Translation Model and Summarization Model

March 9, 2025

Jou-Yi Lee
jlee1039@ucsc.edu
University of California, Santa Cruz

Abstract

This report presents an analysis of both a translation and a summarization model. For the translation model, we examine the published accuracy metrics (BLEU and chr-F) and detail the corresponding evaluation code. For the summarization model (PEGASUS-XSum), we analyze its performance based on ROUGE metrics and explain how the evaluation and self-confidence scores are computed. We further provide an example of an "obvious" incorrect summarization result, highlighting the model's shortcomings when processing complex inputs.

1 Translation Model Accuracy

`Helsinki-NLP/opus-mt-zh-en` From Hugging Face:

Published Accuracy

Testset	BLEU	chr-F
Tatoeba-test.zho.eng	36.1	0.548

Table 1: Published accuracy of the model on the Tatoeba-test.zho.eng dataset.

1.0.1 Explanation of Published Accuracy Metric Computation

The published accuracy metric for the translation model is computed using the `sacreBLEU` tool, which calculates both the BLEU and chr-F scores. The following sections describe the key code segments from the Makefile and explain how they work. :contentReference[oaicite:0]index=0

1.0.2 Preparing the Reference File

The reference translation file is prepared with the following command:

```
${ZCAT} ${patsubst %.${SRC}.gz \
,%.${TRG}.gz,${TESTSET}} > $@.ref
```

This command does the following:

- **`patsubst`**: Replaces the source language file extension (.en.gz) with the target language extension (.de.gz) in the test set file name.
- The output is redirected to a temporary file (`$@.ref`) which will be used as the reference translation during evaluation.

1.0.3 Computing the BLEU Score

The BLEU score is computed by piping the translation output into `sacreBLEU`:

```
cat $< | sacrebleu $@.ref > $@
```

1.0.4 Computing the chr-F Score

Following the BLEU score computation, the Makefile calculates the chr-F score with:

```
cat $<|sacrebleu --metrics=chrf \
--width=3 $@.ref >> $@
```

1.0.5 Cleaning Up

After the metrics are computed, the temporary reference file is removed:

```
rm -f $@.ref
```

Summary

In summary, the published accuracy metric is computed via the following steps:

1. A reference file is generated by decompressing and renaming the target language test set file.

2. The model's translation output is compared to the reference using `sacreBLEU` to compute the BLEU score.

3. The same output is used to compute the chr-F score, with the appropriate parameters, and both results are consolidated.

4. Temporary files are removed after computation.

1.1 Self-Confidence:

The self-confidence score is computed within the MarianNMT decoder during beam search. The decoder (configured via `decoder.yml`) accumulates the token probabilities (or log-probabilities) for each candidate translation.

1.2 Formula for Self-Confidence:

The overall confidence score is given by:

$$S = \prod_{i=1}^{n} p(w_i)$$

In practice, the log-probabilities are summed to avoid numerical underflow:

$$\log S = \sum_{i=1}^{n} \log p(w_i)$$

Optionally, a length normalization may be applied:

$$\text{Average Log-Prob} = \frac{1}{n} \sum_{i=1}^{n} \log p(w_i)$$

1.3 Obvious incorrect result:

Input: 画蛇添足
Expected Translation: `"to overdo something" or "to add unnecessary details"`
Model Output: `"Draw the snake's feet."`
Average Token Confidence: 0.24107

Analysis:

The idiom 画蛇添足 is commonly used to indicate that adding extra, unnecessary elements ruins an otherwise good result. An average person would interpret it idiomatically rather than literally. However, the model returns a literal translation, which is clearly incorrect. Despite this error, the low average token confidence (0.24107) indicates that the model is not very sure about its output.

Rationale:

This instance was selected because idioms often defy literal translation. The simplifying assumption made by the model is that each word's translation can be combined to form the overall meaning. In this case, that assumption fails for idiomatic expressions, leading to an "obvious" error.

2 Summarization Model

2.1 Published Accuracy

For a summarization model `google/pegasus-xsum`, the published accuracy on the target task is typically reported using ROUGE metrics. For example, on the XSum dataset the self-reported scores are:

- **ROUGE-1:** 46.862
- **ROUGE-2:** 24.453
- **ROUGE-L:** 39.055

2.2 Identification of the Evaluation Code

In the PEGASUS repository, the published accuracy metrics for summarization (e.g., ROUGE scores) are computed in the file `text_eval.py`. This file is responsible for reading the model ' s generated summaries and reference summaries, computing the evaluation metrics, and aggregating the results.

How the Published Accuracy Metric is Computed

Code Location:

The file `text_eval.py` uses the `rouge_scorer.RougeScorer` from the `rouge_score` package to compute ROUGE metrics. It also integrates additional metrics such as BLEU, repetition rates, and length statistics.

Key Steps in the Code:

1. **Scorer Initialization:** A dictionary of scorers is created. For ROUGE, the code initializes:

```
scorers_dict[_ROUGE_METRIC] = /
rouge_scorer.RougeScorer(
  ["rouge1", "rouge2", "rougeL", "rougeLsum"], /
    / use_stemmer=True)
```

2. **Score Computation:** For each example in the evaluation dataset, the code decodes the predictions and targets, then computes the scores:

   ```
   scores_i = scorer.score(targets, preds)
   ```

   The scores for each metric (e.g., ROUGE-1, ROUGE-2, etc.) are calculated by comparing the generated summary (`preds`) with the reference summary (`targets`).

   Summary:

The published accuracy of the model (as reported by ROUGE scores) is computed in `text_eval.py` by:

- Decoding the generated outputs and references,
- Using `rouge_scorer.RougeScorer` to calculate n-gram overlaps and longest common subsequences,
- Aggregating the per-example scores to obtain final metrics with confidence bounds.

2.3 Self-Confidence Computation

In the file `estimator_metrics.py`, the function `_create_generative_metrics` computes the loss using

```
loss = tf.losses.sparse_softmax_cross_entropy
(labels, logits, weights=weights)
```

and then reports a metric called `"metrics/perplexity_pad_masked"` via

```
"metrics/perplexity_pad_masked": tf.metrics.mean(loss)
```

While this is not labeled as a "self-confidence score," the perplexity can be interpreted as a proxy for the model's confidence. A lower perplexity indicates that the model assigns higher probabilities to the correct tokens, implying higher confidence.

2.4 Formula for Self-Confidence

The underlying idea is that the model’s confidence in a generated sequence is derived from the probabilities it assigns to each token.

$$S = \prod_{i=1}^{n} p(w_i)$$

Because the product of many probabilities can become numerically unstable, it is common to work in the logarithmic domain:

$$\log S = \sum_{i=1}^{n} \log p(w_i)$$

A normalized version is the average log-probability:

$$\text{AvgLogProb} = \frac{1}{n} \sum_{i=1}^{n} \log p(w_i)$$

In practice, the model computes the loss for each token, and the mean loss is used to compute perplexity:

$$\text{Perplexity} = \exp\left(-\frac{1}{n} \sum_{i=1}^{n} \log p(w_i) \right)$$

Thus, a lower perplexity implies higher average token confidence. Although the code in `estimator_metrics.py` reports perplexity rather than an explicit "self-confidence score," this value serves as an inverse measure of the model's self-confidence in its output.

2.5 Obvious Incorrect Result

Input Text:

"Recently, the government announced a series of measures to reduce urban pollution, including stricter emission controls and the expansion of green spaces. It is predicted that these measures will lower air pollution by 30% within one year. Environmental groups have praised the initiative, while some industry representatives have raised concerns about potential negative economic impacts."

Expected Summary:

"The government announced new measures to reduce urban pollution through stricter emission controls and expanding green spaces, aiming to lower pollution by 30% within a year. Environmental groups support the plan, but industry worries about economic downsides."

Model Output:

"The Chinese government has announced a series of measures to reduce urban pollution, including stricter emission controls and the expansion of green spaces."

2.5.1 Problem

1. The model incorrectly labels the government as "Chinese government" even though the input does not specify a particular country.

2. The generated summary is overly simplified and omits critical information such as the goal of reducing pollution by 30%.

These errors are obvious to a person with common sense. Additionally, the erroneous inclusion of "Chinese government" makes the output particularly amusing.

2.6 Self-Confidence Score

The average token self-confidence score is approximately **0.1674**. This low value indicates that the model is not very confident about the generated (and incomplete) summary.

2.7 Rationale for Problem Instance Selection

The input text contains multiple key pieces of information (e.g., emission controls, expansion of green spaces, a 30% reduction in pollution, and differing reactions from various groups), which would allow an average reader to synthesize a comprehensive summary easily. However, the model-generated summary only includes part of the information and incorrectly labels the government as "Chinese government." This discrepancy clearly violates common sense and results in an obviously incorrect, even entertaining, output. The selection of this instance leverages the assumption that the model may overly rely on prominent keywords or patterns, leading it to ignore less frequent yet essential details, thus revealing its shortcomings in understanding the overall context.