上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

SJTU

# Adaptive Huffman Coding: Analysis and Applications

Jiaxin Li, Dingyao Tao, Benhao Huang

Monday, May 22, 2023

饮 水 思 源 • 爱 国 荣 校

# Content

# Introduction

## Our Research is inspired by the Bad Wine in Cover's Textbook

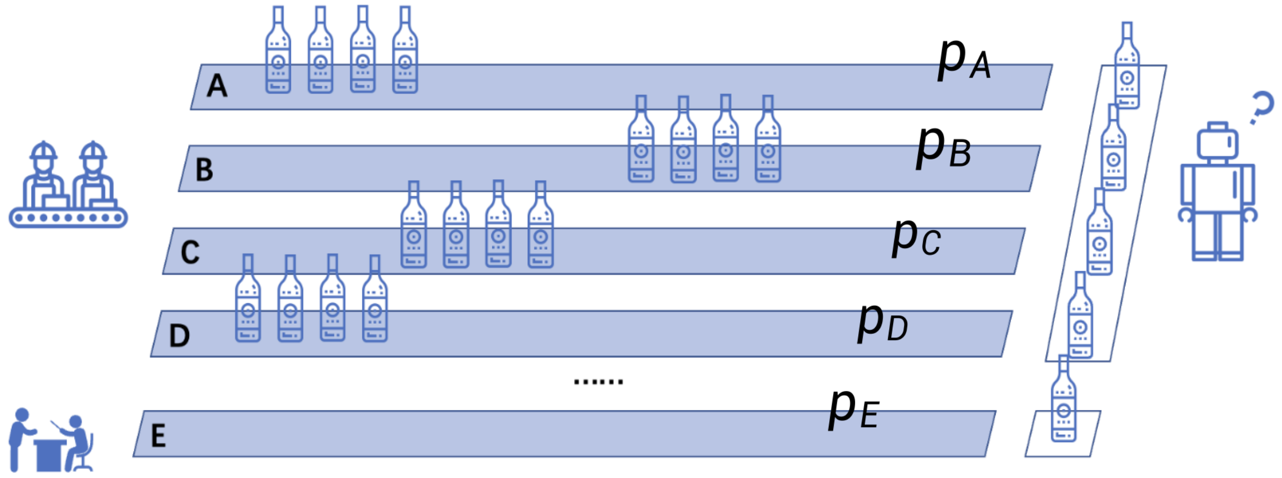
One bottle of wine is bad, and we can quickly find it by mixing several bottles of wine and tasting them, once given the probability of each wine being bad.

# What if the probability is dynamic?

## Probability Distribution is Unknown!

*Bad Wine Pipeline*

What's checking strategy to quickly find the bad wine pipeline?

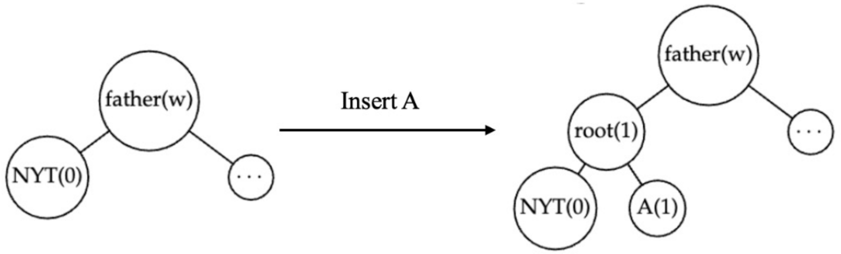


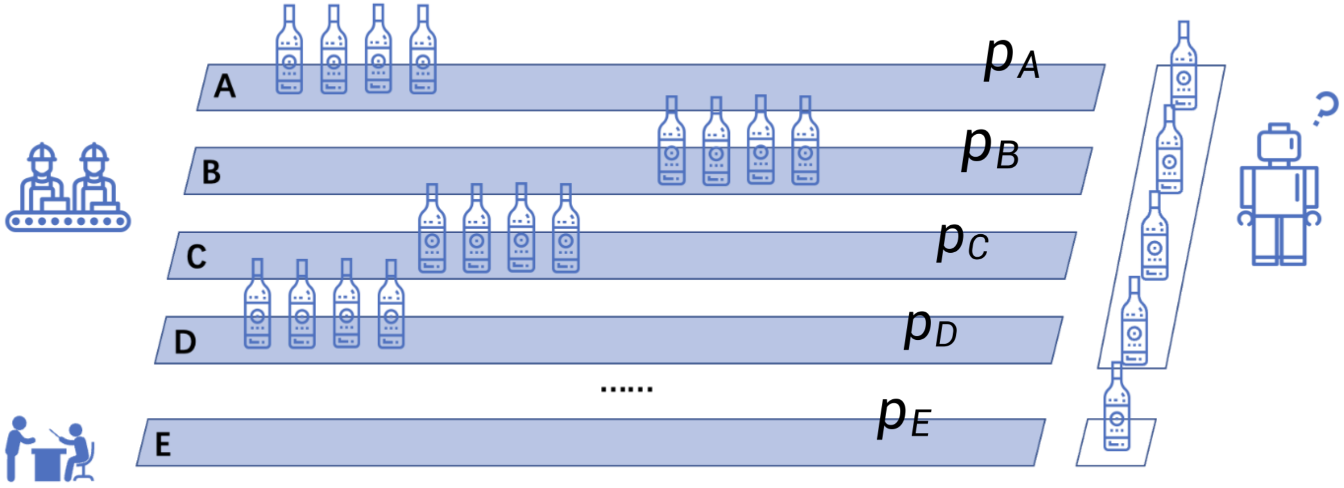

receive exactly one bad wine at a time

**We could use Adaptive Huffman Coding !**

# Solutiion: Adaptive Huffman Tree

Equal to insert a symbol A into Adaptive Huffman Tree !




Every time robot spots a bad wine in pipeline A, pA should increase, and we should correspondingly change our strategy.

# Adaptive Huffman Coding Algorithm

**Definition 1.** *NYT node stands for 'Not Yet Transmitted'. In the building procedure of an adaptive Huffman tree, it serves as auxiliary node. In data compression and transmission procedure, it's an escape code.*

**Definition 2.** *The weight of a node is the number of times it appears, which is denoted by the number in the bracket of the node.*

**Definition 3.** *Block: a group of nodes who have the same weight.*

**Definition 4.** *Node Number: The number of a node is up to the structure of Adaptive Huffman Tree. Starting at 1, it increases from left to the right, bottom to the top.*

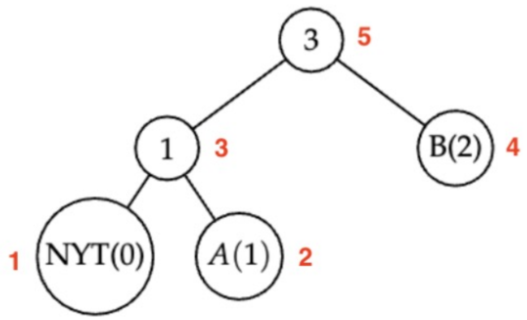
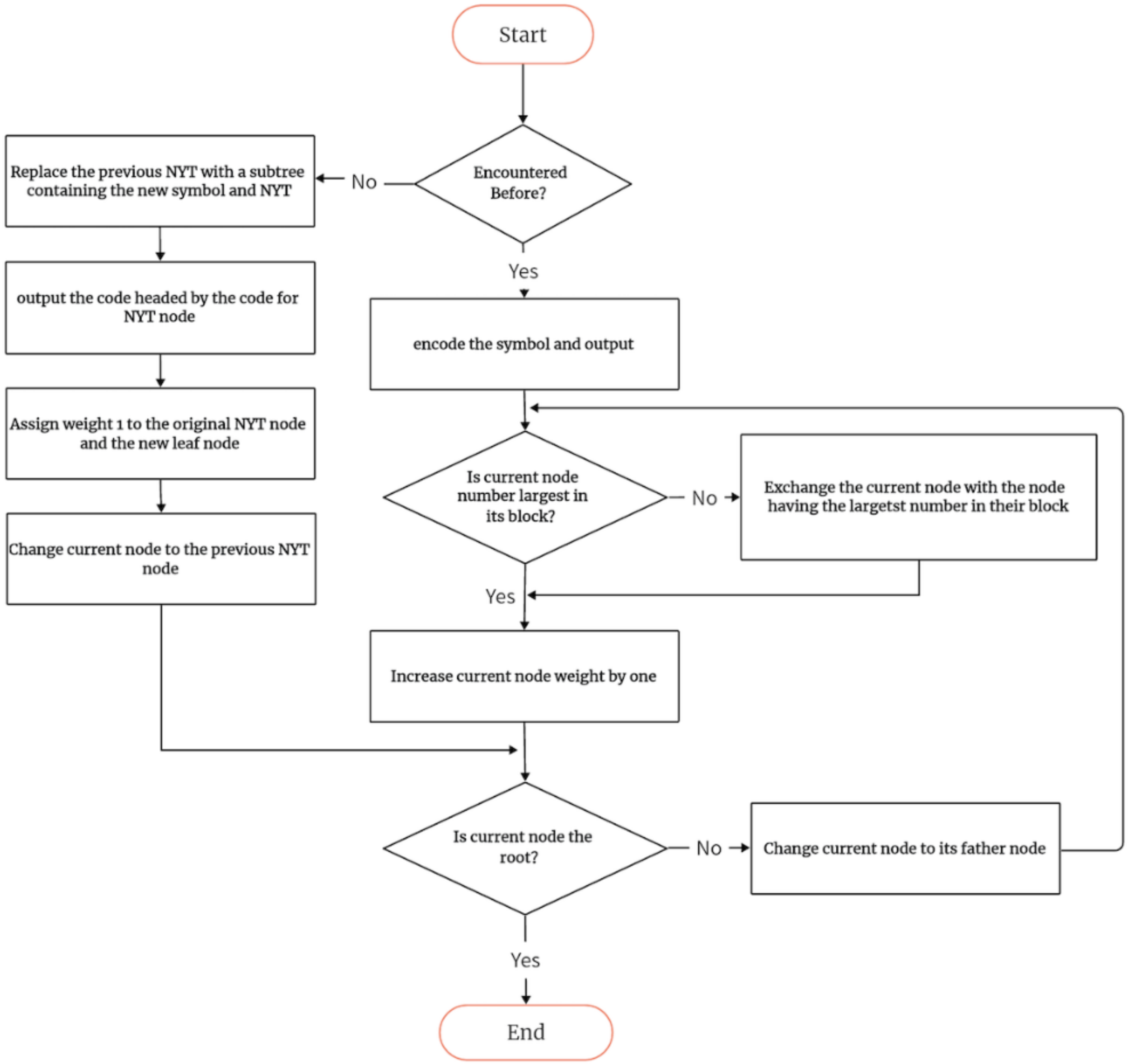
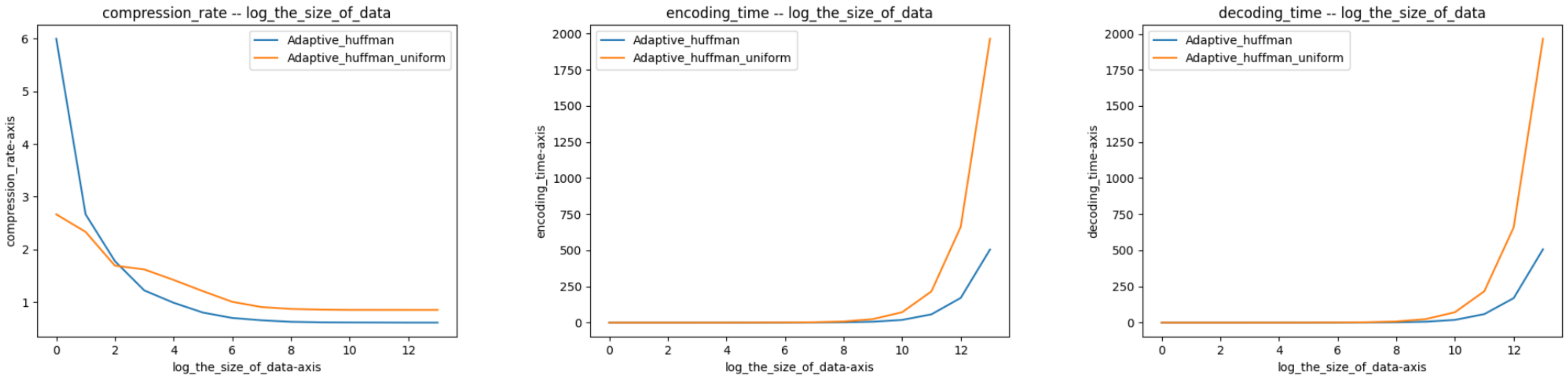



Figure 3: the number of nodes



# Perfomance from an encoding perspective

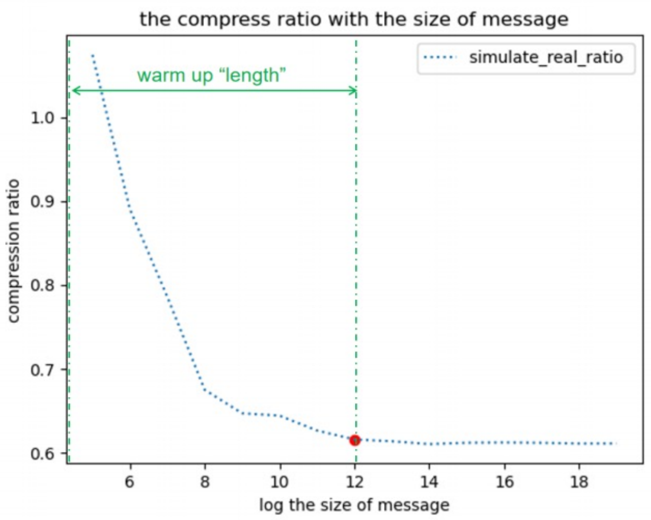
The best iterval for the size of message is $[2^7, 2^{12}]$ in our experiment environment!



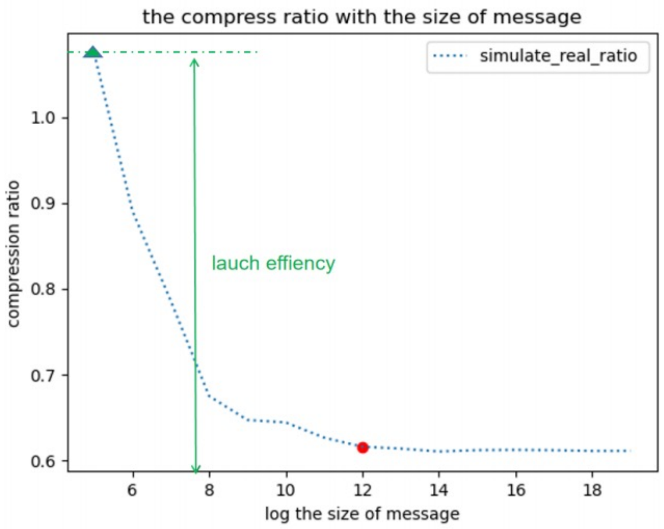

compression rate | encoding time | decoding time

# Metrics: How to measure the "learning" velocity

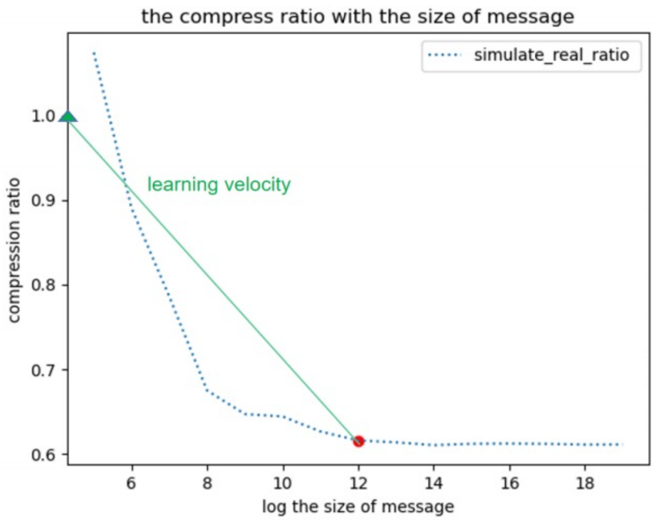
**Definition: the convergence state**: when the length of message is doubled, the decrease of compression ratio is less than 0.01 continuously.




Warm-up length




Launch efficiency




Learning velocity

# Experiment: The Affect Factor- letter Distribution

CONCLUSION:

1) The **continuity** of letters makes warm-up easy!
2) The **uniform** distribution makes the launch efficiency worse!
3) The **regularity** of letters makes learning fast!

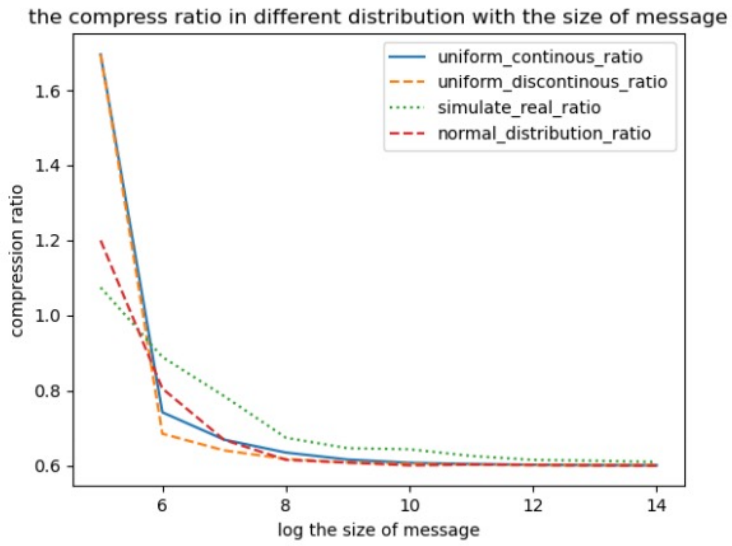
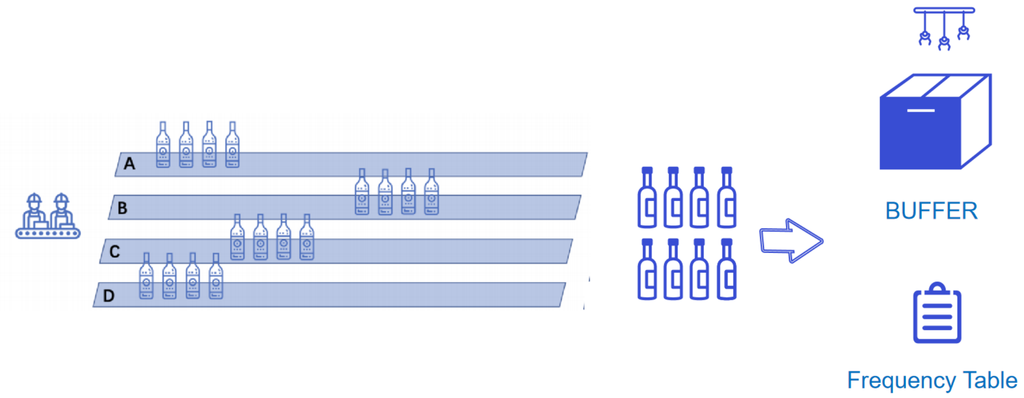



Figure 8: The compression ratio with the size of message in different distribution

Table 1: The metrics of different probability distribution

| distribution | $L_{warm}/(\text{byte})$ | $R_v/1$ | $E_{launch}/1$ | $v_{learning}/log(byte)^{-1}$ |
|---|---|---|---|---|
| continuous uniform | $2^9$ | 0.383 | 1.698 | 0.0425 |
| discontinuous uniform | $2^{18}$ | 0.381 | 1.698 | 0.0477 |
| simulation | $2^{12}$ | 0.384 | 1.074 | 0.0320 |
| normal | $2^8$ | 0.384 | 1.2 | 0.0480 |

# Accelerate the learning velocity: Buffer makes surprise!




**Algorithm 1** Algorithm of Apative with Buffer

**Input:** message ,buffersize
**Output:** deliver code to the decoder

```
while  message.remain.size > 0  do
    while Buffer.size <= BufferMaxSize and message.remain.size > 0  do
        Buffer.Push( message.remain.pop())
        Get the Statistical frequency table of the elements in buffer .
        Deliver this table to the receiver.
        Using this statistical frequency table to update the Huffman Tree in decoder and
        encoder.
        Using the updated Huffman Tree to encode the elements in buffer. And deliver the
        code to the decoder.
    end while
end while
```

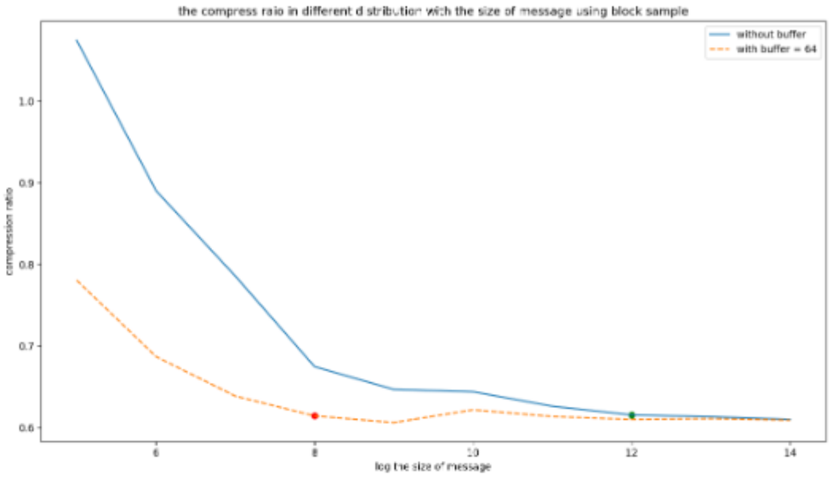



Figure 9: The compression ratio with the size of message

Table 2: The metrics of different probability distribution

| distribution | $L_{warm}$/(byte) | $R_v$/1 | $E_{launch}$/1 | $v_{learning}/log(byte)^{-1}$ |
| --- | --- | --- | --- | --- |
| with buffer | $2^8$ | 0.387 | **0.78** | **0.0483** |
| without buffer | $2^{12}$ | 0.384 | 1.074 | 0.0320 |

# Application Scenario: Hierarchical Softmax in CBOW

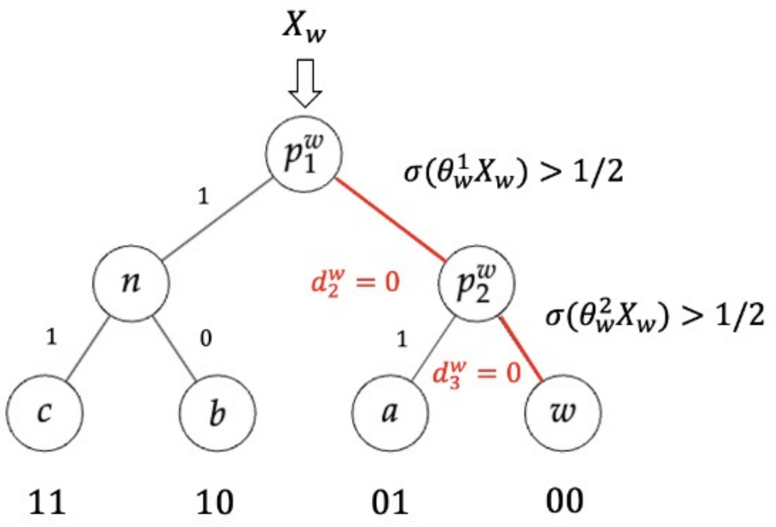
CBOW: Continuous Bag-of-Words

Hierarchical Softmax: a method to improve performance of vec2word models

## Biased Walk on Huffman Tree

Input the information of word W, we try to output W by walk on the corresponding Huffman Tree of word set.

$$\begin{cases} \sigma(\theta_w^1 X_w) > 1/2 & \text{Go Right} \\ \textbf{ELSE:} & \text{Go Left} \end{cases}$$



# Trials on Incremental Learning

It's costly to rebuild Huffman Tree and retrain model when adding new datas.....

## Incremental Huffman Tree

M. Nilufar and A. Abhari, "Incremental text clustering algorithm for cloud-based data management in scientific research papers," in 2022 Annual Modeling and Simulation Conference (ANNSIM), 2022, pp. 778–789

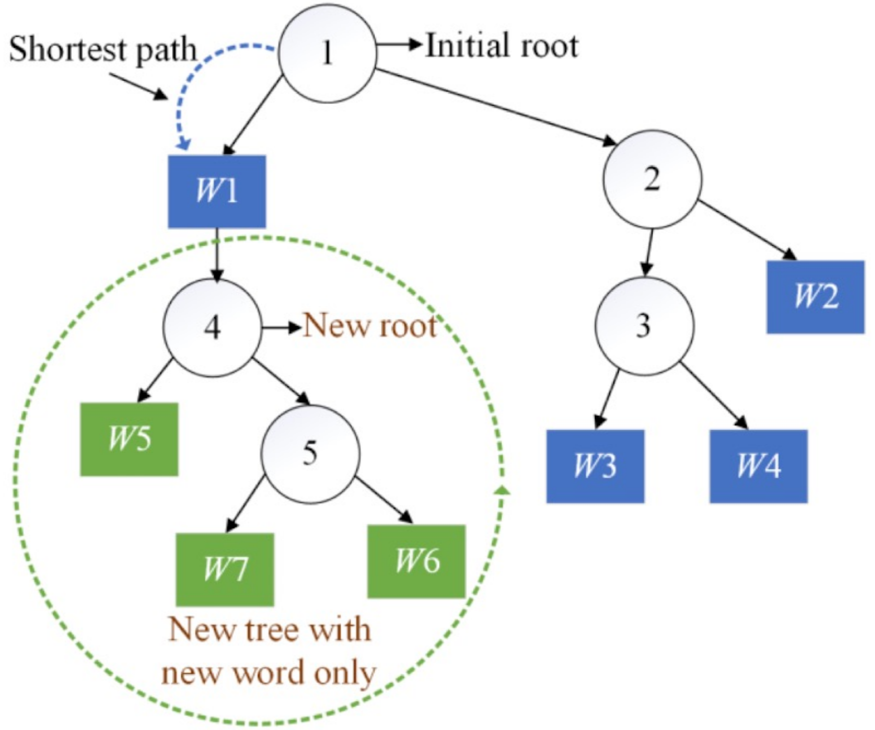
L. Tian, X. Wen, Z. Song et al., "An online word vector generation method based onincremental huffman tree merging," Tehniˇcki vjesnik, vol. 28, no. 1, pp. 52–57, 2021.i

**Algorithm 2** Incremental Huffman Tree

**Input:** previous Huffman Tree $T_p$ , new dataset

**Output:** updated "Huffman Tree" $T_u$

1: **for** word in new dataset and not in previous Huffman Tree **do**
2: add word into new word set
3: **end for**
4: build a new Huffman Tree $T_n$ on new word set
5: **Merge:** find the shortest path of $T_p$ and make its leaf node $n_s$ the root of $T_n$
6: update the Huffman Code of leaf nodes in $T_n$ by adding the code of $n_s$ as prefix

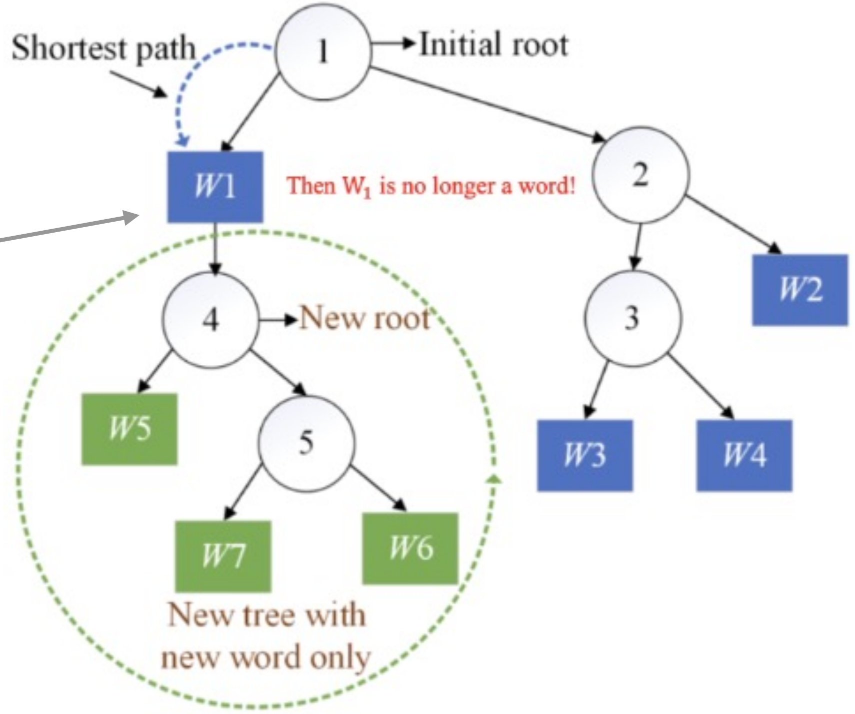


# Some deficiencies we found

Incremental Huffman Tree has several drawbacks....

**Problem 1.** *The accuracy of the model will decrease with this training method.*

A frequently used word being removed！

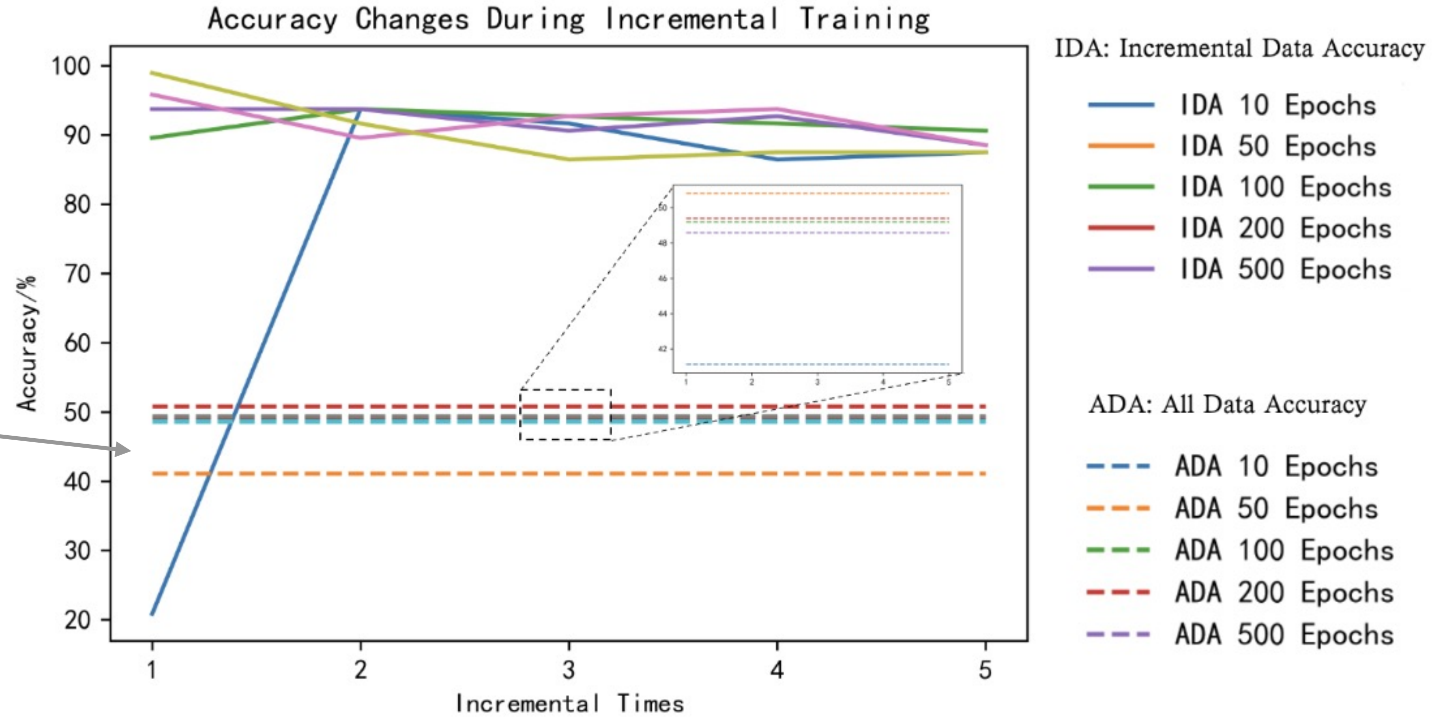


# Some deficiencies we found

Incremental Huffman Tree has several drawbacks....

**Problem 2.** *This method can't really implement incremental learning.*

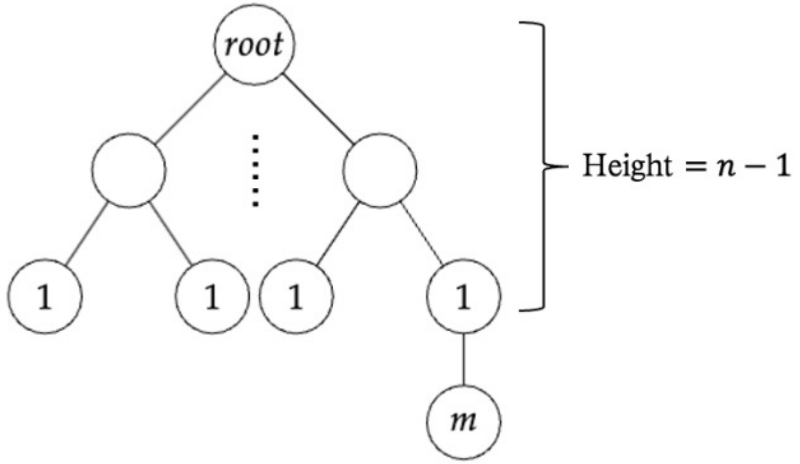
**Bad Accuracy on the whole dataset !**



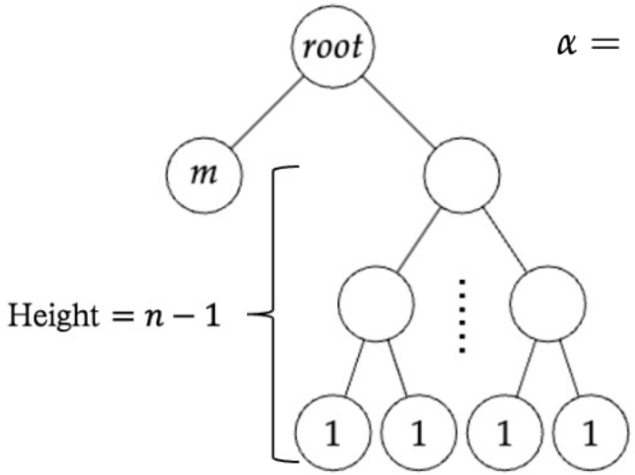
# Some deficiencies we found

Incremental Huffman Tree has several drawbacks....

**Problem 3.** *Incremental Huffman Tree's code length could be very bad, as data set becoming very large, which also means Incremental Huffman Tree has large height.*




Incremental Method




Naive Huffman

$$\alpha = \frac{2^{n-1}(n-1)+nm}{2^{n-1}n+m} \longrightarrow n, \text{with } m \to \infty,$$

# Our Refinement: PIWA Algorithm

How about changing incremental to adaptive ?

**Algorithm 3** PIWA:Partial Incremental With Adaptive Huffman

**Input:** previous Huffman Tree $T_p$ , new dataset

**Output:** updated Huffman Tree $T_u$

1: **for** word in new dataset **do**

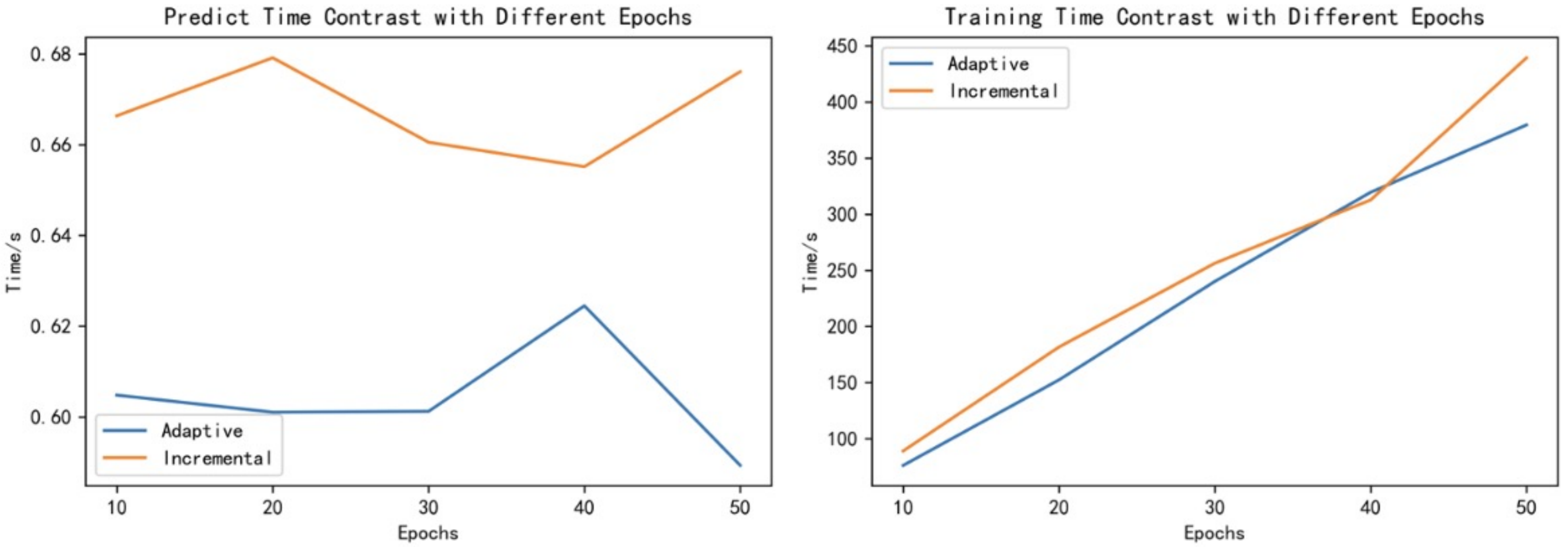
2:     use Adaptive Huffman Tree algorithm to update the tree

3: **end for**

4: $dataset \leftarrow old\ dataset \cup new\ dataset$

5: Retrain the model on *dataset*

# PIWA: Training and Predicting Time



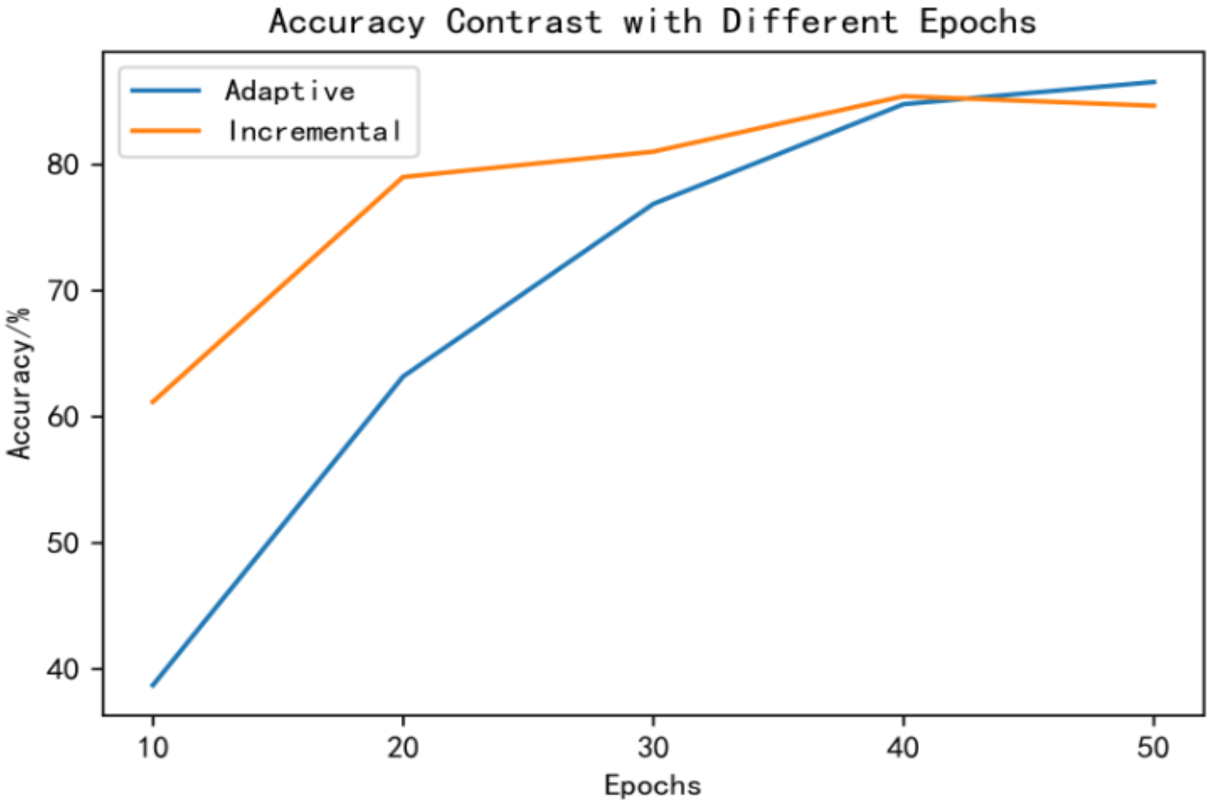

- predicting time is mainly related to the average tree height, PIWA has smaller tree height
- training time of the PIWA increases more steadily with the increase of the epoch, while the Incremental method has a change in slope.

# PIWA: Accuracy

## Observations

- With the increase of epoch,the accuracy of Incremental method increases gently.
- When epoch=50, it is overtaken by Adaptive method.
- incremental method is verified to reduce accuracy, though having high accuracy at begining.

**Our PIWA is Better!**



# Future Work

## Adaptive Huffman Code With Buffer

- We just implemented the process of encode and analyzed its performance. The process of decoder can be implemented in the future to make the Adaptive Huffman Code With Buffer complete.
- To truly implement Incremental Learning, we may have to modify the whole structure of thehierarchical softmax model.

## Modification on PIWA

- Combine the research on improving the performance of Adaptive Huffman Tree with PIWA.
- To truly implement Incremental Learning, we may have to modify the whole structure of thehierarchical softmax model.

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

# Thanks For Listening


饮 水 思 源　爱 国 荣 校