Context-Based Compression

* Explain Dynamic Markov Compression.

=> Dynamic Markov Compression is a technique used in data compression to reduce the size of data.

In Dynamic Markov Model, Model is continuously updated and refined new symbols are processed.

Dynamic Markov Compression is type of lossless data compression algorithm.

Dynamic Markow Model works by modeling the data as a Markov chain.

A Markov chain model is shows the a sequence of events, where the probability of each event depends on events.

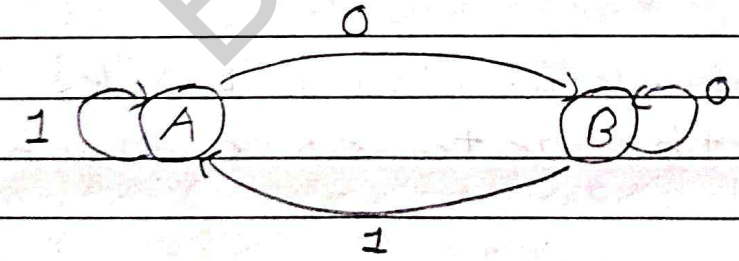
This uses the Markov chain model to predict the next bits of the data in the sequence.

DMC is required more memory or space for implement.

Dynamic Markov Compression has a good compression ratio and similar to ppm algorithm.

It is particularly effective for compressing text data compression.

Example:



→ Application:

1 Text Compression: Widely used in compressing text documents.

2 Data Transmission: It is used in applications where efficient data transmission is crucial.

3 Storage Optimization: DMC is applied in storage systems to optimize disk space usage.

4 Embedded Systemts: DMC is utilized in embedded system with limited storage.

-> Limitations:

1 Context Dependency: DMC relies on context-based modeling, which means it performs best when symbol have clear statistical dependencies.

2 Memory Usage: Maintaining dynamic models and updating probabilities require more memory.

3 Lossless Compression Only.

* Explain Partial Predictive Matching (PPM) with example.

=> PPM is an adaptive statistical compression algorithm used for compressing text data.

PPM uses the context model and prediction to efficiently encode symbols.

=> Steps:

1 First Assume the n symbols which are already encoded.

2 After that Assume the most higher Order context (Always we take context as 2).

3 Create Context-based table For higher order.

For Context 2, we have to create -1, 0, 1 and 2 context-order table.

4 Take Lower Limit L = 000000 = 0
Upper Limit U = 111111 = 63

Formula,

Lower Limit $L = L + \left([U-L+1] \frac{CC-1}{TC} \right)$

Upper Limit $U = L + \left[(U-L+1) \frac{CC}{TC} \right] - 1$

CC = Cum. Count
TC = Total Count

=> Example : this ␣ is ␣ the ␣

Here, We Assume, 7 Symbol's are already encoded.

Encoded Symbol : this ␣ is

High Order Context = 2, we have to create Four Table. (-1, 0, 1, 2).

-> For Context order = -1 -> we have to take whole string

Letters	Count	Cum_Count
t	1	1
h	1	2 (1+1)
i	1	3 (2+1)
s	1	4
e	1	5
b	1	6

For -1 order:
- Only Distinct character
- For -1 Order Count = 1
- Cum_Count = Count + Cum_Count

-> For Context Order = 0 -> Take only Assume encoded sequence this b is

Letters	Count	Cum_Count
t	1	1
Distinct h	1	2
character i	2	4
s	2	6
b	1	7
-> <ESC> Add at last	1	8

Count = How many time come in Assume string

Cum_Count = Count + Cum_Count

-> For Context Order = 1 -> Assume string - this is

Context	Letter	Count	Cum_Count
t	h	1	1
	<ESC>	1	2
		TC= 2	
h	i	1	1
	<ESC>	1	2
		TC=2	
i	s	2	2
	<ESC>	1	3
		TC=3	
s	␣	1	1
	<ESC>	1	2
		TC=2	
␣	i	1	1
	<ESC>	1	2
		TC=2	

Context = Every distinct character
letter = Context letter Previous letter
Count = How many time occured After context, letter.
After every letter add <ESC>.

-> For Context Order = 2

Context	letter	Count	Cum_Count
th	i	1	1
	<ESC>	1	2
		TC= 2	
hi	S	1	1
	<ESC>	1	2
		TC= 2	
is	K	1	1
	<ESC>	1	2
		TC=2	
SK	i	1	1
	<ESC>	1	2
		TC=2	
Ki	S	1	1
	<ESC>	1	2
		TC=2	

Context = First Two letter together
letter = After context letter.

SMVS
Page No.
Date : / /

Here, We Assume 7 character already encoded.

Remaning string will be encoded using the formula.