Team Members

Name	Email
Anushri Suresh	asures13@jh.edu
Suhas Sasetty	ssasett1@jh.edu
Yogeeshwar Selvaraj	yselvar1@jh.edu

Motivation

We implement beam search as a decoding algorithm for sequence generation tasks, such as machine translation, to efficiently explore multiple translation hypotheses. Our goal is to achieve an optimal balance between translation accuracy and computational efficiency. Beam search maintains a limited number of hypotheses, controlled by a beam width parameter, allowing us to prune less likely candidates while focusing on the most promising ones.

In our implementation, we integrate a language model (LM) to evaluate the fluency of generated English translations and a translation model (TM) to handle phrase translations. This approach enables us to decode sentences from a foreign language into English effectively.

Beam Search Algorithm Description

1. **Input Parsing and Setup:** We begin by parsing command-line arguments, including the input sentences to translate, the translation model, and the language model. The beam width (b) parameter controls the number of hypotheses considered at each step, and the distortion limit (d) sets the maximum reordering of phrases allowed during decoding.

2. **Data Structures:** We define a hypothesis structure using Python's `namedtuple`. This structure holds:

   - `logprob`: The cumulative log probability of the hypothesis.
   - `lm_state`: The current state of the language model.
   - `predecessor`: The previous hypothesis (for backtracking).
   - `phrase`: The current translated phrase.
   - `coverage`: A tuple tracking which words in the foreign sentence are covered by the translation.

3. **Initialization:** Each sentence is processed individually. We initialize the decoding process with an empty hypothesis where no words have been translated, and the LM state is at the beginning.

4. **Beam Search Process:** The algorithm iterates through each position in the foreign sentence and explores translation hypotheses:

   - **Phrase Translation:** At each step, the algorithm considers all possible phrase translations for unaligned segments of the foreign sentence. The translation model provides log probabilities for each phrase.
   - **Language Model Scoring:** For each phrase in English, the language model assigns a log probability to each word, contributing to the overall score of the hypothesis.
   - **Coverage and Reordering:** The algorithm updates the coverage vector, marking the positions of words already translated. It ensures that phrases do not overlap by skipping those that have already been covered. Beam search allows for limited reordering of phrases, constrained by the distortion limit d.

5. **Hypothesis Recombination and Pruning:** For each new hypothesis generated, we either:

   - Add it to the current stack of hypotheses, or
   - Recombine it with an existing hypothesis if it covers the same words and has a higher log probability. At each stage, only the top `beam_width` hypotheses are retained to limit the search space.

6. **Decoding and Extraction:** Once all words in the foreign sentence are covered, the best hypothesis is chosen from the final stack. The English translation is extracted by following the chain of predecessors from the final hypothesis back to the initial one.

Mathematical Description

Given a foreign sentence $f = \{f_1, f_2, \ldots, f_n\}$ and its corresponding English translation $e = \{e_1, e_2, \ldots, e_m\}$, beam search works by maximizing the following score for each hypothesis h:

$$\text{Score}(h) = \sum_{\text{phrases}} \log P(e_{\text{phrase}} \mid f_{\text{phrase}}) + \sum_{\text{words}} \log P(e_i \mid \text{LM\_state})$$

Where:

- $P(e_{\text{phrase}} \mid f_{\text{phrase}})$ is the probability of the English phrase given the foreign phrase, provided by the translation model.
- $P(e_i \mid \text{LM\_state})$ is the probability of the next English word according to the language model.

We apply beam pruning by retaining only the top hypotheses at each step, where the number of hypotheses is limited by the beam width parameter b.

Modifications

We explored the following modifications to the basic beam search:

- **Distortion Limit (d):** This restricts the reordering of phrases during decoding. We experimented with different distortion limits to balance between reordering flexibility and alignment accuracy.
- **Beam Width (b):** A higher beam width allows the algorithm to consider more hypotheses, potentially improving accuracy but at the cost of increased computation. We experimented with beam widths ranging from 500 to 2000.

Experiments and Results

We conducted several experiments by varying the beam width and distortion limit. The best results were obtained with a beam width of 1000 and distortion limit of 10. These parameters provided the optimal balance between translation quality and computational efficiency.

Usage

To run the beam search algorithm:

```
python decode--beam--search --b 1000 --d 10 > translations
```

To generate translations for a dataset of sentences, the output will be saved in the `translations` file.

Greedy Decoding Algorithm Description

In this implementation, we apply a greedy decoding strategy to translate sentences from a foreign language into English. The translation process uses a translation model (TM) to handle phrase translations and a language model (LM) to evaluate the fluency of the generated English translations.

1. **Input Parsing and Setup:** The program begins by parsing command-line arguments, which include the input sentences to translate, the translation model, and the language model. Unlike beam search, greedy decoding has no explicit beam width parameter; instead, it relies on choosing the highest-probability translation at each step.

2. **Data Structures:** We use Python's `namedtuple` to define a hypothesis structure, which holds:
    - `logprob`: The cumulative log probability of the hypothesis.
    - `lm_state`: The current state of the language model.
    - `predecessor`: The previous hypothesis, allowing for backtracking.
    - `phrase`: The current translated phrase.
3. **Initialization:** For each sentence, we initialize the decoding process with an empty hypothesis, where no words have been translated, and the language model is in its initial state.
4. **Greedy Search Process:** The greedy decoding algorithm processes one sentence at a time and makes local decisions at each step based on the highest-probability translation.
    - **Phrase Translation:** At each step, the algorithm considers all possible phrase translations from the foreign sentence, as provided by the translation model.
    - **Language Model Scoring:** The algorithm uses the language model to score the English phrases, adding the log probability of each word to the total hypothesis score.
    - **Commitment to the Best Hypothesis:** Unlike beam search, which keeps multiple hypotheses, greedy decoding commits to the single best hypothesis based on the current local decision. The search moves forward by the length of the phrase that has been translated.
    - **Termination:** The algorithm terminates when the entire foreign sentence is translated. The final hypothesis is selected as the best translation.
5. **Decoding and Extraction:** Once the decoding process is complete, the algorithm traces back from the final hypothesis to the initial hypothesis by following the predecessor links, thus reconstructing the English translation.

Mathematical Description

Given a foreign sentence

$$\mathbf{f} = \{f_1, f_2, \ldots, f_n\}$$

and its corresponding English translation

$$\mathbf{e} = \{e_1, e_2, \ldots, e_m\},$$

greedy decoding works by maximizing the following score for each hypothesis h:

$$\text{Score}(h) = \sum_{\text{phrases}} \log P(e_{\text{phrase}} \mid f_{\text{phrase}}) + \sum_{\text{words}} \log P(e_i \mid \text{LM\_state})$$

Where:

- $P(e_{\text{phrase}} \mid f_{\text{phrase}})$ is the probability of the English phrase given the foreign phrase, as provided by the translation model.
- $P(e_i \mid \text{LM\_state})$ is the probability of the next English word according to the language model.

In greedy decoding, we do not store or prune multiple hypotheses. Instead, at each step, we greedily choose the highest-probability phrase, which can make this approach less robust to suboptimal local decisions.

Conclusion

In conclusion, beam search provides a more comprehensive exploration of possible translations by maintaining multiple hypotheses, which often results in higher translation accuracy, especially for complex sentences requiring phrase reordering. However, this comes at the cost of increased computational resources and time.

Greedy decoding is significantly faster and more resource-efficient, as it commits to the highest-probability translation at each step without maintaining multiple hypotheses. While this can lead to suboptimal translations due to its myopic nature, it is well-suited for applications where speed is critical and computational resources are limited.

.