11868 LLM Systems
Tokenization & Decoding

Lei Li

Carnegie Mellon University
Language Technologies Institute

Today's Topic

- How to construct a vocabulary for a large corpus
  - Tokenization: how to break text into units?

- How to generate text at inference time

Tokenization

- Break sentences into tokens, basic elements of processing

- Word-level Tokenization
  - Break by space and punctuation.
  - English, French, German, Spanish

The most eager is Oregon which is enlisting 5,000 drivers in the country's biggest experiment.

  - Special treatment: numbers replaced by special token [number]
  - How large is the Vocabulary? Cut-off by frequency, the rest replaced by [UNK]

What is a word?

How many words?

Bob's handy man is a do-it-yourself kinda guy, isn't he?

Clitic

noun-noun compound

multi-word expression

contraction

Words

- Orthographic definition
  - strings separated by white spaces
  - spoken language: units corresponding to written word separated by pause
  - problem: Bob’s handy man is a do-it-yourself kinda guy, isn’t he?
- What about languages that do not use white spaces?

他昨天晚上去看了消失的她

he yesterday night watched lost in stars

Pros and Cons of Word-level Tokenization

- Easy to implement

- Cons:
  - Out-of-vocabulary (OOV) or unknown tokens, e.g. Covid
  - Tradeoff between parameters size and unknown chances.
    - Smaller vocab => fewer parameters to learn, easier to generate (deciding one word from smaller dictionary), more OOV
    - Larger vocab => more parameters to learn, harder to generate, less OOV
  - Hard for certain languages with continuous script: Japanese, Chinese, Korean, Khmer, etc. Need separate word segmentation tool (can be neural networks)

Character-level Tokenization

T	h	e	m	o	s	t	e	a	g	e	r	i	s	O	r	e	g	…

- Each letter and punctuation is a token
- Pros:
  - Very small vocabulary (except for some languages, e.g. Chinese)
  - No Out-of-Vocabulary token
- Cons:
  - A sentence can be longer sequence
  - Tokens do not representing semantic meaning

Subword-level Tokenization

- Goal:
  - moderate size vocabulary
  - no OOV

- Idea:
  - represent rare words (OOV) by sequence of subwords

- Byte Pair Encoding (BPE)
  - not necessarily semantic meaningful
  - Originally for data compression Philip Gage. A New Algorithm for Data Compression, 1994

Byte-Pair-Encoding Tokenization

1. starting from chars
2. repeatedly, merge most frequent pairs to form new tokens
3. until reaching a fixed size.

raw word	freq.
cat	90
catch	50
rat	80
rattle	40

a
c
e
h
l
t

merge
(‘a’, ’t’)

a
c
e
h
l
t
at

merge
(‘c’, ’at’)

a
c
e
h
l
t
at
cat

merge
(‘r’, ’at’)

a
c
e
h
l
t
at
cat
rat

merge
(‘cat’, ’c’)

a
c
e
h
l
t
at
cat
rat
catc

Neural Machine Translation of Rare Words with Subword Units. Sennrich et al. ACL 2016

Byte Pair Encoding (BPE) for Text Tokenization

1. Initialize vocabulary with all characters as tokens (also add end-of-word symbol) and frequencies

2. Loop until vocabulary size reaches capacity
   1) Count successive pairs of tokens in corpus
   2) Rank and select the top frequent pair
   3) Combine the pair to form a new token, add to vocabulary

3. Output final vocabulary and tokenized corpus

Rico Sennrich et al. Neural Machine Translation of Rare Words with Subword Units. 2016

More Subword Tokenization

- Wordpiece:
    - like BPE
    - but instead of merge with most frequent pairs, merge a and b, if $p(b|a)$ will be maximized

- SentencePiece:
    - Uniform way to treat space, punctuation
    - Use the raw sentence, replacing space ‘ ’ with _ (U+2581)
    - Then split character and do BPE

Kudo and Richardson, SentencePiece, 2018

Code Example

- https://github.com/llmsystem/llmsys_code_examples/blob/main/tokenization/tokenization.ipynb

Sequence Decoding

$$\underset{y}{\mathrm{arg}max} P(y|\mathrm{x}) = f_\theta(x, y)$$

- naive solution: exhaustive search
  - too expensive

- Sampling

- Beam search
  - (approximate) dynamic programming

Sampling

- Instead of $\mathrm{argmax}_{\mathrm{y}} P(y|x) = f_\theta(x, y)$
- Generate samples of translation Y from the distribution P(Y|X)
- Q: how to generate samples from a discrete distribution?

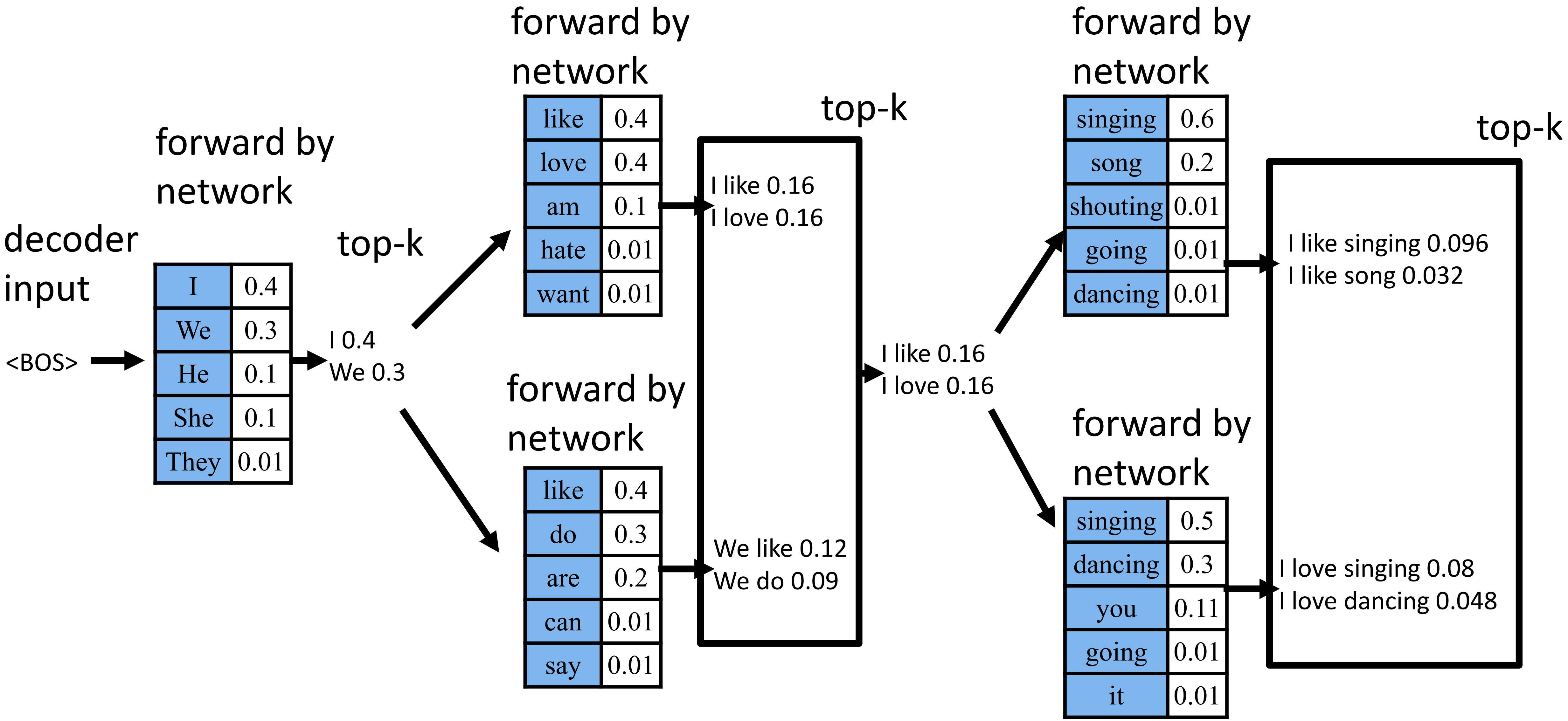
Discrete Sampling

- sample n value x from k categories, with prob. p1, p2, … pk
- Direct sampling: O(nk)
- BinarySearch: O(k + n logk)
- Alias sampling: O(k logk + n)

Beam Search

1. start with empty S
2. at each step, keep k best partial sequences
3. expand them with one more forward generation
4. collect new partial results and keep top-k

Beam Search



decoder input
<BOS>
forward by network
I 0.4
We 0.3
He 0.1
She 0.1
They 0.01
top-k
I 0.4
We 0.3
forward by network
like 0.4
love 0.4
am 0.1
hate 0.01
want 0.01
forward by network
like 0.4
do 0.3
are 0.2
can 0.01
say 0.01
top-k
I like 0.16
I love 0.16
We like 0.12
We do 0.09
I like 0.16
I love 0.16
forward by network
singing 0.6
song 0.2
shouting 0.01
going 0.01
dancing 0.01
forward by network
singing 0.5
dancing 0.3
you 0.11
going 0.01
it 0.01
top-k
I like singing 0.096
I like song 0.032
I love singing 0.08
I love dancing 0.048

Beam Search (pseudocode)

```
best_scores = []
add {[0], 0.0} to best_scores # 0 is for
beginning of sentence token
for i in 1 to max_length:
  new_seqs = PriorityQueue()
  for (candidate, s) in best_scores:
    if candidate[-1] is EOS:
      prob = all -inf
      prob[EOS] = 0
    else:
      prob = using model to take candidate and
compute next token probabilities (logp)
```

Pruning for Beam Search

- Relative threshold pruning
  - prune candidates with too low score from the top one
  - Given a pruning threshold rp and an active candidate list C, a candidate cand $\in$ C is discarded if: score(cand) $\leq$ rp $*$ max{score(c)}
- Absolute threshold pruning:
  - score(cand) $\leq$ max{score(c)} – ap
- Relative local threshold pruning

Freitag & Al-Onaizan. Beam Search Strategies for Neural Machine Translation. 2017.

Combine Sample and Beam Search

- Sample the first tokens
- continue beam search for the later
- why?

https://belladoreai.github.io/llama-tokenizer-js/example-demo/build/

Question

- How to implement tokenization efficiently?
- Some idea:
  - Binary search
  - Max heap

Project

- https://llmsystem.github.io/llmsystem2024spring/docs/Projects
- Proposal due: 2/28
- Mid term Report: 4/1
- Poster Project Presentation: 4/29
- Final Report: 4/30

Project Proposal

- What LLM System problem are you planning to address?
  - what are the **system challenges**?
- What are the existing state-of-art methods on this problem? Is the source code/model available?
- Possible directions for going forward.
- How do you evaluate the performance? what kind of workload?
- Who is your team and how are you planning to split the workload between team members?
- A rough timeline/milestones
- What CPU, GPU and storage infrastructure do need for this project? Please estimate the amount of computation time required.

Project Report Requirement

- Introduction/Motivation: This essentially lays out the problem definition, motivation, talks about why we need to work on it, the key contributions expected/presented in the work.
- Related Work/Background: This talks about key papers/works that provide context to your current work. Instead of listing down multiple past works, talk about the ones that minimally differ from your work, and how.
- Methodology: This section talks about your method, raises research questions and how you are going to address them.
- Experiments: This section can describe your experiments and the results you obtain.
- Analysis/Ablations: Typically, you would have multiple factors involved in your experimental setting. Analysis sections help you probe deeper into the results and help piece out contributions from individual modeling decisions made.
- Conclusion/Discussion: This would list the main takeaways from your work, discuss some future ideas (if any) and engage in discussion.
- Limitations: This section lays out some known limitations of your work.
- [final report only] Team Member Contributions List out each individual's contributions in this section.