid t_{<i}\big), \tag{4}$$

and the **perplexity** is defined as

$$\mathrm{PPL}(\boldsymbol{t}; P) := \exp\big(H(\boldsymbol{t}; P)\big) \tag{5}$$

5.1 Perplexity

Remark

Note that differences in tokenization can lead to different results. This is also true for several other evaluation metrics.

Example (PPL Calculation Example (Rigorous Setting and Step-by-Step Calculation))

Formal Setting:

- Vocabulary $\mathcal{V} = \{a, b\}$
- Sequence $\boldsymbol{t} = (t_1, t_2, t_3) = (a, b, a)$
- Probabilities are given as:

$$P(a \mid ()) = 0.8$$
$$P(b \mid (a)) = 0.5$$
$$P(a \mid (a, b)) = 0.25$$

Here $n = 3$. Let's calculate the perplexity.

Calculation Steps:

First, we write down the formula for the average negative log-likelihood, $H(\boldsymbol{t}; P)$:

$$H(\boldsymbol{t}; P) = -\frac{1}{3}\Big(\log P(t_1{=}a \mid ()) + \log P(t_2{=}b \mid (a)) + \log P(t_3{=}a \mid (a, b))\Big)$$

Calculation Steps:

Next, we plug in the given probability values:

$$
\begin{aligned}
H(\boldsymbol{t}; P) &= -\frac{1}{3}\Big(\log P(t_1{=}a \mid ()) + \log P(t_2{=}b \mid (a)) + \log P(t_3{=}a \mid (a, b))\Big) \\
&= -\frac{1}{3}\big(\log 0.8 + \log 0.5 + \log 0.25\big)
\end{aligned}
$$

Calculation Steps:

Using the property that the sum of logs is the log of the product:

$$
\begin{aligned}
H(\boldsymbol{t}; P) &= -\frac{1}{3}\big(\log 0.8 + \log 0.5 + \log 0.25\big) \\
&= -\frac{1}{3}\log(0.8 \times 0.5 \times 0.25) = -\frac{1}{3}\log 0.1.
\end{aligned}
$$

This is the value for $H(\boldsymbol{t}; P)$.

5.1 Perplexity

Calculation Steps:

Finally, we compute the perplexity by taking the exponential of H:

$$\mathrm{PPL}(\boldsymbol{t}; P) = \exp\left(-\tfrac{1}{3}\log 0.1\right) = 0.1^{-1/3} \approx 2.154. \tag{6}$$

The perplexity is approximately 2.154.

Exercise (PPL Exercise)

For $\boldsymbol{t} = (a, a)$, with $P(a \mid ()) = 0.6$ and $P(a \mid (a)) = 0.3$, calculate the PPL step-by-step.

Answer

Step-by-step Calculation: The length of the sequence is $n = 2$.

$$
\begin{aligned}
H(\boldsymbol{t}; P) &= -\frac{1}{2}\Big(\log P(t_1{=}a \mid ()) + \log P(t_2{=}a \mid (a))\Big) \\
&= -\frac{1}{2}(\log 0.6 + \log 0.3) = -\frac{1}{2}\log(0.18).
\end{aligned}
$$

Therefore,

$$
\mathrm{PPL}(\boldsymbol{t}; P) = \exp(H) = (0.18)^{-1/2} \approx 2.357.
$$

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Input/Output Format. For each question j, a prompt (context) $\boldsymbol{c}_j$, a set of choices $\mathcal{A}_j = \{\boldsymbol{a}_{j,1}, \ldots, \boldsymbol{a}_{j,K_j}\}$, and a correct index $y_j \in [1, K_j]_{\mathbb{Z}}$ are given.

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Input/Output Format. For each question j, a prompt (context) $\boldsymbol{c}_j$, a set of choices $\mathcal{A}_j = \{\boldsymbol{a}_{j,1}, \ldots, \boldsymbol{a}_{j,K_j}\}$, and a correct index $y_j \in [1, K_j]_{\mathbb{Z}}$ are given.

The model P provides the **conditional likelihood** for each choice (e.g., the sum of token log-likelihoods).

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Example (HellaSwag [10] Data Example)

Choose the most plausible continuation for a given context. **Input (stem)**: [header] How to know what to expect on a newborn's skin [title] Note your newborn's skin tone. [step] At birth, a newborn's skin may be reddish or pinkish. However, the baby's hands and feet may be bluish... **Choices**:

- 0. As the newborn's circulatory system opens, this bluish color will subside...
- 1. This means they will not produce much oxygen...
- 2. Therefore, it's important to measure the temperature and tone...
- 3. A newborn's skin color is linked to their normal appearance...

Output (gold label index): 0

Example (ARC-Easy [1] Data Example)

Science question (multiple-choice)

Input (question): Which is the function of the gallbladder?
Input (choices): A: `store bile`, B: `produce bile`, C: `store digestive enzymes`, D: `produce digestive enzymes`
Output (gold label): A

Example (ARC-Challenge Data Example)

More difficult science question (multiple-choice)

Input (question): Cities control the amount of pollution... How does this most likely help people?
Input (choices): A: `The air stays cleaner.`, B: `Cars can travel at faster speeds.`, ...

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Example (MMLU (all) [3] Data Example)

Academic problems from various fields (multiple-choice)

Input (question): A state built a casino and issued bonds to finance its construction... **Output (gold index)**: 1

Example (TruthfulQA (MC1) Data [5] Example)

Multiple-choice questions to measure the tendency to imitate misinformation

Input (question): Can a husband and wife be arrested for the same crime in the U.S.?
Output (gold index): 0

Motivation for Introduction.

Since scoring free-form generation is difficult, framing the task as **selecting from choices** facilitates automatic evaluation.

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Motivation for Introduction.

Since scoring free-form generation is difficult, framing the task as **selecting from choices** facilitates automatic evaluation.

By explicitly defining a selection procedure based on the **probabilistic likelihood** that a language model assigns to each choice, we can perform an evaluation that does not depend on superficial features like string length.

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Definition (Rigorous Definition of Accuracy based on Most Likely Option)

For question j, given context $\boldsymbol{c}_j$ and choices $\{\boldsymbol{a}_{j,k}\}$, we tokenize each choice $\boldsymbol{a}_{j,k}$ into a sequence $(t_1^{(j,k)}, \ldots, t_{n_{j,k}}^{(j,k)})$.

Definition (Rigorous Definition of Accuracy based on Most Likely Option)

For question j, given context $\boldsymbol{c}_j$ and choices $\{\boldsymbol{a}_{j,k}\}$, we tokenize each choice $\boldsymbol{a}_{j,k}$ into a sequence $(t_1^{(j,k)}, \ldots, t_{n_{j,k}}^{(j,k)})$. The **log-likelihood sum score** for each choice is defined as:

$$S_{j,k} := \sum_{i=1}^{n_{j,k}} \log P\big(t_i^{(j,k)} \mid t_{<i}^{(j,k)}, \boldsymbol{c}_j\big) \tag{7}$$

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Definition (Rigorous Definition of Accuracy based on Most Likely Option)

For question j, given context $\boldsymbol{c}_j$ and choices $\{\boldsymbol{a}_{j,k}\}$, we tokenize each choice $\boldsymbol{a}_{j,k}$ into a sequence $(t_1^{(j,k)}, \ldots, t_{n_{j,k}}^{(j,k)})$. The **log-likelihood sum score** for each choice is defined as:

$$S_{j,k} \coloneqq \sum_{i=1}^{n_{j,k}} \log P\big(t_i^{(j,k)} \mid t_{<i}^{(j,k)}, \boldsymbol{c}_j\big) \tag{7}$$

The **predicted label** $\hat{y}_j$ is the choice with the maximum score:

$$\hat{y}_j \coloneqq \arg \max_{k \in [1, K_j]_{\mathbb{Z}}} S_{j,k} \tag{8}$$

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Definition (Rigorous Definition of Accuracy based on Most Likely Option)

For question j, given context $\boldsymbol{c}_j$ and choices $\{\boldsymbol{a}_{j,k}\}$, we tokenize each choice $\boldsymbol{a}_{j,k}$ into a sequence $(t_1^{(j,k)}, \ldots, t_{n_{j,k}}^{(j,k)})$. The **log-likelihood sum score** for each choice is defined as:

$$S_{j,k} \coloneqq \sum_{i=1}^{n_{j,k}} \log P\big(t_i^{(j,k)} \mid t_{<i}^{(j,k)}, \boldsymbol{c}_j\big) \tag{7}$$

The **predicted label** $\hat{y}_j$ is the choice with the maximum score:

$$\hat{y}_j \coloneqq \arg \max_{k \in [1, K_j]_{\mathbb{Z}}} S_{j,k} \tag{8}$$

The **accuracy** for N questions is the fraction of correct predictions:

$$\text{Accuracy} \coloneqq \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\big[\hat{y}_j = y_j\big] \tag{9}$$

Example (Log-Likelihood Selection in a 4-choice setting (Rigorous Setting))

Setting:

- For a single question ($N = 1$), we have $K_1 = 4$ choices.
- The log-likelihood sum scores for each choice are given as:

$$S = (S_{1,1}, S_{1,2}, S_{1,3}, S_{1,4}) = (-5.1,\ -4.2,\ -4.9,\ -6.0).$$

Example (Log-Likelihood Selection in a 4-choice setting (Rigorous Setting))

Setting:

- For a single question ($N = 1$), we have $K_1 = 4$ choices.
- The log-likelihood sum scores for each choice are given as:

$$S = (S_{1,1}, S_{1,2}, S_{1,3}, S_{1,4}) = (-5.1,\ -4.2,\ -4.9,\ -6.0).$$

Step-by-step Calculation:

- Find the index of the maximum score:

$$\hat{y}_1 = \arg\max\{-5.1,\ -4.2,\ -4.9,\ -6.0\} = 2.$$

Example (Log-Likelihood Selection in a 4-choice setting (Rigorous Setting))

Setting:

- For a single question ($N = 1$), we have $K_1 = 4$ choices.
- The log-likelihood sum scores for each choice are given as:

$$S = (S_{1,1}, S_{1,2}, S_{1,3}, S_{1,4}) = (-5.1,\ -4.2,\ -4.9,\ -6.0).$$

Step-by-step Calculation:

- Find the index of the maximum score:

$$\hat{y}_1 = \arg\max\{-5.1,\ -4.2,\ -4.9,\ -6.0\} = 2.$$

- Compare with the gold label y_1. If $y_1 = 2$, then the prediction is correct

Exercise (Prediction from Log-Likelihoods)

Given scores $S = (-10.0,\ -9.9,\ -10.5,\ -9.7)$, and the correct answer is the 4th option. Calculate the accuracy step-by-step (for this single question).

Answer

Step-by-step Calculation:

- First, we find the model's prediction by taking the argmax of the scores:

$$\hat{y}_1 = \arg\max\{-10.0,\ -9.9,\ -10.5,\ -9.7\} = 4.$$

Answer

Step-by-step Calculation:

- First, we find the model's prediction by taking the argmax of the scores:

$$\hat{y}_1 = \arg\max\{-10.0,\ -9.9,\ -10.5,\ -9.7\} = 4.$$

- This prediction, 4, matches the correct gold label, which is also $y_1 = 4$.

Answer

Step-by-step Calculation:

- First, we find the model's prediction by taking the argmax of the scores:

$$\hat{y}_1 = \arg\max\{-10.0,\ -9.9,\ -10.5,\ -9.7\} = 4.$$

- This prediction, 4, matches the correct gold label, which is also $y_1 = 4$.
- Therefore, the indicator function is $\mathbf{1}[\hat{y}_1 = y_1] = 1$.

5.2 Accuracy of the Most Likely Option in Multiple-Choice

Answer

Step-by-step Calculation:

- First, we find the model's prediction by taking the argmax of the scores:

$$\hat{y}_1 = \arg\max\{-10.0,\ -9.9,\ -10.5,\ -9.7\} = 4.$$

- This prediction, 4, matches the correct gold label, which is also $y_1 = 4$.
- Therefore, the indicator function is $\mathbf{1}[\hat{y}_1 = y_1] = 1$.
- Since it is a single question ($N = 1$), the accuracy is $\text{Accuracy} = \frac{1}{1} \times 1 = 1$.

Evaluation of Natural Language String Input/Output

Motivation. In situations where we want to evaluate **local word order and co-occurrence** (such as machine translation), it is necessary to measure the degree of **substring match** between a candidate sentence and a reference sentence.

6.1 Motivation, Rigorous Definition, and Intuition of n-grams

Motivation. In situations where we want to evaluate **local word order and co-occurrence** (such as machine translation), it is necessary to measure the degree of **substring match** between a candidate sentence and a reference sentence.

- Exact matching of the whole sentence is too strict.
- Matching only the sets of words ignores word order.

6.1 Motivation, Rigorous Definition, and Intuition of n-grams

Motivation. In situations where we want to evaluate **local word order and co-occurrence** (such as machine translation), it is necessary to measure the degree of **substring match** between a candidate sentence and a reference sentence.

- Exact matching of the whole sentence is too strict.
- Matching only the sets of words ignores word order.

Therefore, we use n**-grams** (contiguous token sequences of length n).

Definition (n-gram Expansion)

Fix a tokenizer tok. For a token sequence $(t_1, t_2, \ldots, t_m)$, the **expansion function to an n-gram sequence** G_n is defined as:

$$\mathrm{G}_n\big((t_1, \ldots, t_m)\big) := \big((t_1, \ldots, t_n),\ (t_2, \ldots, t_{n+1}),\ \ldots,\ (t_{m-n+1}, \ldots, t_m)\big) \tag{10}$$

(if $m < n$, it results in an empty sequence $()$).

Example (Concrete Example of n-gram Expansion)

Setting: The English sentence $s =$ "the quick brown fox jumps over the lazy dog" is tokenized as:

$$\mathsf{tok}(\boldsymbol{s}) = (\text{the},\ \text{quick},\ \text{brown},\ \text{fox},\ \text{jumps},\ \text{over},\ \text{the},\ \text{lazy},\ \text{dog})$$

The number of tokens is $m = 9$.

1-gram Expansion (Unigrams).

$$\mathrm{G}_1\big(\mathsf{tok}(\boldsymbol{s})\big) = \big((\text{the}),\ (\text{quick}),\ (\text{brown}),\ \ldots,\ (\text{dog})\big)$$

The length is $m = 9$. This is just the sequence of individual tokens.

2-gram Expansion (Bigrams).

$$\mathrm{G}_2\big(\mathsf{tok}(\boldsymbol{s})\big) = \big((\text{the}, \text{quick}),\ (\text{quick}, \text{brown}),\ (\text{brown}, \text{fox}),\ \ldots,\ (\text{lazy}, \text{dog})\big)$$

The length is $m - 1 = 8$.

3-gram Expansion (Trigrams).

$$\mathrm{G}_3\big(\mathsf{tok}(\boldsymbol{s})\big) = \big((\mathrm{the}, \mathrm{quick}, \mathrm{brown}),\ (\mathrm{quick}, \mathrm{brown}, \mathrm{fox}),\ \ldots,\ (\mathrm{the}, \mathrm{lazy}, \mathrm{dog})\big)$$

The length is $m - 2 = 7$.

To use n-grams for evaluation, we need to count their occurrences.

6.1 Motivation, Rigorous Definition, and Intuition of n-grams

To use n-grams for evaluation, we need to count their occurrences.

Definition (Histogram)

For any sequence $\boldsymbol{z} = (z_1, \ldots, z_L)$, the **occurrence count (histogram) function** is:

$$\mathsf{Hist}_{\boldsymbol{z}}(u) := \left|\left\{\, i \in [1, L]_{\mathbb{Z}} \;\middle|\; z_i = u \,\right\}\right|$$

6.1 Motivation, Rigorous Definition, and Intuition of n-grams

To use n-grams for evaluation, we need to count their occurrences.

Definition (Histogram)

For any sequence $\boldsymbol{z} = (z_1, \ldots, z_L)$, the **occurrence count (histogram) function** is:

$$\mathsf{Hist}_{\boldsymbol{z}}(u) := \left|\left\{\, i \in [1, L]_{\mathbb{Z}} \;\middle|\; z_i = u \,\right\}\right|$$

We also define the **element-wise minimum** $(f \wedge h)(u) := \min\{f(u), h(u)\}$ and **element-wise maximum** $\left(\bigvee_{r \in \mathcal{R}} f_r\right)(u) := \max_{r \in \mathcal{R}} f_r(u)$.

6.1 Motivation, Rigorous Definition, and Intuition of n-grams

To use n-grams for evaluation, we need to count their occurrences.

Definition (Histogram)

For any sequence $\boldsymbol{z} = (z_1, \ldots, z_L)$, the **occurrence count (histogram) function** is:

$$\mathsf{Hist}_{\boldsymbol{z}}(u) := \left|\{\, i \in [1, L]_{\mathbb{Z}} \;\middle|\; z_i = u \,\}\right|$$

We also define the **element-wise minimum** $(f \wedge h)(u) := \min\{f(u), h(u)\}$ and **element-wise maximum** $\left(\bigvee_{r \in \mathcal{R}} f_r\right)(u) := \max_{r \in \mathcal{R}} f_r(u)$.

The **1-norm** of a histogram function is the total count:

$$\|f\|_1 := \sum_u f(u)$$

Remark (Intuition)

- $n = 1$ reflects **frequency matching** of vocabulary.

Remark (Intuition)

- $n = 1$ reflects **frequency matching** of vocabulary.
- $n = 2$ reflects **local matching of word order**.

Remark (Intuition)

- $n = 1$ reflects **frequency matching** of vocabulary.
- $n = 2$ reflects **local matching of word order**.
- a large n reflects **matching of long phrases**.

Remark (Intuition)

- $n = 1$ reflects **frequency matching** of vocabulary.
- $n = 2$ reflects **local matching of word order**.
- a large n reflects **matching of long phrases**.

A weighted average that mixes different n values measures multiple levels of fidelity simultaneously.

Input/Output Format. Context C, question Q, set of correct short answers $\mathcal{Y} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(m)}\}$. The model's output is a text sequence $\hat{y}$.

Example (SQuAD v1.1 Data Example [9])

Context snippet: In cpDNA, there are several A $\rightarrow$ G deamination gradients. DNA becomes susceptible... **Question**: How does the secondary theory say most cpDNA is structured? **Gold answer text(s)**: [`linear`, `linear`, `linear`]

Motivation for Introduction.

In span extraction QA, even if the strings do not match perfectly, they are often **nearly identical at the word level**.

6.2 QA (SQuAD): Exact Match and Token-level F1

Motivation for Introduction.

In span extraction QA, even if the strings do not match perfectly, they are often **nearly identical at the word level**.

Simple EM (exact match) is too strict, so Token-level F1, which **gives points for partial matches**, is used in conjunction.

6.2 QA (SQuAD): Exact Match and Token-level F1

Definition (Rigorous Definition of EM and F1)

Fix a string normalization function norm. **Exact Match** (EM) is the maximum score over all references:

$$\mathrm{EM}(\hat{\boldsymbol{y}}, \mathcal{Y}) \coloneqq \max_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{1}\Big[\mathrm{norm}(\hat{\boldsymbol{y}}) = \mathrm{norm}(\boldsymbol{y})\Big]. \tag{11}$$

Definition (Rigorous Definition of EM and F1)

Fix a string normalization function norm. **Exact Match** (EM) is the maximum score over all references:

$$\mathrm{EM}(\hat{\boldsymbol{y}}, \mathcal{Y}) \coloneqq \max_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{1}\left[\mathrm{norm}(\hat{\boldsymbol{y}}) = \mathrm{norm}(\boldsymbol{y})\right]. \tag{11}$$

Using unigram ($n = 1$) histograms, we define token-level precision, recall, and F1 for a single reference $\boldsymbol{y}$:

$$\mathrm{Prec}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\left\| \mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})} \wedge \mathsf{Hist}_{\mathsf{tok}(\boldsymbol{y})} \right\|_1}{\left\| \mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})} \right\|_1}, \tag{12}$$

$$\mathrm{Rec}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\left\| \mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})} \wedge \mathsf{Hist}_{\mathsf{tok}(\boldsymbol{y})} \right\|_1}{\left\| \mathsf{Hist}_{\mathsf{tok}(\boldsymbol{y})} \right\|_1}, \tag{13}$$

$$\mathrm{F1}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{2\,\mathrm{Prec} \cdot \mathrm{Rec}}{\mathrm{Prec} + \mathrm{Rec}}. \tag{14}$$

6.2 QA (SQuAD): Exact Match and Token-level F1

Example (Manual Calculation of SQuAD-style F1)

Setting:

- Predicted answer: $\hat{y} =$ “the red apple”
- Reference answer: $y =$ “red apple”
- (Assume strings are already normalized, tokenizer is space-splitting)

6.2 QA (SQuAD): Exact Match and Token-level F1

Step 1: Unigram Histograms

- Prediction $\hat{\boldsymbol{y}}$ tokens: $(\mathrm{the}, \mathrm{red}, \mathrm{apple})$

$$\mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})} : \ \{\mathrm{the}, \mathrm{red}, \mathrm{apple}\} \mapsto \{1, 1, 1\}$$

Total predicted tokens: $\|\mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})}\|_1 = 3$.

6.2 QA (SQuAD): Exact Match and Token-level F1

Step 1: Unigram Histograms

- Prediction $\hat{\boldsymbol{y}}$ tokens: $(\text{the}, \text{red}, \text{apple})$

$$\mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})}: \ \{\text{the}, \text{red}, \text{apple}\} \mapsto \{1, 1, 1\}$$

  Total predicted tokens: $\|\mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})}\|_1 = 3$.

- Reference $\boldsymbol{y}$ tokens: $(\text{red}, \text{apple})$

$$\mathsf{Hist}_{\mathsf{tok}(\boldsymbol{y})}: \ \{\text{red}, \text{apple}\} \mapsto \{1, 1\}$$

  Total reference tokens: $\|\mathsf{Hist}_{\mathsf{tok}(\boldsymbol{y})}\|_1 = 2$.

6.2 QA (SQuAD): Exact Match and Token-level F1

Step 2: Element-wise Minimum (Common Tokens)

We find the minimum count for each token across both histograms.

- $\min(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(\text{red}),\ \mathsf{Hist}_{\boldsymbol{y}}(\text{red})) = \min(1, 1) = 1$
- $\min(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(\text{apple}),\ \mathsf{Hist}_{\boldsymbol{y}}(\text{apple})) = \min(1, 1) = 1$
- $\min(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(\text{the}),\ \mathsf{Hist}_{\boldsymbol{y}}(\text{the})) = \min(1, 0) = 0$

6.2 QA (SQuAD): Exact Match and Token-level F1

Step 2: Element-wise Minimum (Common Tokens)

We find the minimum count for each token across both histograms.

- $\min(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(\text{red}),\ \mathsf{Hist}_{\boldsymbol{y}}(\text{red})) = \min(1,1) = 1$
- $\min(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(\text{apple}),\ \mathsf{Hist}_{\boldsymbol{y}}(\text{apple})) = \min(1,1) = 1$
- $\min(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(\text{the}),\ \mathsf{Hist}_{\boldsymbol{y}}(\text{the})) = \min(1,0) = 0$

The total number of common tokens is the sum of these minimums:

$$\left\|\mathsf{Hist}_{\mathsf{tok}(\hat{\boldsymbol{y}})} \wedge \mathsf{Hist}_{\mathsf{tok}(\boldsymbol{y})}\right\|_1 = 1 + 1 + 0 = 2.$$

Step 3: Calculate Precision, Recall, and F1

- **Precision** = (Common Tokens) / (Total Predicted Tokens)

$$\mathrm{Prec} = \frac{2}{3}$$

6.2 QA (SQuAD): Exact Match and Token-level F1

Step 3: Calculate Precision, Recall, and F1

- **Precision** = (Common Tokens) / (Total Predicted Tokens)

$$\text{Prec} = \frac{2}{3}$$

- **Recall** = (Common Tokens) / (Total Reference Tokens)

$$\text{Rec} = \frac{2}{2} = 1$$

6.2 QA (SQuAD): Exact Match and Token-level F1

Step 3: Calculate Precision, Recall, and F1

- **Precision** = (Common Tokens) / (Total Predicted Tokens)

$$\text{Prec} = \frac{2}{3}$$

- **Recall** = (Common Tokens) / (Total Reference Tokens)

$$\text{Rec} = \frac{2}{2} = 1$$

- **F1 Score** = $2 \cdot \frac{\text{Prec}\cdot\text{Rec}}{\text{Prec}+\text{Rec}}$

$$\text{F1} = \frac{2 \cdot (2/3) \cdot 1}{(2/3) + 1} = \frac{4/3}{5/3} = 0.8.$$

6.2 QA (SQuAD): Exact Match and Token-level F1

Exercise (EM/F1 Exercise)

Let predicted answer $\hat{y} =$ “capital of France”, and one reference $y =$ “the capital of France”. Assume a normalization rule removes articles (like "the") and lowercases everything. Let the tokenizer be space-splitting. Calculate EM and F1 step-by-step.

6.2 QA (SQuAD): Exact Match and Token-level F1

Answer

Normalization and EM: After removing "the" and lowercasing, both strings become identical:

$$\mathrm{norm}(\hat{\boldsymbol{y}}) = \mathrm{norm}(\boldsymbol{y}) = \text{“capital of france”}$$

Therefore, $\mathrm{EM} = 1$.

6.2 QA (SQuAD): Exact Match and Token-level F1

Answer

Normalization and EM: After removing "the" and lowercasing, both strings become identical:

$$\mathrm{norm}(\hat{\boldsymbol{y}}) = \mathrm{norm}(\boldsymbol{y}) = \text{“capital of france”}$$

Therefore, $\mathrm{EM} = 1$.

F1 Calculation (on original tokens):

- Predicted tokens: 3 (‘capital‘, ‘of‘, ‘france‘).
- Reference tokens: 4 (‘the‘, ‘capital‘, ‘of‘, ‘france‘).
- Common tokens: 3 (‘capital‘, ‘of‘, ‘france‘).

Answer

Normalization and EM: After removing "the" and lowercasing, both strings become identical:

$$\mathrm{norm}(\hat{\boldsymbol{y}}) = \mathrm{norm}(\boldsymbol{y}) = \text{“capital of france”}$$

Therefore, $\mathrm{EM} = 1$.

F1 Calculation (on original tokens):

- Predicted tokens: 3 (‘capital‘, ‘of‘, ‘france‘).
- Reference tokens: 4 (‘the‘, ‘capital‘, ‘of‘, ‘france‘).
- Common tokens: 3 (‘capital‘, ‘of‘, ‘france‘).

- $\mathrm{Prec} = 3/3 = 1$
- $\mathrm{Rec} = 3/4 = 0.75$

Input/Output Format. Source sentence x, candidate translation $\hat{y}$, and a set of references $\mathcal{R} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(M)}\}$ [7].

Example (WMT14 (en→de) Data Example)

Input (English): It has always taken place.
Output (gold German): Das war schon immer so.

Motivation for Introduction.

In translation, there are **paraphrases** and **differences in word order**, making simple accuracy or EM unhelpful.

Motivation for Introduction.

In translation, there are **paraphrases** and **differences in word order**, making simple accuracy or EM unhelpful.

BLEU aggregates **n-gram precisions** at multiple levels and also penalizes **outputs that are too short** with a brevity penalty.

6.3 Machine Translation: BLEU

<u>Motivation for Introduction.</u>

In translation, there are **paraphrases** and **differences in word order**, making simple accuracy or EM unhelpful.

BLEU aggregates n**-gram precisions** at multiple levels and also penalizes **outputs that are too short** with a brevity penalty.

This measures both **fluency and adequacy**.

Definition (Rigorous Definition of BLEU-N)

The **clipped n-gram precision** p_n is the ratio of candidate n-grams that appear in any reference, where the count of each candidate n-gram is "clipped" by its maximum count in any single reference.

$$p_n = \frac{\sum_g \min\Big(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(g),\ \max_m \mathsf{Hist}_{\boldsymbol{y}^{(m)}}(g)\Big)}{\sum_g \mathsf{Hist}_{\hat{\boldsymbol{y}}}(g)} \tag{15}$$

Definition (Rigorous Definition of BLEU-N)

The **clipped n-gram precision** p_n is the ratio of candidate n-grams that appear in any reference, where the count of each candidate n-gram is "clipped" by its maximum count in any single reference.

$$p_n = \frac{\sum_g \min\Big(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(g),\ \max_m \mathsf{Hist}_{\boldsymbol{y}^{(m)}}(g)\Big)}{\sum_g \mathsf{Hist}_{\hat{\boldsymbol{y}}}(g)} \tag{15}$$

The **brevity penalty** BP is:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } |\hat{\boldsymbol{y}}| > r, \\ \exp(1 - r/|\hat{\boldsymbol{y}}|) & \text{if } |\hat{\boldsymbol{y}}| \leq r, \end{cases} \tag{16}$$

where r is the length of the closest reference.

Definition (Rigorous Definition of BLEU-N)

The **clipped n-gram precision** p_n is the ratio of candidate n-grams that appear in any reference, where the count of each candidate n-gram is "clipped" by its maximum count in any single reference.

$$p_n = \frac{\sum_g \min\Big(\mathsf{Hist}_{\hat{\boldsymbol{y}}}(g),\ \max_m \mathsf{Hist}_{\boldsymbol{y}^{(m)}}(g)\Big)}{\sum_g \mathsf{Hist}_{\hat{\boldsymbol{y}}}(g)} \tag{15}$$

The **brevity penalty** BP is:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } |\hat{\boldsymbol{y}}| > r, \\ \exp(1 - r/|\hat{\boldsymbol{y}}|) & \text{if } |\hat{\boldsymbol{y}}| \leq r, \end{cases} \tag{16}$$

where r is the length of the closest reference.

Example (Manual Calculation of BLEU-2 (1 reference, Rigorous Procedure))

Setting:

- Reference $\boldsymbol{y}$ = "the cat is on the mat" (length 6)
- Candidate $\hat{\boldsymbol{y}}$ = "the cat the cat on the mat" (length 7)
- We will calculate BLEU-2, with equal weights $w_1 = w_2 = \frac{1}{2}$.

6.3 Machine Translation: BLEU

Unigram Precision (p_1):

- Candidate unigrams: ‘the‘: 3, ‘cat‘: 2, ‘on‘: 1, ‘mat‘: 1. Total: 7.
- Reference unigrams: ‘the‘: 2, ‘cat‘: 1, ‘is‘: 1, ‘on‘: 1, ‘mat‘: 1.

6.3 Machine Translation: BLEU

Unigram Precision (p_1):

- Candidate unigrams: ‘the‘: 3, ‘cat‘: 2, ‘on‘: 1, ‘mat‘: 1. Total: 7.
- Reference unigrams: ‘the‘: 2, ‘cat‘: 1, ‘is‘: 1, ‘on‘: 1, ‘mat‘: 1.

Clipping:

- ‘the‘: candidate count is 3, max reference count is 2 $\implies$ clipped count is 2.
- ‘cat‘: candidate count is 2, max reference count is 1 $\implies$ clipped count is 1.
- ‘on‘: clipped count is 1.
- ‘mat‘: clipped count is 1.

6.3 Machine Translation: BLEU

Unigram Precision (p_1):

- Candidate unigrams: ‘the‘: 3, ‘cat‘: 2, ‘on‘: 1, ‘mat‘: 1. Total: 7.
- Reference unigrams: ‘the‘: 2, ‘cat‘: 1, ‘is‘: 1, ‘on‘: 1, ‘mat‘: 1.

Clipping:

- ‘the‘: candidate count is 3, max reference count is 2 $\implies$ clipped count is 2.
- ‘cat‘: candidate count is 2, max reference count is 1 $\implies$ clipped count is 1.
- ‘on‘: clipped count is 1.
- ‘mat‘: clipped count is 1.

Total clipped count = $2 + 1 + 1 + 1 = 5$.

$$p_1 = \frac{\text{Total Clipped Count}}{\text{Total Candidate Count}} = \frac{5}{7}.$$

Bigram Precision (p_2):

- Candidate bigrams: ‘the cat‘ (2), ‘cat the‘ (1), ‘cat on‘ (1), ‘on the‘ (1), ‘the mat‘ (1). Total: 6.
- Reference bigrams: ‘the cat‘ (1), ‘cat is‘ (1), ‘is on‘ (1), ‘on the‘ (1), ‘the mat‘ (1).

6.3 Machine Translation: BLEU

Bigram Precision (p_2):

- Candidate bigrams: ‘the cat‘ (2), ‘cat the‘ (1), ‘cat on‘ (1), ‘on the‘ (1), ‘the mat‘ (1). Total: 6.
- Reference bigrams: ‘the cat‘ (1), ‘cat is‘ (1), ‘is on‘ (1), ‘on the‘ (1), ‘the mat‘ (1).

Clipping:

- ‘the cat‘: candidate count 2, reference 1 $\implies$ clipped count 1.
- ‘on the‘: candidate count 1, reference 1 $\implies$ clipped count 1.
- ‘the mat‘: candidate count 1, reference 1 $\implies$ clipped count 1.
- Other candidate bigrams don’t appear in the reference, so their clipped count is 0.

6.3 Machine Translation: BLEU

Bigram Precision (p_2):

- Candidate bigrams: ‘the cat‘ (2), ‘cat the‘ (1), ‘cat on‘ (1), ‘on the‘ (1), ‘the mat‘ (1). Total: 6.
- Reference bigrams: ‘the cat‘ (1), ‘cat is‘ (1), ‘is on‘ (1), ‘on the‘ (1), ‘the mat‘ (1).

Clipping:

- ‘the cat‘: candidate count 2, reference 1 $\implies$ clipped count 1.
- ‘on the‘: candidate count 1, reference 1 $\implies$ clipped count 1.
- ‘the mat‘: candidate count 1, reference 1 $\implies$ clipped count 1.
- Other candidate bigrams don’t appear in the reference, so their clipped count is 0.

Total clipped count = $1 + 1 + 1 = 3$.

Brevity Penalty (BP) and Final Score:

- Candidate length $|\hat{\boldsymbol{y}}| = 7$.
- Reference length $r = |\boldsymbol{y}| = 6$.
- Since $|\hat{\boldsymbol{y}}| > r$, the brevity penalty is $\mathrm{BP} = 1$.

6.3 Machine Translation: BLEU

Brevity Penalty (BP) and Final Score:

- Candidate length $|\hat{\boldsymbol{y}}| = 7$.
- Reference length $r = |\boldsymbol{y}| = 6$.
- Since $|\hat{\boldsymbol{y}}| > r$, the brevity penalty is $\mathrm{BP} = 1$.

Final BLEU-2 Score:

$$\begin{aligned}\text{BLEU-2} &= \mathrm{BP} \cdot \exp(w_1 \log p_1 + w_2 \log p_2)\\ &= 1 \cdot \exp\big(\tfrac{1}{2}\log\tfrac{5}{7} + \tfrac{1}{2}\log\tfrac{1}{2}\big)\\ &= \sqrt{\frac{5}{7} \times \frac{1}{2}} = \sqrt{\frac{5}{14}} \approx 0.5976.\end{aligned}$$

Exercise (BLEU-1 Exercise)

Let reference y: “a b c d” (length 4), and candidate $\hat{y}$: “a c e” (length 3). Calculate BLEU-1 (which only uses unigrams) including the brevity penalty, step-by-step.

Answer

Unigram Precision (p_1):

- Candidate unigrams: $\{\mathrm{a}, \mathrm{c}, \mathrm{e}\}$. Total: 3.
- Reference unigrams: $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}$.
- Common unigrams are $\{\mathrm{a}, \mathrm{c}\}$. Total clipped count: 2.
- $p_1 = 2/3$.

Answer

Unigram Precision (p_1):

- Candidate unigrams: $\{\mathrm{a}, \mathrm{c}, \mathrm{e}\}$. Total: 3.
- Reference unigrams: $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}$.
- Common unigrams are $\{\mathrm{a}, \mathrm{c}\}$. Total clipped count: 2.
- $p_1 = 2/3$.

Brevity Penalty (BP):

- Candidate length $|\hat{\boldsymbol{y}}| = 3$. Reference length $r = 4$.
- Since $3 \leq 4$, $\mathrm{BP} = \exp(1 - 4/3) = \exp(-1/3)$.

Answer

Unigram Precision (p_1):

- Candidate unigrams: $\{\mathrm{a}, \mathrm{c}, \mathrm{e}\}$. Total: 3.
- Reference unigrams: $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}$.
- Common unigrams are $\{\mathrm{a}, \mathrm{c}\}$. Total clipped count: 2.
- $p_1 = 2/3$.

Brevity Penalty (BP):

- Candidate length $|\hat{\boldsymbol{y}}| = 3$. Reference length $r = 4$.
- Since $3 \leq 4$, $\mathrm{BP} = \exp(1 - 4/3) = \exp(-1/3)$.

Final Score:

- $\text{BLEU-1} = \mathrm{BP} \cdot p_1 = \exp(-1/3) \cdot \frac{2}{3} \approx 0.7165 \times 0.6667 \approx 0.478$.

6.4 Machine Translation: chrF

Input/Output Format. Candidate $\hat{\boldsymbol{y}}$, reference $\boldsymbol{y}$ (based on character n-grams) [8].

6.4 Machine Translation: chrF

Input/Output Format. Candidate $\hat{\boldsymbol{y}}$, reference $\boldsymbol{y}$ (based on character n-grams) [8].

Motivation for Introduction.

For morphologically rich languages or in situations with unstable tokenization, word-level n-grams are not robust.

6.4 Machine Translation: chrF

Input/Output Format. Candidate $\hat{y}$, reference $\boldsymbol{y}$ (based on character n-grams) [8].

Motivation for Introduction.

For morphologically rich languages or in situations with unstable tokenization, word-level n-grams are not robust.

chrF, based on **character n-grams**, is robust to **spelling differences and inflectional changes**, capturing fine-grained matches.

Definition (Rigorous Definition of chrF)

Based on character n-grams (for $n = 1, \ldots, N_c$), we define **micro-averaged precision and recall**:

$$\mathrm{Prec}_{\mathrm{chr}} = \frac{\sum_{n=1}^{N_c} (\text{common char } n\text{-grams})}{\sum_{n=1}^{N_c} (\text{candidate char } n\text{-grams})} \tag{18}$$

$$\mathrm{Rec}_{\mathrm{chr}} = \frac{\sum_{n=1}^{N_c} (\text{common char } n\text{-grams})}{\sum_{n=1}^{N_c} (\text{reference char } n\text{-grams})} \tag{19}$$

6.4 Machine Translation: chrF

Definition (Rigorous Definition of chrF)

Based on character n-grams (for $n = 1, \ldots, N_c$), we define **micro-averaged precision and recall**:

$$\mathrm{Prec}_{\mathrm{chr}} = \frac{\sum_{n=1}^{N_c}(\text{common char } n\text{-grams})}{\sum_{n=1}^{N_c}(\text{candidate char } n\text{-grams})} \tag{18}$$

$$\mathrm{Rec}_{\mathrm{chr}} = \frac{\sum_{n=1}^{N_c}(\text{common char } n\text{-grams})}{\sum_{n=1}^{N_c}(\text{reference char } n\text{-grams})} \tag{19}$$

Then, for a parameter $\beta > 0$, we define the F-score:

$$\mathrm{chrF}_{\beta} = \frac{(1+\beta^2)\,\mathrm{Prec}_{\mathrm{chr}} \cdot \mathrm{Rec}_{\mathrm{chr}}}{\mathrm{Rec}_{\mathrm{chr}} + \beta^2 \mathrm{Prec}_{\mathrm{chr}}} \tag{20}$$

Typically, $N_c = 6,\ \beta = 2$ are used (weighting recall more heavily).

Example (Complete Manual Calculation of chrF$_{\beta=2}$ ($N_c = 3$, Rigorous Procedure))

Setting:

- Reference: $\boldsymbol{y} =$ "color"
- Candidate: $\hat{\boldsymbol{y}} =$ "colour"
- We will use character n-grams up to $N_c = 3$, and $\beta = 2$.

6.4 Machine Translation: chrF

Step 1: Count n-grams and common n-grams

- $n = 1$:
  - $\boldsymbol{y}$: (c,o,l,o,r). Total 5.
  - $\hat{\boldsymbol{y}}$: (c,o,l,o,u,r). Total 6.
  - Common: (c,o,l,o,r). Total 5.
- $n = 2$:
  - $\boldsymbol{y}$: {co,ol,lo,or}. Total 4.
  - $\hat{\boldsymbol{y}}$: {co,ol,lo,ou,ur}. Total 5.
  - Common: {co,ol,lo}. Total 3.
- $n = 3$:
  - $\boldsymbol{y}$: {col,olo,lor}. Total 3.
  - $\hat{\boldsymbol{y}}$: {col,olo,lou,our}. Total 4.
  - Common: {col,olo}. Total 2.

Step 2: Micro-average to get Precision and Recall

- Total common n-grams = $5 + 3 + 2 = 10$.
- Total candidate n-grams = $6 + 5 + 4 = 15$.
- Total reference n-grams = $5 + 4 + 3 = 12$.

Step 2: Micro-average to get Precision and Recall

- Total common n-grams = $5 + 3 + 2 = 10$.
- Total candidate n-grams = $6 + 5 + 4 = 15$.
- Total reference n-grams = $5 + 4 + 3 = 12$.

$$\mathrm{Prec}_{\mathrm{chr}} = \frac{\text{Total common}}{\text{Total candidate}} = \frac{10}{15} = \frac{2}{3}$$
$$\mathrm{Rec}_{\mathrm{chr}} = \frac{\text{Total common}}{\text{Total reference}} = \frac{10}{12} = \frac{5}{6}$$

Step 3: Calculate the Final F-score

Using $\beta = 2$, $\mathrm{Prec}_{\mathrm{chr}} = 2/3$, and $\mathrm{Rec}_{\mathrm{chr}} = 5/6$:

$$
\begin{aligned}
\mathrm{chrF}_{\beta=2} &= \frac{(1+2^2)\cdot \mathrm{Prec}\cdot \mathrm{Rec}}{\mathrm{Rec} + 2^2 \cdot \mathrm{Prec}} \\
&= \frac{(1+4)\cdot(2/3)\cdot(5/6)}{(5/6) + 4\cdot(2/3)} \\
&= \frac{5\cdot(10/18)}{(5/6)+(8/3)} = \frac{50/18}{21/6} \\
&= \frac{25/9}{7/2} = \frac{50}{63} \approx 0.794.
\end{aligned}
$$

Exercise (chrF$_{\beta=2}$ Exercise ($N_c = 3$))

Let reference $\boldsymbol{y} =$ “center” and candidate $\hat{\boldsymbol{y}} =$ “centre”. Using $N_c = 3$ and $\beta = 2$, calculate chrF$_{\beta=2}$ step-by-step.

Answer

n**-gram Counts**:

- $n = 1$: common 6, candidate 6, reference 6.
- $n = 2$: common 3, candidate 5, reference 5.
- $n = 3$: common 2, candidate 4, reference 4.

Answer

n**-gram Counts**:

- $n = 1$: common 6, candidate 6, reference 6.
- $n = 2$: common 3, candidate 5, reference 5.
- $n = 3$: common 2, candidate 4, reference 4.

Micro-averaging:

- Total common $= 6 + 3 + 2 = 11$.
- Total candidate $= 6 + 5 + 4 = 15$.
- Total reference $= 6 + 5 + 4 = 15$.
- $\text{Prec}_{\text{chr}} = \text{Rec}_{\text{chr}} = 11/15$.

Answer

n**-gram Counts**:

- $n = 1$: common 6, candidate 6, reference 6.
- $n = 2$: common 3, candidate 5, reference 5.
- $n = 3$: common 2, candidate 4, reference 4.

Micro-averaging:

- Total common $= 6 + 3 + 2 = 11$.
- Total candidate $= 6 + 5 + 4 = 15$.
- Total reference $= 6 + 5 + 4 = 15$.
- $\mathrm{Prec}_{\mathrm{chr}} = \mathrm{Rec}_{\mathrm{chr}} = 11/15$.

Final Score: When $\mathrm{Prec} = \mathrm{Rec}$, the F-score is equal to precision and recall.

6.5 Lexical Semantic Similarity: BERTScore

Input/Output Format. Candidate $\hat{y}$ and reference y are tokenized, and each token is mapped to a vector embedding from a pre-trained language model [11].

6.5 Lexical Semantic Similarity: BERTScore

Input/Output Format. Candidate $\hat{y}$ and reference $\boldsymbol{y}$ are tokenized, and each token is mapped to a vector embedding from a pre-trained language model [11].

Motivation for Introduction.

n-gram based methods cannot sufficiently capture **synonyms and paraphrases**.

Input/Output Format. Candidate $\hat{y}$ and reference y are tokenized, and each token is mapped to a vector embedding from a pre-trained language model [11].

Motivation for Introduction.

n-gram based methods cannot sufficiently capture **synonyms and paraphrases**.

BERTScore measures the semantic consistency between the candidate and reference using **similarity in a continuous vector space**, enabling an evaluation that does not depend on superficial lexical matching.

6.5 Lexical Semantic Similarity: BERTScore

However, when considering the average similarity, the behavior of frequent but semantically unimportant words can become too dominant.

6.5 Lexical Semantic Similarity: BERTScore

However, when considering the average similarity, the behavior of frequent but semantically unimportant words can become too dominant.

IDF weighting emphasizes information-rich words and relatively reduces the contribution of **common words**.

6.5 Lexical Semantic Similarity: BERTScore

However, when considering the average similarity, the behavior of frequent but semantically unimportant words can become too dominant.

IDF weighting emphasizes information-rich words and relatively reduces the contribution of **common words**.

Definition (Inverse Document Frequency (IDF))

Let $\mathcal{D}$ be a collection of documents. The **inverse document frequency** for a token u is:

$$\mathrm{idf}_{\mathcal{D}}(u) := \log\left(\frac{|\mathcal{D}| + 1}{\mathrm{df}_{\mathcal{D}}(u) + 1}\right)$$

where $\mathrm{df}_{\mathcal{D}}(u)$ is the number of documents in $\mathcal{D}$ containing token u.

Definition (Complete Rigorous Definition of BERTScore (F_1))

Let the token embeddings for the candidate be $\{\tilde{\boldsymbol{h}}_i\}$ and for the reference be $\{\tilde{\boldsymbol{g}}_j\}$ (normalized to unit length).

6.5 Lexical Semantic Similarity: BERTScore

Definition (Complete Rigorous Definition of BERTScore (F_1))

Let the token embeddings for the candidate be $\{\tilde{\boldsymbol{h}}_i\}$ and for the reference be $\{\tilde{\boldsymbol{g}}_j\}$ (normalized to unit length). The similarity matrix is $s_{i,j} := \tilde{\boldsymbol{h}}_i^\top \tilde{\boldsymbol{g}}_j$.

6.5 Lexical Semantic Similarity: BERTScore

Definition (Complete Rigorous Definition of BERTScore (F_1))

Let the token embeddings for the candidate be $\{\tilde{\boldsymbol{h}}_i\}$ and for the reference be $\{\tilde{\boldsymbol{g}}_j\}$ (normalized to unit length). The similarity matrix is $s_{i,j} := \tilde{\boldsymbol{h}}_i^\top \tilde{\boldsymbol{g}}_j$. The precision and recall are calculated as IDF-weighted sums of maximal similarities:

$$\mathrm{Prec}_{\mathrm{BERT}} = \frac{\sum_i \mathrm{idf}(\hat{w}_i) \cdot \max_j s_{i,j}}{\sum_i \mathrm{idf}(\hat{w}_i)} \tag{21}$$

$$\mathrm{Rec}_{\mathrm{BERT}} = \frac{\sum_j \mathrm{idf}(w_j) \cdot \max_i s_{i,j}}{\sum_j \mathrm{idf}(w_j)} \tag{22}$$

Definition (Complete Rigorous Definition of BERTScore (F_1))

Let the token embeddings for the candidate be $\{\tilde{\boldsymbol{h}}_i\}$ and for the reference be $\{\tilde{\boldsymbol{g}}_j\}$ (normalized to unit length). The similarity matrix is $s_{i,j} := \tilde{\boldsymbol{h}}_i^\top \tilde{\boldsymbol{g}}_j$. The precision and recall are calculated as IDF-weighted sums of maximal similarities:

$$\mathrm{Prec}_{\mathrm{BERT}} = \frac{\sum_i \mathrm{idf}(\hat{w}_i) \cdot \max_j s_{i,j}}{\sum_i \mathrm{idf}(\hat{w}_i)} \tag{21}$$

$$\mathrm{Rec}_{\mathrm{BERT}} = \frac{\sum_j \mathrm{idf}(w_j) \cdot \max_i s_{i,j}}{\sum_j \mathrm{idf}(w_j)} \tag{22}$$

$\mathbf{F}_1$ Aggregation: $\mathrm{F1}_{\mathrm{BERT}} = \dfrac{2\,\mathrm{Prec}_{\mathrm{BERT}} \cdot \mathrm{Rec}_{\mathrm{BERT}}}{\mathrm{Prec}_{\mathrm{BERT}} + \mathrm{Rec}_{\mathrm{BERT}}}$.

Example (Complete Manual Calculation of BERTScore (F_1))

Setting:

- Reference $\boldsymbol{y}$ = "red apple"
- Candidate $\hat{\boldsymbol{y}}$ = "the red apples"
- IDF is disabled for simplicity ($u_i = v_j \equiv 1$).
- Word embeddings (already ℓ_2-normalized) are given as:

$$\begin{aligned} \text{the} &= (0, 0, 1) \\ \text{red} &= (1, 0, 0) \\ \text{apple} &= (0, 1, 0) \\ \text{apples} &= (0, 0.8, 0.6) \end{aligned}$$

6.5 Lexical Semantic Similarity: BERTScore

Step 1: Similarity Matrix $s_{i,j}$ (rows: candidate, columns: reference):

The similarity is the dot product of the normalized vectors.

	red $(1, 0, 0)$	apple $(0, 1, 0)$
the $(0, 0, 1)$	0	0
red $(1, 0, 0)$	1	0
apples $(0, 0.8, 0.6)$	0	0.8

Step 2: Precision

For each candidate token (row), find its maximum similarity with any reference token (column). Then average these maximums.

	red	apple	$\max_j s_{i,j}$
the	0	0	0
red	1	0	1
apples	0	0.8	0.8

$$\mathrm{Prec}_{\mathrm{BERT}} = \frac{0 + 1 + 0.8}{3} = 0.6.$$

Step 3: Recall

For each reference token (column), find its maximum similarity with any candidate token (row). Then average these maximums.

	red	apple
the	0	0
red	1	0
apples	0	0.8
$\max_i s_{i,j}$	1	0.8

$$\mathrm{Rec}_{\mathrm{BERT}} = \frac{1+0.8}{2} = 0.9.$$

Step 4: F$_1$ Score

Now we calculate the harmonic mean of Precision and Recall.

$$\mathrm{F1}_{\mathrm{BERT}} = \frac{2 \cdot \mathrm{Prec} \cdot \mathrm{Rec}}{\mathrm{Prec} + \mathrm{Rec}} = \frac{2 \cdot 0.6 \cdot 0.9}{0.6 + 0.9} = \frac{1.08}{1.5} = 0.72.$$

Exercise (BERTScore (F_1) Exercise)

Given word embeddings (normalized) as:

$$\begin{aligned} \text{fast} &= (1, 0, 0) \\ \text{car} &= (0, 1, 0) \\ \text{quick} &= (0.9, 0.1, 0) \\ \text{automobile} &= (0, 1/\sqrt{2}, 1/\sqrt{2}) \end{aligned}$$

Let reference $\boldsymbol{y} =$ “fast car” and candidate $\hat{\boldsymbol{y}} =$ “quick automobile”. IDF is disabled. Calculate BERTScore (F_1) step-by-step.

Answer

Similarity Matrix $s_{i,j}$:

	fast	car
quick	0.9	0.1
automobile	0	$1/\sqrt{2} \approx 0.707$

Answer

Similarity Matrix $s_{i,j}$:

	fast	car
quick	0.9	0.1
automobile	0	$1/\sqrt{2} \approx 0.707$

Precision (avg of row maxes):

$$\mathrm{Prec} = \frac{\max(0.9, 0.1) + \max(0, 1/\sqrt{2})}{2} = \frac{0.9 + 1/\sqrt{2}}{2} \approx 0.8036.$$

Answer

Similarity Matrix $s_{i,j}$:

	fast	car
quick	0.9	0.1
automobile	0	$1/\sqrt{2} \approx 0.707$

Precision (avg of row maxes):

$$\text{Prec} = \frac{\max(0.9, 0.1) + \max(0, 1/\sqrt{2})}{2} = \frac{0.9 + 1/\sqrt{2}}{2} \approx 0.8036.$$

Recall (avg of col maxes):

$$\text{Rec} = \frac{\max(0.9, 0) + \max(0.1, 1/\sqrt{2})}{2} = \frac{0.9 + 1/\sqrt{2}}{2} \approx 0.8036.$$

Answer

Similarity Matrix $s_{i,j}$:

	fast	car
quick	0.9	0.1
automobile	0	$1/\sqrt{2} \approx 0.707$

Precision (avg of row maxes):

$$\text{Prec} = \frac{\max(0.9, 0.1) + \max(0, 1/\sqrt{2})}{2} = \frac{0.9 + 1/\sqrt{2}}{2} \approx 0.8036.$$

Recall (avg of col maxes):

$$\text{Rec} = \frac{\max(0.9, 0) + \max(0.1, 1/\sqrt{2})}{2} = \frac{0.9 + 1/\sqrt{2}}{2} \approx 0.8036.$$

Input/Output Format. Candidate summary $\hat{y}$, reference y [4].

Example (XSum Summarization Data Example)

Input (article snippet): The 29-year-old committed two fouls but jumped 7.90m with his last effort to go through as the 10th of 12 qualifiers... **Output (gold summary)**: Great Britain's Greg Rutherford sneaked into Saturday's long jump final to maintain his hopes of defending his Olympic crown.

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Motivation for Introduction.

In summarization, both **content selection** and **word order preservation** are important.

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Motivation for Introduction.

In summarization, both **content selection** and **word order preservation** are important.

ROUGE-L, based on the **Longest Common Subsequence (LCS)**, evaluates consistency that cannot be measured solely by the number of overlapping words, while gently respecting word order.

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Definition (Rigorous Definition of ROUGE-L F$_1$)

Let $\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ be the length of the **longest common subsequence** of the tokens in $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$.

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Definition (Rigorous Definition of ROUGE-L F_1)

Let $\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ be the length of the **longest common subsequence** of the tokens in $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$. Then, we define precision and recall based on this length:

$$\mathrm{Prec} = \frac{\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})}{|\hat{\boldsymbol{y}}|} \quad \text{(num tokens in candidate)} \tag{23}$$

$$\mathrm{Rec} = \frac{\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})}{|\boldsymbol{y}|} \quad \text{(num tokens in reference)} \tag{24}$$

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Definition (Rigorous Definition of ROUGE-L F$_1$)

Let $\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ be the length of the **longest common subsequence** of the tokens in $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$. Then, we define precision and recall based on this length:

$$\mathrm{Prec} = \frac{\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})}{|\hat{\boldsymbol{y}}|} \quad \text{(num tokens in candidate)} \tag{23}$$

$$\mathrm{Rec} = \frac{\mathrm{LCS}(\hat{\boldsymbol{y}}, \boldsymbol{y})}{|\boldsymbol{y}|} \quad \text{(num tokens in reference)} \tag{24}$$

The ROUGE-L score is the F-score of these values (typically with $\beta = 1$):

$$\text{ROUGE-L} = \frac{(1 + \beta^2)\,\mathrm{Prec} \cdot \mathrm{Rec}}{\mathrm{Rec} + \beta^2 \mathrm{Prec}} \tag{25}$$

Example (Step-by-Step Calculation of ROUGE-L)

Setting:

- Reference $y =$ "the quick brown fox" (4 tokens)
- Candidate $\hat{y}$="quick brown fox" (3 tokens)

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Example (Step-by-Step Calculation of ROUGE-L)

Setting:

- Reference $y =$ “the quick brown fox” (4 tokens)
- Candidate $\hat{y}$=“quick brown fox” (3 tokens)

LCS: The longest common subsequence is “quick brown fox”. Its length is $\mathrm{LCS} = 3$.

Example (Step-by-Step Calculation of ROUGE-L)

Setting:

- Reference $\boldsymbol{y}$ = "the quick brown fox" (4 tokens)
- Candidate $\hat{\boldsymbol{y}}$="quick brown fox" (3 tokens)

LCS: The longest common subsequence is "quick brown fox". Its length is $\mathrm{LCS} = 3$.

Calculation ($\beta = 1$):

$$\mathrm{Prec} = \frac{3}{3} = 1$$
$$\mathrm{Rec} = \frac{3}{4} = 0.75$$
$$\mathrm{F}_1 = \frac{2 \cdot 1 \cdot 0.75}{1 + 0.75} = \frac{1.5}{1.75} \approx 0.857.$$

Exercise (LCS Exercise)

Let reference $\boldsymbol{y} =$ “a b c d” and candidate $\hat{\boldsymbol{y}} =$ “a c d”. Calculate ROUGE-L (F_1) step-by-step.

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Answer

LCS:

- The longest common subsequence is “a c d”.
- Its length is $\mathrm{LCS} = 3$.

Answer

LCS:

- The longest common subsequence is “a c d”.
- Its length is $\mathrm{LCS} = 3$.

Lengths:

- Candidate length $|\hat{\boldsymbol{y}}| = 3$.
- Reference length $|\boldsymbol{y}| = 4$.

6.6 Summarization: ROUGE-L (Longest Common Subsequence)

Answer

LCS:

- The longest common subsequence is “a c d”.
- Its length is $\mathrm{LCS} = 3$.

Lengths:

- Candidate length $|\hat{\boldsymbol{y}}| = 3$.
- Reference length $|\boldsymbol{y}| = 4$.

Metric Calculation ($\beta = 1$):

$$\mathrm{Prec} = 3/3 = 1$$

$$\mathrm{Rec} = 3/4 = 0.75$$

$$F = \frac{2 \cdot 1 \cdot 0.75}{} = \frac{1.5}{} \approx 0.857$$

6.7 Math QA (GSM8K): Numeric Accuracy

Input/Output Format. A natural language problem statement q and a **numerical correct answer** $y \in \mathbb{R}$ (or a stringified number) [2].

Example (GSM8K Data Example)

Question: Jared is trying to increase his typing speed. He starts with 47 words per minute... what will be the average of the three measurements?

Gold numeric answer: 52

6.7 Math QA (GSM8K): Numeric Accuracy

Motivation for Introduction.

In numerical response tasks, there are variations in notation (digit separators, decimal points, units).

6.7 Math QA (GSM8K): Numeric Accuracy

Motivation for Introduction.

In numerical response tasks, there are variations in notation (digit separators, decimal points, units).

Instead of **string matching**, we determine if they are **numerically equivalent** by defining **numerical normalization** before calculating accuracy.

Definition (Rigorous Definition of Numeric Accuracy)

Fix a **numeric normalization** function $\mathrm{num} : \mathsf{Str} \cup \mathbb{R} \to \mathbb{R}$.

6.7 Math QA (GSM8K): Numeric Accuracy

Definition (Rigorous Definition of Numeric Accuracy)

Fix a **numeric normalization** function $\mathrm{num} : \text{Str} \cup \mathbb{R} \to \mathbb{R}$. This function converts various string representations of numbers (including fractions, percentages, units, etc.) into a canonical real number format.

Definition (Rigorous Definition of Numeric Accuracy)

Fix a **numeric normalization** function $\mathrm{num} : \mathsf{Str} \cup \mathbb{R} \to \mathbb{R}$. This function converts various string representations of numbers (including fractions, percentages, units, etc.) into a canonical real number format. For N questions, the accuracy is simply the fraction of questions where the normalized prediction matches the normalized correct answer:

$$\mathrm{Accuracy} := \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\left[\mathrm{num}(\hat{y}_j) = \mathrm{num}(y_j)\right] \tag{26}$$

6.7 Math QA (GSM8K): Numeric Accuracy

Example (Normalization without decimals or units)

Setting:

- Gold answer: $y = 3.5$
- Predicted answer: $\hat{y} =$ “3.5000”

6.7 Math QA (GSM8K): Numeric Accuracy

Example (Normalization without decimals or units)

Setting:

- Gold answer: $y = 3.5$
- Predicted answer: $\hat{y} =$ "3.5000"

Normalization:

- $\mathrm{num}(3.5) = 3.5$
- $\mathrm{num}($"3.5000"$) = 3.5$ (e.g., remove trailing zeros and convert to float)

6.7 Math QA (GSM8K): Numeric Accuracy

Example (Normalization without decimals or units)

Setting:

- Gold answer: $y = 3.5$
- Predicted answer: $\hat{y} =$ "3.5000"

Normalization:

- $\mathrm{num}(3.5) = 3.5$
- $\mathrm{num}($"3.5000"$) = 3.5$ (e.g., remove trailing zeros and convert to float)

Judgment: Since $3.5 = 3.5$, it is a correct answer.

6.7 Math QA (GSM8K): Numeric Accuracy

Exercise (Unit Normalization Exercise)

Let the gold answer be $y = 100$ (meters), and the predicted answer be $\hat{y} =$ “0.1 km”. Assume the normalization function num converts everything to SI base units (meters). Show the accuracy judgment with a step-by-step calculation.

6.7 Math QA (GSM8K): Numeric Accuracy

Answer

Normalization:

- The gold answer is already in the base unit: $\mathrm{num}(y) = 100$.
- The predicted answer needs conversion: $\mathrm{num}(\hat{y}) = \mathrm{num}(\text{“0.1 km”}) = 0.1 \times 1000 = 100$.

Answer

Normalization:

- The gold answer is already in the base unit: $\mathrm{num}(y) = 100$.
- The predicted answer needs conversion: $\mathrm{num}(\hat{y}) = \mathrm{num}(\text{"0.1 km"}) = 0.1 \times 1000 = 100$.

Judgment:

- We compare the normalized real numbers: $100 = 100$.
- The equality holds, so $\mathbf{1}[\mathrm{num}(\hat{y}) = \mathrm{num}(y)] = 1$.
- The accuracy for this single question is 1.

Summary

7. Summary

Let's summarize the key points corresponding to the learning objectives of this lecture.

- **Non-triviality**: Natural language output has infinitely many semantically equivalent solutions, making the application of classical evaluation methods difficult.

7. Summary

Let's summarize the key points corresponding to the learning objectives of this lecture.

- **Non-triviality**: Natural language output has infinitely many semantically equivalent solutions, making the application of classical evaluation methods difficult.
- **Distinction**: We clearly distinguished between the **evaluation of the language model itself** (PPL, most likely option) and the **evaluation of the string output** (EM/F1, BLEU, ROUGE-L, chrF, BERTScore, Numeric Accuracy).

7. Summary

Let's summarize the key points corresponding to the learning objectives of this lecture.

- **Non-triviality**: Natural language output has infinitely many semantically equivalent solutions, making the application of classical evaluation methods difficult.
- **Distinction**: We clearly distinguished between the **evaluation of the language model itself** (PPL, most likely option) and the **evaluation of the string output** (EM/F1, BLEU, ROUGE-L, chrF, BERTScore, Numeric Accuracy).
- **Application**: We rigorously defined each metric and understood the calculation procedures through concrete examples and exercises.

Next Time

8. Next Time

In this lecture, we dealt with **absolute** evaluation metrics that target a single probabilistic language model.

8. Next Time

In this lecture, we dealt with **absolute** evaluation metrics that target a single probabilistic language model.

In the next lecture, we will address **relative** evaluation metrics that quantify the **deviation** from a given **reference probabilistic language model** (e.g., distances and divergences between distributions).

[1] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord.
Think you have solved question answering? try arc, the ai2 reasoning challenge.
In Proc. of NAACL-HLT, 2018.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman.
Training verifiers to solve math word problems.
In arXiv preprint arXiv:2110.14168, 2021.
GSM8K dataset.

[3] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt.
Measuring massive multitask language understanding.
In International Conference on Learning Representations (ICLR), 2021.

[4] Chin-Yew Lin.
Rouge: A package for automatic evaluation of summaries.
In Proc. of Workshop on Text Summarization Branches Out, 2004.

[5] Stephanie Lin, Jacob Hilton, and Owain Evans.
Truthfulqa: Measuring how models mimic human falsehoods.
In Proc. of ACL, 2022.

References iii

[6] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher.
Pointer sentinel mixture models.
In International Conference on Learning Representations (ICLR) Workshop, 2017.
Introduces WikiText datasets (WikiText-2/WikiText-103).

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu.
Bleu: a method for automatic evaluation of machine translation.
In Proc. of ACL, 2002.

[8] Maja Popović.
chrf: Character n-gram f-score for automatic mt evaluation.
In Proc. of WMT, 2015.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang.
Squad: 100,000+ questions for machine comprehension of text.
In Proc. of EMNLP, 2016.

[10] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi.
Hellaswag: Can a machine really finish your sentence?
In Proc. of ACL, 2019.

[11] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi.
Bertscore: Evaluating text generation with bert.
In Proc. of ICLR, 2020.