Machine Translation HW3 Math Description

Beam Search Decoding with Reordering:

The basic idea behind beam search decoding in machine translation is to find the most likely translation given a source sentence, by exploring multiple translation hypotheses simultaneously. It's a heuristic search algorithm designed to explore the most promising paths in a search space efficiently, which is especially useful in scenarios where exhaustive search is infeasible due to computational constraints.

Beam search decoding with reordering is an extension of the standard beam search algorithm used in machine translation. It is designed to handle reordering of words or phrases in the translation process, which is common when translating between languages with different word orders.

Implementation:

1. **Imports and Setup:**

   The code begins by importing necessary libraries and setting up data structures, including importing translation and language models, and defining a hypothesis namedtuple to represent translation hypotheses.

2. **Command-Line Options:**

   The get_options function parses command-line options using the optparse library. These options include input file paths, stack size, reordering limits, and threshold values.

3. **Precompute Future Costs:**

   The precompute_future_costs function calculates a future cost table for spans of the French sentence. This table is used to estimate the cost of translating remaining uncovered words, incorporating reordering constraints.

4. **Main Decoding Function (decode):**

   The decode function is the core of the decoding algorithm. It initializes an initial hypothesis with a log probability of 0.0 and starts the beam search process.

5. **Beam Data Structure:**

   A beam data structure (beams) is used to maintain a list of hypotheses at each decoding step. The size of the beam is determined by the stack-size option.

6. **Decoding Loop:**

   The code enters a loop where it iterates through the hypotheses in the beam. For each hypothesis, it generates possible extensions by considering the next word or phrase to translate.

7. **Extending Hypotheses:**

   The extend_hypothesis function is called to extend a given hypothesis with a new phrase. This involves selecting the next word or phrase to translate and determining its position in the target sentence. Reordering is considered during this step.

8. **Scoring and Pruning:**

   The extended hypotheses are scored based on log probabilities, including LM scores, TM scores, and reordering scores. The hypotheses are sorted by score, and the top-K hypotheses are retained, where K is determined by the beam width.

9. **Early Stopping:**

   The algorithm performs early stopping if a hypothesis has covered all words in the source sentence, indicating that the translation is complete.

10. **Backtracking:**

    Once the decoding is complete, the algorithm backtracks through the best hypothesis to reconstruct the final translation.

11. **Extracting the Best Translation:**

    The extract_best_translation function is used to extract the best translation from the final beams.

12. **Output and Unknown Words:**

    The main function reads French sentences from an input file, handles unknown words, and prints the decoded English translations.

13. Execution:

The code executes the main function when run as a script.

Algorithms used in our Beam Search Decoder with Reordering:

Language Model Score (LM Score):

The LM score represents the likelihood of a sequence of English words given the context. It is computed as the sum of log probabilities for each word in the sequence based on the language model.

$$LM\ SCORE\ = Log(P(word_1|context)) + Log(P(word_2|context, word_1)) + Log(P(word_n|context, word_1 .... word_{n-1}))$$

$P(word_i \mid context)$ is the probability of $word_i$ given the preceding words in the sequence.

Translation Model Score (TM Score):

The TM score quantifies how well a translation hypothesis aligns with the source sentence and includes translation phrase probabilities and reordering probabilities.

$$TM\ Score(H) = \sum[Log(P(phrase))\ for\ each\ phrase\ in\ H] + \sum[Log(P(reordering))\ for\ each\ reordering\ decision\ in\ H]$$

H represents the hypothesis, and $P(phrase)$ is the probability assigned to a translation phrase in the hypothesis. $P(reordering)$ is the probability of reordering decisions made during translation.

Reordering Score (Reordering Probability):

The reordering score reflects the likelihood of reordering words or phrases in the target sentence relative to their positions in the source sentence. The specific equation depends on the reordering model used but generally considers alignment probabilities and distortion penalties.

Log Probability of a Hypothesis:

Each hypothesis in the beam has a log probability score, which is the sum of the LM score, the TM score (including translation and reordering), and any other relevant scores:

$$Hypothesis\ Log\ Probability\ = LM\ Score + TM\ Score + Reordering\ Score + Other\ Sources...$$

Pruning Threshold (Delta):

The algorithm employs a pruning threshold (delta) to limit the number of hypotheses kept in the beam. Only hypotheses with log probabilities greater than or equal to max_logprob - delta are considered.

Future Cost Estimation:

The future_cost_estimate function estimates the cost of translating the remaining uncovered words in the source sentence. It sums the costs of untranslated spans, considering the current coverage.

Beam Search:

The algorithm performs beam search, exploring multiple translation hypotheses in parallel. It iterates through hypotheses in the beam, extending them with new translation phrases while considering reordering constraints.

Backtracking and Extraction:

After decoding, the algorithm backtracks through the best hypothesis to extract the final English translation.

Beam Width (Stack Size):

The beam width, determined by the stack-size option, controls the number of hypotheses retained at each decoding step. It influences the trade-off between search efficiency and translation quality.

Recombination Check:

The algorithm checks for recombination to avoid keeping multiple hypotheses with the same language model state and coverage. It compares the log probabilities of hypotheses with the same state and coverage.

<u>Motivation:</u>

The algorithm aims to find the translation hypothesis with the highest log probability while considering language models, translation models, reordering, and pruning.

Conclusion:

The code presented here implements a beam search decoding algorithm with coverage and reordering capabilities for machine translation. By utilizing coverage vectors and imposing reordering limits, it enhances translation quality by controlling word reordering and efficiently exploring translation hypotheses. This approach contributes to the generation of more accurate and fluent translations in machine translation systems. Overall, this mathematical description captures the core calculations and processes involved in the beam search decoding algorithm for machine translation with reordering.

Beam Search Decoding with Vector Coverage:

Abstract:

This code implements a beam search decoding algorithm with coverage for machine translation. The algorithm efficiently explores the search space to generate translations from a source language (French) to a target language (English). It employs coverage vectors to manage word reordering and prevent over-translation.

Algorithm:

1. **Initialization:**

- Input: Source sentence in French, translation model (TM), language model (LM), and decoding parameters.
- Initialize coverage vector C as a binary vector of zeros with a length equal to the number of words in the source sentence.

2. **Hypothesis Representation:**

- Represent translation hypotheses as tuples: (logprob, lm_state, predecessor, phrase, C, start, end).
- logprob: Log probability of the hypothesis.
- lm_state: Language model state.
- predecessor: Reference to the previous hypothesis.
- phrase: Translated phrase.
- C: Coverage vector indicating which words are covered.
- start and end: Indices indicating the range of words covered by the hypothesis.

3. **Coverage Hypothesis:**

- Define the coverage_hypothesis function to extend hypotheses based on translation phrases and coverage.
- Calculate the log probability of a new hypothesis by considering translation probabilities, language model scores, and coverage penalties.
- Update the coverage vector C to mark newly covered words.

4. **Beam Search Decoding:**

- Initialize stacks for hypotheses, one stack for each position in the source sentence, along with an additional stack.
- Start with an initial hypothesis with an empty coverage vector and add it to the appropriate stack.
- For each position x in the source sentence and each hypothesis h in the current stack, extend the hypothesis in both directions by considering different phrases.
- Maintain beam width and prune hypotheses with low probabilities.
- Update the stacks with new hypotheses, ensuring that coverage information is correctly maintained.

5. **Winning Hypothesis and Translation Extraction:**

- Identify the winning hypothesis with the highest log probability in the last stack.
- Extract the English translation by recursively following the predecessor links.

Motivation:

- **Coverage Control:** The use of coverage vectors (C) is motivated by the need to control word reordering during translation. By tracking which words have been covered, the decoder ensures that translations respect the source word order.

- **Preventing Over-Translation:** Coverage prevents over-translation by disallowing the retranslation of already covered words. This helps generate more fluent and accurate translations.

- **Efficient Beam Search:** Beam search efficiently explores the translation hypothesis space, considering a limited number of top hypotheses at each position. Pruning based on probabilities and coverage helps focus on more promising hypotheses.

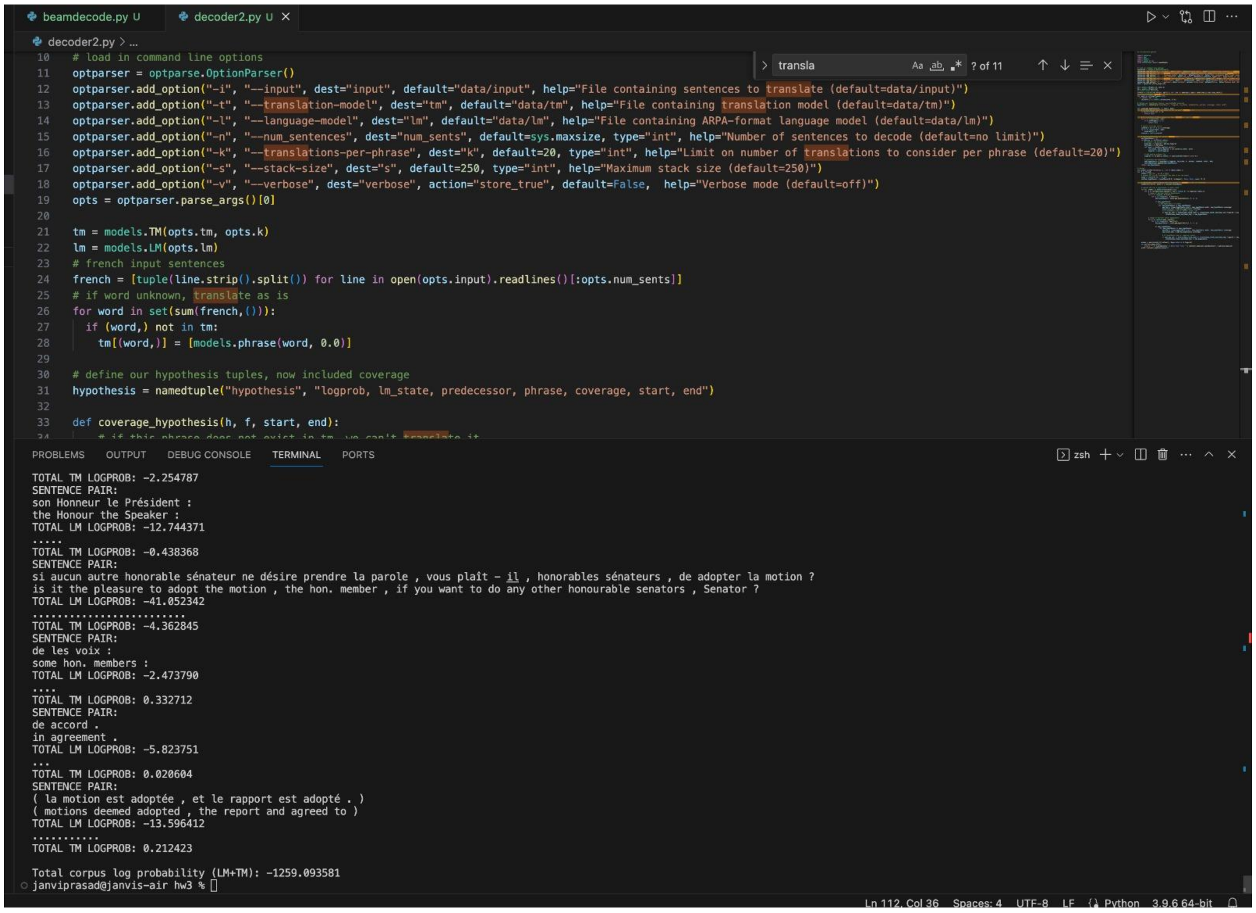
- **Translation Completeness:** The coverage vector also serves as an indicator of translation completeness. When all elements of C are set to 1, it signifies that the entire source sentence has been translated.

Conclusion:

The code presented here demonstrates an effective beam search decoding algorithm with coverage for machine translation. Through the use of coverage vectors, it manages word reordering, prevents over-translation, and ensures translation completeness. This approach contributes to the generation of high-quality translations in machine translation systems.

Results

The code results should produce a total corpus log probability (LM+TM) of -1259.093581 (see figure below)



```python
# load in command line options
optparser = optparse.OptionParser()
optparser.add_option("-i", "--input", dest="input", default="data/input", help="File containing sentences to translate (default=data/input)")
optparser.add_option("-t", "--translation-model", dest="tm", default="data/tm", help="File containing translation model (default=data/tm)")
optparser.add_option("-l", "--language-model", dest="lm", default="data/lm", help="File containing ARPA-format language model (default=data/lm)")
optparser.add_option("-n", "--num_sentences", dest="num_sents", default=sys.maxsize, type="int", help="Number of sentences to decode (default=no limit)")
optparser.add_option("-k", "--translations-per-phrase", dest="k", default=20, type="int", help="Limit on number of translations to consider per phrase (default=20)")
optparser.add_option("-s", "--stack-size", dest="s", default=250, type="int", help="Maximum stack size (default=250)")
optparser.add_option("-v", "--verbose", dest="verbose", action="store_true", default=False,  help="Verbose mode (default=off)")
opts = optparser.parse_args()[0]

tm = models.TM(opts.tm, opts.k)
lm = models.LM(opts.lm)
# french input sentences
french = [tuple(line.strip().split()) for line in open(opts.input).readlines()[:opts.num_sents]]
# if word unknown, translate as is
for word in set(sum(french,())):
  if (word,) not in tm:
    tm[(word,)] = [models.phrase(word, 0.0)]

# define our hypothesis tuples, now included coverage
hypothesis = namedtuple("hypothesis", "logprob, lm_state, predecessor, phrase, coverage, start, end")

def coverage_hypothesis(h, f, start, end):
    # if this phrase does not exist in tm, we can't translate it
```

```
TOTAL TM LOGPROB: -2.254787
SENTENCE PAIR:
son Honneur le Président :
the Honour the Speaker :
TOTAL LM LOGPROB: -12.744371
.....
TOTAL TM LOGPROB: -0.438368
SENTENCE PAIR:
si aucun autre honorable sénateur ne désire prendre la parole , vous plaît - il , honorables sénateurs , de adopter la motion ?
is it the pleasure to adopt the motion , the hon. member , if you want to do any other honourable senators , Senator ?
TOTAL LM LOGPROB: -41.052342
.........................
TOTAL TM LOGPROB: -4.362845
SENTENCE PAIR:
de les voix :
some hon. members :
TOTAL LM LOGPROB: -2.473790
....
TOTAL TM LOGPROB: 0.332712
SENTENCE PAIR:
de accord .
in agreement .
TOTAL LM LOGPROB: -5.823751
...
TOTAL TM LOGPROB: 0.020604
SENTENCE PAIR:
( la motion est adoptée , et le rapport est adopté . )
( motions deemed adopted , the report and agreed to )
TOTAL LM LOGPROB: -13.596412
...........
TOTAL TM LOGPROB: 0.212423

Total corpus log probability (LM+TM): -1259.093581
janviprasad@janvis-air hw3 %
```