Lecture 8: Language model and Natural Language Generation

Language Model

— Task of predicting the word coming to the next based on the given word

— a probabilistic model which predicts the probability that a sequence of tokens belongs to a language

- Language Modeling in Natural Language Generation

  — Conditional Language Modelling:

  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 LM

  — collect and learn the model with the corpus that includes documents about the domain that your system/application will be used

Traditional LM

Statistical Language Model (SLM): $p(A|B) = \frac{p(A \cap B)}{p(B)} = \frac{p(A.B)}{p(A)}$

- Conditional probability

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

- Conditional Language Modeling:

  — Predicting the next word, given the words so far, and also some other input x

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

N-gram LM

- N-gram: a sequence of N words=a chunk if n consecutive words
- N-gram model predicts the probability of a given N-gram within any sequence of words in the language
- Assumption: the next word, $x^{(t+1)}$, depends only on the preceding n-1 words

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

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

- Limitations
  - Trade-off Issue

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

    n-1 words only

    (3-1) words only

    — OOV (small n) or Model Size(big n)

— Optimal n

- Zero count issue:

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

  1. ***'is speeding w'*** never occur in the coupurs:

     — Consequence: the probability will be 0

     — Solution: Smoothing (add small value to the count for every w in the courpus)

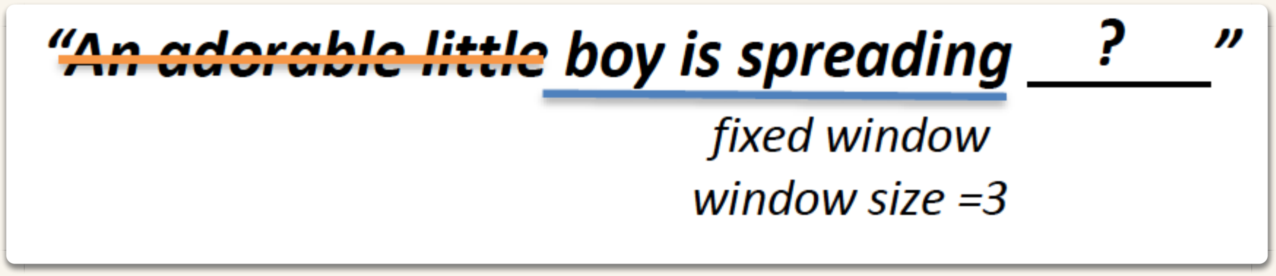
  2. ***'is spreding'*** never occur in the corpus

     — Consequence: impossible to calculate the porbability for any w

     — Solution: Backoff (condition on ***'spreading'***)

Neural Language model

Window-based NLM



- Pros

  — No Trade-off issue

- Cons

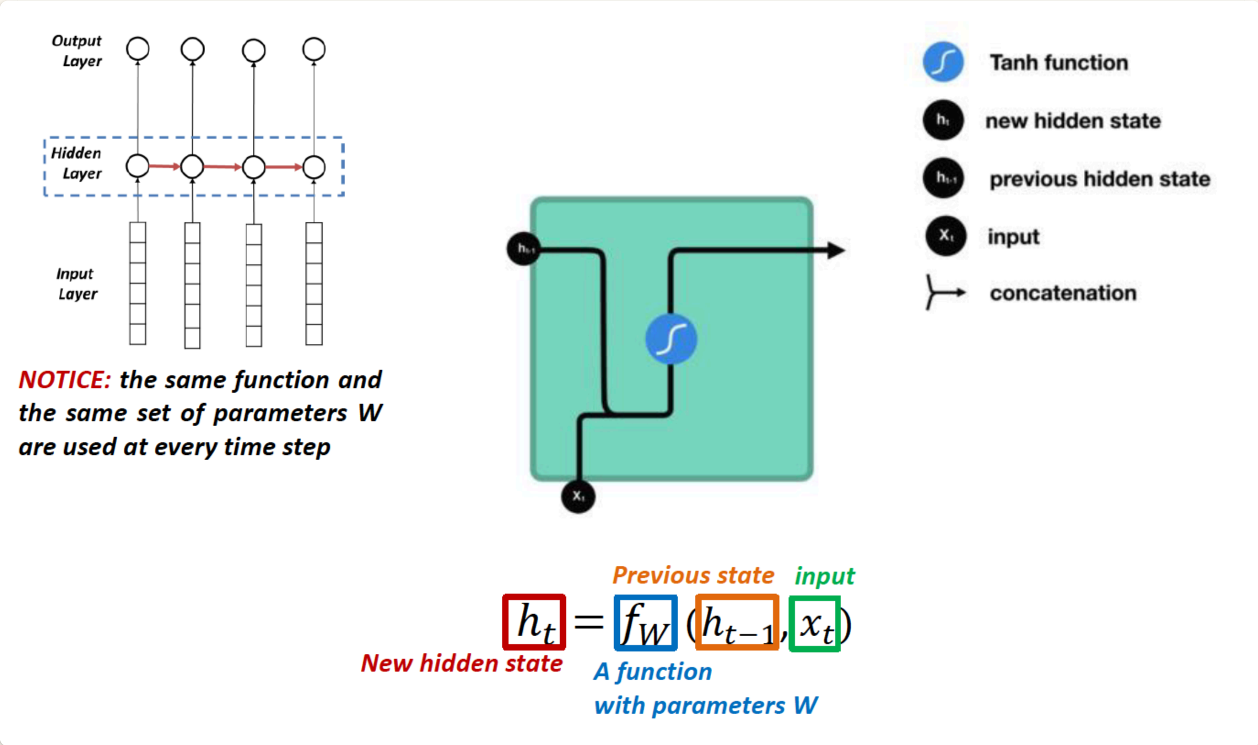
  — window size selection issue : incresing window size elarge W

  — Input vectors are multiplied by completely different wieghts in W: No symmetry in how the inputs are processed

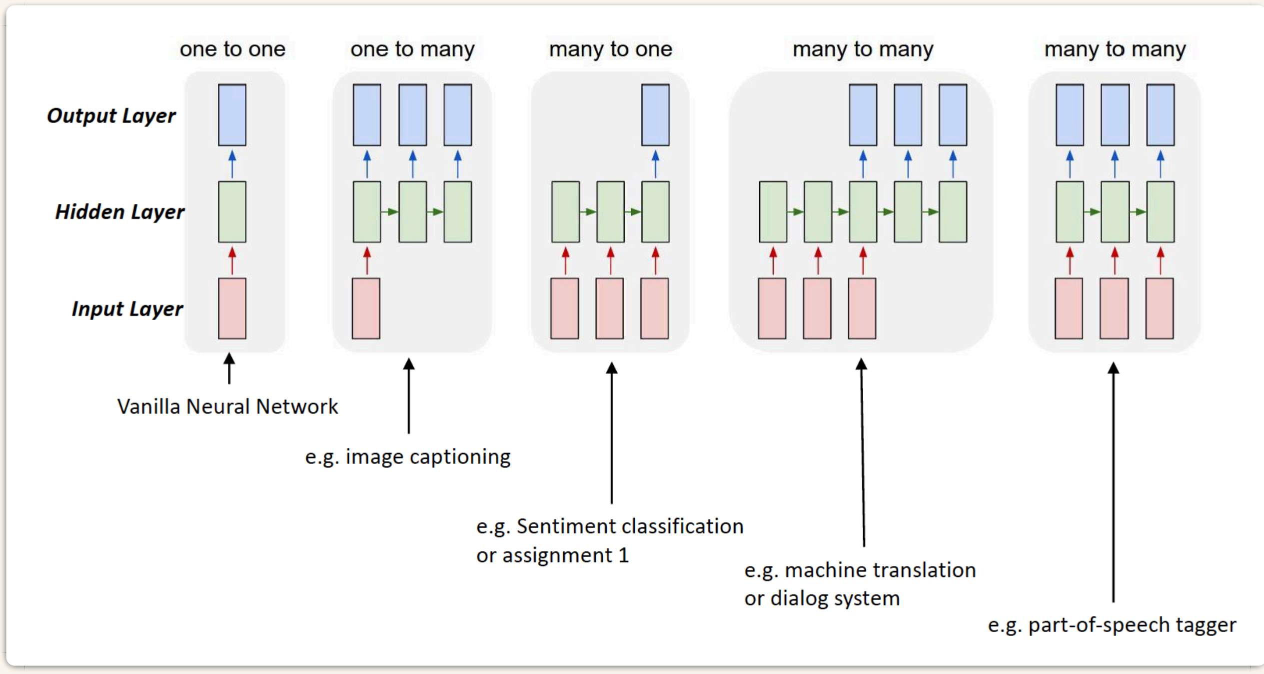
RNN-based LM

Neural Network + Memory= Recurrent Neural Network

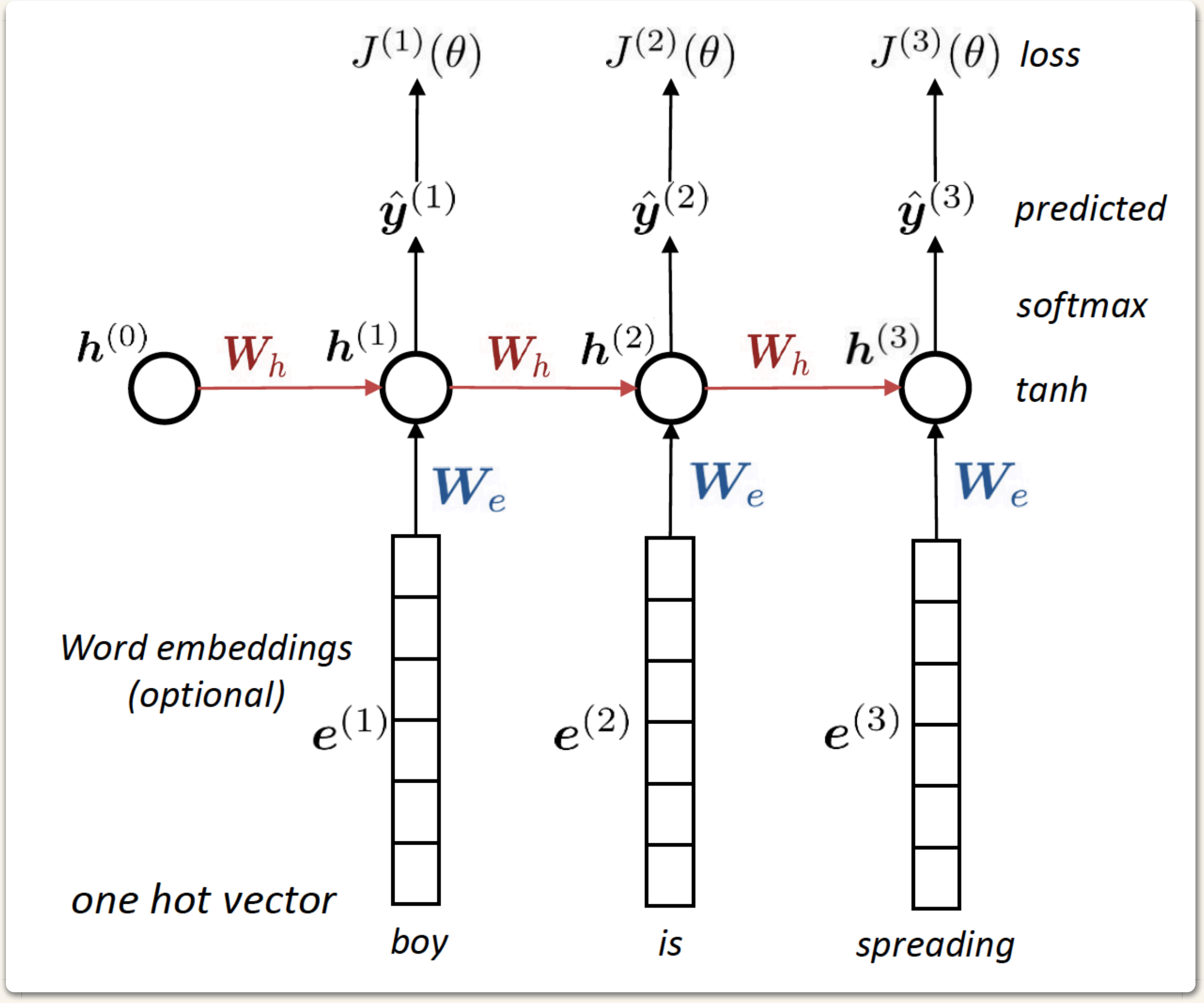
- Basic architecture



- Types of RNN



- Example of prediction



- Pros

— can process any length input (sequence 2 sequence structure)

— can use information from many step back (hidden state)

— model size does not increase (sequence padding)

— same weights applied on every time step (symmetry structure: share parameters)

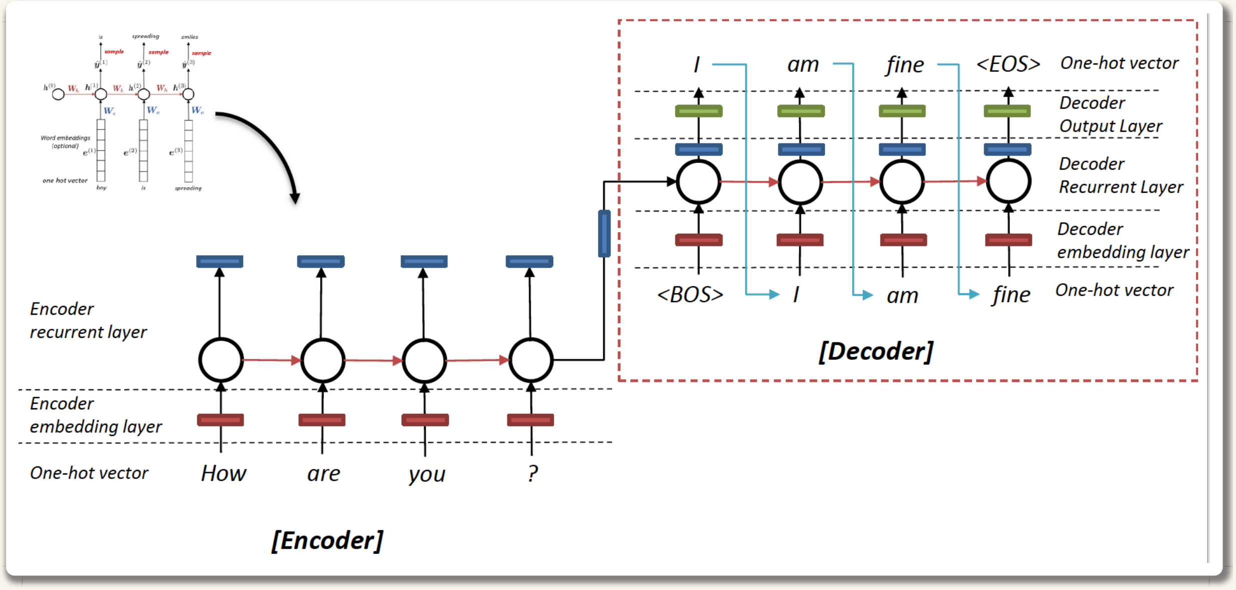
- Cons

— Slow computation (Neural nets)

— Difficult to access information from many step back (coverage sequence no.)

Seq2Seq Model with trained language Model

- Teacher reforming: feed the gold target sentence into the decoder, regardess of what the decoder predicts

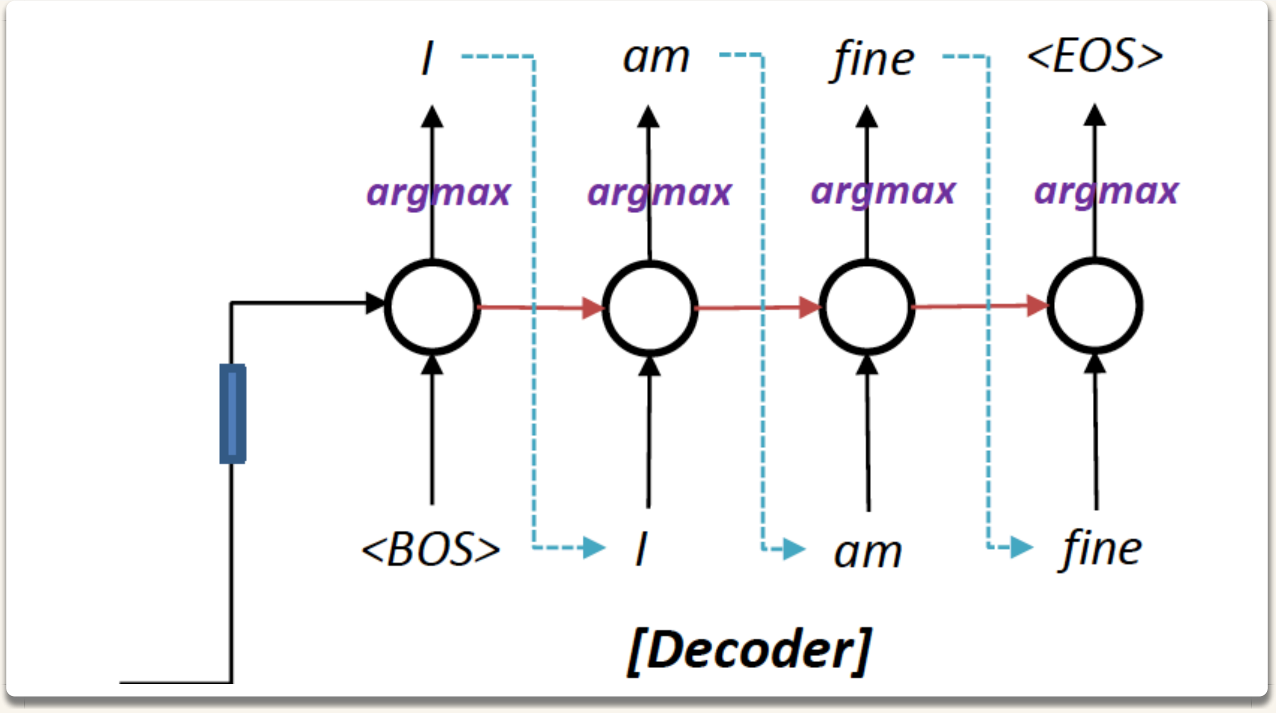


Natural Language Generation

Decoding Algorithm

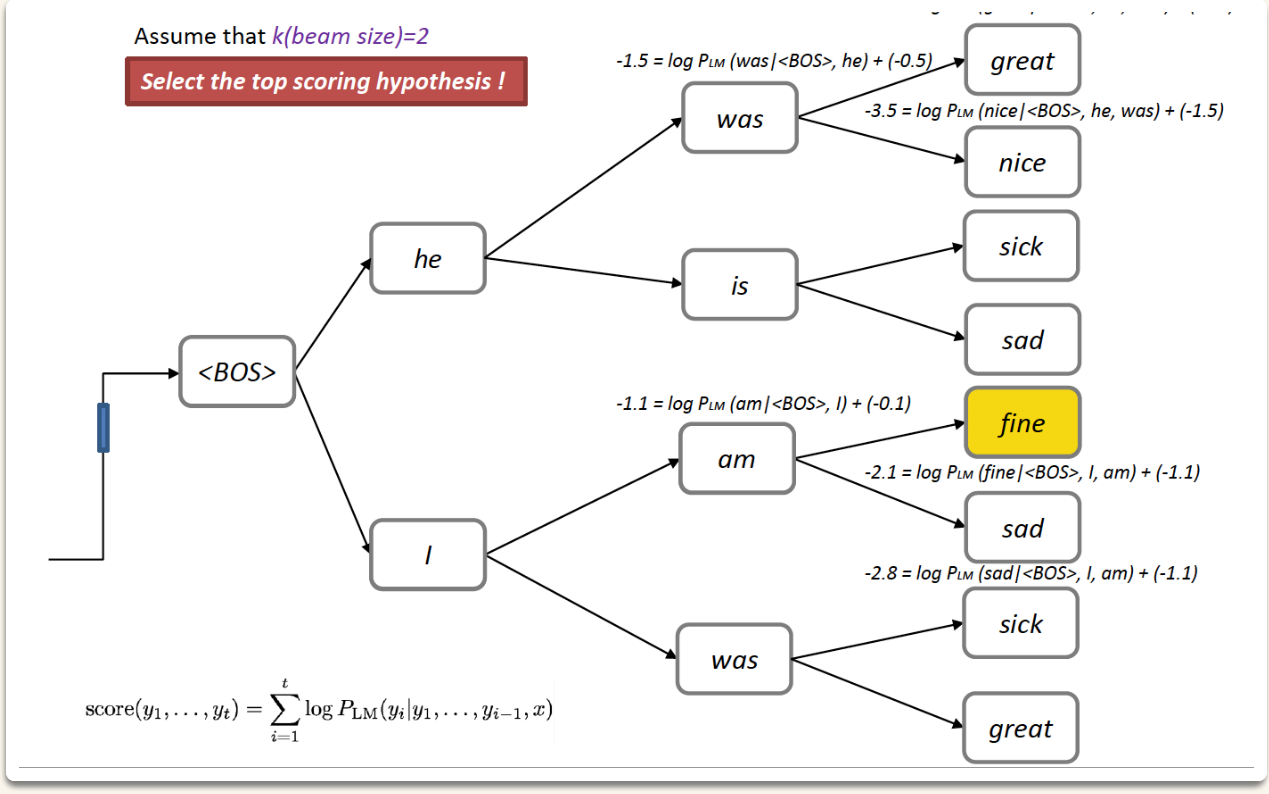
Greedin Decoding

- Generate the sentence by taking argmax on each step of the decoder
  - Take the most probable word on each step
- Use that as the previous argmax ouput as the next word and feed it as input on the next step
- Keep going until you produce (end token)
- LIMITATION : cannot backtracking (no way to undo decisions)–> ungrammatical and unnatural
  - Solution: Exhaustive search decoding —> computing all possible sequences



Beam Search

- A search algorithm which aims to find a high-probability sequence by tracking multiple possible sequences at once
- One each step of decoder, keep track of the k most probable partial sequences (which we call hypotheses)
  - K: beam size (usually 5-10)
- After reaching stopping criterion, choose the sequence with the highest probability (factoring in some adjustment for length)



- The effect of beam size k
  - Small k: similar problems to greedy decoding
  - Large k: consider more hypothesis

    — solve the issues in greedy decoding

    — Conputationallu expensive

    — open-ended tasls like chit-chat dialogue, large k can make output more generic

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?

Lower beam size
More on topic but non-sensical

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

Machine Answer

Sampling-based decoding

Pure sampling

— Each step t, randomly sample from the probability distribution p_t to obtain next word. — Like greedy decoding, but using sample instead of argmax

Top-n sampling

— 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

Other NLG Approaches

Neural based NLG in Dialog: Issue

A naive application of standard seq2seq methods has serious pervasive deficiency

— Either because it's generic

— Or because changing the subject to something unrelated

— Boring response

— Repetition problem

— Lack of consistent persona problem

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 comeercial system
- Used in conjunction with concatenative TTS to make natural sounding output
- Pros
  - Conceputally simple : No specialized knowledge required to develope
  - Tailored to the domain, so often good quality
- Cons
  - Lacks generality: Repeatedly encode linguistic rules
  - Little variation in style
  - Difficult to grow/maintain: each utterace must be manually added

- Solution
  - Deeper utterance representations
  - Linguistic rules to manipulate them

Rule-based Generation

- Content planning
  - What information must be communicated?

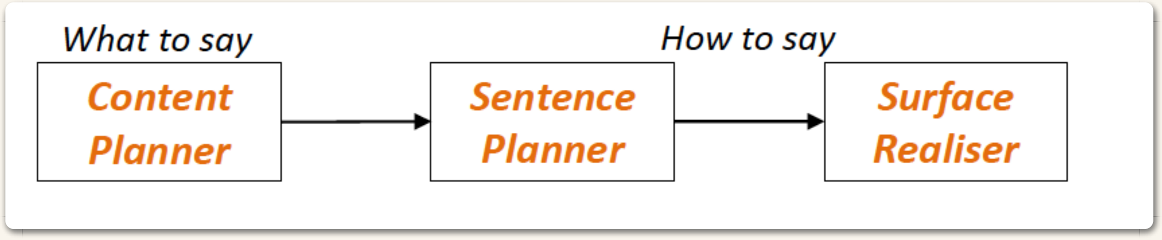
    — Content selection ordering

- Sentence planning
  - What words and ysntactic constructions will be used for describing the content

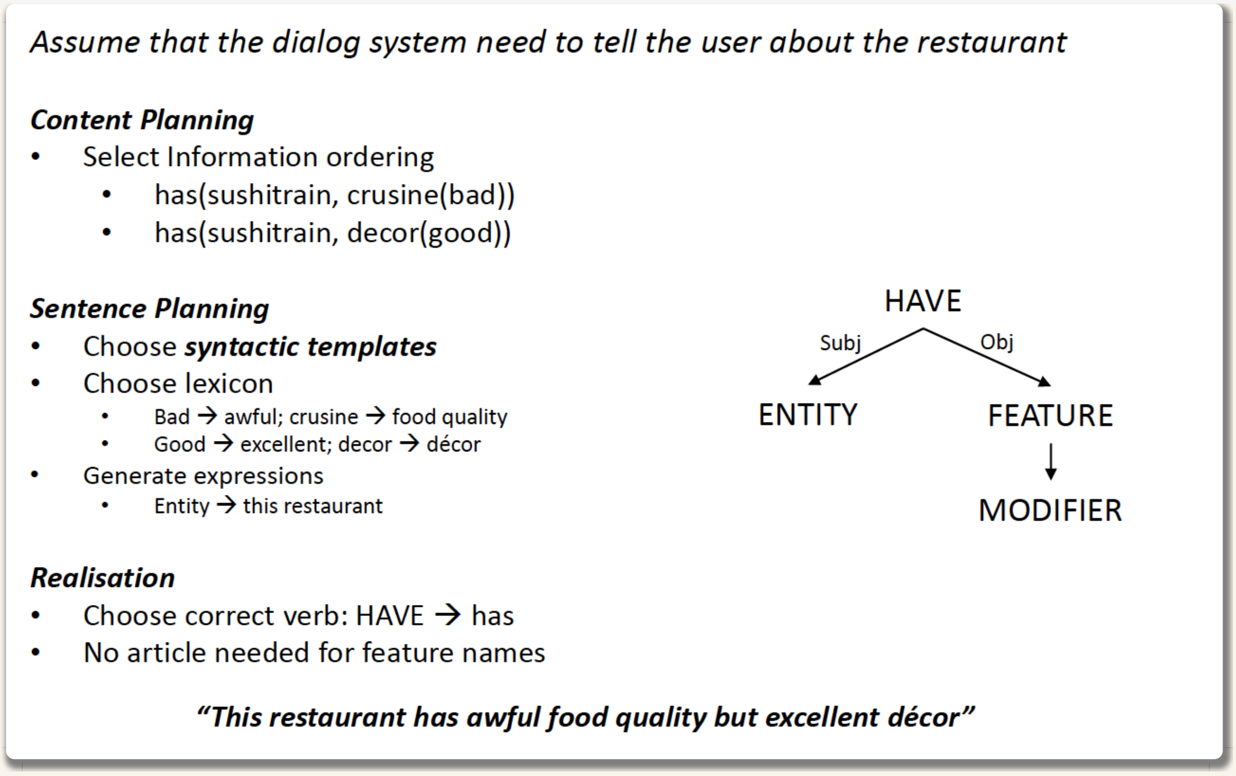
    — Aggretation: what elements can be grouped together for more natural-sounding, succinct output

    — 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



- Example:



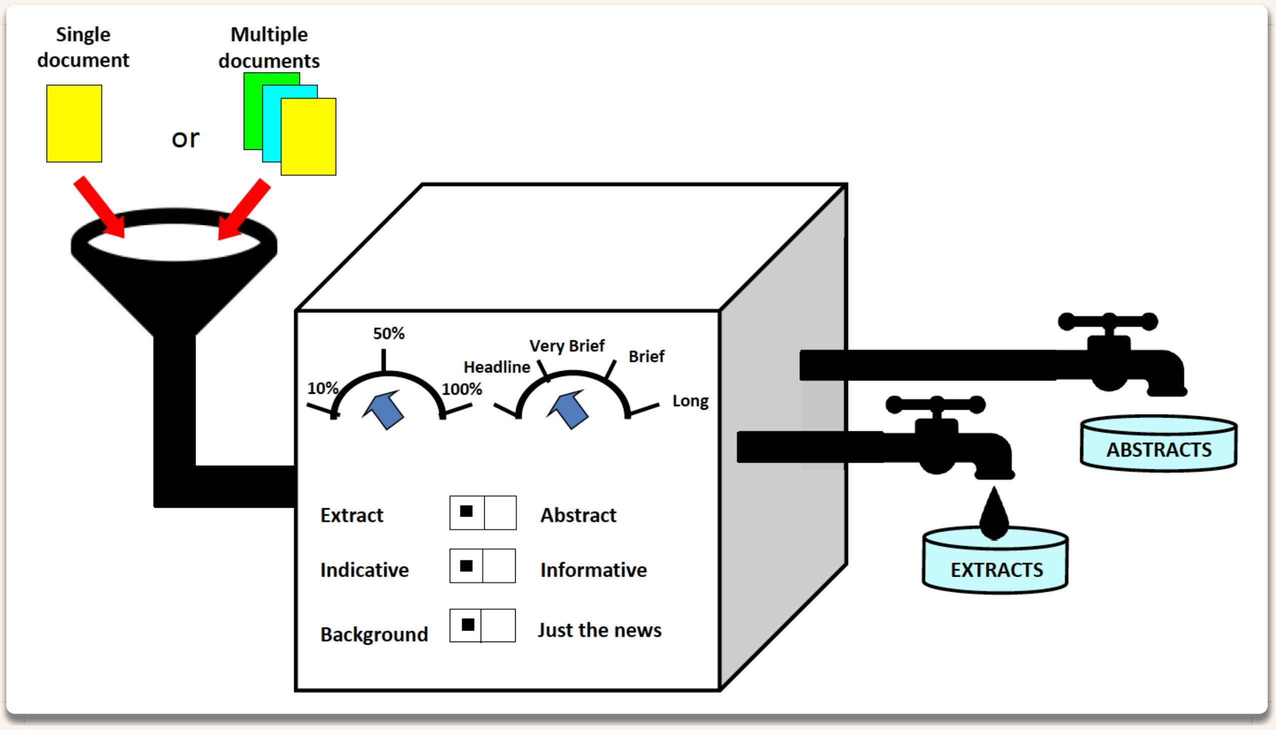
Summarisation: two strategies

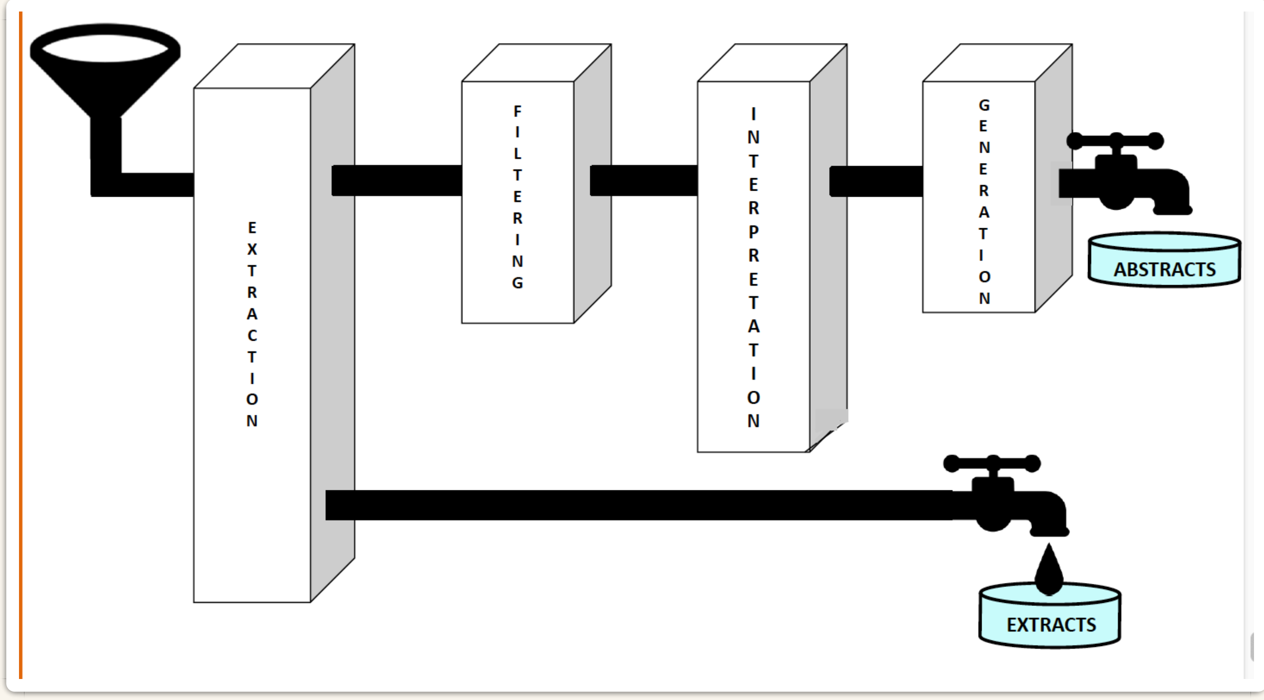
Extractive Summarisation

- slect parts (sentences) of the orginal text to form a summary

Abstraction Summarisation

- Generate new text using natural laguange generation techniques





Language Model and NLG Evaluation

Evaluation

- Perplexity
  - only caputre how powerful the model it is, but not the generation
  - = Exponential of the cross-entropy loss: Lower perplexity is better

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

Normalized by number of words

Inverse probability of corpus, according to Language Model

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

- No automatic metrics to adequately capture overall quality
- Some metrics to capture particular aspects of generated text
  - Fluency: compute probability–well-trained LM
  - Correct style: LM tranined on targert corpus
  - Diversity: rare word usage, uniqueness of n-grams
  - Relevance to input: semantic similarity measure

- Simple things like length and repetition:
- Task-specific metrics: compression rate for summarization
- Human evaluation