Natural Language Generation

Natural Language Generation (NLG)

- Natural Language Generation refers to any setting in which we *generate* (i.e. write) new text.

- NLG is a subcomponent of:
  - Machine Translation
  - (Abstractive) Summarization
  - Dialogue (chit-chat and task-based)
  - Creative writing: storytelling, poetry-generation
  - Freeform Question Answering (i.e. answer is generated, not extracted from text or knowledge base)
  - Image captioning
  - …

Recap

- **Language Modeling**: the task of predicting the next word, given the words so far

$$P(y_t|y_{1,\dots,}y_{t-1})$$

- A system that produces this probability distribution is called a **Language Model**

- If that system is an RNN, it's called a **RNN-LM**

Recap

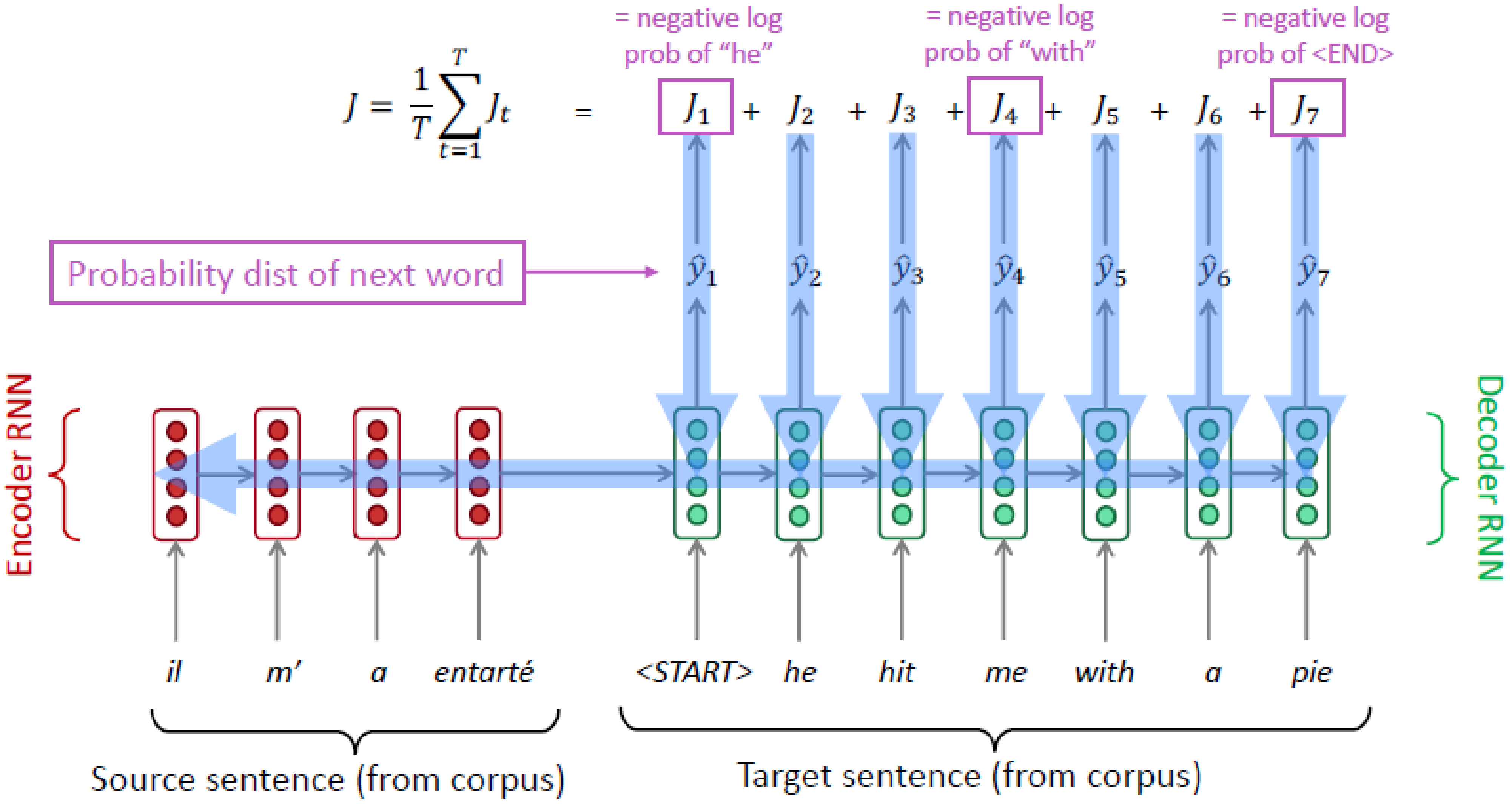
- **Conditional** Language Modeling: the task of predicting the next word, given the words so far, and also some other input *x*

$$P(y_t | y_{1,\ldots,} y_{t-1}, x)$$

- Examples of conditional language modeling tasks:
  - Machine Translation (x=source sentence, y=target sentence)
  - Summarization (x=input text, y=summarized text)
  - Dialogue (x=dialogue history, y=next utterance)
  - …

Recap: training a (conditional) RNN-LM

This example: Neural Machine Translation



During training, we feed the gold (aka reference) target sentence into the decoder, regardless of what the decoder predicts. This training method is called ***Teacher Forcing***.

Recap: decoding algorithms

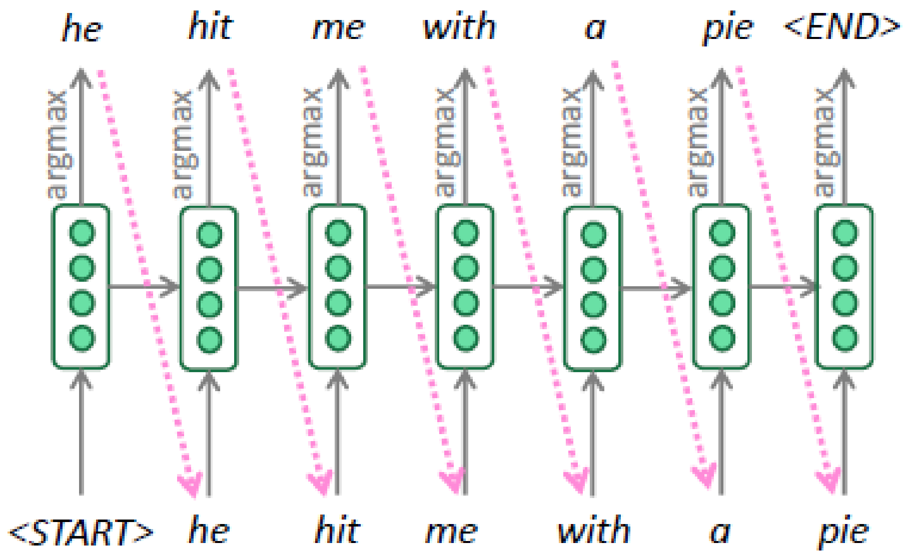
- Question: Once you've trained your (conditional) language model, how do you use it to generate text?

- Answer: A *decoding algorithm* is an algorithm you use to generate text from your language model

- We've learnt about two decoding algorithms:
  - Greedy decoding
  - Beam search

Recap: greedy decoding

- A simple algorithm
- On each step, take the most probable word (i.e. argmax)
- Use that as the next word, and feed it as input on the next step
- Keep going until you produce <END> (or reach some max length)



- Due to lack of backtracking, output can be poor
  (e.g. ungrammatical, unnatural, nonsensical)

Recap: beam search decoding

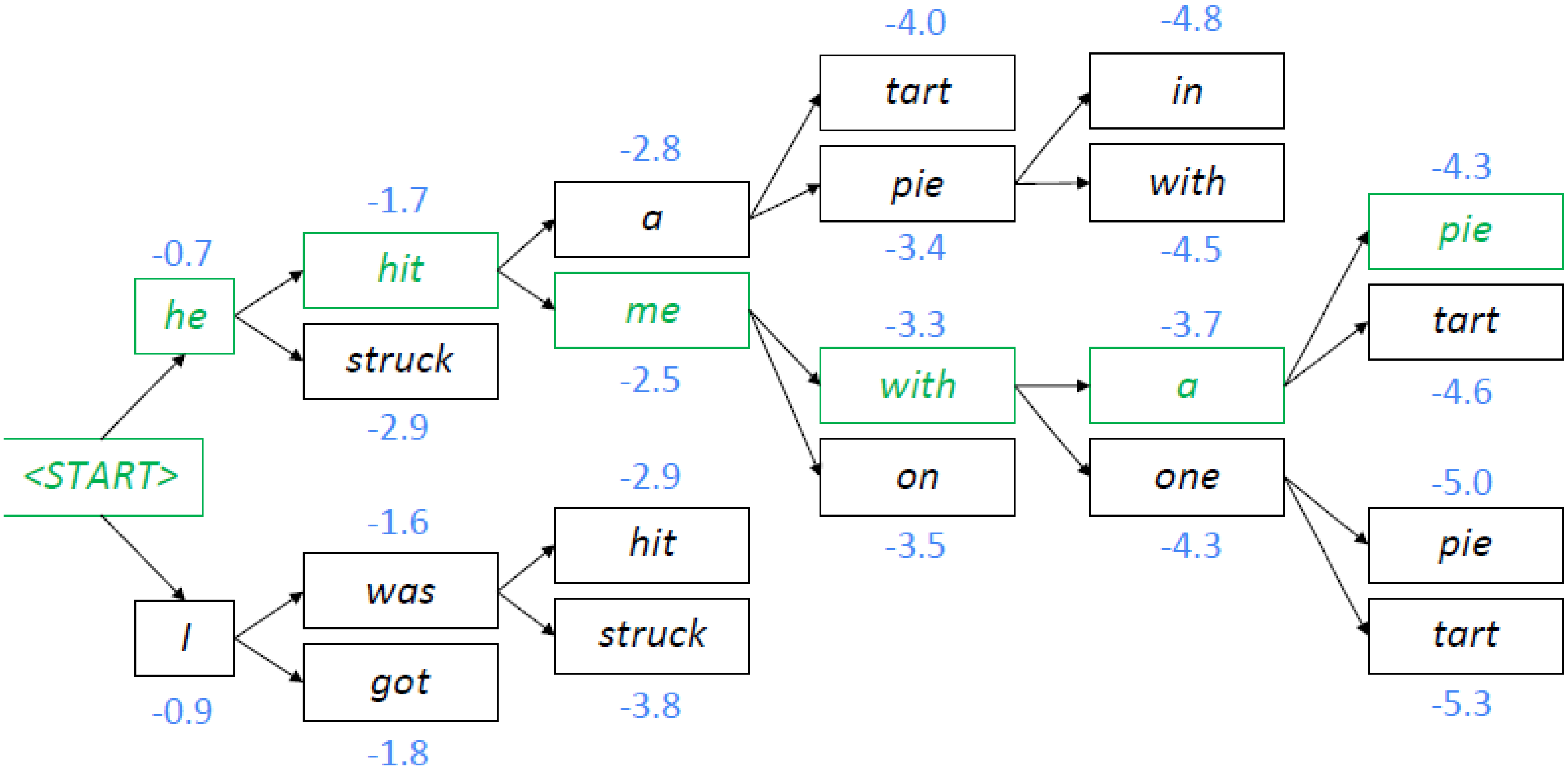
- A search algorithm which aims to find a high-probability sequence (not necessarily the optimal sequence, though) by tracking multiple possible sequences at once.

- Core idea: On each step of decoder, keep track of the k most probable partial sequences (which we call *hypotheses*)
  - k is the beam size

- After you reach some stopping criterion, choose the sequence with the highest probability (factoring in some adjustment for length)

Recap: beam search decoding

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



What's the effect of changing beam size *k*?

- Small *k* has similar problems to greedy decoding (*k*=1)
  - Ungrammatical, unnatural, nonsensical, incorrect

- Larger *k* means you consider more hypotheses
  - Increasing *k* reduces some of the problems above
  - Larger *k* is more computationally expensive
  - But increasing *k* can introduce other problems:
    - For NMT, increasing *k* too much decreases BLEU score (Tu et al, Koehn et al). This is primarily because large-k beam search produces too-short translations (even with score normalization!)
    - In open-ended tasks like chit-chat dialogue, large *k* can make output more generic (see next slide)

Neural Machine Translation with Reconstruction, Tu et al, 2017 https://arxiv.org/pdf/1611.01874.pdf
Six Challenges for Neural Machine Translation, Koehn et al, 2017 https://arxiv.org/pdf/1706.03872.pdf

Effect of beam size in chitchat dialogue

Human chit-chat partner

Beam size	Model response
1	*I love to eat healthy and eat healthy*
2	*That is a good thing to have*
3	*I am a nurse so I do not eat raw food*
4	*I am a nurse so I am a nurse*
5	*Do you have any hobbies?*
6	*What do you do for a living?*
7	*What do you do for a living?*
8	*What do you do for a living?*

Low beam size:
More on-topic but nonsensical;
bad English

High beam size:
Converges to safe, "correct" response, but it's generic and less relevant

Sampling-based decoding

Both of these are **more efficient** than beam search – no multiple hypotheses

- Pure sampling
  - On each step *t*, randomly sample from the probability distribution P_t to obtain your next word.
  - Like greedy decoding, but sample instead of argmax.

- Top-n sampling*
  - On each step *t*, randomly sample from P_t, restricted to just the top-n most probable words
  - Like pure sampling, but truncate the probability distribution
  - *n=1* is greedy search, *n=V* is pure sampling
  - Increase *n* to get more diverse/risky output
  - Decrease *n* to get more generic/safe output

*Usually called top-*k* sampling, but here we're avoiding confusion with beam size *k*

Softmax temperature

- Recall: On timestep t, the LM computes a prob dist P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$

$$P_t(w) = \frac{\exp(s_w)}{\sum_{w' \in V} \exp(s_{w'})}$$

- You can apply a *temperature hyperparameter* τ to the softmax:

$$P_t(w) = \frac{\exp(s_w/\tau)}{\sum_{w' \in V} \exp(s_{w'}/\tau)}$$

- Raise the temperature τ: P_t becomes more uniform
  - Thus more diverse output (probability is spread around vocab)
- Lower the temperature τ: P_t becomes more spiky
  - Thus less diverse output (probability is concentrated on top words)

Note: softmax temperature is not a decoding algorithm!
It's a technique you can apply at test time, in conjunction with a decoding algorithm (such as beam search or sampling)

Definition of the temperature parameter

- The temperature parameter is a hyperparameter that can be used to control the randomness and creativity of the generated text in a generative language model. It is used to adjust the probabilities of the predicted words in the softmax output layer of the model. The temperature parameter is defined as the inverse of the scaling factor used to adjust the logits before the softmax function is applied.

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Impact on the randomness and creativity of generated text

- When the temperature is set to a low value, the probabilities of the predicted words are sharpened, which means that the most likely word is selected with a higher probability. This results in more conservative and predictable text, as the model is less likely to generate unexpected or unusual words. On the other hand, when the temperature is set to a high value, the probabilities of the predicted words are flattened, which means that all words are more equally likely to be selected. This results in more creative and diverse text, as the model is more likely to generate unusual or unexpected words.

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Example 1: Temperature = 0.1

- Prompt: "Please write two sentences about why ice cream is better than cake. Keep the tone light-hearted and amusing with a temperature of 0.1."
- Result: "Ice cream is better than cake because it's cold, creamy, and refreshing. Cake, on the other hand, is just a baked dessert with frosting."

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Example 2: Temperature = 0.7

- ***Prompt:*** "Please write two sentences about why ice cream is better than cake. Keep the tone light-hearted and amusing with a temperature of 0.7."
- ***Result:*** "Ice cream triumphs over cake because it's like a refreshing vacation for your taste buds, whisking you away to an endless world of flavors and textures. Cake, while tasty, is more of a staycation that doesn't quite reach the adventurous heights that ice cream offers."

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Example 3: Temperature = 1.0

- <u>Prompt:</u> "Please write two sentences about why ice cream is better than cake. Keep the tone light-hearted and amusing with a temperature of 1.0."
- <u>Result:</u> "Ice cream outshines cake as the ultimate dessert, as it offers a deliciously cool escape from reality, transforming ordinary moments into whimsical, frosty adventures. Cake, although scrumptious, remains anchored in tradition, lacking the same level of excitement and novelty that ice cream brings to the table."

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Example 4: Temperature = 1.5

- Prompt: "Please write two sentences about why ice cream is better than cake. Keep the tone light-hearted and amusing with a temperature of 1.5."
- Result: "Ice cream reigns supreme in the dessert kingdom, not only for its frosty allure but for its uncanny ability to shape-shift into fantastic forms like sundae mountains, banana split valleys, and milkshake rivers. Cake, though a trusty standby, pales in comparison, its static layers and familiar frosting unable to compete with the boundless creativity that ice cream inspires."

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Temperature parameter

- These examples demonstrate how the temperature parameter can influence the creativity, tone, and content of the generated text in generative language models. By adjusting the temperature, users can control the output to match their specific needs and preferences.

Reference: https://www.linkedin.com/pulse/understanding-leveraging-temperature-parameter-language-ziga-bracko/

Decoding algorithms: in summary

- Greedy decoding is a simple method; gives low quality output
- Beam search (especially with high beam size) searches for high-probability output
  - Delivers better quality than greedy, but if beam size is too high, can return high-probability but unsuitable output (e.g. generic, short)
- Sampling methods are a way to get more diversity and randomness
  - Good for open-ended / creative generation (poetry, stories)
  - Top-n sampling allows you to control diversity
- Softmax temperature is another way to control diversity
  - It’s not a decoding algorithm! It's a technique that can be applied alongside any decoding algorithm.