

Notes

20 June 2024 22:26

Outline:

Today: Natural Language Generation

1. What is NLG?
2. A review: neural NLG model and training algorithm
3. Decoding from NLG models
4. Training NLG models
5. Evaluating NLG Systems
6. Ethical Considerations

Stanford

What is Natural language Generation?

- NLP = Natural Language Understanding + Natural Language Generation
- NLG deals with part of the system that generates the text output

Examples of Natural Language Generation:

Example Uses of Natural Language Generation

Machine Translation systems
input: utterances in source languages
output: translated text in target languages.

Digital assistant (dialogue) systems use NLG
input: dialog history
output: text that respond / continue the conversation

Summarization systems (for research articles, email, meetings, documents) use NLG
input: long documents
output: summarization of the long documents

Stanford

SOTA NLG system

ChatGPT is an NLG system!
It's general purpose and can do many things.
e.g., Poetry Generation:

Sorting algorithms, oh so wise,
Bringing order to chaotic ties.
Bubble sort, quick sort, merge sort too,
Each one unique, with their own special view.

Bubble sort, gentle as can be,
Compares and swaps, so efficiently.
It moves like waves, across the sea,
Bringing order, so beautifully.

Quick sort, so fast and strong,
Divides and conquers all day long.
With each partition, it grows so wide,
Order is found, on every side.

Merge sort, so neat and trim,
Divides and merges, one by one.
It builds a bridge, to conquer all.
Order is its overall call.

Stanford

Categorization of NLG tasks:

- Categories wrt. The openness of the task – how much dependency an NLG system has on the input to generate the output?
- e.g. Machine translation is closed end task – as it is almost fully dependent on the input
- Summarization is a bit more open ended than translation – as there is scope for creativity but semantics of inputs are still very much necessary.
- Chat-got or story generation are bit more open ended tasks as they are lot of possibilities to diverge from the input semantics

Categorization of NLG tasks

Less Open-ended More Open-ended

Machine Translation Summarization Task-driven Dialog ChitChat Dialog Story Generation

Open-ended generation: the output distribution still has high freedom

Non-open-ended generation: the input mostly determines the output generation.

- More open ended high entropy

Review of NLG:

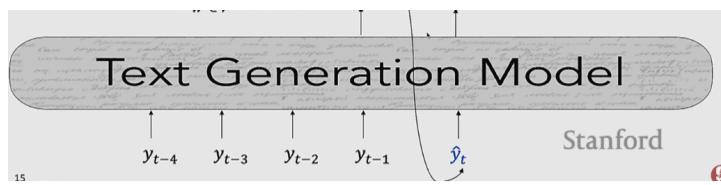
- Autoregressive generation: mostly used for open ended task

Basics of natural language generation

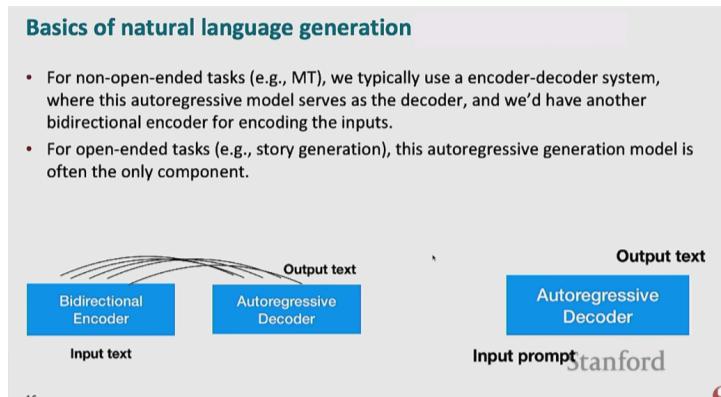
- In autoregressive text generation models, at each time step t , our model takes in a sequence of tokens as input $\{y\}_{<t}$ and outputs a new token, \hat{y}_t
- For model $f(\cdot)$ and vocab V , we get scores $S = f(\{y_{<t}\}, \theta) \in \mathbb{R}^V$

$$P(y_t | \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

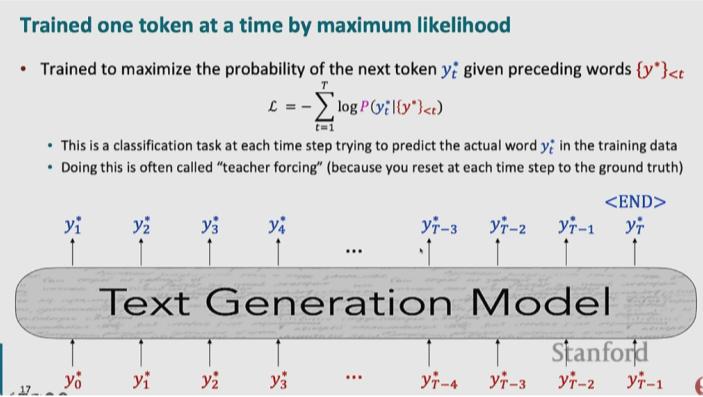
\hat{y}_t



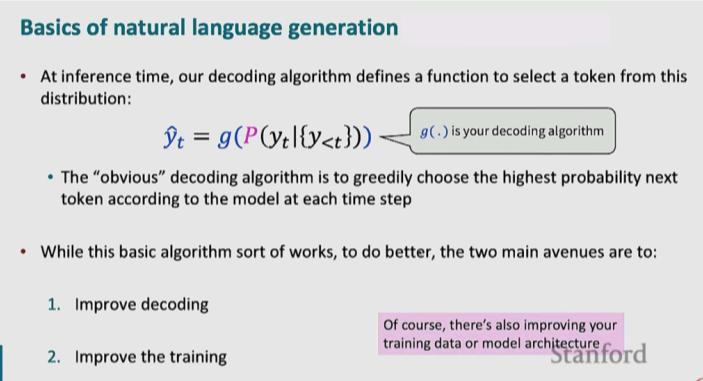
- Encoder-decoder tasks: mostly used by close ended tasks



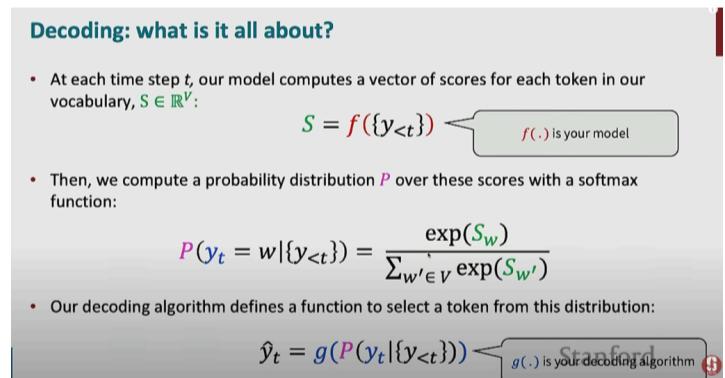
- Training of text generation models:
 - Teacher forcing way



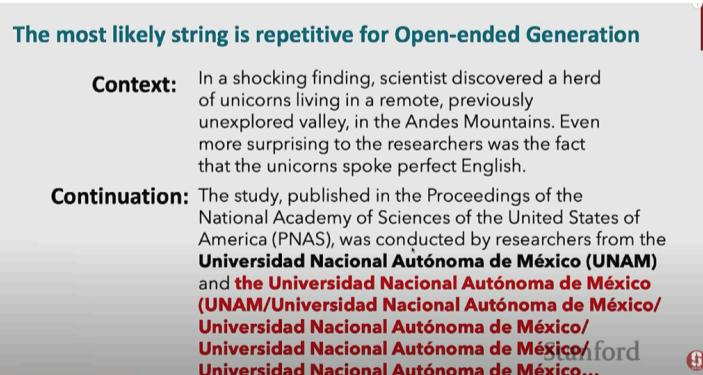
- Decoding:



Decoding from NLG models:



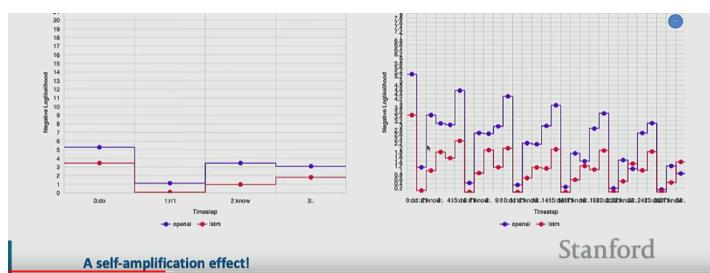
- From Lecture 7 – recall greedy decoding and beam search. These both techniques directly or indirectly are trying to generate the token with highest probability.
 - Such decoding techniques are good for close ended tasks.
 - For open ended task, they end up producing the repetitive tokens



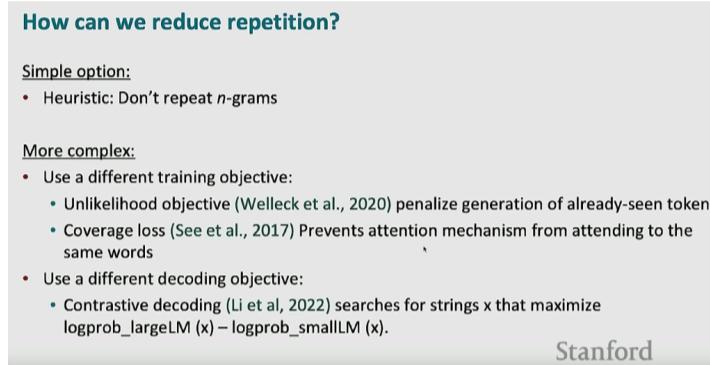
- Why does this repetition happen?
 - Model generates the "best applicable" token at current moment without considering previous/subsequent tokens. As token gets repeated the negative log likelihood decreases /probability of that token increases.

Why does repetition happen?

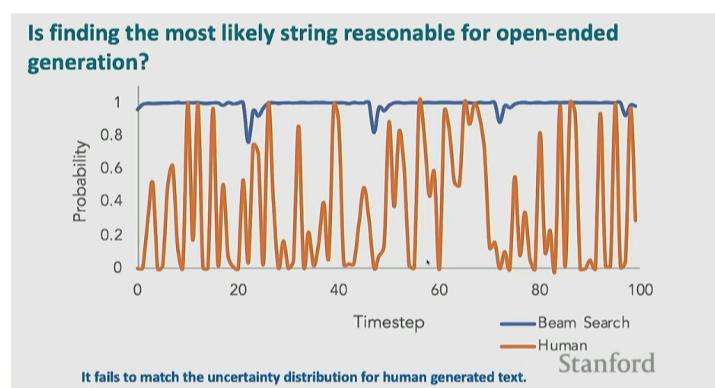
I don't know. I don't know.



- How to reduce repetition?
 - Use n-grams blocking: e.g. if model has decoded I'm happy previously and now you see I'm being decoded the simply block "happy" being decoded. This is not always feasible.
 - Use different training objectives:
 - Unlikelihood loss: penalize the generation of already-seen token during training
 - Coverage loss: repetition happens because model has similar attention patterns over and over again. Coverage loss prevents the model attending to same words
 - Different decoding objective:
 - Contrastive decoding:



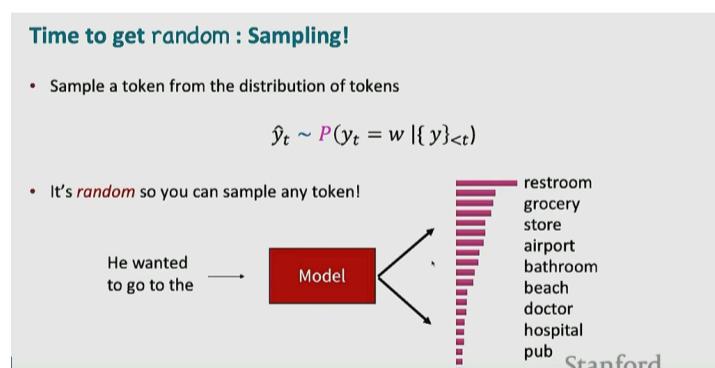
- Is greedy-type of decoding the best way for open ended tasks? ---- NO
 - Here you can see that human "generation" had varying probabilities across the time but the model was always sure about the next word.



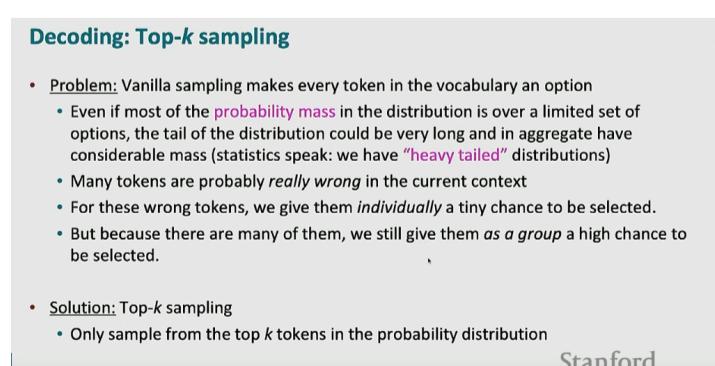
- If not greedy decoding then what???

(random) Sampling based decoding:

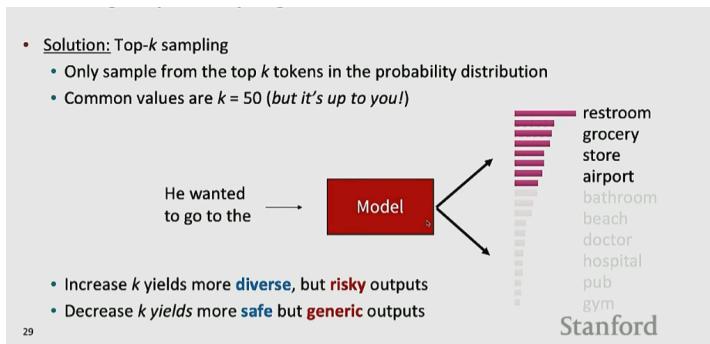
- Greedy decoding would select restroom here (assuming restroom has highest probability) but with random sampling now any of the options from vocab are open



- Problems with Sampling: We never zero out any tokens so all the tokens are valid options!!!! Even undesired/wrong words can be thrown out by the model. ---- Use top-K sampling
- Top-K sampling selects the next token randomly from top-K token only and neglects other tokens having very low prob. (cuts out the tail)
- K is small for close ended tasks and large for open ended tasks

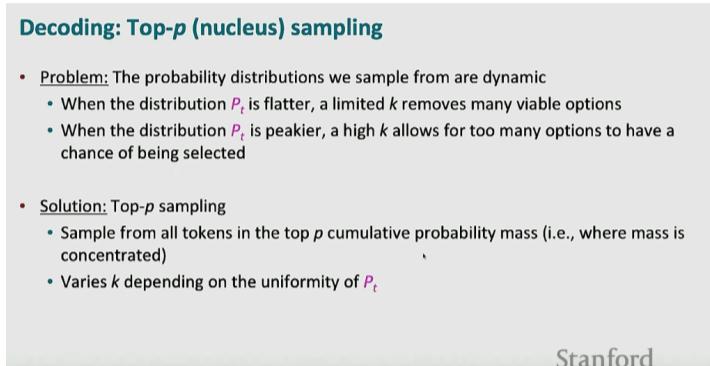


Decoding: Top-k sampling

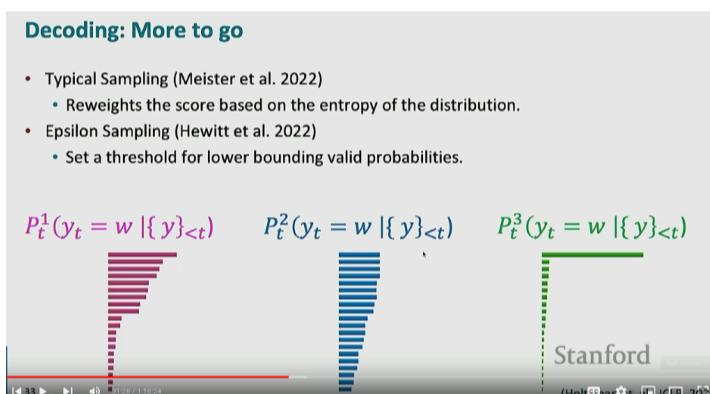


- Is top K sampling good? Nah, not really. It suffers from bad recall/precision depending upon the probability distribution produced by the model.

- Deciding good k value is difficult. We can instead use probability information to decide K for each token...

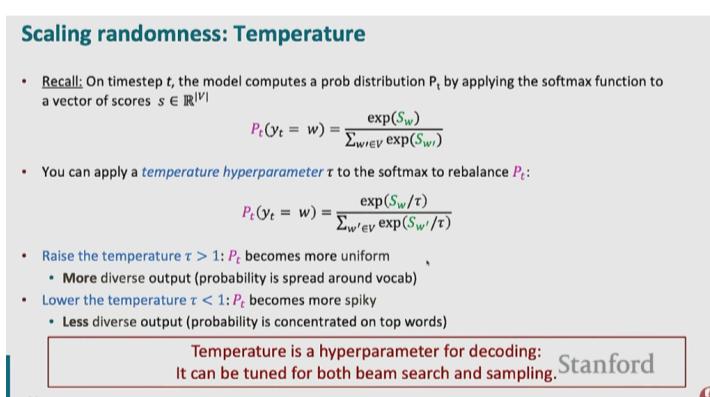


- Other decoding algorithms:
 - Typical sampling: try to generate the tokens with probability closer to the negative entropy of the data distribution.
 - Epsilon Sampling: limit the probabilities being assigned by the model, hence tokens with prob lower than the bound will not be an option to when choosing the next token

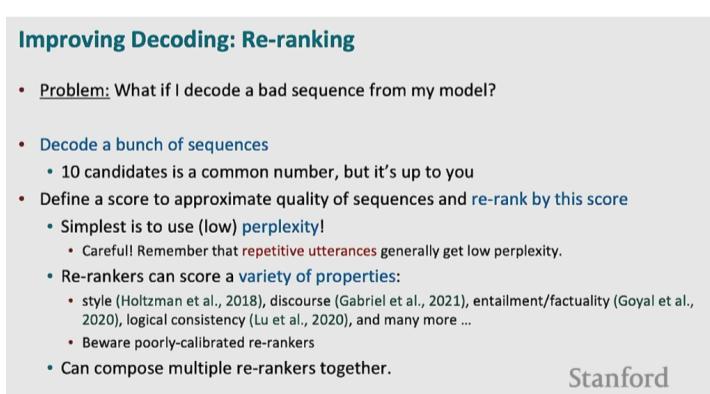


Scaling Randomness: Temperature Parameter

- Temperature parameter does not change the absolute probabilities values. It only scales the relative values i.e. difference in probability values might be more/less depending upon the temperature



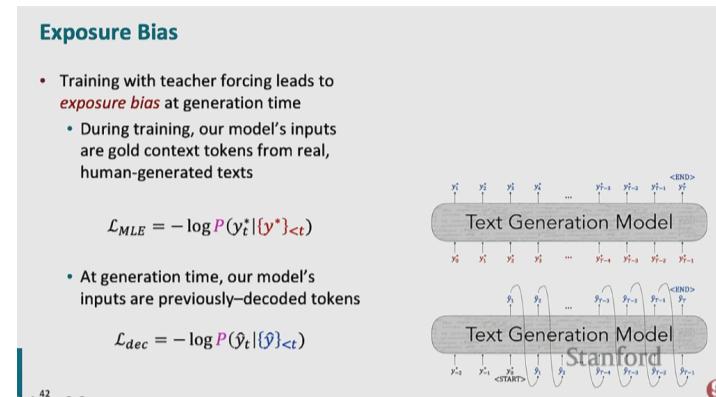
Improving decoding by Reranking



Training the NLG systems

Repetition (during decoding) and Training methodology:

- Models learnt with teacher forcing tend to focus on Mode of the distribution i.e. such models assign high probability to the most repeating string in the dataset. Why does this happen?
--- due to exposure bias
- Exposure Bias:
 - Discrepancy of what model sees during training and testing



- To reduce exposure bias:

- **Scheduled sampling** (Bengio et al., 2015)
 - With some probability p , decode a token and feed that as the next input, rather than the **gold token**.
 - Increase p over the course of training
 - Leads to improvements in practice, but can lead to **strange training objectives**
 - **Dataset Aggregation** (DAgger; Ross et al., 2011)
 - At various intervals during training, generate sequences from your current model
 - Add these sequences to your training set as additional examples
- Basically, variants of the same approach; see: <https://nlpers.blogspot.com/2016/03/a-dagger-by-any-other-name-scheduled.html>

- (the favorite solution) Retrieval Augmented Generation:

- For RAG: (sort of) gold standard tokens are also available during decoding

Exposure Bias Solutions

- **Retrieval Augmentation** (Guu*, Hashimoto*, et al., 2018)
 - Learn to retrieve a sequence from an existing corpus of human-written prototypes (e.g., dialogue responses)
 - Learn to edit the retrieved sequence by adding, removing, and modifying tokens in the prototype – this will still result in a more “human-like” generation
- **Reinforcement Learning**: cast your text generation model as a Markov decision process
 - State s is the model’s representation of the preceding context
 - Actions a are the words that can be generated
 - Policy π is the decoder
 - Rewards r are provided by an external score
 - Learn behaviors by rewarding the model when it exhibits them – go study CS 234

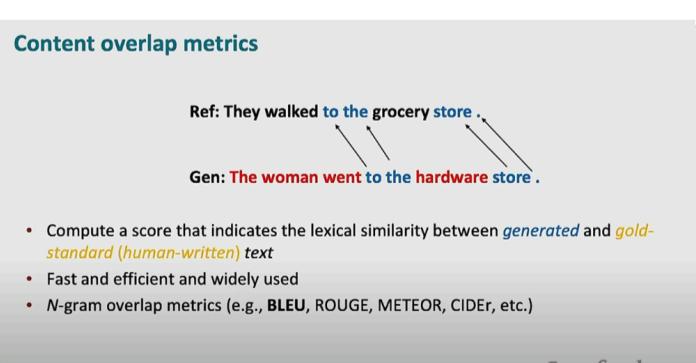
Training: Takeaways

- Training: Takeaways**
- **Teacher forcing** is still the main algorithm for training text generation models
 - **Exposure bias** causes text generation models to **lose coherence** easily
 - Models must learn to recover from their own bad samples
 - E.g., scheduled sampling, DAgger
 - Or not be allowed to generate bad text to begin with (e.g., retrieval + generation)
 - Training with RL can allow models to learn behaviors that are preferred by human preference / metrics.

Evaluating NLG Systems

- Mainly 3 methods
 - Content Overlap
 - Model-based metrics
 - Human Evaluation

Content Overlap:



- Drawbacks of N-gram Overlap metrics:

- They don't have a concept of semantic relatedness. I.e. If correct answer to the question is "Heck Yes" then "Heck No" is also rated as correct because of the overlap of "heck".
- Similarly, "yup" will be considered as wrong because of no overlap with "heck yes"

N-gram overlap metrics

Word overlap-based metrics (BLEU, ROUGE, METEOR, CIDEr, etc.)

- They're **not ideal** for machine translation
- They get progressively **much worse** for tasks that are more open-ended than machine translation
 - **Worse** for **summarization**, as longer output texts are harder to measure
 - **Much worse** for **dialogue**, which is more open-ended than summarization
 - **Much, much worse** **story generation**, which is also open-ended, but whose sequence length can make it seem you're getting decent scores!

Stanford

Model-based metrics:

Model-based metrics to capture more semantics

- Use **learned representations** of words and sentences to compute semantic similarity between generated and reference texts
- No more **n-gram bottleneck** because text units are represented as **embeddings**!
- The embeddings are **pretrained**, distance metrics used to measure the similarity can be **fixed**

Stanford

- How to go from word embeddings to sentence embedding?
 - Averaging the word embeddings (yikes!!)

Model-based metrics: Word distance functions

Vector Similarity
Embedding based similarity for semantic distance between text.

- Embedding Average (Liu et al., 2016)
- Vector Extrema (Liu et al., 2016)
- MEANT (Lo, 2017)
- YISI (Lo, 2019)

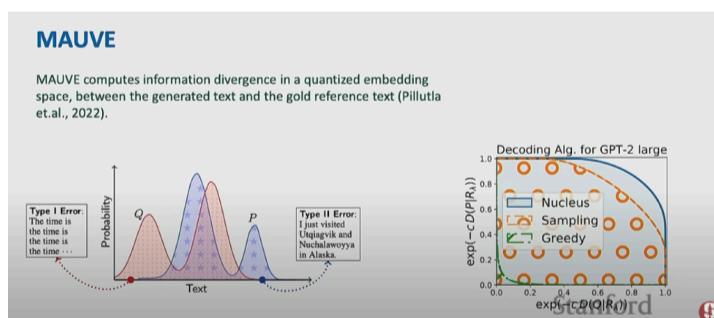
Word Mover's Distance
Measures the distance between two sequences (e.g., sentences, paragraphs, etc.), using word embedding similarity matching.
(Kusner et al., 2015; Zhao et al., 2019)

BERTSCORE
Uses pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity.
(Zhang et al. 2020)

Stanford

Evaluating open-ended NLG systems:

MAUVE Score:



- (Continuous) embeddings are discretized by using e.g. KNN. In such discretized space, we can then plot histograms, find TP/FP etc

