Advanced NLP Assignment 2 - Report 1

Evaluation Metrics for NLP

Question 1: Evaluation Metrics

(a) ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It was created for evaluating automatic summarization systems by measuring overlap between generated summaries and reference summaries.

Types:

- **ROUGE-1**: Looks at individual words (unigrams). If your summary has many words from the reference, you get a high score.
- **ROUGE-2**: Checks pairs of consecutive words (bigrams). This is stricter than ROUGE-1.
- **ROUGE-L**: Uses longest common subsequence - finds the longest sequence of words that appear in both texts in the same order (but gaps are okay).

Why it's useful:

ROUGE is popular because it's fast to compute and works reasonably well for summarization tasks. If a summary covers the important points from the reference, ROUGE will catch that. It's been the standard metric in summarization competitions for years.

Problems with ROUGE:

The biggest issue is that ROUGE only matches exact words. If you write "the physician examined the patient" but the reference says "the doctor checked the patient," ROUGE scores this poorly even though the meaning is the same. It can't understand synonyms or paraphrasing.

Another problem - ROUGE doesn't care about grammar or sentence structure beyond n-grams. You could shuffle words around in weird ways and still get decent scores if the words match. It also encourages copying directly from the source instead of genuinely summarizing.

(b) BLEU

BLEU (Bilingual Evaluation Understudy) was designed for machine translation. Instead of measuring recall like ROUGE, BLEU focuses on precision - how much of what you generated appears in the reference.

Types:

- **BLEU-1, BLEU-2, BLEU-3, BLEU-4**: These look at 1-gram, 2-gram, 3-gram, and 4-gram matches respectively. Standard BLEU combines all four.

BLEU also includes a "brevity penalty" that punishes very short translations. Without this, a system could just output a few safe words and get perfect precision.

Why it's important:

Before BLEU, evaluating translation systems required expensive human evaluation. BLEU gave researchers a quick automatic way to compare different approaches. It works across languages without modification - you just need reference translations.

Limitations:

BLEU has several issues. First, it works poorly with just one reference translation because there are many valid ways to translate the same sentence. Professional translators would produce different outputs, but BLEU penalizes any variation from the single reference you provide.

Second, BLEU treats all words equally. In reality, getting content words right ("cat," "running") matters more than function words ("the," "a").

Third, BLEU can give misleading scores at the sentence level. It works better when averaging over many sentences. If you're trying to evaluate individual translations, BLEU isn't reliable.

(c) BERT Score

BERTScore is much newer (2019) and takes a completely different approach. Instead of counting matching words, it uses BERT's neural network to compare the meaning of words and sentences.

How it works:

BERT creates embeddings (mathematical representations) for each word that capture meaning and context. BERTScore compares these embeddings between your output and the reference. Words that mean similar things get high similarity scores even if they're spelled differently.

Why it's better than n-gram metrics:

The main advantage is handling paraphrasing. "The doctor arrived" and "The physician showed up" get a high BERTScore because BERT understands "doctor" and "physician" mean the same thing, and "arrived" and "showed up" are similar.

BERTScore also handles context. The word "bank" near "river" gets different embeddings than "bank" near "money," so it can tell them apart.

Downsides:

The biggest problem is speed. BERTScore needs to run BERT (a large neural network) on all the text, which is slow and requires a GPU. Calculating ROUGE or BLEU takes milliseconds, but BERTScore can take seconds or minutes for the same text.

Another issue is that results depend on which BERT model you use. Scores from BERT-base will differ from RoBERTa-large. This makes comparison across papers tricky unless everyone uses the same model.

Also, BERTScore can be fooled. If a system generates fluent but factually wrong text using similar vocabulary to the reference, it might score well even though the content is incorrect.

Question 2: Reference-Free Evaluation

What are reference-free evaluations?

Reference-free metrics evaluate text quality without comparing to any ”gold standard” reference. They judge the text on its own merits - is it fluent, grammatical, coherent?

Example: Perplexity

Perplexity measures how ”surprised” a language model is by text. Lower perplexity means the model finds the text more natural and expected.

The formula is:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, ..., w_{i-1})\right)$$

Basically, you train a language model on lots of text, then see how well it predicts new text. Text that follows normal language patterns gets low perplexity. Grammatically broken or nonsensical text gets high perplexity.

Advantages compared to ROUGE/BLEU/BERTScore:

1. **No references needed**: This is huge. Many tasks don't have natural references - creative writing, dialogue, open-ended QA. Perplexity works on any text.
2. **Measures fluency directly**: Reference-based metrics only care about matching the reference. Perplexity checks if the language itself makes sense.
3. **Works for any language**: As long as you have a language model, you can compute perplexity.

Disadvantages:

1. **Ignores correctness**: Perplexity can't tell if your content is accurate. A fluent lie scores better than an awkwardly phrased truth.
2. **Favors boring text**: Language models assign higher probability to common phrases. Generic sentences like ”It was a nice day” get low perplexity even if they're not very informative.
3. **Depends on your language model**: Different models give different perplexity scores. If your model is biased or poorly trained, perplexity is unreliable.
4. **Can't evaluate task success**: For translation or summarization, you need to know if the output captures the right content. Perplexity only checks fluency, not whether the task was done correctly.

Bonus: Custom Evaluation Metric

Semantic Coherence Score (SCS)

I'm proposing a metric that combines three different aspects: semantic meaning, word overlap, and fluency.

The idea:

Most metrics only look at one thing. ROUGE/BLEU check word matching. BERTScore checks meaning but is slow. Perplexity checks fluency but ignores content. What if we combined all three?

How it works:

For a generated text C and reference R, calculate:

1. **Semantic Similarity**: Use a sentence embedding model (like SentenceTransformers) to get embeddings for the whole sentence, then compute cosine similarity. This is much faster than BERTScore because you only need one forward pass, not one per word.

2. **Structural Similarity**: Simple Jaccard similarity on the words - how many words overlap divided by total unique words. This ensures we still reward matching important terms.

3. **Fluency Score**: Average word length. English words average around 5-6 letters. If words are too short (tokenization errors) or too long (concatenation errors), fluency score drops.

Combine them:

$$\text{SCS} = 0.5 \times \text{Semantic} + 0.3 \times \text{Structural} + 0.2 \times \text{Fluency}$$

Why it's better than existing metrics:

Better than ROUGE/BLEU:

- Handles paraphrasing through semantic similarity
- Works reasonably with single references
- Doesn't just count matching words

Better than BERTScore:

- Much faster - one embedding per sentence instead of per word
- Still captures meaning through embeddings
- Adds fluency component that BERTScore lacks

Better than Perplexity:

- Actually compares to reference content
- Can tell if output is on-topic, not just fluent

Example:

Reference: "The cat sat on the mat and looked very comfortable."
Hypothesis 1: "A feline was resting comfortably on a rug."
Hypothesis 2: "A dog ran in the park."

- ROUGE-1 would score Hypothesis 1 low (few matching words)
- BERTScore might score both relatively high (both are fluent)
- SCS would score Hypothesis 1 high (semantic match + good fluency) and Hypothesis 2 low (different meaning + structural mismatch)

The weights (0.5, 0.3, 0.2) can be tuned for different tasks. For summarization, you might increase the structural weight to emphasize content coverage. For paraphrasing tasks, increase semantic weight.

Limitations:

Obviously SCS isn't perfect. The fluency heuristic is pretty simple - just word length doesn't tell you much about actual grammar. The semantic component depends on which embedding model you use. And like all automatic metrics, it can't really judge creativity or detect subtle errors in logic.

But for practical applications where you need something between simple n-gram matching and expensive BERT-based evaluation, SCS offers a reasonable middle ground.