FoLT Tutorial 10 Summary

NLG Evaluation: We are gonna use *BLEU* and *ROUGE*

PART 1: BLEU

- BLEU (BiLingual Evaluation Understudy) is a metric for the automatic evaluation of the quality of machine-translated texts in the field of machine translation
- It was developed to make the evaluation of translations more efficient and consistent
- BLEU allows for the objective evaluation and comparability of machine translations
- BLEU is based on the matching of n-grams between the machine translation and a reference translation
- The matching is calculated for various n-gram sizes
- How to use BLEU?
  - `bleu = nltk.translate.bleu_score.sentence_bleu`
- Code + Examples:

```
def calculate_bleu_score(reference_sent: str, translation_sent : str, custom_weight:
list) -> float:
    bleu_score = 0
    bleu = nltk.translate.bleu_score.sentence_bleu
    tokenizer = nltk.tokenize.word_tokenize
    reference_sent = [tokenizer(reference_sent)]
    translation_sent = tokenizer(translation_sent)
    bleu_score = bleu(reference_sent, translation_sent, weights=custom_weight)
    return bleu_score

ref_sent = "we need to pump those numbers up, those numbers are rookie numbers"
trans_sent_one = "we need to drive these figures up, these figures are beginner figures"
trans_sent_two = "we need to pump those numbers up"

w1 = [1/2, 1/2]
w2 = [1/3, 1/3, 1/3]
w3 = [1/4, 1/4, 1/4, 1/4]
for w in (w1, w2, w3):
    print("Weights: ", w)
    print(calculate_bleu_score(ref_sent, trans_sent_one, w))
    print(calculate_bleu_score(ref_sent, trans_sent_two, w))
```

```
Weights:  [0.5, 0.5]
0.33968311024337877
0.42437284567694994
Weights:  [0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
0.21890301363223727
0.42437284567694994
Weights:  [0.25, 0.25, 0.25, 0.25]
3.908633169762327e-78
0.42437284567694994
```

- Comparing the scores:
  - As the N grows in the respective n-grams, the score drifts more apart in favor of the second translation.
  - This occurs since there are some key words, which are translated with synonyms in the first sentence, while the second translation only has the complete first part of the reference.
  - BLEU favors a better shorter translation over a longer one, which has some key words translated differently. One needs to be aware of this, while using BLEU to evaluate.

PART 2: ROUGE

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a metric used for the automatic evaluation of the quality of summaries and machine generated content.
- Similar to BLEU it is used for comparing the performance of different machine generated summaries in regards to human generated summaries, which in order can be used to evaluate summarization models.
- Similar to BLEU, ROUGE utilizes the overlap of n-grams (sequences of consecutive words) between the machine-generated summary and reference summaries. It evaluates both precision and recall for various n-gram lengths.
- How does it work (formula):
  - Where n stands for the length of the n-gram, gram_n
  - Countmatch(gram_n) is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

- Code:

```
def rouge_n(summaries : list, candidate_summary: str, n: int):
    rouge_score = 0.0
    tokenizer = nltk.tokenize.word_tokenize
    ngramer = nltk.ngrams
    summaries = [tokenizer(summary) for summary in summaries]
    candidate_summary = tokenizer(candidate_summary)
    summaries = [list(ngramer(summary, n)) for summary in summaries]
    candidate_summary = list(ngramer(candidate_summary, n))
    numerator = 0
    denominator = 0
    for s in summaries:
        set_s = set(s)
        for gram in set_s:
            denominator += list.count(s, gram)
        for gram in set(candidate_summary):
            if gram in set_s:
                numerator += list.count(s, gram)
    rouge_score = numerator / denominator
    return rouge_score
```

- Examples are too long, so this is how they look:

```
hq1 = "The Wolf of Wall Street is a biographical film directed by Martin Scorsese, portraying the rise and fall of stockbroker J
hq2 = "A cinematic masterpiece directed by Martin Scorsese, The Wolf of Wall Street recounts the true story of Jordan Belfort's
mq1 = "The Wolf of Wall Street is a movie about a stockbroker named Jordan Belfort. It shows his success in finance, his extrava
lq1 = "The Wolf of Wall Street is a movie about a guy named Jordan Belfort who does stock stuff. It's directed by Martin Scorses
summaries = [hq1, hq2, mq1, lq1]


candidate1 = "The Wolf of Wall Street is a film by Martin Scorsese. It's about Jordan Belfort, who becomes rich and then not so
candidate2 = "Directed by Martin Scorsese, The Wolf of Wall Street is a gripping biographical drama that chronicles the ascent a
```

- Example:

```
rouge_n(summaries, candidate1, 2)
✓ 0.0s

0.2623762376237624
```

```
rouge_n(summaries, candidate2, 2)
✓ 0.0s

0.35148514851485146
```

- The numbers meaning:
  - The second candidate gets a higher rouge-n score, since there is a bigger overlap of ngrams to the references.
  - This tells us, the the second summary should be closer to the quality of the reference summaries than the first summary.
  - However we this is still only rouge-n which is essentially recall and therefore not that good of a metric.
- We are currenlty using ROUGE-N, so why is it not the only metric used for ROUGE:
  - Since Rouge-N is essentially just a n-gram based recall metric, it lacks consideration for semantic content and overall coherence.
  - Incorporating additional metrics, such as ROUGE-L, is crucial to assess the quality of summaries comprehensively.
  - ROUGE-L emphasizes the longest common subsequence, providing a more nuanced evaluation that goes beyond mere word matching, ensuring a better reflection of summary informativeness and fluency.