Section 10

Lavender Jiang

See references for sources of images

11/13/2023

PART 01

Prompt Engineering

Temperature

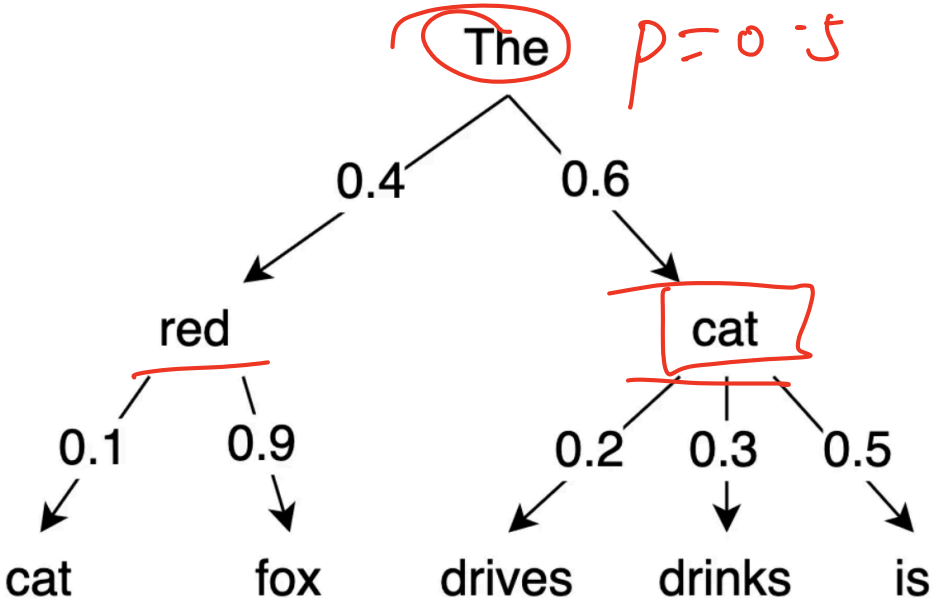
$$\sum p(x_i) = \frac{\sum \cdots}{\sum} = 1$$

$$p(x_i) = \frac{e^{\frac{x_i}{T}}}{\sum_{j=1}^{V} e^{\frac{x_j}{T}}}$$

What happens when T approach 0? How about infinity?

$P(x_i) \sim \text{uniform}$

Top-p



How does the following work on the left example:

1. Greedy search
2. beam search with n_beam=2
3. Top-k sampling
4. Top-p sampling

Max Length, Stop Sequences

Predict common sense results of the following actions.
==
Action: I didn't water the plant for 3 weeks.
Result: The plant died.
==
Action: I went to school.
Result: I got a diploma.
==
Action: I left the AC on all day.
Result: I got a high utility bill.
==
Action: I helped my neighbors when their car broke down.
Result: My neighbors were grateful.
==
Action: I put the ice cream outside for an hour.
Result: **The ice cream melted.**
==

Frequency Penalty

"Write a poem where every word starts with Z"

Frequency Penalty = 0	Frequency Penalty = 2
Zebras zigzagging zealously, Zephyrs zipping, zesty, zestfully. Zodiac's zenith, zeal's zodiac, Zinnias zigzag, zircon's zodiac. Zeppelin zooming, zigzag	Zealous zephyrs zoom, zigzagging zestily, Zinnia zones, zenith zeppelins' zone precisely. Zenith's ziggurats zealously zap, Zirconium zebras zip-zap on Zanz

Basic Prompt

Prompt

```
The sky is
```

Output:

```
blue

The sky is blue on a clear day. On a cloudy day, the sky may be
gray or white.
```

Prompt:

instruction

```
Complete the sentence:
The sky is
```

Output:

```
so  beautiful today.
```

Prompt Formatting

```
This is awesome! // Positive
This is bad! // Negative
Wow that movie was rad! // Positive
What a horrible show! //
```

Negative / positive

```
<Question>?
<Answer>
<Question>?
<Answer>
<Question>?
<Answer>
<Question>?
```

Elements of Prompts

Instruction - a specific task or instruction you want the model to perform

Add 2 7-digit numbers.

Context - external information or additional context that can steer the model to better responses

examples of addition
rules of addition

Input Data - the input or question that we are interested to find a response for

"a+b" "100 + 200"
1,0,0 + 2,0,0

Output Indicator - the type or format of the output.

"Answer: 300"

take 1st line
split by :
int(s)

Homework: 7-digit addition

Instruction - control with your_prompt
Context - control with your_prompt and your_post_processing
Input Data - given by autograder, can format with your_pre_processing
Output Indicator - controlled by your choice of prompt. Can be further processed with your_post_processing

Other knobs: your_config
Metrics: accuracy, mean absolute error, prompt length

Autograder is slow. Recommend local testing with test_prompts.py
(Exact same code as autograder, but different random seeds)
Note: for submission **do not read prompts from files**, autograder cannot find it

PART 02

Huffman Coding

Compression

JPEG

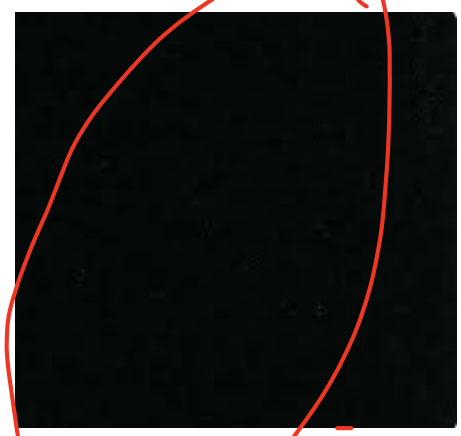
Which is harder to compress?

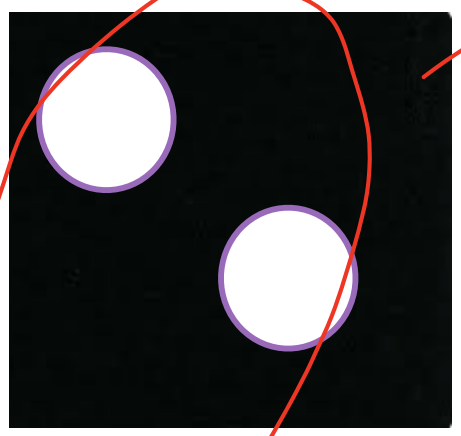
random ↑ harder ↑

all black

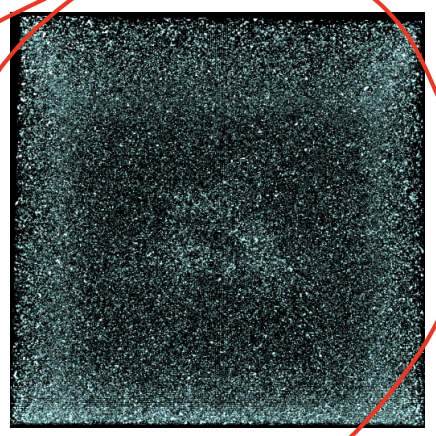
256×256

0





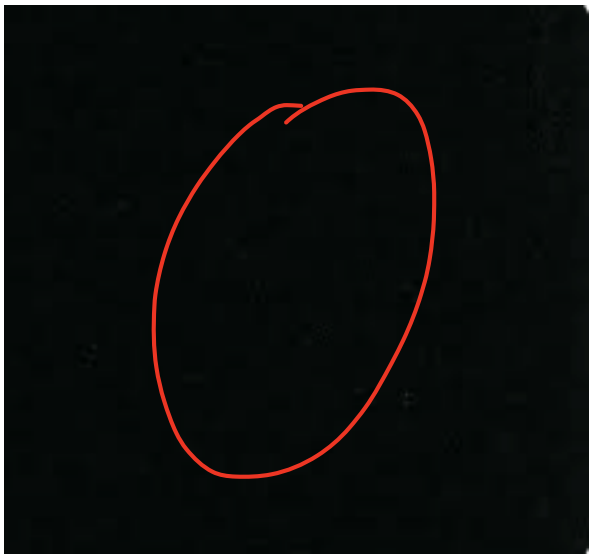
hardest



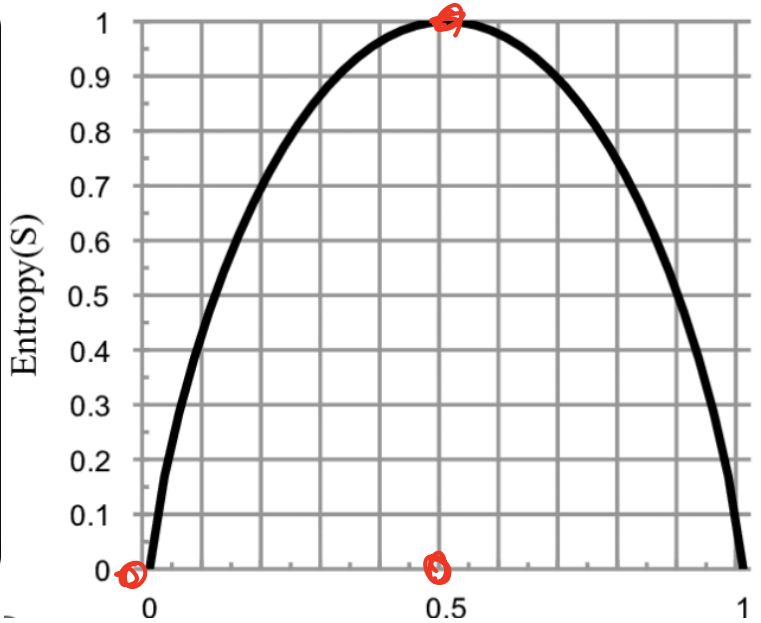
NYU

Entropy

$$H = -\sum p(x) \log p(x)$$

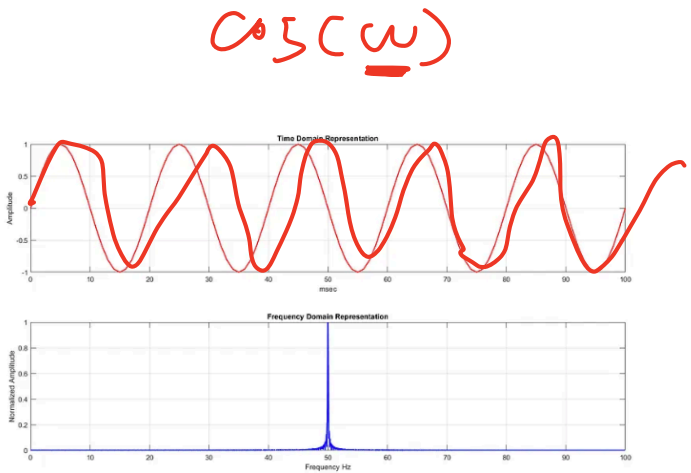


p(x) = black

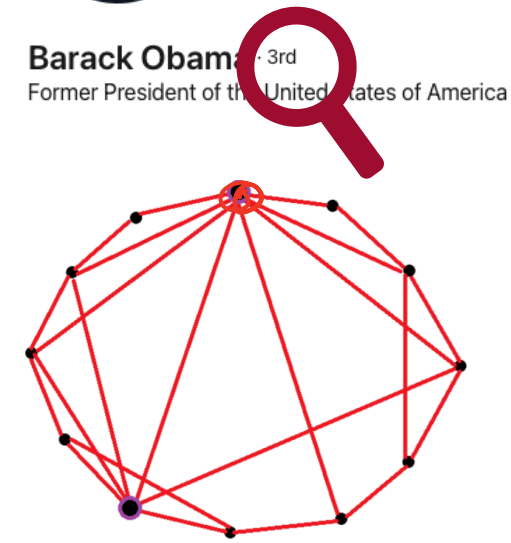


Redundancy

Smoothness/locality prior for image



Periodic prior for sound



Small-world / six-degree prior for social networks

Redundancy in English Text & Entropy of English

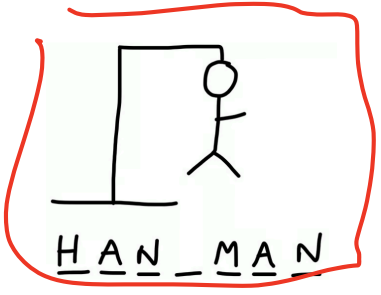
Example Rules:

1. i before e except after c.
Hippie. Fries, Field. cake

2. q must always be followed by a u
Quick, quiche, question, quarrel

3. grammar
Cannot do “subject subject” as a sentence

4. dictionary
Hufamomina is not a word



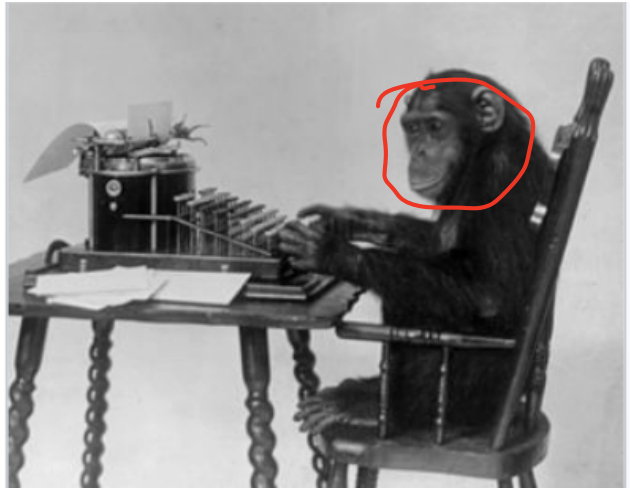
Shannon tried calculating English’s entropy using n-gram p(x_{n}| x_{<n}). Is this a good approach? O complexity?

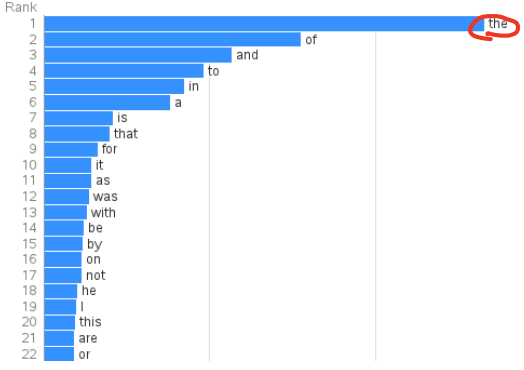
Another approach: take a set of English words (8000), calculate the word-level entropy based on the subset. Then divide by average number of characters to get Character-level entropy.

Maximum entropy: ~4.7 bit/letter
Approximated entropy of English: 2.63 bit/letter (why?)
Is English redundant? Is it good or bad?

Infinite Monkey Theorem

Almost surely, he would type up Shakespeare.





Zero-order approximation	XFOML RXKHRJFFJUJ ALPWXFWJXYJ FFJEYVJCQSGHYD QPAAMKBZAACIBZLKJQD
First-order approximation	OCRO HLO RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL
Second-order approximation	ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE
Third-order approximation	IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE
First-order word approximation	REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE
Second-order word approximation	THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

bigram

Compression relies on redundancy

Fixed length v.s. variable length encoding

	a	b	c	d	e	f
Freq in '000s	45	13	12	16	9	5
a fixed-length	000	001	010	011	100	101
a variable-length	0	101	100	111	1101	1100

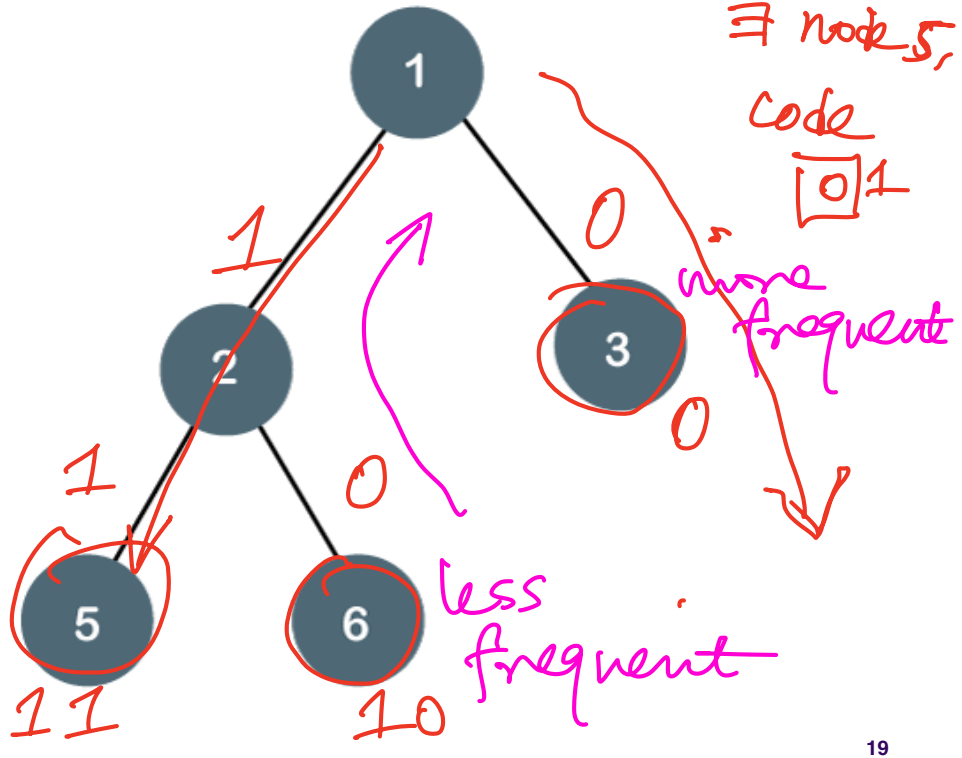
Which scheme uses fewer bits to encode the corpus?
How do we encode "bad"?
How do we decode 11000101? (Is the decoding unique? Why?)

Designing Unique Prefixes with Prefix Tree

Idea:
Greedy bottom-up construction of tree
Read encoding based on path from root to leaves.

Why it works:
Each token traces the path of a leave.
A leave has no children.
So each token has unique prefix.

Why it's efficient:
Greedy algorithm always look for least frequent token
Less frequent tokens become leaves earlier
Less frequent tokens -> longer path -> longer code



Walkthrough (unigram)

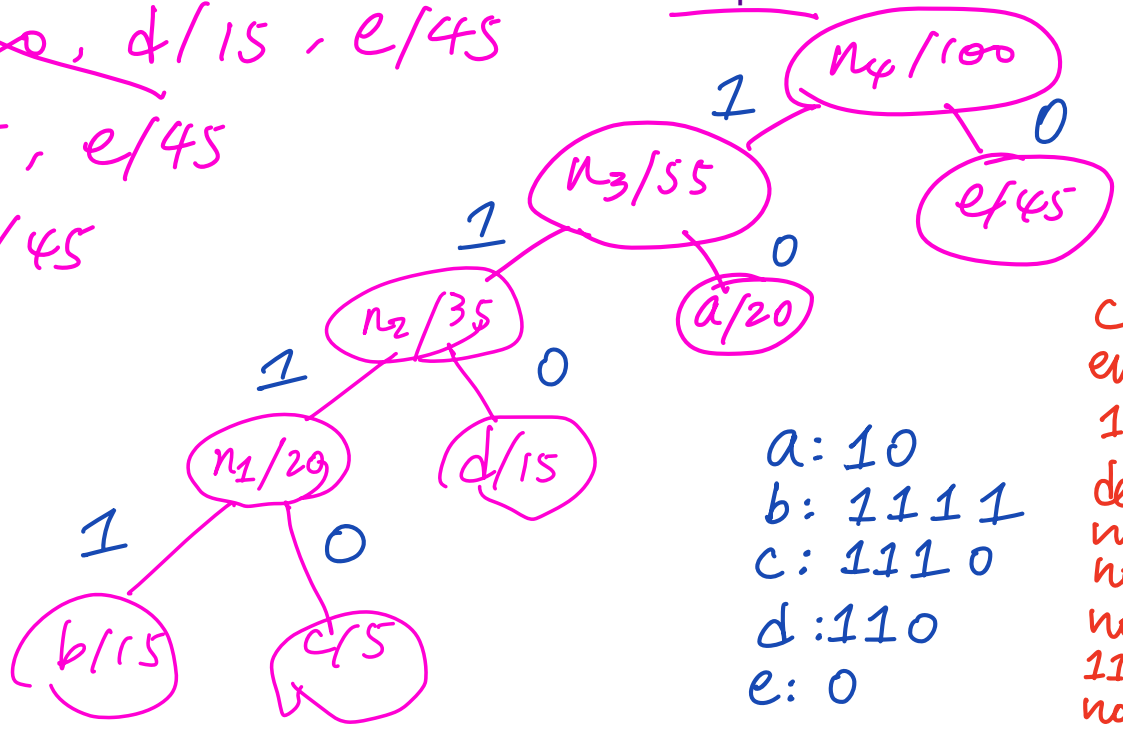
$a/20, b/15, c/5, d/15, e/45$

a/20, n_1/20, d/15, e/45

a/20, n_2/35, e/45

n_3/55, e/45

1. Pick two least frequent words (token)
2. Use them as leaves of a subtree
3. Merge frequency on their common parent
4. Add common parent back to list
5. Repeat



a: 10
b: 1111
c: 1110
d: 110
e: 0

check:
encode "bad"
1111|10|110
decode:
no 1
no 11
no 111
1111 → b
no 1
10 → a
110 → d

Walkthrough (Bigram)

$$a/20, b/15, c/5, d/15, e/45$$

The tokens here (a,b,c,d,e) can be bigrams!
e.g., a = cat | the, b = on | was

hierarchical

1. Pick two least frequent ~~words~~ joint
2. Use them as leaves of a subtree
3. Merge frequency on their common parent
4. Add common parent back to list
5. Repeat

Let's rank them by joint!
only need code for non-deterministic.

no code
example: $P(X_{t+1} \mid X_t = \langle bos \rangle) = \begin{cases} 1, & \text{if } X_{t+1} = a \\ 0, & \text{otherwise} \end{cases}$

code ✓ $P(X_{t+1} \mid X_t = a) = \begin{cases} \frac{1}{2}, & \text{if } X_{t+1} \in \{b, c\} \\ 0, & \text{otherwise} \end{cases}$

✗ $P(X_{t+1} \mid X_t = b) = \begin{cases} 1, & \text{if } X_{t+1} = \langle eos \rangle \\ 0, & \text{otherwise} \end{cases}$

✗ $P(X_{t+1} \mid X_t = c) = \begin{cases} \frac{2}{3}, & \text{if } X_{t+1} = \langle eos \rangle \\ \frac{1}{3}, & \text{if } X_{t+1} = a \\ 0, & \text{otherwise} \end{cases}$

Suppose $P(a) = \frac{1}{3}$, $P(b) = \frac{1}{4}$

$b|a$ / $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6} = \frac{2}{12}$

$c|a$ / $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6} = \frac{2}{12}$

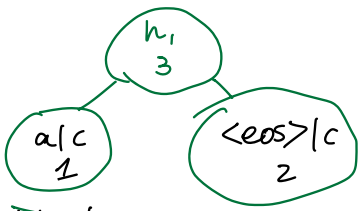
$\langle eos \rangle|c$ / $\frac{2}{3} \times \frac{1}{4} = \frac{1}{6} = \frac{2}{12}$

$a|c$ / $\frac{1}{3} \times \frac{1}{4} = \frac{1}{12}$

NYU

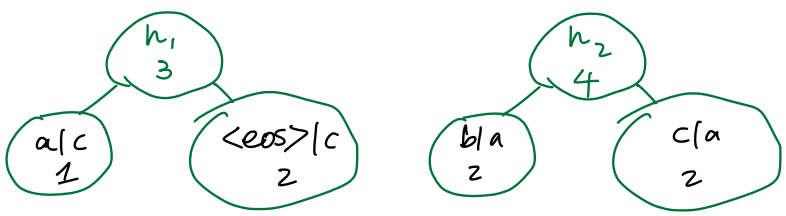
Step 1: $\mathcal{L}$ = { b|a/2, c|a/2, <eos>|c/2, a|c/1 }

pick 2 least frequent, a|c and <eos>|c



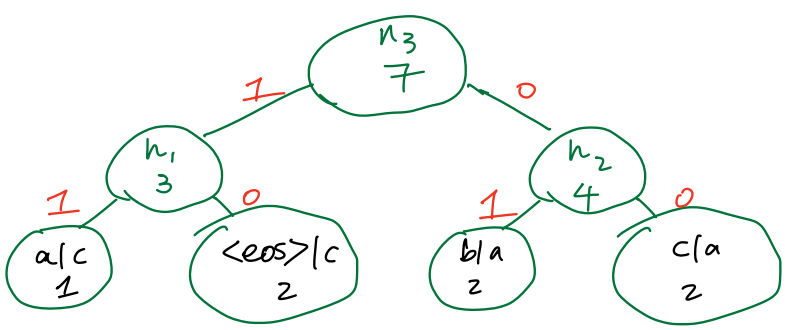
Step 2: $\mathcal{L}$ = { b|a/2, c|a/2, n_1/3 }

pick 2 least frequent, b|a and c|a



Step 3: $\mathcal{L}$ = { n_1/3, n_2/4 }

pick 2 least frequent. n_1/3 and n_2/4



Step 4: left edge 1, right edge 0 (order is arbitrary)
read code by tracing path from root to leave.

a|c: 11 <eos>|c: 10 b|a: 01 c|a: 00

Test: encode <bos>acab<eos> → 00 11 01

a|c: 11 → b|a: 01

no code deterministic

c|a: 00

decode: ① <bos>a ② no 0, 00 → c|a

100% start with this

③ no 1, 11 → a|c ④ no 0, 01 → b|a

⑤ b → 100% <eos> next

out: <bos>acab<eos>

References

https://medium.com/@lazyprogrammerofficial/what-is-temperature-in-nlp-llms-aa2a7212e687
https://medium.com/nlplanet/two-minutes-nlp-most-used-decoding-methods-for-language-models-9d44b2375612
https://docs.ai21.com/docs/when-the-generation-stops
https://towardsdatascience.com/guide-to-chatgpts-advanced-settings-top-p-frequency-penalties-temperature-and-more-b70bae848069
https://www.promptingguide.ai/introduction/basics
https://www.mathworks.com/help/images/jpeg-image-deblocking-using-deep-learning.html
https://www.researchgate.net/figure/Top-The-CNN-trained-for-the-task-of-full-face-detection-Bottom-The-CNN-trained-for-the_fig2_308944615
https://cs.stanford.edu/people/eroberts/courses/soco/projects/1999-00/information-theory/entropy_of_english_9.html
https://home.cse.ust.hk/faculty/golin/COMP271Sp03/Notes/MyL17.pdf