COMP5046
Natural Language Processing

Lecture 8: Language Model and Natural Language Generation

Semester 1, 2019

School of Computer Science

The University of Sydney, Australia

THE UNIVERSITY OF SYDNEY

Caren Han
Caren.Han@sydney.edu.au

0 LECTURE PLAN

Lecture 8: Language Model and Natural Language Generation

1. **Language Model**
2. Traditional Language Model
3. Neural Language Model
4. Natural Language Generation
5. Other NLG Approaches
6. Language Model and NLG Evaluation

1 Language Model

What is Language Model

- is the task of predicting what word comes next based on the given words.
- is a probabilistic model which predicts the probability that a sequence of tokens belongs to a language.

Can you come _______

here
there
close
hear
their
further
...

$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ $\qquad$ x^1, x^2, x^3

Given a sequence of words, ***Can, you, come,*** compute the probability distribution of the next word.

$\boldsymbol{x}^{(t+1)} \longrightarrow$ *can be any word in the vocabulary*

P(Can you please come here?) $>$ *P(Can you please come there?)*

$$P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

e.g. compare ***P(here|can, you, come)*** *and* ***P(there|can, you, come)*** *and …*

1 Language Model

What is Language Model

- is the task of predicting what word comes next based on the given words.
- is a probabilistic model which predicts the probability that a sequence of tokens belongs to a language.

P(Can, you, please, come, here,?) $>$ *P(Can, you, please, come, there,?)*

compare ***P(here|can, you, come)*** *and* ***P(there|can, you, come)***

$$\begin{aligned} P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) &= P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} |\, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} |\, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)}) \\ &= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} |\, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)}) \end{aligned}$$

Conditional Probability

THE UNIVERSITY OF SYDNEY

Language Modeling in NLP

- The probabilities returned by a language model are mostly useful to compare the likelihood that different sentences are "good sentences". This is useful in many practical tasks, for example:

Spell correction/Automatic Speech Recognition

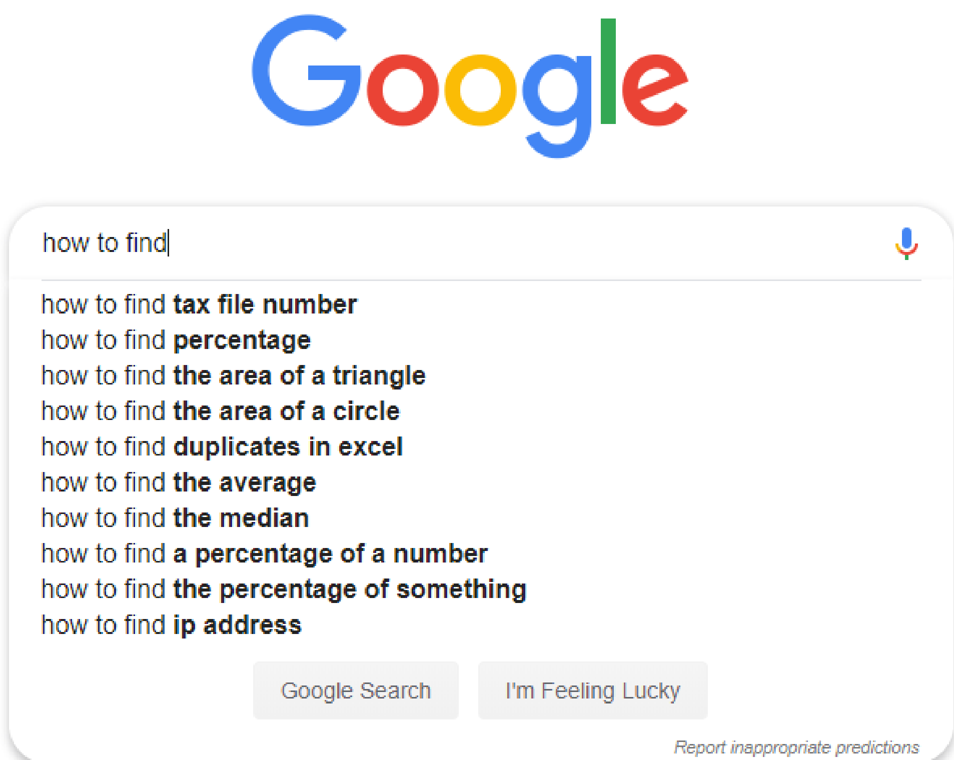
- I would like to read that ***boot*** *Closest words= [book, boog, boat, …]*

Natural Language Generation

- Dialogue (chit chat and task-based)
- Abstractive Summarisation
- Machine Translation
- Creative Writing: Story Telling, …

1 Language Model

Do we use Language Model?

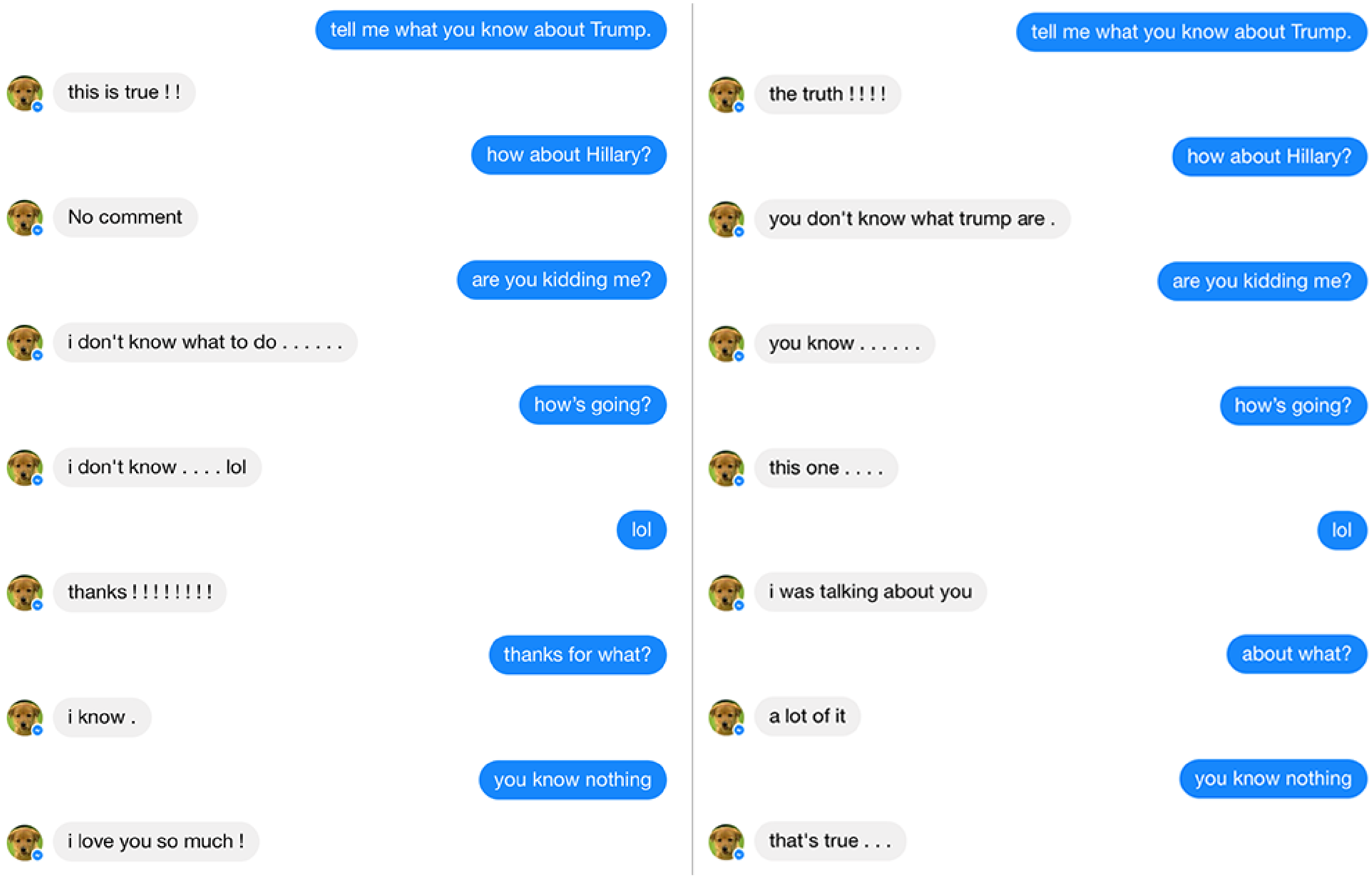


1 Language Model

Yes but sometimes fail..

1 Language Model

Language Model in Dialog System



Language Modeling in Natural Language Generation

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

Natural Language Generation

- Dialogue (chit chat and task-based)

  x=dialogue history, y=next utterance

- Abstractive Summarisation

  x=input text, y=summarized text

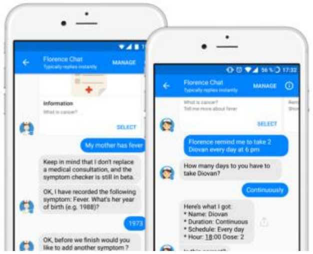
- Machine Translation

  x=source sentence, y=target sentence

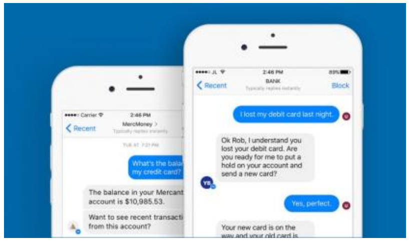
Tips for using Language Model (you already knew!)

It is extremely important to collect and learn the model with the corpus that includes documents about the domain that your system/application will be used.

Medical Documents



Financial Documents



0 LECTURE PLAN

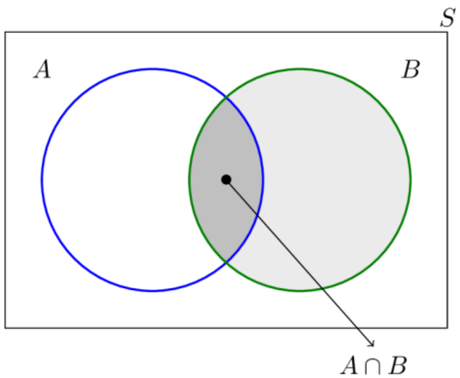
Lecture 8: Language Model and Natural Language Generation

1. Language Model
2. **Traditional Language Model**
3. Neural Language Model
4. Natural Language Generation
5. Other NLG Approaches
6. Language Model and NLG Evaluation

Statistical Language Model (SLM)

- ***Conditional Probability***: the probability of an event (A), given that another (B) has already occurred.

Conditional Probability



$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$p(B|A) = P(A,B)/P(A)$$

$$\mathbf{P(A,B) = P(A)*P(B|A)}$$

Traditional Language Models

Statistical Language Model (SLM) *P(A,B) = P(A)*P(B|A)*

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

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} |\, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} |\, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} |\, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

"An adorable little boy is spreading smiles"

P(An, adorable, little, boy, is, spreading, smiles)
= P(An) × P(adorable|An) × P(little|An adorable) × P(boy|An adorable little) × P(is|An adorable little boy) × P(spreading|An adorable little boy is) × P(smiles|An adorable little boy is spreading)

THE UNIVERSITY OF SYDNEY

Statistical Language Model (SLM) *P(A,B) = P(A)*P(B|A)*

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

$$P(\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)}|\ \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)}|\ \boldsymbol{x}^{(T-1)},\ldots,\boldsymbol{x}^{(1)})$$
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)}|\ \boldsymbol{x}^{(t-1)},\ldots,\boldsymbol{x}^{(1)})$$

"An adorable little boy is"

P(is|An adorable little boy)?

Trained Corpus

An adorable little boy is ....
..
.... An adorable little boy is
..
..
An adorable little boy laughed
..
..
..

Simplest method

= Count(An adorable little boy is)/Count(An adorable little boy)
= 30/100 =0.3

Q: What if there is no 'An adorable little boy is' phrase in the corpus?

2 Traditional Language Models

N-gram Language Models

- *An N-gram is a sequence of N words.*
- *An N-gram model predicts the probability of a given N-gram within any sequence of words in the language.*

"An adorable little boy is spreading smiles"

A ***n*-gram** is a chunk of *n* consecutive words.

- **uni**grams : an, adorable, little, boy, is, spreading, smiles
- **bi**grams : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles
- **tri**grams : an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles
- **4-**grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles

N-gram Language Models: Exercise

- *Assume that we learn a **tri**gram language model*

"~~An adorable little boy~~ is spreading ___?___"

n-1 words only

(3-1) words only

P(w|is spreading) =

Count(is spreading w) / Count(is spreading)

Trained Corpus

boy is spreading smile………………
…………………………………………………
………….. boy is spreading rumours
…………………………………………………
…………………………………………………
An adorable little boy is spreading
…………………………………………………
…………………………………………………
…………………………………………………
…………………………………………………
…………………………………………………

P(rumours|is spreading)
= Count(is spreading rumours)/Count(is spreading)
= 500/1000 =0.5

P(smiles|is spreading)
= Count(is spreading smiles)/Count(is spreading)
= 200/1000 =0.2

THE UNIVERSITY OF SYDNEY

N-gram Language Models: Beautiful Formula ☺

- ***Simplifying assumption**: the next word, $\boldsymbol{x}^{(t+1)}$,, depends only on the* ***preceding n-1 words**.*

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}})$$

- *How do we get these n-gram and (n-1)-gram probabilities?*

Counting them!

$$\approx \frac{\mathrm{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\mathrm{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

N-gram Language Model Limitation: Trade-off Issue

- *Mostly, n=2 works better than n=1 in n-gram language model*
- *We learned a **tri**gram language model*

"~~An adorable little boy~~ is spreading ___?___"

n-1 words only

(3-1) words only

- *Find the optimal **n is important!** (OOV issue or Model Size issue)*

P(w|is spreading) =

Count(is spreading w) / Count(is spreading)

Need to store count for all n-grams that you saw in the corpus.

If you increase n or corpus,
the model size will be increased!

N-gram Language Model Limitation: Zero Count Issue

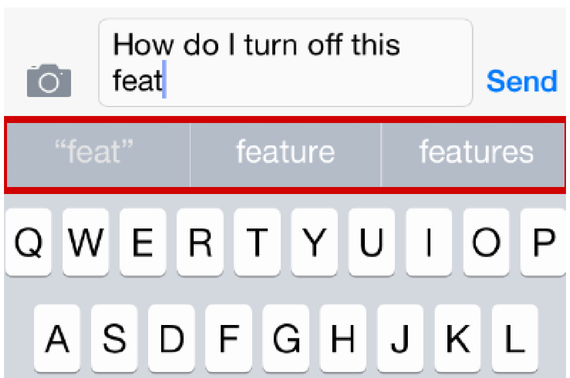
P(w|is spreading) =
*Count(**is spreading w**) / Count(**is spreading**)*

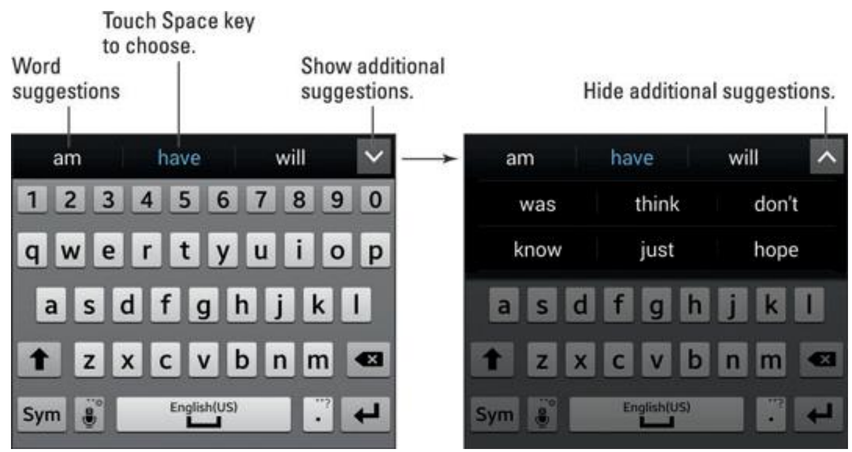
1. *What if the '**is spreading w**' phrase never occurred in the corpus? The probability will be 0.*
   - *Alternative solution: Smoothing (Add small δ to the count for every w in the corpus)*

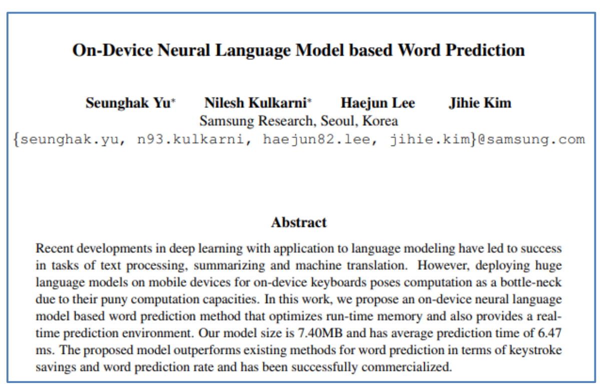
2. *What if the '**is spreading**' phrase never occurred in the corpus? It is impossible to calculate the probability for any **w**.*
   - *Alternative solution: Backoff (Just condition on "spreading" instead)*

Try some Language Model – Word Prediction

Generating text with a n-gram







On-Device Neural Language Model based Word Prediction

Seunghak Yu* Nilesh Kulkarni* Haejun Lee Jihie Kim
Samsung Research, Seoul, Korea
{seunghak.yu, n93.kulkarni, haejun82.lee, jihie.kim}@samsung.com

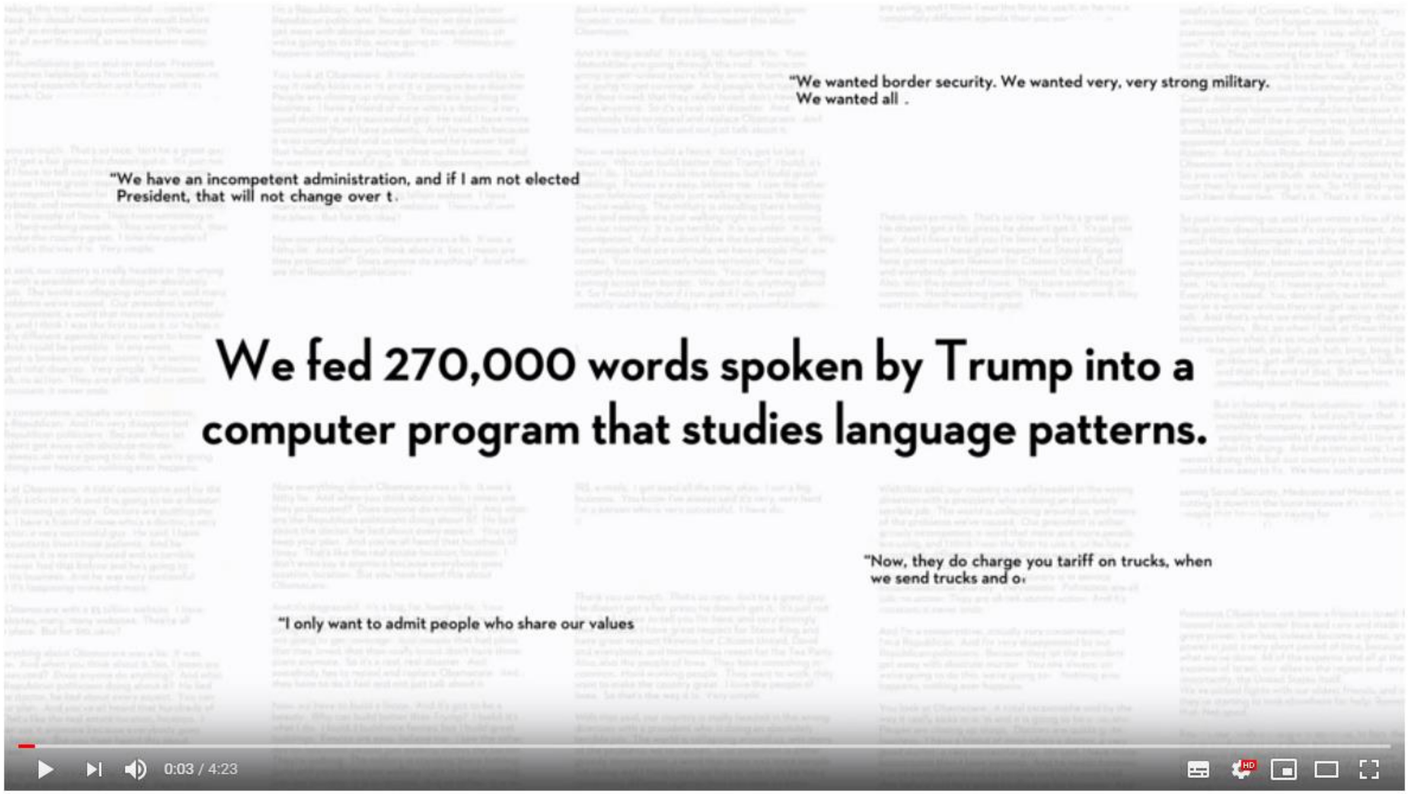
Abstract

Recent developments in deep learning with application to language modeling have led to success in tasks of text processing, summarizing and machine translation. However, deploying huge language models on mobile devices for on-device keyboards poses computation as a bottle-neck due to their puny computation capacities. In this work, we propose an on-device neural language model based word prediction method that optimizes run-time memory and also provides a real-time prediction environment. Our model size is 7.40MB and has average prediction time of 6.47 ms. The proposed model outperforms existing methods for word prediction in terms of keystroke savings and word prediction rate and has been successfully commercialized.

Traditional Language Models

Try some Language Model – Computer Generated

Generating text with a neural language model



https://youtu.be/EFHyzuqjaok

0 LECTURE PLAN

Lecture 8: Language Model and Natural Language Generation

1. Language Model
2. Traditional Language Model
3. **Neural Language Model**
4. Natural Language Generation
5. Other NLG Approaches
6. Language Model and NLG Evaluation

3 Neural Language Model

(Window-based) Neural Language Model

“~~An adorable little~~ boy is spreading ___?___”

fixed window
window size =3

Pros

No Trade-off issue

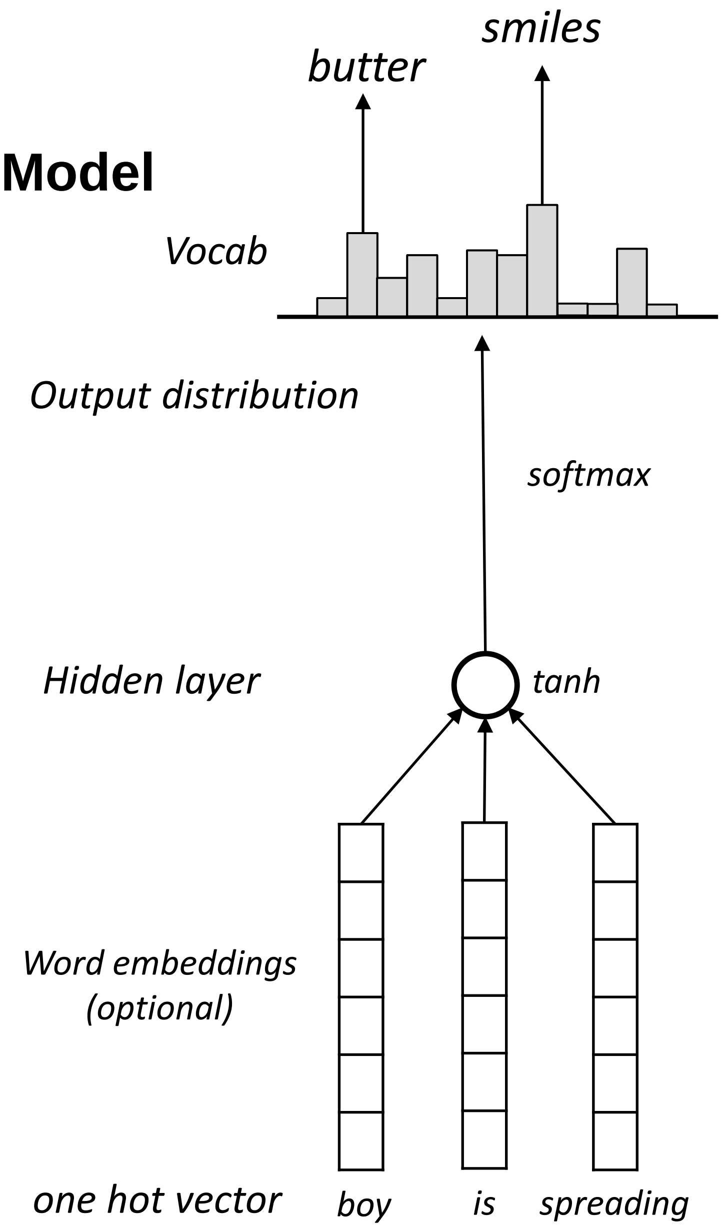
Cons

- *Window size selection issue*

(increasing window size enlarges W)

- *Input vectors are multiplied by completely different weights in W*

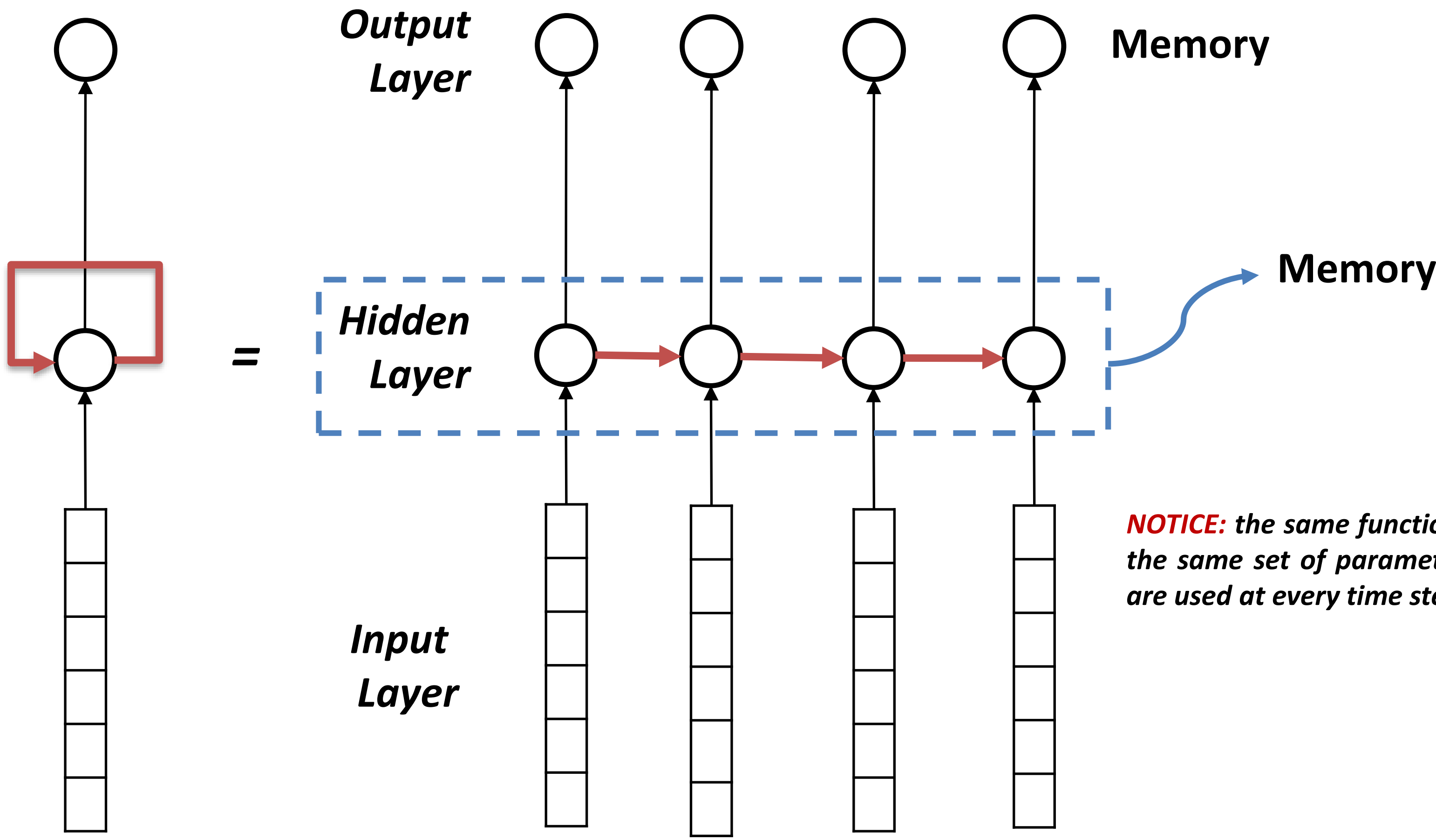
(No symmetry in how the inputs are processed)



Neural Language Model

Recap: RNN (Recurrent Neural Network)

Neural Network + Memory = Recurrent Neural Network

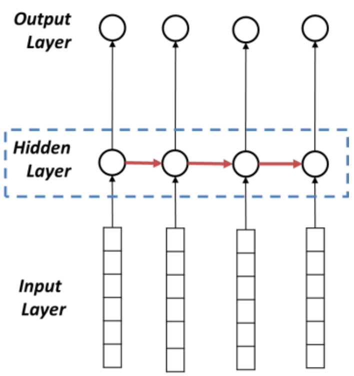


***NOTICE:** the same function and the same set of parameters W are used at every time step*

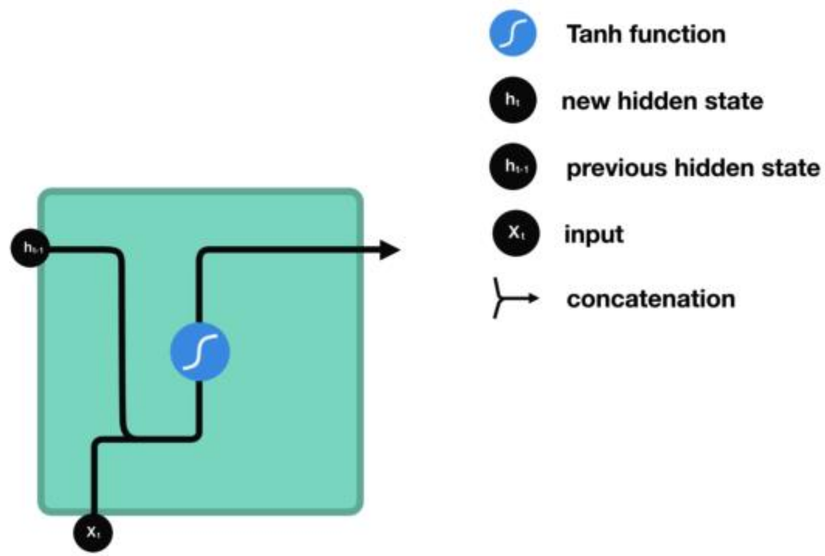
3 Neural Language Model

Recap: RNN (Recurrent Neural Network)

Neural Network + Memory = Recurrent Neural Network



NOTICE: ***the same function and the same set of parameters W are used at every time step***



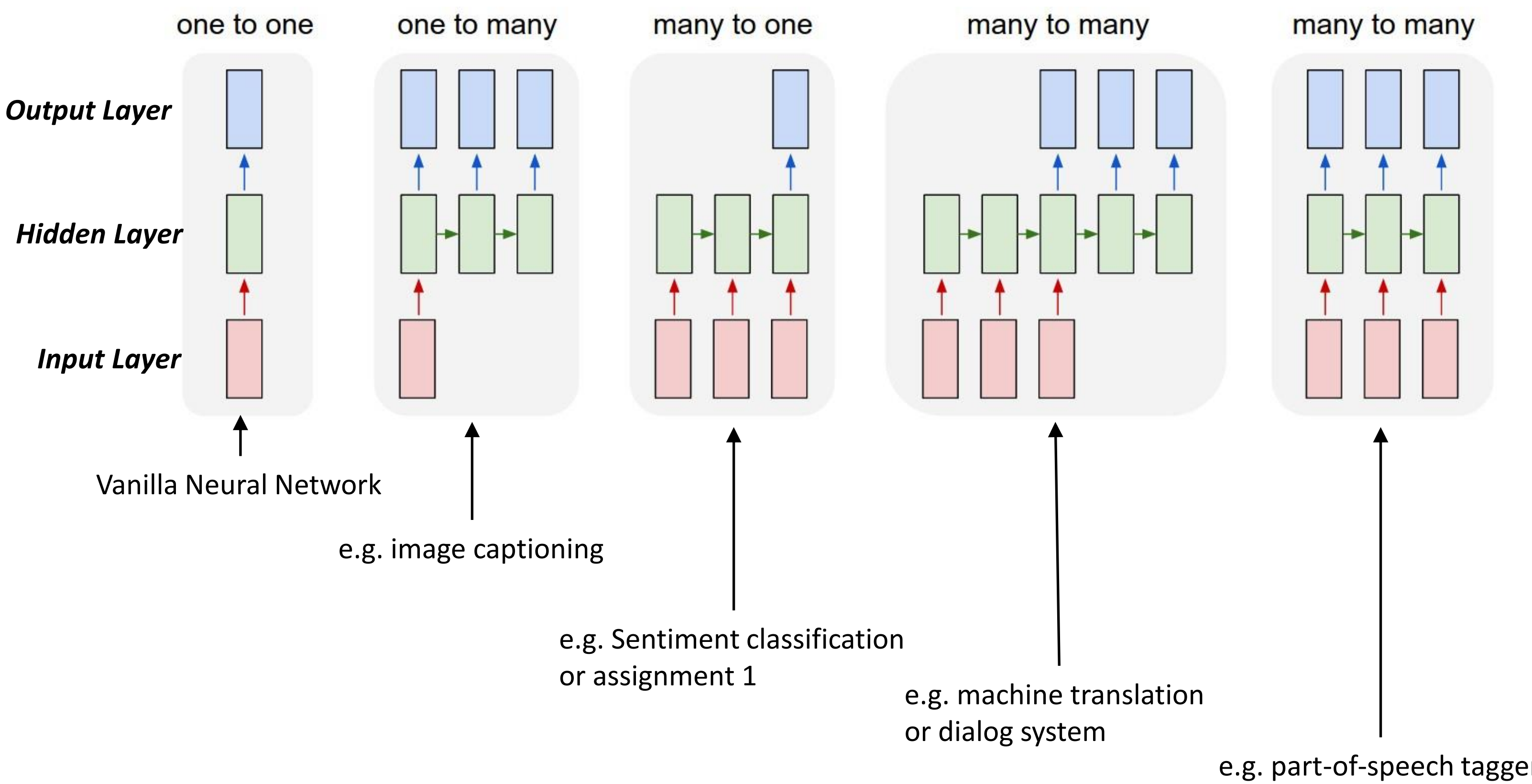
Previous state *input*

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

New hidden state *A function with parameters W*

Recap: RNN Types

Neural Network + Memory = Recurrent Neural Network

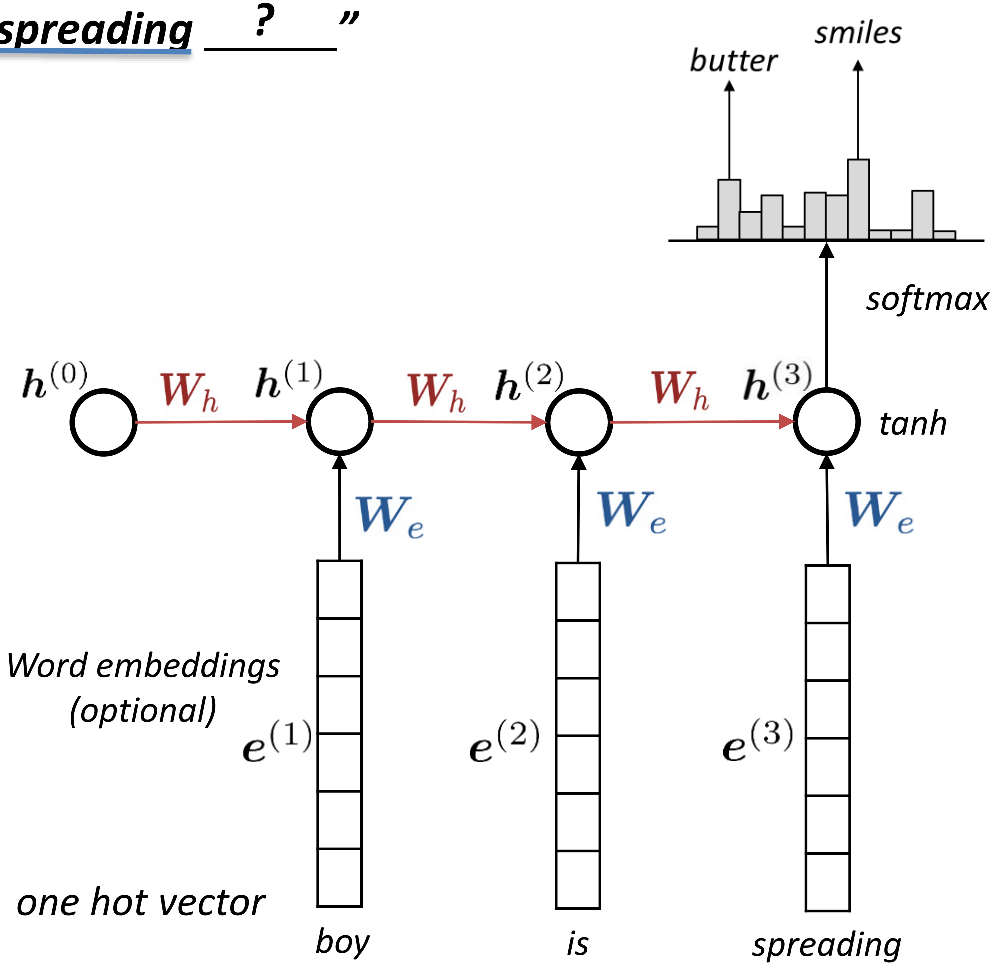


3 Neural Language Model

RNN-based Language Model

"~~An adorable little~~ boy is spreading __?__ "

$$h^t = tanh(W_h h^{t-1} + W_e e^t + b_1)$$



RNN-based Language Model

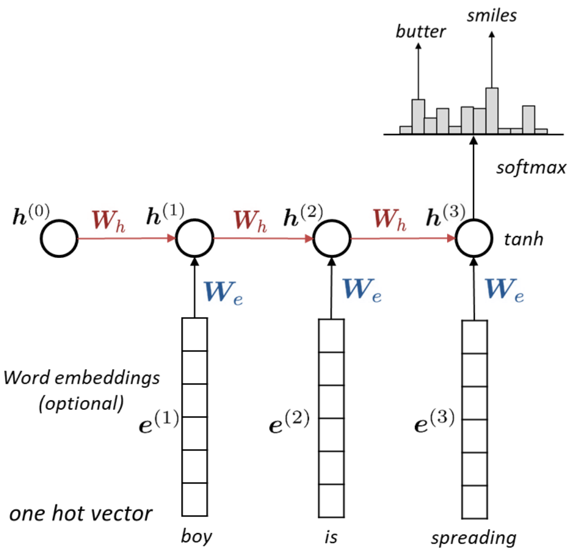
"~~An adorable little~~ boy is spreading ___?___"

Pros

- *Can process any length input*
- *Can use information from many step back*
- *Model size does not increase*
- *Same weights applied on every time step (Symmetry)*

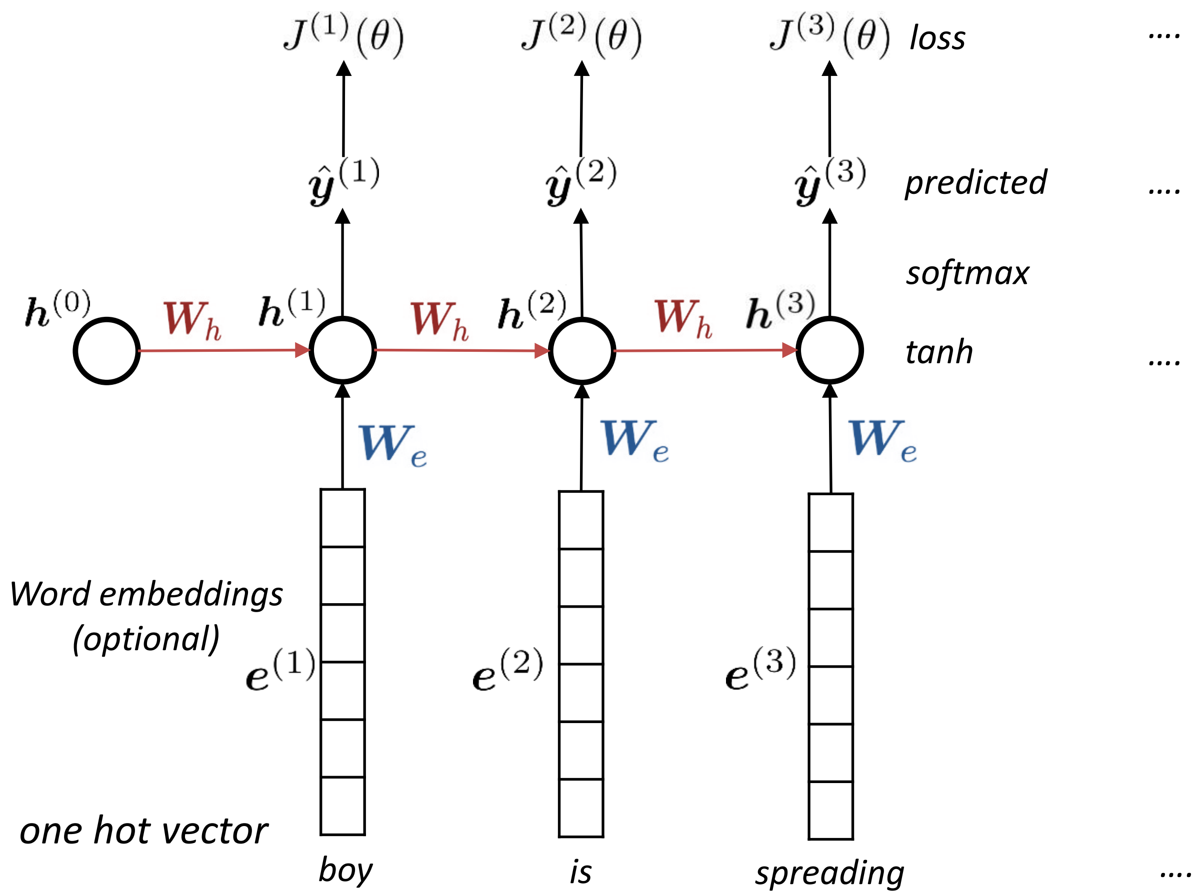
Cons

- *Slow computation*
- *Difficult to access information from many step back (remember what we learned in lecture 4?)*



Training a RNN-based Language Model

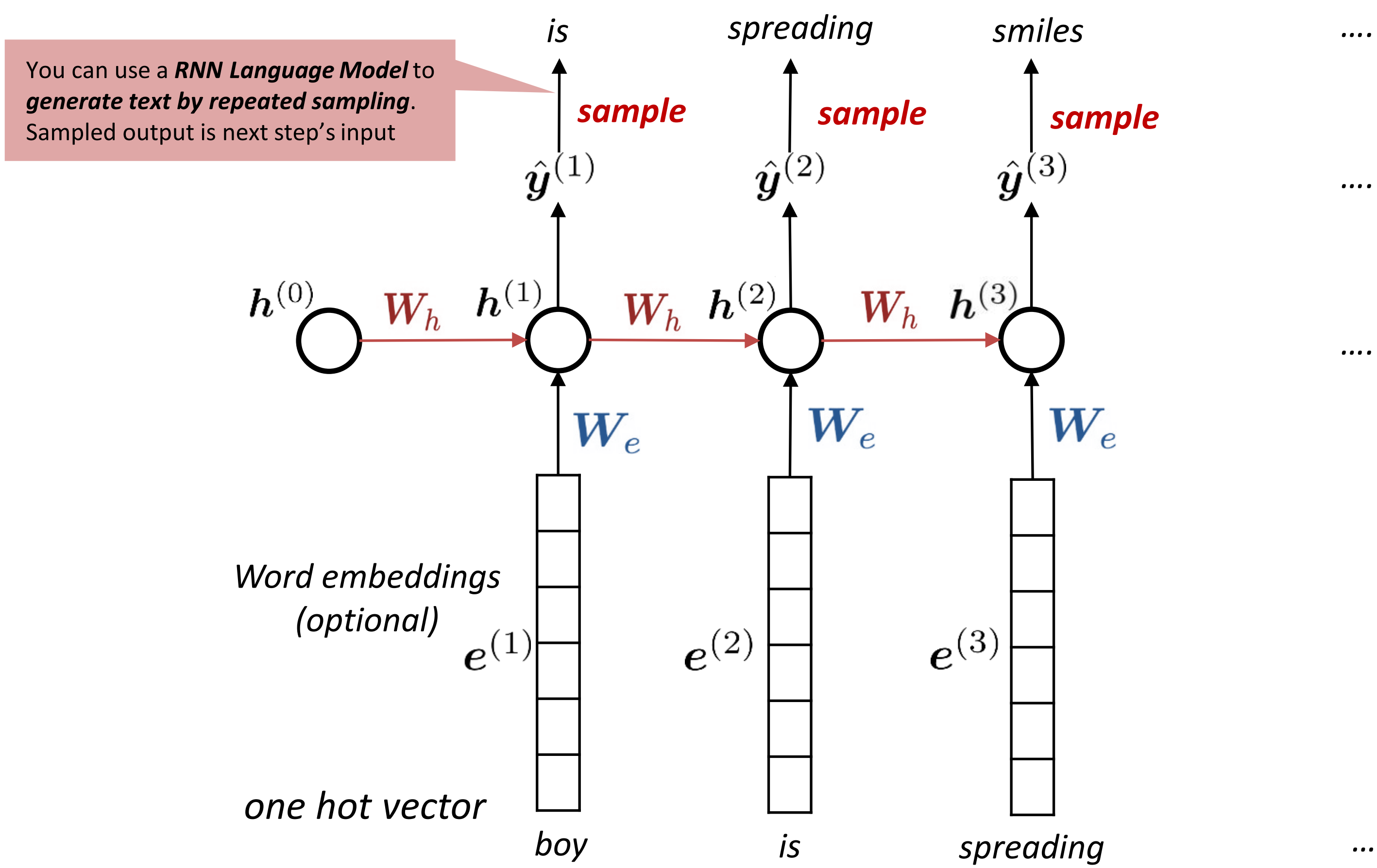
"~~An adorable little~~ boy is spreading ___?___"



3 Neural Language Model

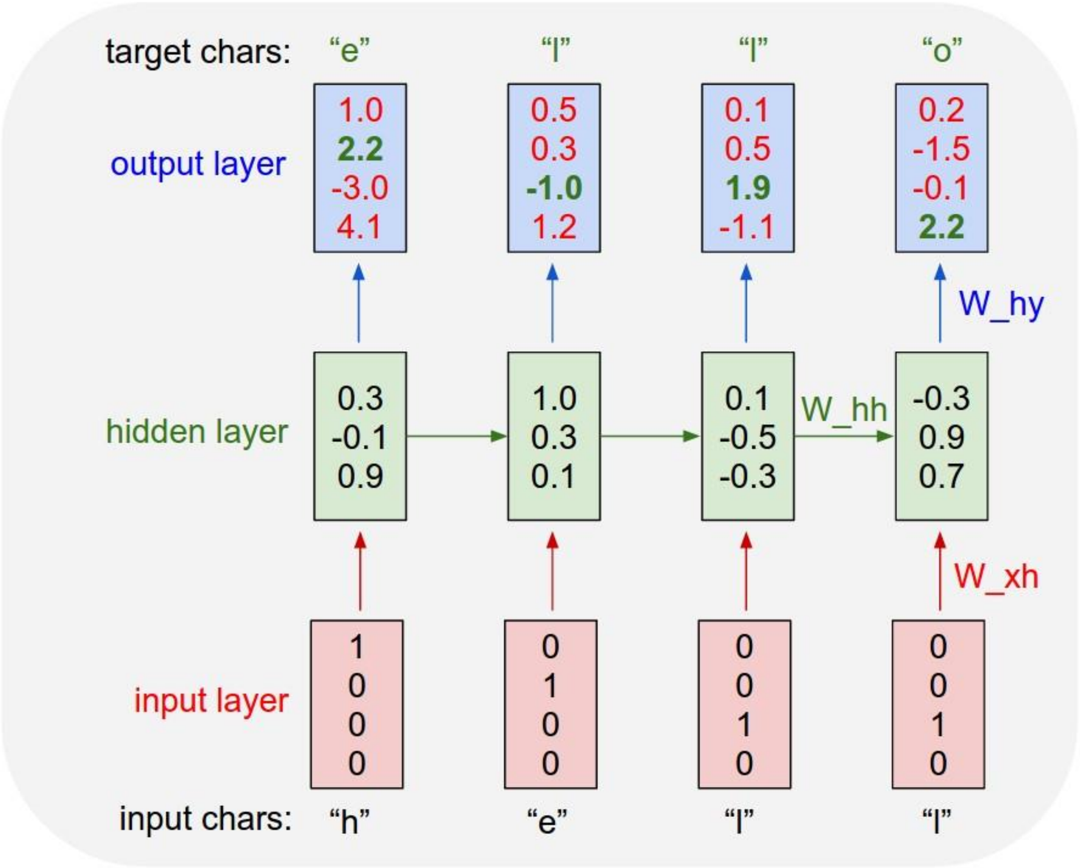
RNN-based Language Model

"~~An adorable little~~ boy is spreading ___?___"

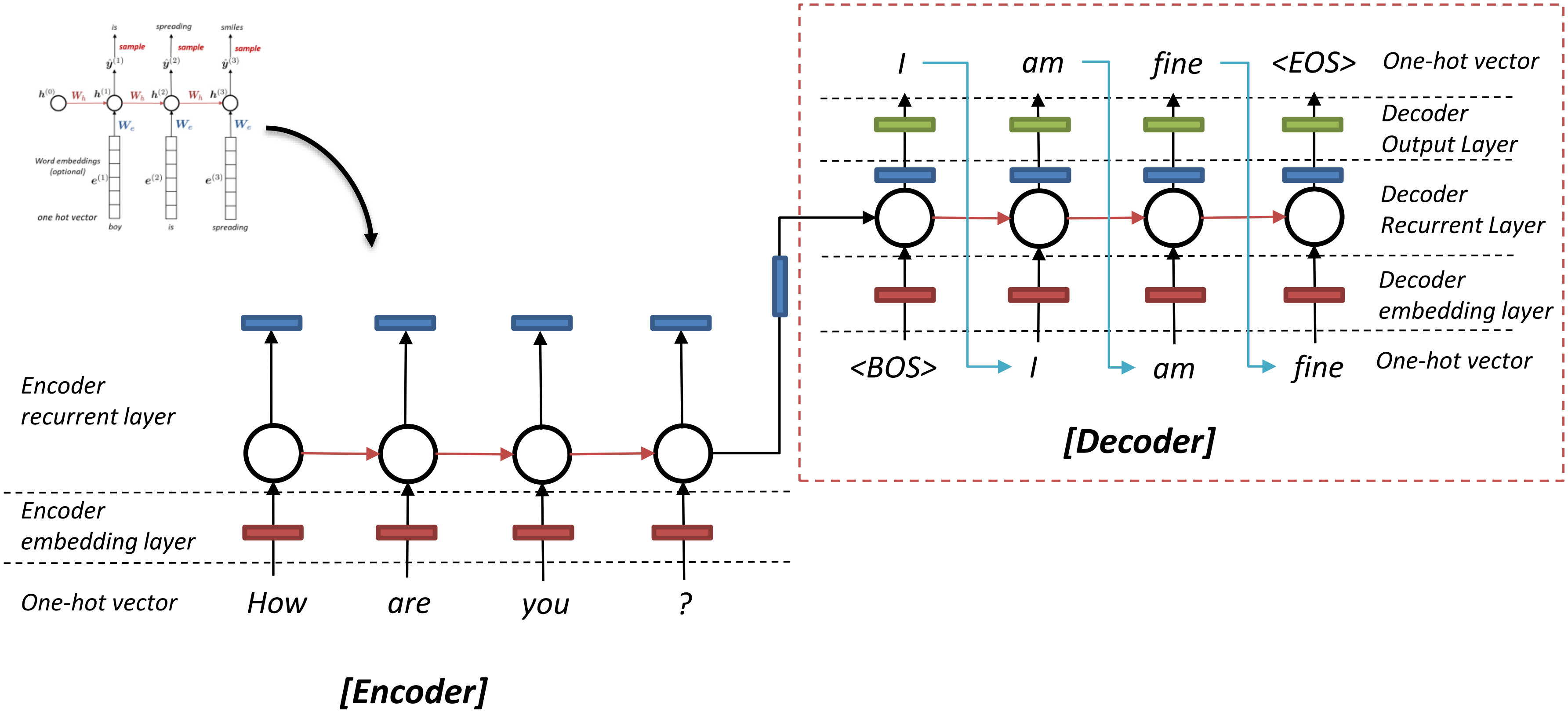


Neural Language Model

Recap: Character-based RNN Language Model



Seq2Seq Model with trained language model



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

0

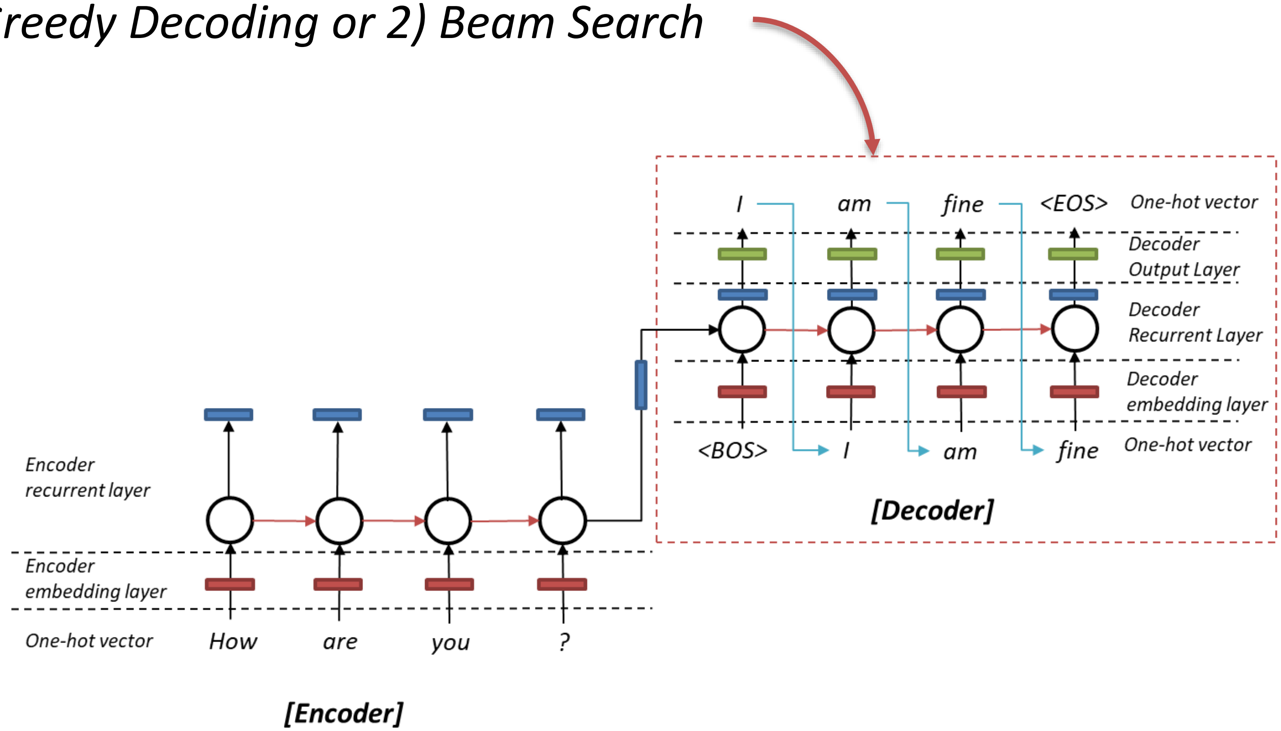
LECTURE PLAN

Lecture 8: Language Model and Natural Language Generation

1. Language Model
2. Traditional Language Model
3. Neural Language Model
4. **Natural Language Generation**
5. Other NLG Approaches
6. Language Model and NLG Evaluation

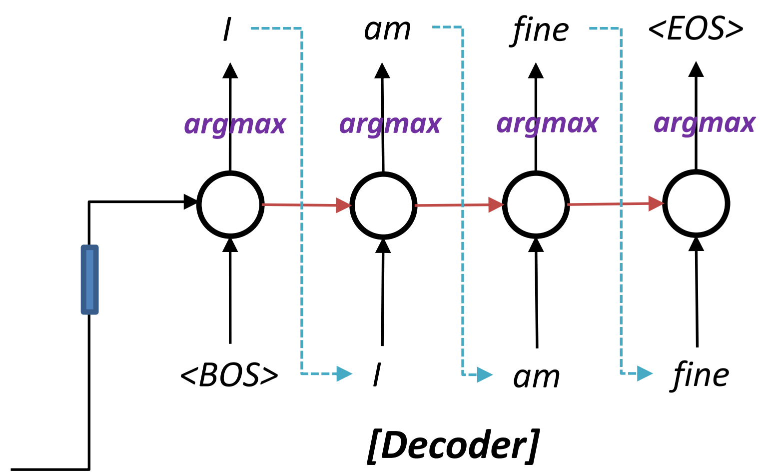
Decoding Algorithm

Now we have trained the conditional language model!
How do we use the language model to generate text?
1) Greedy Decoding or 2) Beam Search



Decoding Algorithm 1: Greedy Decoding

- Generate/decode the sentence by taking ***argmax*** on each step of the decoder
  - Take most probable word on each step
- Use that as the next word, and feed it as input on the next step
- Keep going until you produce *<EOS>*



Issue

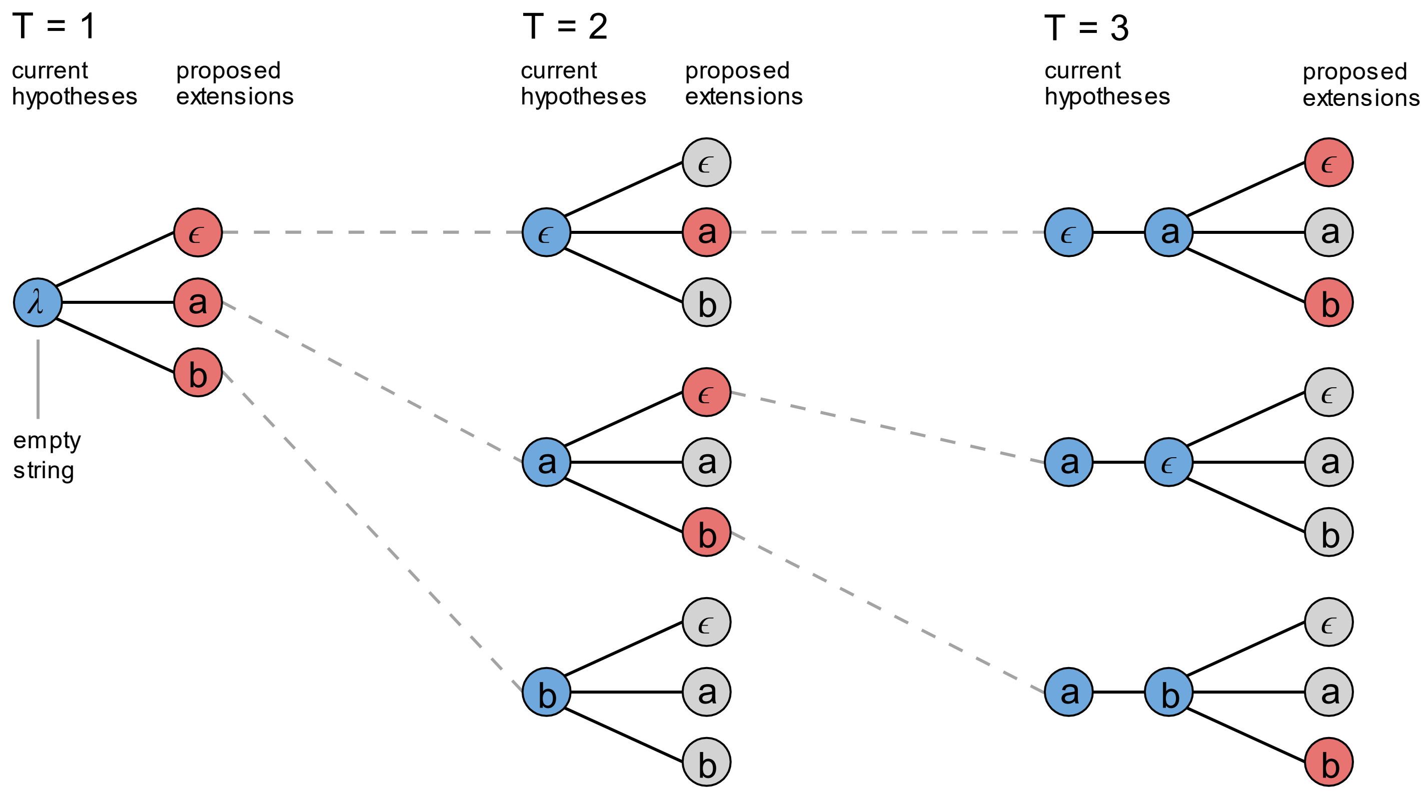
backtracking

- Greedy decoding has no way to undo decisions!! (Ungrammatical, unnatural)
- How to fix this issue?

Exhaustive search decoding: We could try computing all possible sequences

Decoding Algorithm: Beam Search

A standard beam search algorithm with an alphabet of {ϵ,a,b} with a beam size 3.



Decoding Algorithm: Beam Search

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

- On each step of decoder, keep track of the ***k most probable*** partial sequences (which we call *hypotheses*)
  - *K is the* ***beam size*** *(in practice around 5 to 10)*

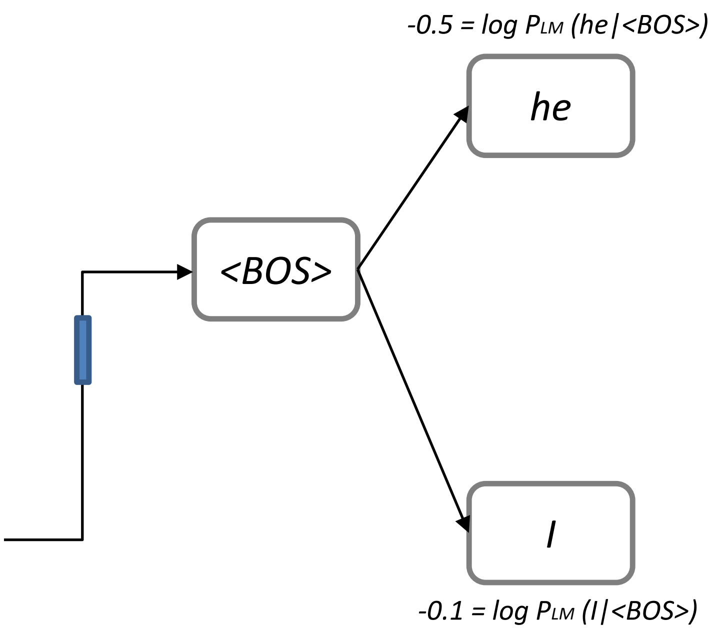
- After you reach some stopping criterion, *choose the sequence with the highest probability* (factoring in some adjustment for length)

4 Natural Language Generation

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Take top k words and compute scores

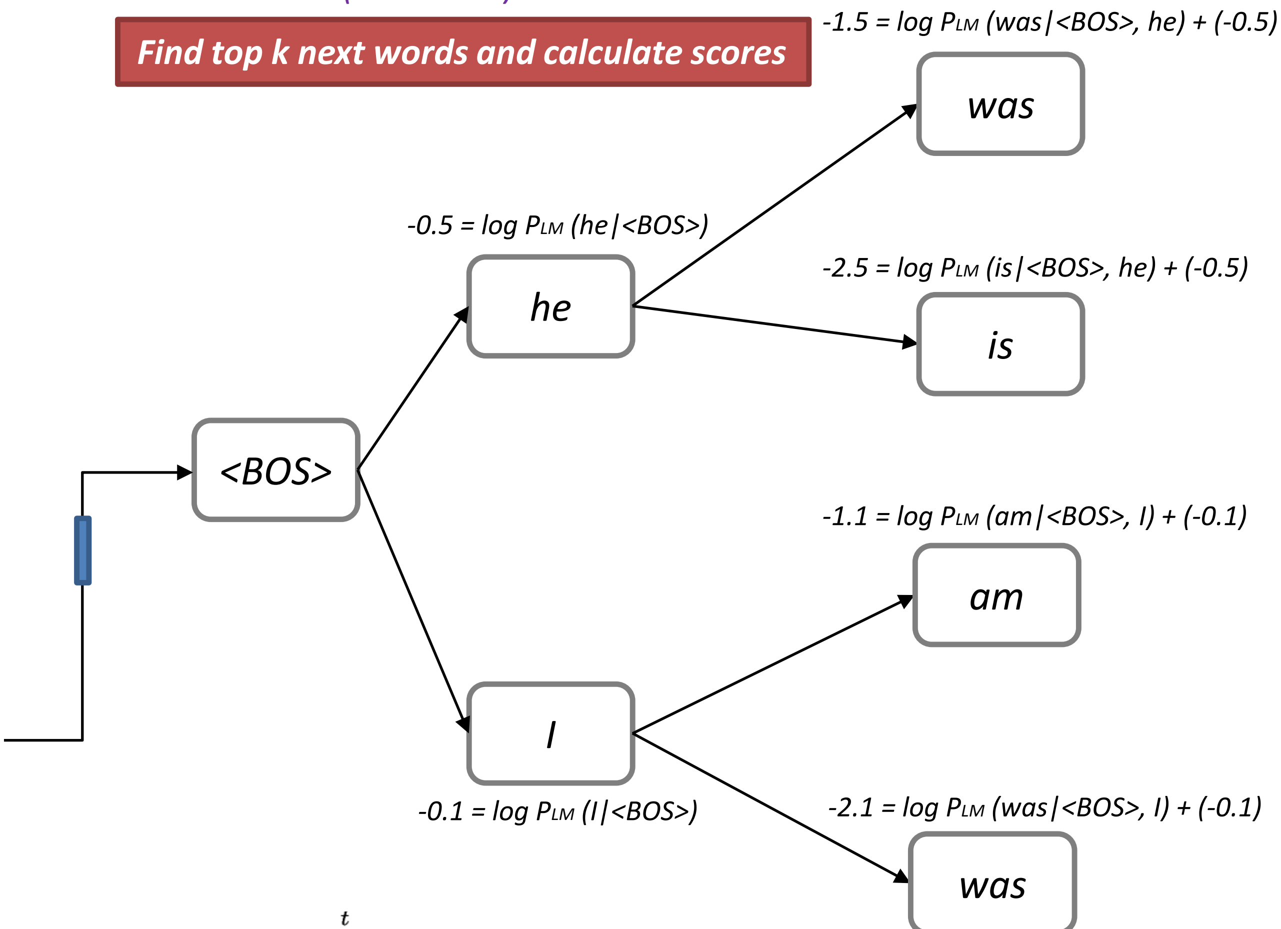


$$\text{score}(y_1,\ldots,y_t)=\sum_{i=1}^{t}\log P_{\text{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Find top k next words and calculate scores



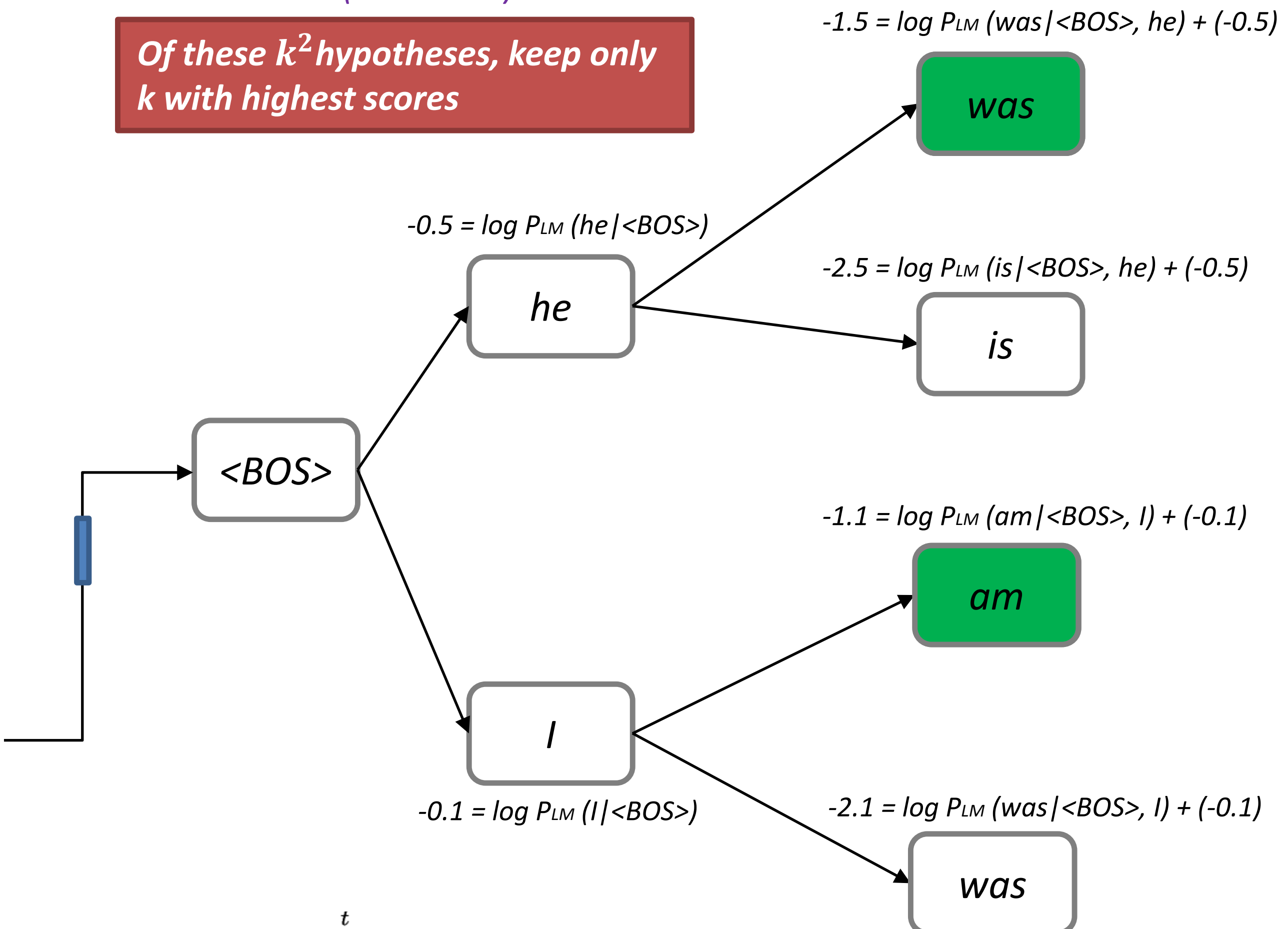
$$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

4 Natural Language Generation

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Of these k^2 hypotheses, keep only k with highest scores

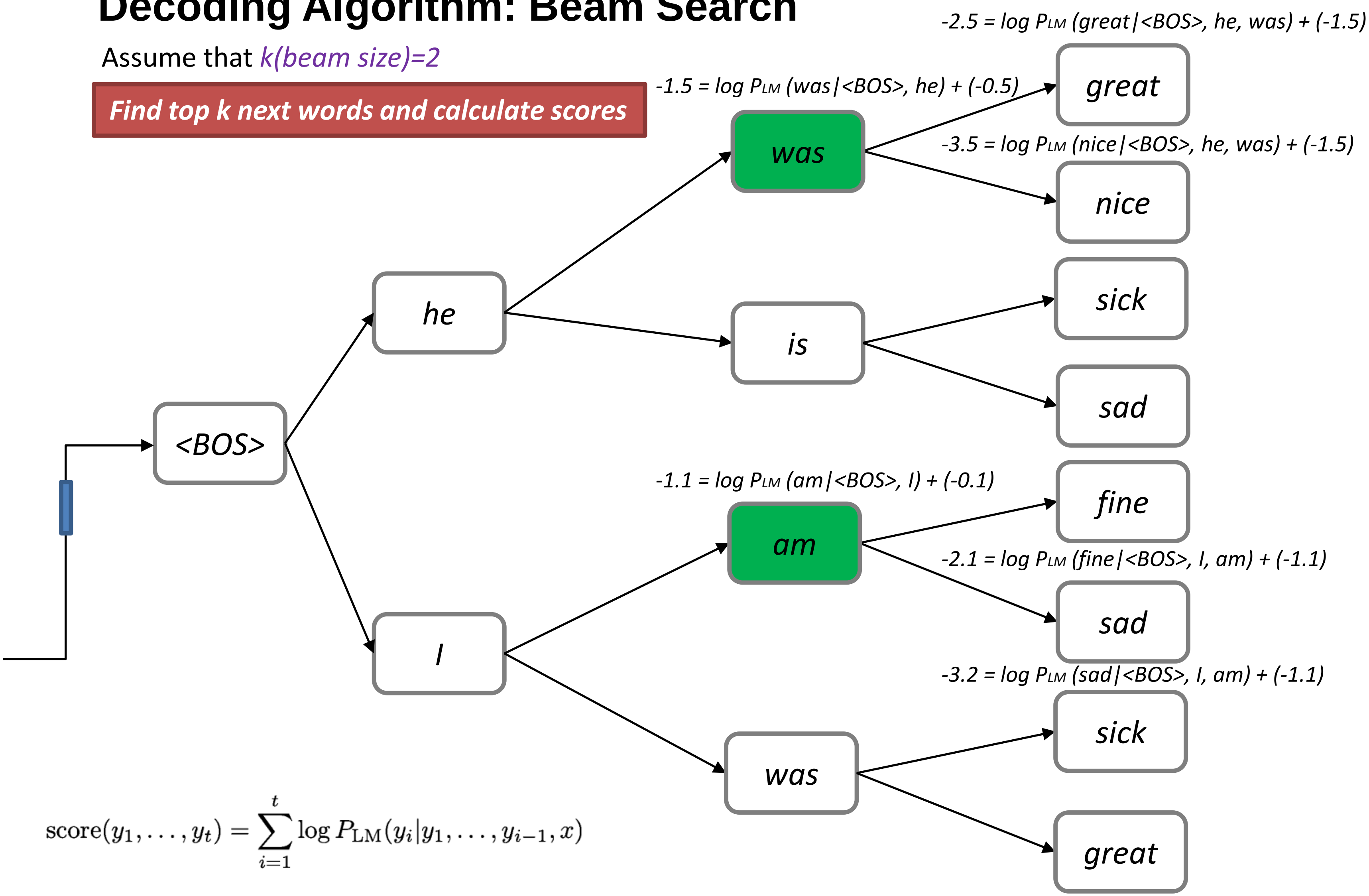


$$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Find top k next words and calculate scores

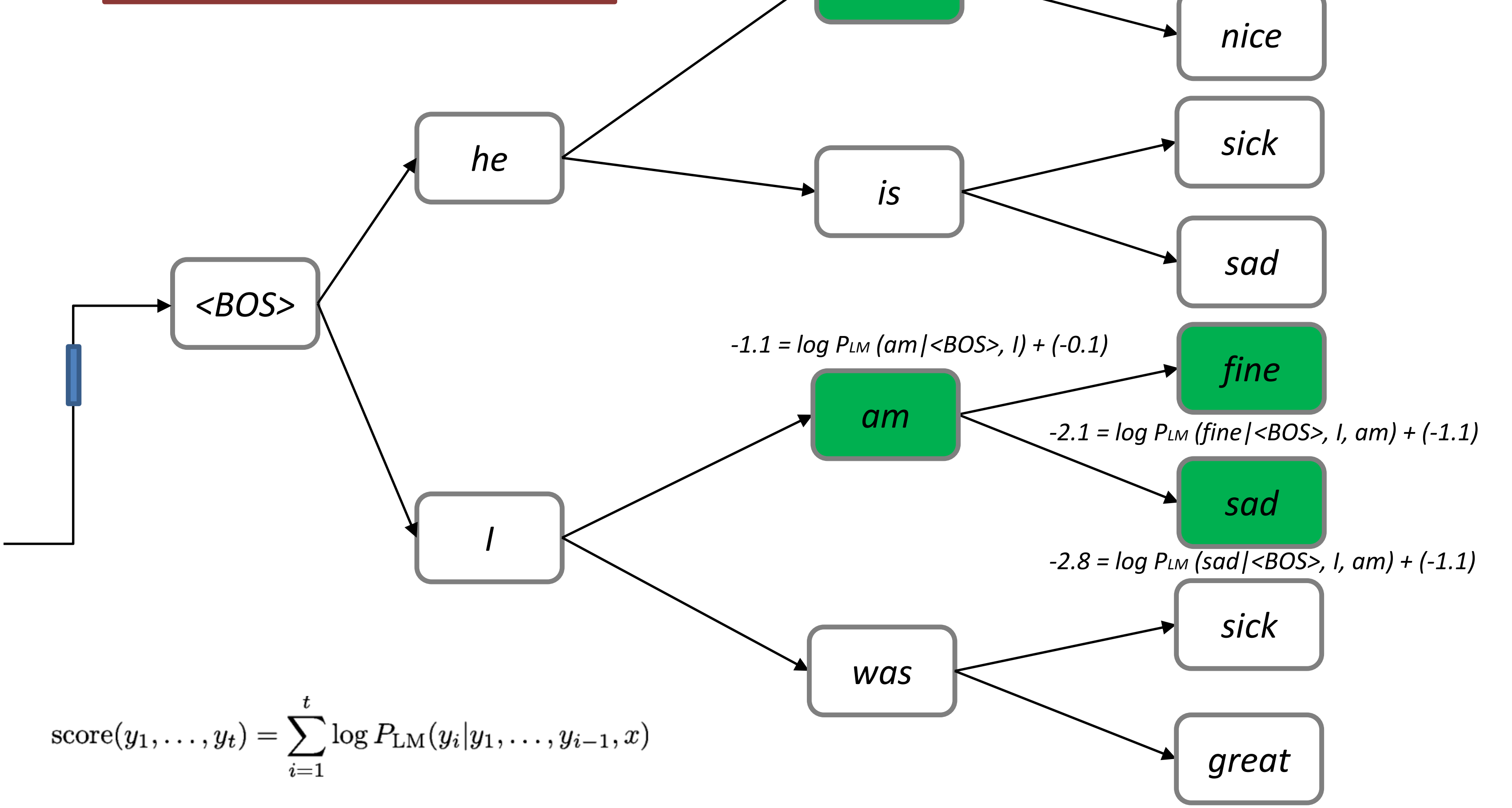


$$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Of these k^2 hypotheses, keep only k with highest scores

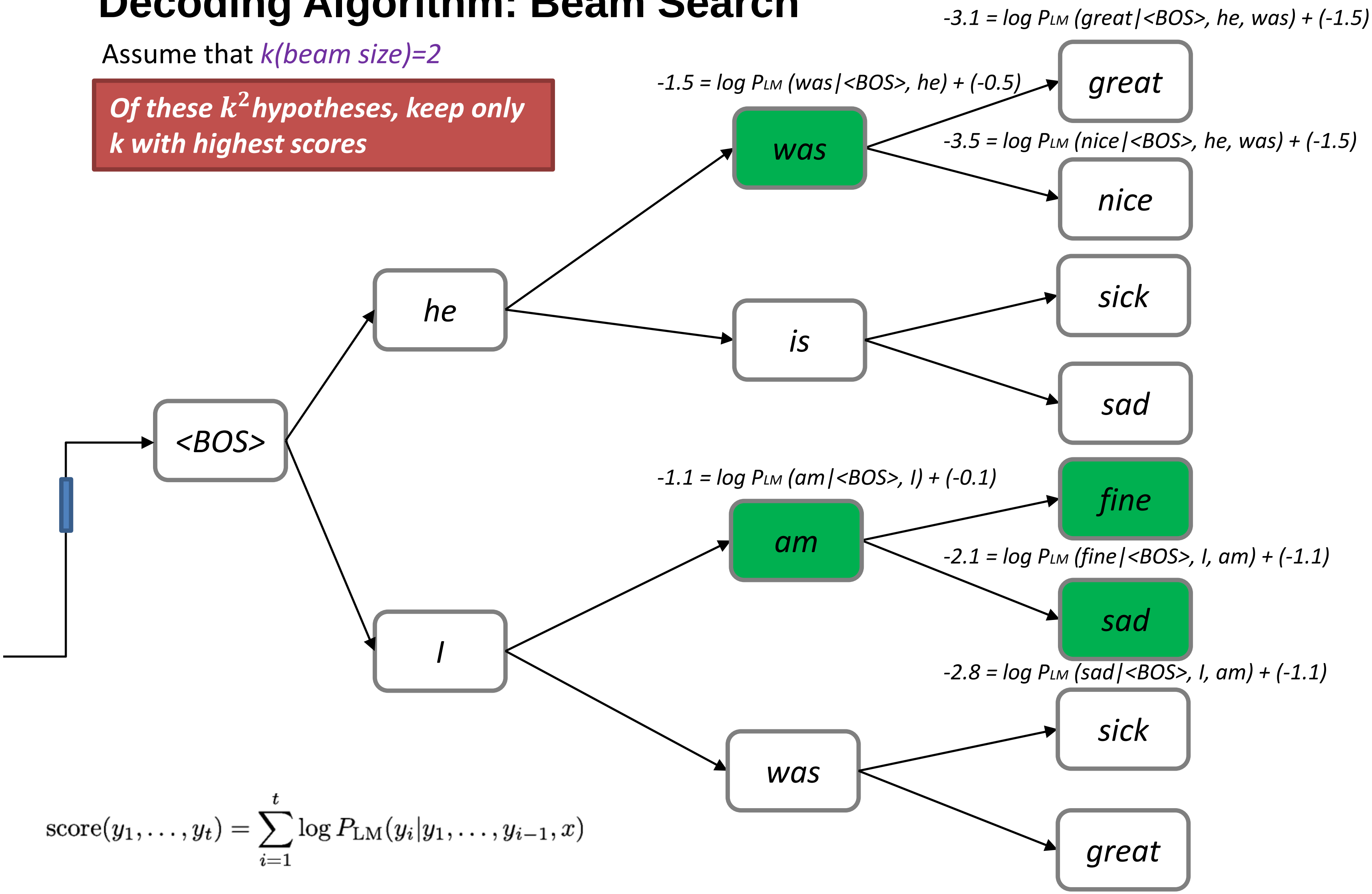


$$\mathrm{score}(y_1,\ldots,y_t)=\sum_{i=1}^{t}\log P_{\mathrm{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Of these k^2 hypotheses, keep only k with highest scores

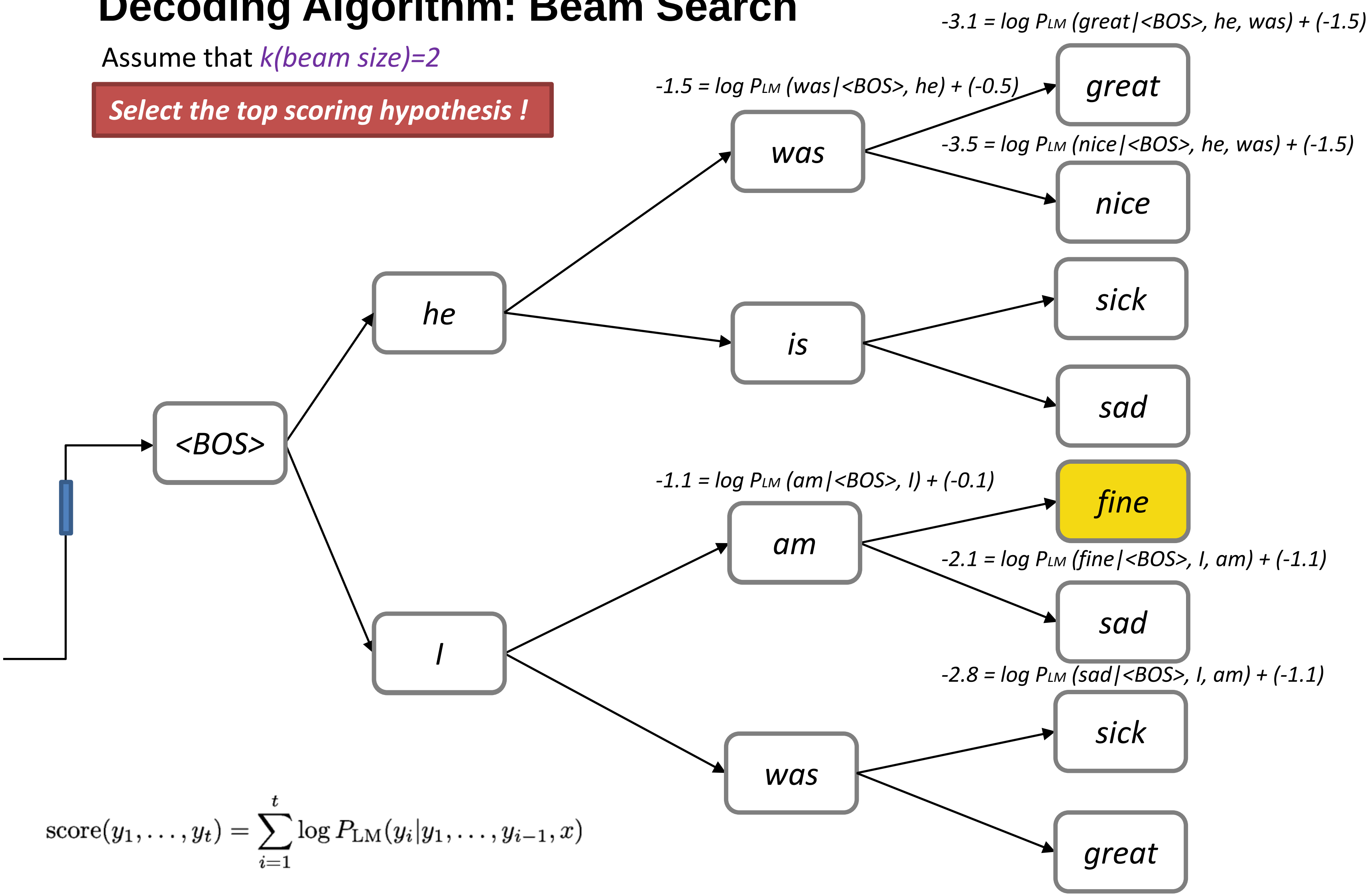


$$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

Decoding Algorithm: Beam Search

Assume that *k(beam size)=2*

Select the top scoring hypothesis !



$$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

The effect of beam size k

- Small k has similar problems to greedy decoding (k=1)
  - Why?

- Large k means you consider more hypotheses
  - Solve the issues in greedy decoding
  - Produce, other issues:
    - Computationally expensive
    - In open-ended tasks like chit-chat dialogue, large k can make output more generic

The effect of beam size k in chit chatbot

Human

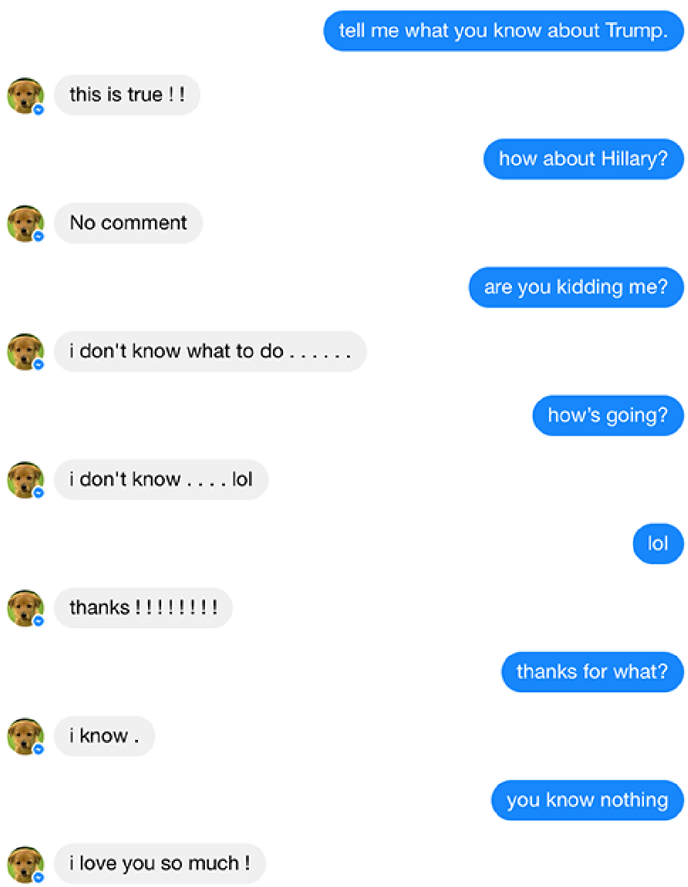
Beam size	Model response
1	I love to eat healthy and eat healthy
2	That is a good thing to have
3	I am a nurse so I do not eat raw food
4	I am a nurse so I am a nurse
5	Do you have any hobbies?
6	What do you do for a living?
7	What do you do for a living?
8	What do you do for a living?

Machine Answer

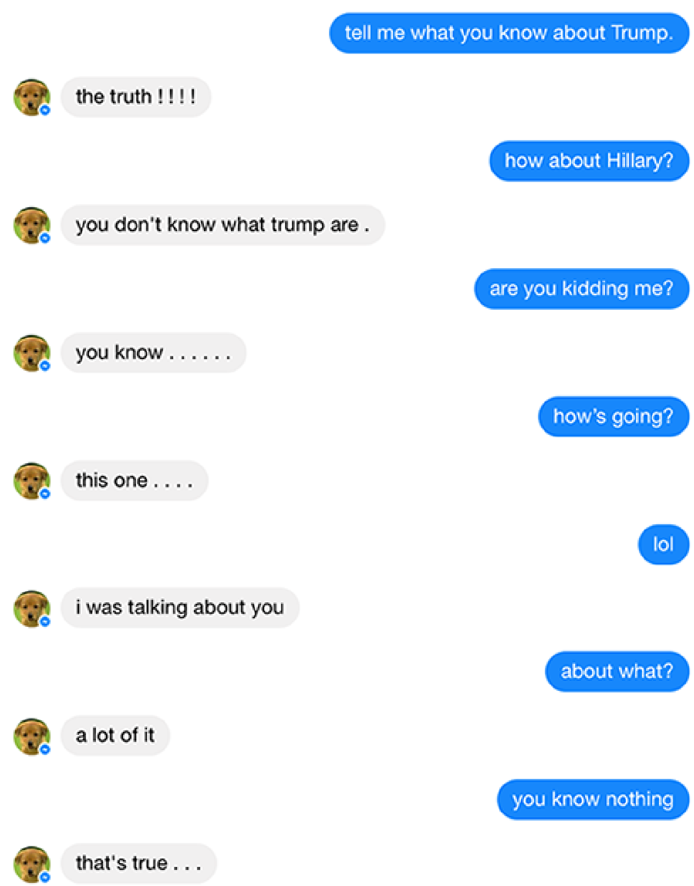
Lower beam size
More on topic but non-sensical

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

The effect of beam size k in chit chatbot



Beam size=10



Beam size=10 and anti-language model

Sampling-based decoding

Pure sampling

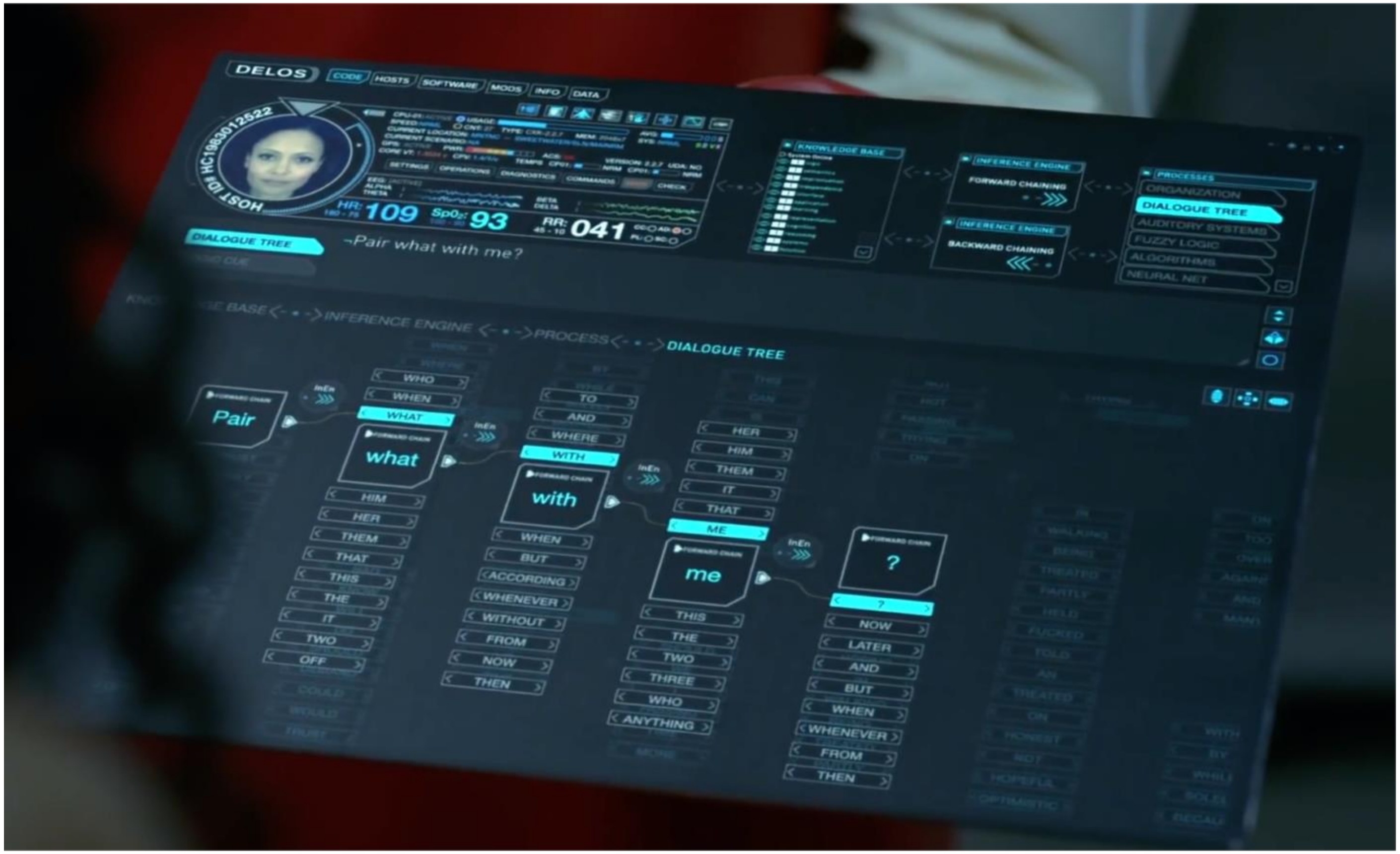
- On each step t, randomly sample from the probability distribution Pt to obtain your next word.
- Like greedy decoding, but using sample instead of argmax

Top-n sampling

- On each step t, randomly sample from Pt, 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

Natural Language Generation

Dialog Tree from Westworld



https://www.youtube.com/watch?v=ZnxJRYit44k&feature=youtu.be&t=28

0 LECTURE PLAN

Lecture 8: Language Model and Natural Language Generation

1. Language Model
2. Traditional Language Model
3. Neural Language Model
4. Natural Language Generation
5. **Other NLG Approaches**
6. Language Model and NLG Evaluation

5 Other NLG Approaches

Neural based NLG in Dialog: Issue

- Problem: became apparent that a naïve application of standard seq2seq methods has serious pervasive deficiency for (chitchat) dialogue:
  - Either because it’s generic (e.g. “I don’t know”)
  - Or because changing the subject to something unrelated
  - Boring response
  - Repetition problem
  - Lack of consistent persona problem

What else do we have?

5 Other NLG Approaches

Template-based generation

- The most common approach in spoken natural language generation.
- In simplest form, words fill in slots:

"Flights from *ORIGIN* to *DEST* on *DEPT_DATE DEPT_TIME*. Just one moment please"

Slot	*Type*	*Question*
ORIGIN	*city*	*What city are you leaving from?*
DEST	*city*	*Where are you going?*
DEPT DATE	*date*	*What day would you like to leave?*
DEPT TIME	*time*	*What time would you like to leave?*
AIRLINE	*line*	*What is your preferred airline?*

- Most common NLG used in commercial systems
- Used in conjunction with concatenative TTS (text-to-speech) to make natural sounding output

5 Other NLG Approaches

Template-based generation

Pros

- Conceptually Simple: No specialized knowledge required to develop
- Tailored to the domain, so often good quality

Cons

- Lacks generality: Repeatedly encode linguistic rules (e.g. subject-verb agreement)
- Little variation in style
- Difficult to grow/maintain: Each utterance must be manually added

Improvement?

- Need deeper utterance representations
- Linguistic rules to manipulate them

5 Other NLG Approaches

Rule-based Generation

What to say — *How to say*

Content Planner → **Sentence Planner** → **Surface Realiser**

Content Planning

- What information must be communicated?
  - Content selection and ordering

Sentence Planning

- What words and syntactic constructions will be used for describing the content?
  - Aggretation: What elements can be grouped together for more natural-sounding, succinct output?
  - Lexicalisation: What word are used to express the various entities?

Realisation

- How is it all combined into a sentence that is syntactically and morphologically correct?

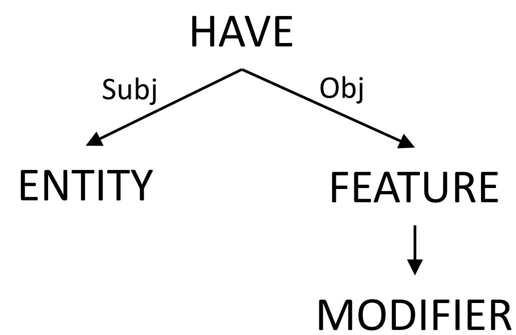
5 Other NLG Approaches

Rule-based Generation

Assume that the dialog system need to tell the user about the restaurant

Content Planning
- Select Information ordering
  - has(sushitrain, crusine(bad))
  - has(sushitrain, decor(good))

Sentence Planning
- Choose ***syntactic templates***
- Choose lexicon
  - Bad → awful; crusine → food quality
  - Good → excellent; decor → décor
- Generate expressions
  - Entity → this restaurant



Realisation
- Choose correct verb: HAVE → has
- No article needed for feature names

"This restaurant has awful food quality but excellent décor"

Other NLG Approaches

Summarisation: two strategies

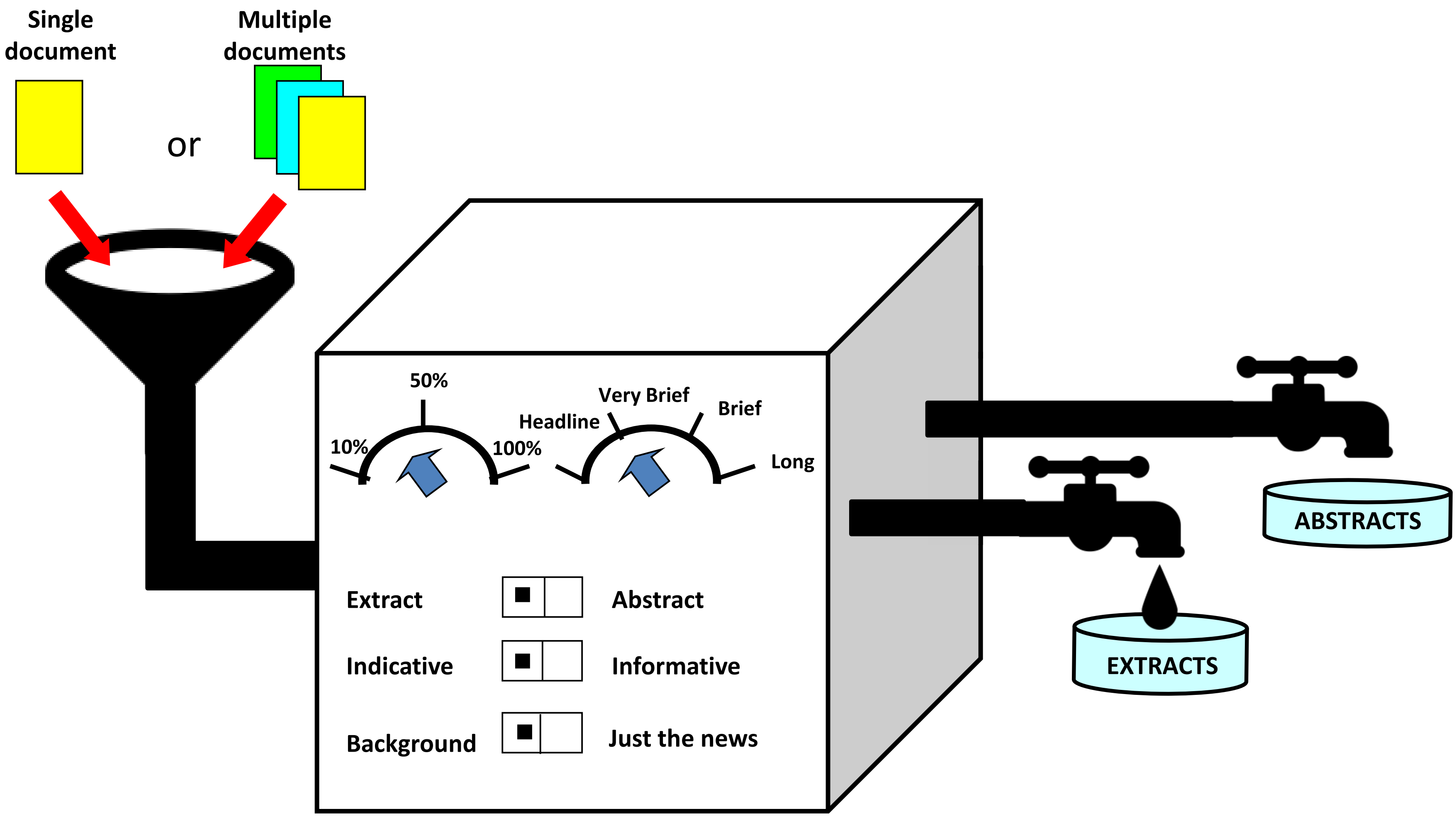
Extractive Summarisation

- Select parts (typically sentences) of the original text to form a summary.

Abstractive Summarisation

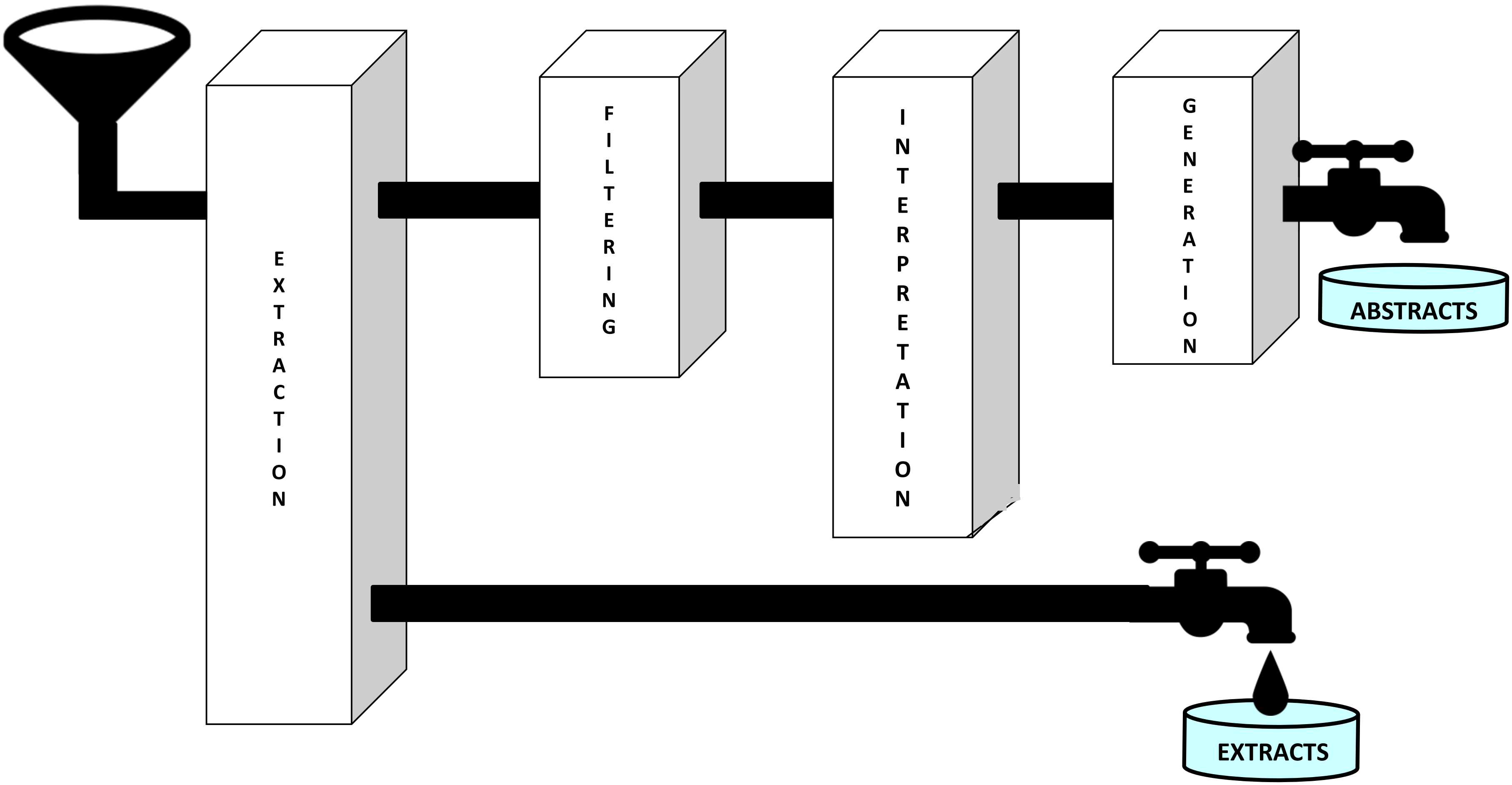
- *Generate new text using natural language generation techniques.*

Summarisation: two strategies



5 Other NLG Approaches

Summarisation: two strategies



0 LECTURE PLAN

Lecture 8: Language Model and Natural Language Generation

1. Language Model
2. Traditional Language Model
3. Neural Language Model
4. Natural Language Generation
5. Other NLG Approaches
6. **Language Model and NLG Evaluation**

How to evaluate the Language Model?

*The standard evaluation metric for Language Models is **perplexity**.*

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

*This is equal to the **exponential of the cross-entropy loss***

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

So, Lower Perplexity is better!

6 NLG Evaluation

How to evaluate the Language Model?

Language Model Approaches Performance Evaluated by Facebook Research

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1	
RNN-2	
Sparse Non-negative Matrix Factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Joz	
2-layer LSTM-8	
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

N-gram model

ural model

Can we use the perplexity for NLG Evaluation?

No. Captures how powerful your LM is,
but doesn't tell you anything about generation

Table 2. Comparison on 1B word in perplexity (lower the better). Note that Jozefowicz et al., uses 32 GPUs for training. We only use 1 GPU.

https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/

How to evaluate the Natural Language Generation?

Unfortunately, No automatic metrics to adequately capture overall quality

There are some metrics to capture particular aspects of generated text:

- *Fluency (compute probability - well-trained Language model)*
- *Correct style (Language Model trained on target corpus)*
- *Diversity (rare word usage, uniqueness of n-grams)*
- *Relevance to input (semantic similarity measures)*
- *Simple things like length and repetition*
- *Task-specific metrics e.g. compression rate for summarization*
- *Though these don't measure overall quality, they can help us track some important qualities that we care about.*

6 NLG Evaluation

How to evaluate the Natural Language Generation?

Human Evaluation

- *Human judgments are regarded as the gold standard*
- *Of course, we know that human eval is slow and expensive*
- *Supposing you do have access to human evaluation: Does human evaluation solve all of your problems?*

Humans …

- *are inconsistent*
- *can be illogical*
- *lose concentration*
- *misinterpret your question*
- *can't always explain why they feel the way they do*

6 NLG Evaluation

Natural Language Generation

/ Reference

Reference for this lecture

- Deng, L., & Liu, Y. (Eds.). (2018). Deep Learning in Natural Language Processing. Springer.
- Rao, D., & McMahan, B. (2019). Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. " O'Reilly Media, Inc.".
- Manning, C. D., Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.
- Manning, C 2018, Natural Language Processing with Deep Learning, lecture notes, Stanford University

- Li, J., Galley, M., Brockett, C., Gao, J., & Dolan, B. (2015). A diversity-promoting objective function for neural conversation models. arXiv preprint arXiv:1510.03055.
- Jiang, S., & de Rijke, M. (2018). Why are Sequence-to-Sequence Models So Dull? Understanding the Low-Diversity Problem of Chatbots. arXiv preprint arXiv:1809.01941.
- Liu, C. W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., & Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. arXiv preprint arXiv:1603.08023.