Multilingual NLP
Lab 5 – Decoding in NMT

AN Ji

January 26, 2025

Notebook and Python script

1 Installing and running JoeyNMT

2 Impact of the beam size

2.1 Using the provided model, translate the test set using beams of size 1 (greedy), 5, 10, 15 and 20. For each decoding report:

– the time needed to decode the test set;

– the BLEU, chrF and $\text{CometKiwi}_{22}^{DA}$ scores;

We simply sample 200 pairs of English/Portuguese sentences to do the job. Table 1 shows the computational costs of JoeyNMT as well as the corresponding scores for different beam sizes. $\text{CometKiwi}_{22}^{DA}$ scores are given by Unbabel/wmt22-cometkiwi-da.

beam_size	time_(s)	bleu	chrf	cometkiwi
1 (greedy)	50.8333	26.75	0.53	0.6346
5	79.6815	25.47	0.50	0.6008
10	187.7228	23.19	0.47	0.5649
15	339.4308	21.95	0.45	0.5494
20	410.9484	20.54	0.43	0.5389

Table 1: Computational costs and evaluation scores for different beam sizes

– the number of hypotheses that are exactly the same as the hypotheses of the greedy decoding;

```python
greedy = "hyp_beam1.txt"
others = ["hyp_beam5.txt", "hyp_beam10.txt", "hyp_beam15.txt", "hyp_beam20.txt"]

with open("hyp_beam1.txt", "r") as f:
    greedy_hyps = [line.strip() for line in f]

results = {}
for file in others:
    with open(file, "r") as f:
        other_hyps = [line.strip() for line in f]
    matches = sum(1 for g, b in zip(greedy_hyps, other_hyps) if g == b)
    results[file] = matches

results
```

```
[Out]:
{'hyp_beam5.txt': 87,
 'hyp_beam10.txt': 81,
 'hyp_beam15.txt': 80,
 'hyp_beam20.txt': 79
 }
```

– the 5 hypotheses with the most differences from those generated by greedy decoding.

```python
import Levenshtein

def compute_diffs(greedy_hyps, other_hyps):
    """compute differences using Levenshtein distance"""
    diffs = []
    for idx, hyp in enumerate(other_hyps):
        distance = Levenshtein.distance(greedy_hyps[idx], hyp)
        diffs.append((idx, distance, greedy_hyps[idx], hyp))
    return sorted(diffs, key=lambda x: x[1], reverse=True)

# compare each file with greedy decoding file
most_different_hypotheses = {}
for file, hyps in other_hyps.items():
    differences = compute_diffs(greedy_hyps, hyps)
    most_different_hypotheses[file] = differences[:5]  # top 5
```

SentIdx	Dist.	Greedy Hyp.	Hyp. (Beam=5)
97	93	sua arma e a nave russa carbine com uma bayonet na frente e é o precursor do ak-47, o rifle militar mundial utilizado pela união soviética e chamado de "arma que mudou o campo de batalha".	sua arma é o precursor do ak-47, o rifle de assalto militar mundial utilizado pela união soviética e chamado "the gun that changed the battlefield.".
197	85	a reality show pode fazer as pessoas celebridades instantâneas, e um jovem competidor do reality show top chef é fazendo a maior parte de seu tempo na spotlight.	reality show can make people instant celebrities, and a young contestant from the reality show top chef is making the most of his time in the spotlight.
188	45	você vai ver as crianças vendendo jasminas árabes na rua.	you'll see the children selling arabian jasmines on the streets.
6	41	não adorava dizer qualquer coisa que uma coisa.	don't worry.
90	35	mas as mudanças propostas contradiaram um outro.	but the proposed changes contradicted one another.

Table 2: Top 5 most different hypotheses for beam size 5

SentIdx	Dist.	Greedy Hyp.	Hyp. (Beam=10)
66	135	o pássaro era um bloco escuro contra o céu azul perfeito por vários segundos, e então, repentinamente desdobrando seus piniões e fechar sua cauda e ele daria downward como uma bomba caiu de um aeroplano .	the bird was a dark blotch against the perfect blue sky for several seconds, and then, suddenly folding his pinions and closing his tail, he darted downward like a bomb dropped from an aeroplane.
97	93	sua arma e a nave russa carbine com uma bayonet na frente e é o precursor do ak-47, o rifle militar mundial utilizado pela união soviética e chamado de "arma que mudou o campo de batalha".	sua arma é o precursor do ak-47, o rifle de assalto militar mundial utilizado pela união soviética e chamado "the gun that changed the battlefield.".
197	85	a reality show pode fazer as pessoas celebridades instantâneas, e um jovem competidor do reality show top chef é fazendo a maior parte de seu tempo na spotlight.	reality show can make people instant celebrities, and a young contestant from the reality show top chef is making the most of his time in the spotlight.
158	67	quando o chamento é concluído, um diretor sênior na sala de situação das notas e transcrição.	when the call is finished, a senior director in the situation room reviews the notes and transcript.
59	48	o vento violenta violentemente sopra e o fogo destrói a ponte antiga.	the wind violently blows through and the fire destroys the ancient bridge.

Table 3: Top 5 most different hypotheses for beam size 10

SentIdx	Dist.	Greedy Hyp.	Hyp. (beam=15)
110	161	mas uma forma que as pessoas podem ajudar como estamos chegando ao pike nas eleições de 2006 é lembrar o efeito que retórica pode ter em nossas tropas na forma de harm, e o efeito que retórica pode ter em emboldagem ou enfraquecendo um inimigo, ele said.	but one way people can help as we are coming down the pike in the 2006 elections is to remember the effect that rhetoric can have on our troops in harm's way and the effect that rhetoric can have in emboldening or weakening an enemy, he said.
66	135	o pássaro era um bloco escuro contra o céu azul perfeito por vários segundos, e então, repentinamente desdobrando seus piniões e fechar sua cauda e ele daria downward como uma bomba caiu de um aeroplano .	the bird was a dark blotch against the perfect blue sky for several seconds, and then, suddenly folding his pinions and closing his tail, he darted downward like a bomb dropped from an aeroplane.
97	93	sua arma e a nave russa carbine com uma bayonet na frente e é o precursor do ak-47, o rifle militar mundial utilizado pela união soviética e chamado de "arma que mudou o campo de batalha".	sua arma é o precursor do ak-47, o rifle de assalto militar mundial utilizado pela união soviética e chamado "the gun that changed the battlefield.".
197	85	a reality show pode fazer as pessoas celebridades instantâneas, e um jovem competidor do reality show top chef é fazendo a maior parte de seu tempo na spotlight.	reality show can make people instant celebrities, and a young contestant from the reality show top chef is making the most of his time in the spotlight.
144	69	um dos poucos empregos disponíveis para jovens homens é se tornar um pilheiro de rodas de rodas.	one of the few jobs available for young men is to become a wheelbarrow pusher.

Table 4: Top 5 most different hypotheses for beam size 15

2.2 Comment

Time: Table 1 shows that computational cost increases from 50.83s (greedy) to 410.95s (beam size 20) as the beam size gets larger, which is expected since greedy decoding only picks the most probable next word candidate, while customizing beam size will let model search for more candidates, thus being more time-consuming.

Metric scores: All three metric scores drop as beam size increases from 1 (greedy) to 20: `BLEU` from 26.75 to 20.54 (6.21), `chrF` from 0.53 to 0.43 (0.10), $\texttt{CometKiwi}^{\texttt{DA}}_{\texttt{22}}$ from 0.6346 to 0.5389 (0.0957).

- `BLEU` and `chrF`: these two metrics require reference translations by comparing word- (`BLEU`) and character-level (`chrF`) n-gram matching.
- $\texttt{CometKiwi}^{\texttt{DA}}_{\texttt{22}}$: independent of reference translations, it is a quality estimation model-based metric that only refers to source sentences to produce an average score based on all segment-level scores, thus not being affected by reference bias.

All metrics share a common trend that in general, the greedy decoding does better than all beam search strategies. If we also take into account the computational cost, the greedy decoding is most worthy of consideration, even though theoretically beam search allows the model to explore more translation possibilities.

Number of identical hypotheses as greedy decoding ones: We can see that as the `beam size` increases, there tends to be slightly fewer identical hypotheses (from 87 to 79) with those in greedy decoding.

5 most different hypotheses from greedy decoding ones: From Tables 2 – 5, compared to the corresponding translations in greedy mode, the most different hypotheses generated using beam search contain more non-decoded English chunks or pieces, showing that the model cannot fully well decode these sentences even with beam search strategy. Note sentences 66, 97 and 197 are the most frequently badly translated ones. Under beam search, 66 and 197 are usually entirely in English, and the last phrase of 97 (*the gun that changed the battlefield*) always remains untouched.

SentIdx	Dist.	Greedy Hyp.	Hyp. (beam=20)
110	161	mas uma forma que as pessoas podem ajudar como estamos chegando ao pike nas eleições de 2006 é lembrar o efeito que retórica pode ter em nossas tropas na forma de harm, e o efeito que retórica pode ter em emboldagem ou enfraquecendo um inimigo, ele said.	but one way people can help as we are coming down the pike in the 2006 elections is to remember the effect that rhetoric can have on our troops in harm's way and the effect that rhetoric can have in emboldening or weakening an enemy, he said.
66	135	o pássaro era um bloco escuro contra o céu azul perfeito por vários segundos, e então, repentinamente desdobrando seus piniões e fechar sua cauda e ele daria downward como uma bomba caiu de um aeroplano .	the bird was a dark blotch against the perfect blue sky for several seconds, and then, suddenly folding his pinions and closing his tail, he darted downward like a bomb dropped from an aeroplane.
148	133	em um lançamento de notícias de que o doj disse obiang usou sua posição como ministro da agricultura e forestry em 2011 "para amass mais de 300 milhões de dólares de bens através de corrupção e dinheiro em violação dos estados unidos e equatoguinean law.".	in a news release, the doj said obiang used his position as minister of agriculture and forestry in 2011 "to amass more than $300 million worth of assets through corruption and money laundering, in violation of both u.s. and equatoguinean law.".
198	119	pelo menos 15 funcionários em uma fabricante de drogas chinesa tem sido detido como parte de uma investigação que produziu registros falsas envolvendo suas vacinas de coelho.	pelo menos 15 funcionários em uma fabricante de drogas.
150	108	mas nas poucas semanas curtas, ela tinha pego mais do que um glimpse da natureza primeval - ela da falsa sangrenta fang, cedo, remorsela, insensate, destruído, já destruído.	but in the few short weeks since, she had caught more than one glimpse of primeval nature,—she of the bloody fang, blind, remorseless, insensate, destroying, ever destroying.

Table 5: Top 5 most different hypotheses for beam size 20

2.3 Are the variations in $\texttt{CometKiwi}_{22}^{DA}$ scores significant (you can look here [2] to know how to interpret $\texttt{CometKiwi}_{22}^{DA}$ scores)?

From Table 1 we know the $\texttt{CometKiwi}_{22}^{DA}$ scores range from 0.5389 to 0.6346, i.e. with a difference slightly less than 0.10. According to Kocmi's MT thresholds tool, an improvement of 0.0957 $\texttt{CometKiwi}_{22}^{DA}$ has the same estimated accuracy as 0.56 **BLEU** or as 0.37 **chrF**.

3 Evaluating Error Propagation

Modify the provided script to implement the experiment for demonstrating exposure bias, i.e. use average 0/1 loss to evaluate two decoders: one with predicted tokens as input (greedy decoding), the other with forced reference history as input (forced decoding).

The updated key parts are as follows. Figure 1 shows the losses for both scenarios.

```python
def transform_sentences(
    src_tokenized: List[List[str]],
    ref_tokenized: List[List[str]],
    model: Model
    ):
    """turn src/ref tokenized sentences into tensors,
    pass src_ids through the encoder to get encoder output"""

    # ref
    ref_ids, ref_lengths = model.trg_vocab.sentences_to_ids(
        ref_tokenized, bos=True, eos=True
    )
    ref_ids     = torch.tensor(ref_ids)
    ref_lengths = torch.tensor(ref_lengths)

    # src
    src_ids, src_lengths = model.src_vocab.sentences_to_ids(
        src_tokenized, bos=False, eos=True
    )

```

```python
    # tensors
    src_ids     = torch.tensor(src_ids)
    src_lengths = torch.tensor(src_lengths)
    src_masks   = (src_ids != model.src_vocab.pad_index).unsqueeze(1)
    # shape: (batch_size, 1, seq_len)

    # pass through the encoder
    model.eval()
    with torch.no_grad():
        src_encoded, _, _, _ = model(
            return_type="encode",
            src=src_ids,
            src_length=src_lengths,
            src_mask=src_masks
        )

    return src_encoded, src_lengths, src_masks, ref_ids, ref_lengths


def greedy_decoding(
        model:          Model,
        src_tokenized:  List[List[str]],
        ref_tokenized:  List[List[str]],
        max_output_size: int,
    ):

    # transform all sentences into tensors
    src_encoded, src_lengths, src_masks, ref_ids, ref_lengths = transform_sentences(
        src_tokenized,
        ref_tokenized,
        model
    )

    ys = src_encoded.new_full([src_encoded.shape[0], 1], BOS_ID, dtype=torch.long)
    trg_masks = src_masks.new_ones([src_encoded.shape[0], 1, 1])

    for t in tqdm(range(max_output_size), desc="Greedy decoding"):
        model.eval()
        with torch.no_grad():
            logits, _, _, _ = model(
                return_type="decode",
                trg_input=ys,  # use predicted tokens so far
                encoder_output=src_encoded,
                encoder_hidden=None,
                src_mask=src_masks,
                unroll_steps=None,
                decoder_hidden=None,
                trg_mask=trg_masks
            )

            logits = logits[:, -1]
            _, pred_tokens = torch.max(logits, dim=1)
            pred_tokens = pred_tokens.unsqueeze(-1)

            ys = torch.cat([ys, pred_tokens], dim=1)

    # compute average loss when all is done
    greedy_loss = 0
    for i in range(ref_ids.shape[0]): # sentence by sentence
        sent_loss = 0
        ref_len = ref_lengths[i].item() # actual ref length <= 78
        pred_ids = ys[i].tolist() # predicted ids

        # actual pred length, <=80
        if EOS_ID in pred_ids:
            pred_len = pred_ids.index(EOS_ID) + 1
        else:
            pred_len = len(pred_ids)  # full length if EOS not found

        # first compare up to the shorter length of ref and pred
        for j in range(min(ref_len, pred_len)):
            sent_loss += int(ys[i, j] != ref_ids[i, j])
        # penalty for extra and missing tokens in pred: count as 1 error each
        sent_loss += abs(pred_len - ref_len)
        sent_loss /= ref_len # normalize by actual ref length

```

```python
        greedy_loss += sent_loss

    mean_loss = greedy_loss / ref_ids.shape[0]  # average loss per sentence
    greedy_hyps = model.trg_vocab.arrays_to_sentences(ys)

    return greedy_hyps[0], mean_loss


def forced_decoding(
        model:          Model,
        src_tokenized:  List[List[str]],
        ref_tokenized:  List[List[str]],
        max_output_size: int
    ):

    # transform all sentences into tensors
    src_encoded, src_lengths, src_masks, ref_ids, ref_lengths = transform_sentences(
        src_tokenized,
        ref_tokenized,
        model
    )

    ys = src_encoded.new_full([src_encoded.shape[0], 1], BOS_ID, dtype=torch.long)
    trg_masks = src_masks.new_ones([src_encoded.shape[0], 1, 1])

    for t in tqdm(range(max_output_size), desc="Forced decoding"):
        model.eval()
        with torch.no_grad():
            logits, _, _, _ = model(
                return_type="decode",
                trg_input=ref_ids[:, :t+1],  # use ref tokens up to t
                encoder_output=src_encoded,
                encoder_hidden=None,
                src_mask=src_masks,
                unroll_steps=None,
                decoder_hidden=None,
                trg_mask=trg_masks
            )

            logits = logits[:, -1]
            _, pred_tokens = torch.max(logits, dim=1)
            pred_tokens = pred_tokens.unsqueeze(-1)

            ys = torch.cat([ys, pred_tokens], dim=1)


    # compute average loss when all is done
    forced_loss = 0
    for i in range(ref_ids.shape[0]): # sentence by sentence
        sent_loss = 0
        ref_len = ref_lengths[i].item() # actual ref length <= 78
        pred_ids = ys[i].tolist() # predicted ids

        # actual pred length, <=80
        if EOS_ID in pred_ids:
            pred_len = pred_ids.index(EOS_ID) + 1
        else:
            pred_len = len(pred_ids)  # full length if EOS not found

        # first compare up to the shorter length of ref and pred
        for j in range(min(ref_len, pred_len)):
            sent_loss += int(ys[i, j] != ref_ids[i, j])
        # penalty for extra and missing tokens in pred: count as 1 error each
        sent_loss += abs(pred_len - ref_len)
        sent_loss /= ref_len # normalize by actual ref length
        forced_loss += sent_loss

    mean_loss = forced_loss / ref_ids.shape[0]
    forced_hyps = model.trg_vocab.arrays_to_sentences(ys)

    return forced_hyps[0], mean_loss
```

Mean greedy decoding loss per sentence: **0.722**
Mean forced decoding loss per sentence: **0.371**

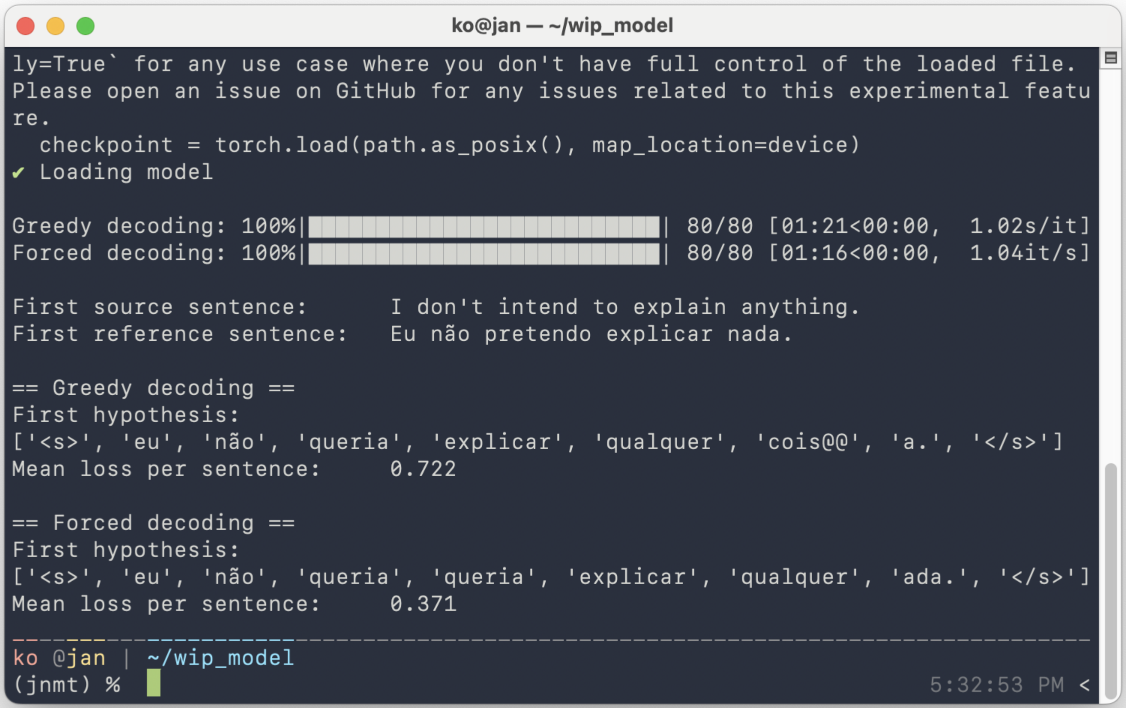

Figure 1: Results of two decoding scenarios in CLI

My choice and thoughts

The major modification of the script is to make the model decode the sentences in batch as well as consider and compare two different decoding scenarios, where we feed different values to the argument `trg_input` of the model and see what differences on prediction loss will be: the first scenario is standard greedy decoding (`greedy_decoding`), where we use model's predicted tokens as target input; while the second one is an enforced process (`forced_decoding`) that provides reference segments up to current timestep ("true context") instead.

As for the implementation of average loss, we accumulate sentence by sentence all normalized-by-reference-length loss values and compute the mean loss per sentence. In the prediction part, we allow the model to decode up to `max_output_size` timesteps (reduced from 100 to 80 after inspecting the maximum length of reference sentences, i.e. 78 tokens including BOS and EOS) for each sentence. Since predicted sentences can be longer or shorter than their corresponding reference sentences, I finally decide to use the following strategy to compute the sentence-wise loss:

1. Extract the valid length of each prediction by indexing the first EOS token. If EOS is not found, take the full length of prediction.
2. Compare tokens up to the shorter one between reference length and prediction length.
3. Finally, for each longer or shorter prediction, penalize their absolute disparity with respect to their reference by counting each of the "extra" or "missing" tokens as 1 error.
4. Normalize sentence-wise loss on reference length and compute the overall loss on all sentences.

As a result, we observe that standard greedy decoding produces a higher loss (0.722) than forced greedy decoding using references (0.371). This indicates that given "correct hints" the model is more likely to be guided to the "right" prediction. It's also a good example to illustrate that the model's generalization ability could be biased due to exposure to the reference data it was given during training process. Therefore, when using its own predicted tokens together with the encoded source data as input, the model would depend more on its own knowledge to predict next tokens. If, at an early timestep, the model predicts a wrong token, this very error tends to have an accumulative impact on the following prediction and cannot be corrected by the model itself, leading to what is called error propagation here.

4 ϵ-sampling Decoding (*to be continued...*)

4.1 Drawing on your experience gained from implementing the previous experiment, implement a decoder using the ϵ-sampling strategy described in this article (section 4.3) with $\epsilon = 0.02$.

4.2 Using this decoding strategy, generate 200 translation hypotheses for each source sentence and determine the mean, max and min $\texttt{CometKiwi}^{\texttt{DA}}_{\texttt{22}}$ scores for each list. What can you conclude from this?

References

[1] Hewitt, J., Manning, C., and Liang, P. (Dec. 2022). "Truncation Sampling as Language Model Desmoothing". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Ed. by Goldberg, Y., Kozareva, Z., and Zhang, Y. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 3414–3427. DOI: `10.18653/v1/2022.findings-emnlp.249`. URL: `https://aclanthology.org/2022.findings-emnlp.249/`.

[2] Kocmi, T. et al. (Aug. 2024). "Navigating the Metrics Maze: Reconciling Score Magnitudes and Accuracies". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Ku, L.-W., Martins, A., and Srikumar, V. Bangkok, Thailand: Association for Computational Linguistics, pp. 1999–2014. DOI: `10.18653/v1/2024.acl-long.110`. URL: `https://aclanthology.org/2024.acl-long.110/`.

[3] Kreutzer, J., Bastings, J., and Riezler, S. (Nov. 2019). "Joey NMT: A Minimalist NMT Toolkit for Novices". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*. Hong Kong, China: Association for Computational Linguistics, pp. 109–114. DOI: `10.18653/v1/D19-3019`. URL: `https://www.aclweb.org/anthology/D19-3019`.

[4] Rei, R. et al. (Dec. 2022). "CometKiwi: IST-Unbabel 2022 Submission for the Quality Estimation Shared Task". In: *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Ed. by Koehn, P. et al. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, pp. 634–645. URL: `https://aclanthology.org/2022.wmt-1.60/`.