Information Compression

روش‌های مبتنی بر محتوا

Context-based compressing

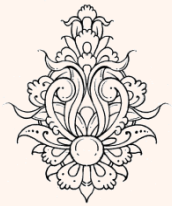
فشرده‌سازی اطلاعات
۴۰-۱۰-۷۰۲-۰۱
بخش سوم

دانشگاه شهید بهشتی
پژوهشکده‌ی فضای مجازی
بهار ۱۳۹۷
احمد محمودی ازناوه

فهرست مطالب

- معرفی
- پیش‌بینی متن
- پیش‌بینی با تطابق محدود (ppm)
  - مثال

- Burrows-Wheeler Transform
- Move To Front
- Run Length Encoding
- Associative Coder of Buyanovsky

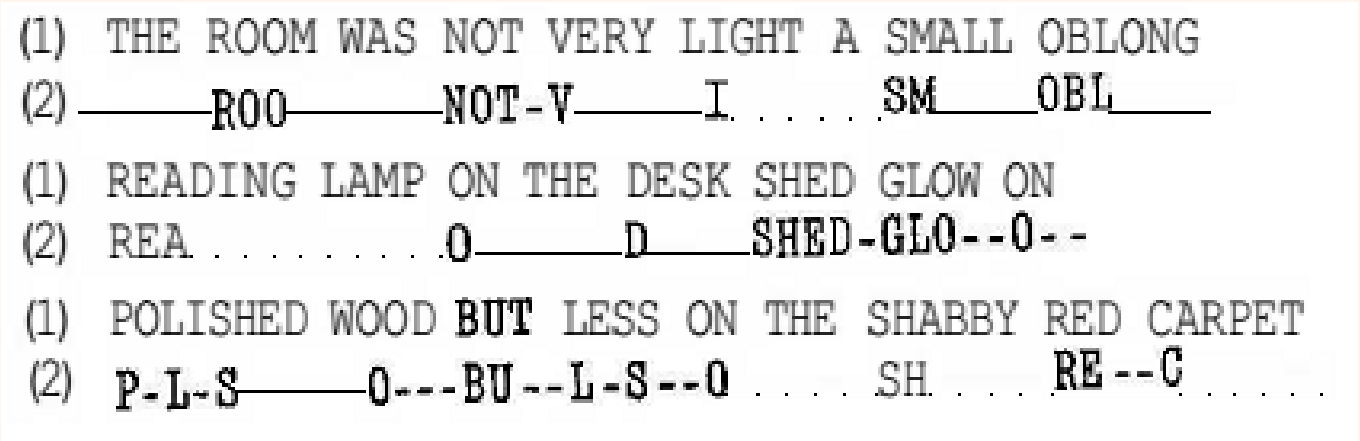


معرفی

- در صورت وجود عدم توازن احتمال وقوع داده(skewness)، نسبت فشرده‌سازی افزایش می‌یابد.
- احتمال توزیع داده‌ها در صورت در اختیار داشتن محتوا(در صورت داشتن همبستگی بالا)، دارای عدم توازن بیشتری خواهد بود.
  - محتوا باید در کدگذار و کدگشا در اختیار باشد.
- در این دسته از روش‌ها فرض بر این است که در مورد ویژگی‌های آماری دادگان اطلاعات کمی در اختیار داریم.
- از محتوای داده‌های کدشده برای کدگذاری استفاده می‌شود.
  - این شیوه وفقی است.
  - بر اساس مدل‌سازی محتوا عمل می‌کند.
  - به جای این هر نماد به صورت مستقل بررسی شود، با توجه به محتوا مورد بررسی قرار می‌گیرد.
  - برای کدگذاری عمدتا از کدگذاری محاسباتی و یا کدگذاری Huffman استفاده می‌شود.

پیش‌بینی متن- آزمایش ۱

- یک متن در نظر گرفته می‌شود و از یک کاربر می‌خواهیم (ادامه‌ی) متن را حدس بزند، در صورت حدس اشتباه نماد درست به او گفته شده و کار ادامه می‌یابد.



- در ۶۹٪ موارد پیش‌بینی درست بوده است.

Shannon, C. E. (1951). "Prediction and entropy of printed English." Bell system technical journal 30(1): 50-64.

پیش‌بینی متن- آزمایش ا(ادامه...)

- چنان‌چه که بتوان در کدگشا(گیرنده) حدس مشابهی زد، می‌توان برای فشرده‌سازی از چنین پیش‌بینی استفاده کرد:

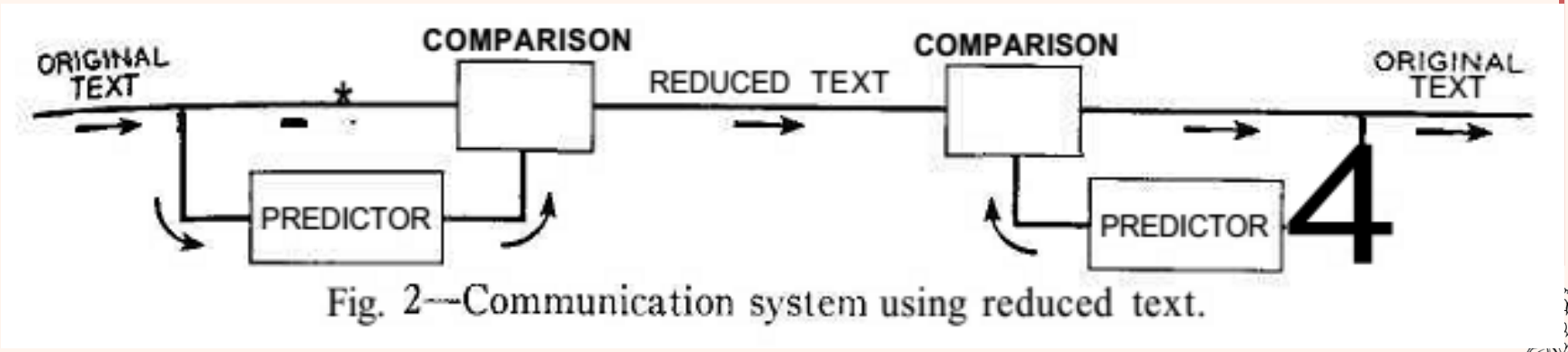
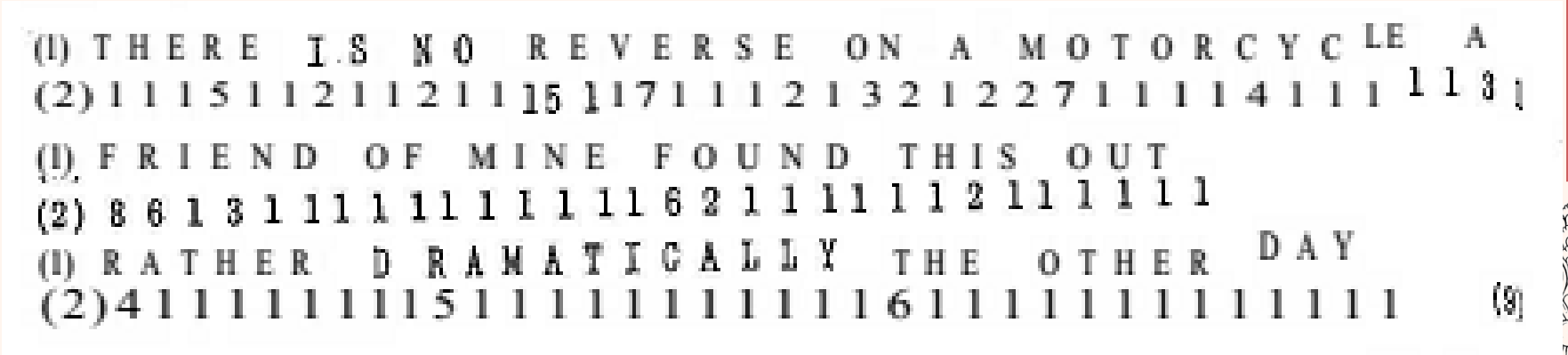


Fig. 2—Communication system using reduced text.

پیش‌بینی متن- آزمایش ۲

- در آزمایش دوم در صورت حدس اشتباه کاربر حدس دیگری را جایگزین می‌کند و این کار آن‌قدر ادامه می‌یابد تا نماد مورد نظر درست حدس زده شود.

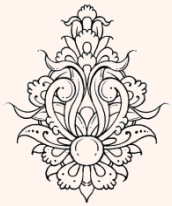


بر اساس این آزمایش برای نمایش الفبای انگلیسی بین ۰,۶ تا ۱,۳ بیت به ازای هر حرف نیاز دارد.

Shannon, C. E. (1951). "Prediction and entropy of printed English." Bell system technical journal 30(1): 50-64.

پیش‌بینی

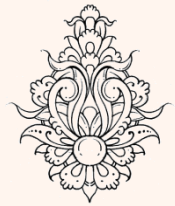
- یک راه نگهداری همه‌ی N-تایی‌ها(N-grams) و حدس بر اساس احتمال وقوع آن‌هاست.
- در این صورت برای M نماد، محتوای مرتبه‌ی اول یک جدول M-تایی، محتوای مرتبه‌ی دوم یک جدول M^2-تایی و ... خواهد بود.
- محتوای مرتبه‌ی ۵ برای یک الفبای ۲۵۶ تایی برابر با $۲۵۶^۵$ خواهد بود.



پیش‌بینی با تطابق محدود

Prediction with Partial Match (ppm)

- به جای آماده‌سازی و ذخیره‌ی جداول شامل احتمال‌های شرطی، می‌توان این جداول را در طی عملیات کدگذاری آماده نمود.
- بدین‌ترتیب فقط دنباله‌هایی که در متن وجود دارد، ذخیره می‌شوند.
- هر چند این کار نیاز به حافظه را کاهش می‌دهد، اما لازم است نمادهایی که تا کنون دیده نشده‌اند، به گونه‌ای مشخص شوند.
  - استفاده از یک نماد توالی-گریز(Escape) لازم است.



Cleary, J. and I. Witten (1984). "Data compression using adaptive coding and partial string matching." IEEE transactions on Communications 32(4): 396-402.

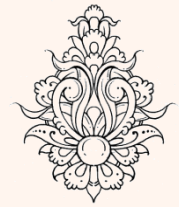
الگوریتم پایه

- در این الگوریتم ابتدا سعی می‌شود پیش‌بینی بر پایه حداکثر طول مورد استفاده انجام شود.
  - حداکثر طول محتوای مورد استفاده از پیش تعیین می‌شود.
- در صورت نبود نماد، در کدگذاری از یک نماد توالی‌گریز استفاده شده و طول محتوای مورد بررسی را یک واحد کاهش داده و همین روند را ادامه می‌دهیم.
- در صورتی که نماد پیش از این دیده نشده بود با احتمال $1/M$ کدگذاری انجام می‌شود.

پس از کد کردن نماد، جداول مربوطه هم به روزرسانی خواهند شد

الگوریتم پایه(ادامه...)

- هدف کد کردن رشته‌ی مقابل است: ***probability***
- با فرض آن که حرف «a» بناست کد شود.
- ابتدا بررسی می‌شود که آیا دنباله‌ی«proba» پیش از این دیده شده است یا نه؟
- در صورت عدم مشاهده، پس از کدکردن کاراکتر توالی‌گریز، دنباله «roba» بررسی می‌شود.
- در صورت عدم مشاهده، کاراکتر توالی‌گریز و به همین ترتیب «oba»
- و در پایان حرف «a» به منزله محتوای مرتبه‌ی صفر



مثال

حداکثر طول محتوا: ۲

this is the tight

این کاراکتر باید کد شود

TABLE 6.2 Count array for zero-order context.

Letter	Count	*Cum_Count*
t	1	1
h	1	2
i	2	4
s	2	6
␣	1	7
⟨*Esc*⟩	1	8
Total_Count		8

TABLE 6.3 Count array for first-order contexts.

Context	Letter	Count	*Cum_Count*
t	*h*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
h	*i*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
i	*s*	2	2
	⟨*Esc*⟩	1	3
	Total_Count		3
␣	*i*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
s	␣	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2

this is the tight

ادامه‌ی مثال

TABLE 6.4 **Count array for second-order contexts.**

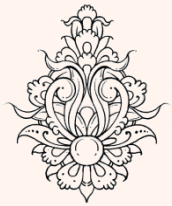
Context	Letter	Count	*Cum_Count*
th	*i*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
hi	*s*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
is	*␣*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
s␣	*i*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2
␣i	*s*	1	1
	⟨*Esc*⟩	1	2
	Total_Count		2

TABLE 6.5 **Updated count array for zero-order context.**

Letter	Count	*Cum_Count*
t	1	1
h	1	2
i	2	4
s	2	6
␣	2	8
⟨*Esc*⟩	1	9
Total_Count		9

پیش‌بینی با تطابق محدود(جمع‌بندی)

- نسبت به LZ کندتر است.
- به حافظه‌ی زیادی احتیاج دارد.
- نمونه‌های پیاده‌سازی جدید آن در زمره‌ی بهترین روش‌ها برای فشرده‌سازی متن‌های شامل زبان طبیعی است.



فشرده‌سازی با BWT

The Burrows-Wheeler Transform

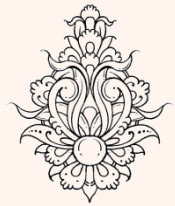
- از محتوا استفاده می‌کند.
- سال ۱۹۹۴ مطرح شد.
  - تبدیل مورد استفاده در سال ۱۹۸۳ توسط wheeler مطرح شده بود.
- برای متن مناسب است.
- لازم است کل داده‌ها پیش از کدگذاری در اختیار باشد.
- در bzip2 استفاده شده است.

Burrows, M. and D. J. Wheeler (1994). A block-sorting lossless data compression algorithm, Digital Systems Research Center.

The Burrows-Wheeler Transform

- یک دنباله شامل N-1 نماد را در نظر بگیرید.
- این دنباله را هر بار به صورت دورانی یک واحد (به سمت/چپ) راست شیفت داده و N-1 دنباله‌ی حاصل را به صورت «الفبایی» مرتب کنید.
- خروجی تبدیل، نمادهای آخر دنباله‌های مرتب شده و موقعیت دنباله‌ی اصلی است.

Lexicographical order



مثال- تبدیل BW

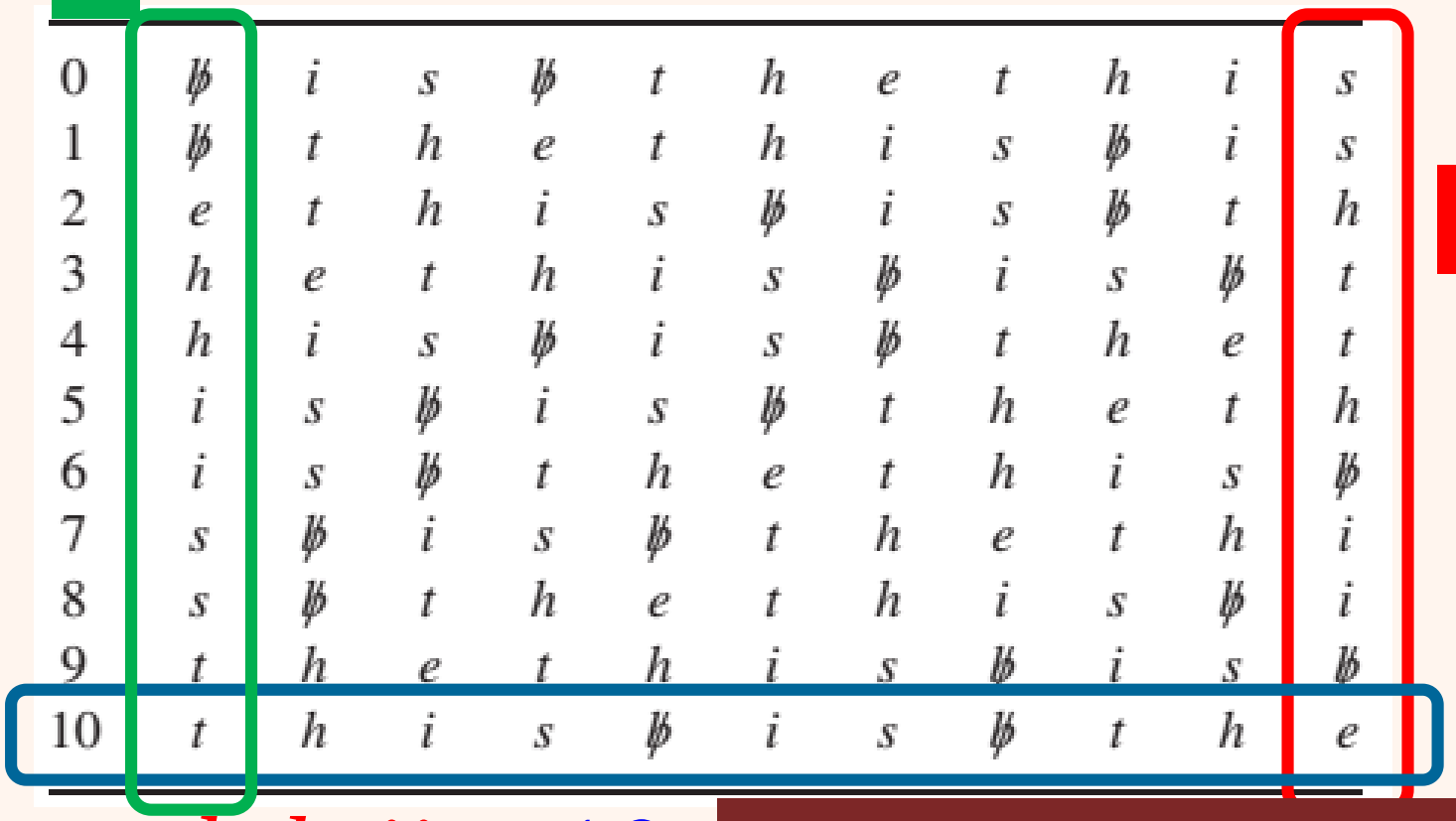
this is the

دنباله‌های حاصل از چرخش دنباله‌ی ورودی:

00	*t*	*h*	*i*	*s*	*ϕ*	*i*	*s*	*ϕ*	*t*	*h*	*e*
01	*h*	*i*	*s*	*ϕ*	*i*	*s*	*ϕ*	*t*	*h*	*e*	*t*
02	*i*	*s*	*ϕ*	*i*	*s*	*ϕ*	*t*	*h*	*e*	*t*	*h*
03	*s*	*ϕ*	*i*	*s*	*ϕ*	*t*	*h*	*e*	*t*	*h*	*i*
04	*ϕ*	*i*	*s*	*ϕ*	*t*	*h*	*e*	*t*	*h*	*i*	*s*
05	*i*	*s*	*ϕ*	*t*	*h*	*e*	*t*	*h*	*i*	*s*	*ϕ*
06	*s*	*ϕ*	*t*	*h*	*e*	*t*	*h*	*i*	*s*	*ϕ*	*i*
07	*ϕ*	*t*	*h*	*e*	*t*	*h*	*i*	*s*	*ϕ*	*i*	*s*
08	*t*	*h*	*e*	*t*	*h*	*i*	*s*	*ϕ*	*i*	*s*	*ϕ*
09	*h*	*e*	*t*	*h*	*i*	*s*	*ϕ*	*i*	*s*	*ϕ*	*t*
10	*e*	*t*	*h*	*i*	*s*	*ϕ*	*i*	*s*	*ϕ*	*t*	*h*

ادامه‌ی مثال

داده‌ها به صورت الفبایی مرتب می‌شوند.



خروجی:

L : sshth ii e 10

- این تبدیل «برگشت‌پذیر» است؛ خروجی برای یافتن ورودی کفایت می‌کند.

مثال- تبدیل معکوس

چگونه می‌توان از خروجی به ورودی رسید؟

- تمام نمادها در رشته‌ی خروجی وجود دارد.

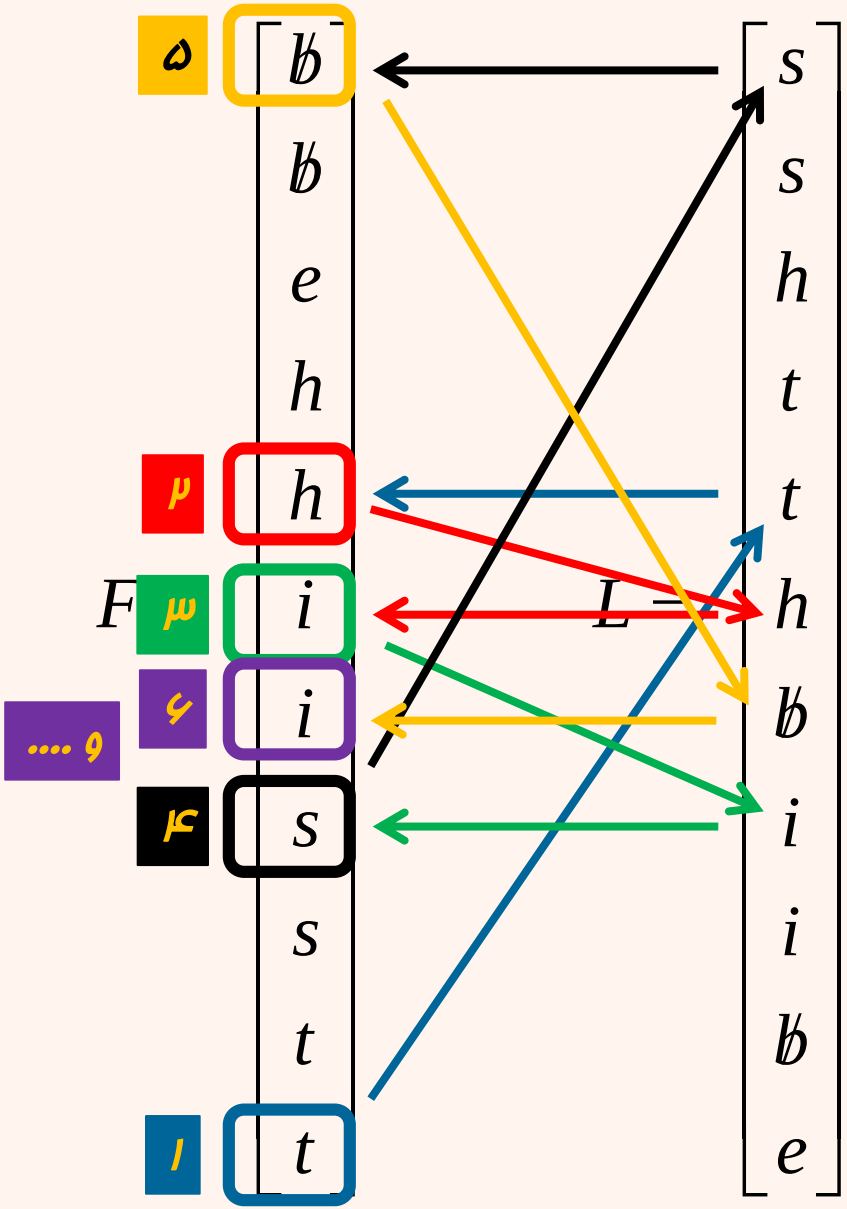
$$L: \ sshth\text{␣}iii\text{␣}be$$

- می‌توان با توجه به رشته‌ی L، رشته‌ی F را به دست آورد، با مرتب کردن الفبایی رشته .

$$F: \ \text{␣␣}behhiisstt$$

0	␣	i	s	␣	t	h	e	t	h	i	s
1	␣	t	h	e	t	h	i	s	␣	i	s
2	e	t	h	i	s	␣	i	s	␣	t	h
3	h	e	t	h	i	s	␣	i	s	␣	t
4	h	i	s	␣	i	s	␣	t	h	e	t
5	i	s	␣	i	s	␣	t	h	e	t	h
6	i	s	␣	t	h	e	t	h	i	s	␣
7	s	␣	i	s	␣	t	h	e	t	h	i
8	s	␣	t	h	e	t	h	i	s	␣	i
9	t	h	e	t	h	i	s	␣	i	s	␣
10	t	h	i	s	␣	i	s	␣	t	h	e

ادامه‌ی مثال



0	␣	i	s	␣	t	h	e	t	h	i	s
1	␣	t	h	e	t	h	i	s	␣	i	s
2	e	t	h	i	s	␣	i	s	␣	t	h
3	h	e	t	h	i	s	␣	i	s	␣	t
4	h	i	s	␣	i	s	␣	t	h	e	t
5	i	s	␣	i	s	␣	t	h	e	t	h
6	i	s	␣	t	h	e	t	h	i	s	␣
7	s	␣	i	s	␣	t	h	e	t	h	i
8	s	␣	t	h	e	t	h	i	s	␣	i
9	t	h	e	t	h	i	s	␣	i	s	␣
10	t	h	i	s	␣	i	s	␣	t	h	e

t h i s b i ...

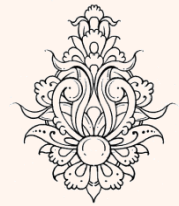
Compression booster

تبدیل BWT و فشرده‌سازی

- تاکنون عملا هیچ‌گونه فشرده‌سازی انجام نشده است.
- در حقیقت با این تبدیل، دنباله‌ای به دست می‌آید که به نسبت بیشتری قابل فشرده‌سازی خواهد بود.
  - در زبان انگلیسی واژه‌ی «the» زیاد استفاده می‌شود، با این روش بعد از عملیات چرخش و مرتب‌سازی، چندین رشته خواهیم داشت که با «he» شروع می‌شود و به «t» ختم می‌شود.
  - در L تعداد زیادی «t» در کنار هم قرار خواهد گرفت.
  - در این حالت رشته‌ای خواهیم داشت که «به صورت محلی همگن» است.

locally homogeneous

  - برای پاسخ مناسب به چندده کیلوبایت داده نیاز دارد.



Move-to-Front Coding(MTF)

book stack

- می‌توان آن را پیش‌پردازش الگوریتم‌های کدگذاری دانست.
- در این شیوه کاراکترهای متوالی شبیه به هم با کاراکتر «0» کد می‌شوند.
- عملکرد به این ترتیب است که الفبا به صورت یک لیست در نظر گرفته می‌شود.
- عنصر بالا با شاخص صفر نشان داده می‌شود و سایر عناصر به ترتیب با اعداد طبیعی
- آخرین عنصری که دیده شده است به بالای لیست منتقل می‌شود.
- بدین ترتیب قسمت‌های همگن با یک‌سری «0» پشت سر هم مدل می‌شود.

مثال ۱

$$A = \{a, b, c, d\}$$

الفبا

پشته‌ی نمادهای دیده‌شده

b
c
a
d

رشته‌ی ورودی

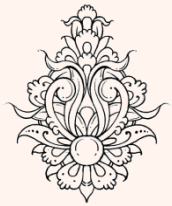
S =

b	a	a	c	b

MTF(S) =

1	1	0	2	2

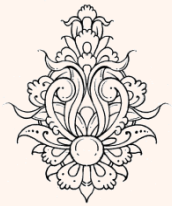
کد خروجی



Move to Front

مثال ۲

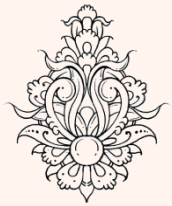
abracadabra		**a**,b,r,c,d



Move to Front

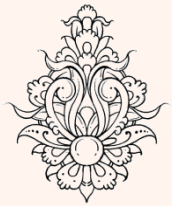
مثال ۲(ادامه...)

abracadabra		**a**,b,r,c,d
a**b**racadabra	0	a,**b**,r,c,d



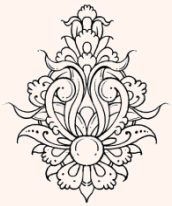
مثال ۲(ادامه...)

abracadabra		**a**,b,r,c,d
a**b**racadabra	0	a,**b**,r,c,d
ab**r**acadabra	0,1	b,a,**r**,c,d



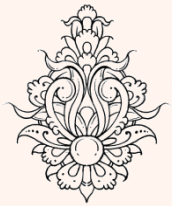
مثال ۲(ادامه...)

abracadabra		**a**,b,r,c,d
a**b**racadabra	0	a,**b**,r,c,d
ab**r**acadabra	0,1	b,a,**r**,c,d
abr**a**cadabra	0,1,2	r,b,**a**,c,d



مثال ۲(ادامه...)

abracadabra		**a**,b,r,c,d
a**b**racadabra	0	a,**b**,r,c,d
ab**r**acadabra	0,1	b,a,**r**,c,d
abr**a**cadabra	0,1,2	r,b,**a**,c,d
abra**c**adabra	0,1,2,2	a,r,b,**c**,d

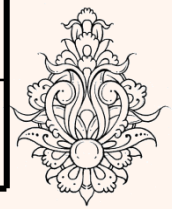


مثال ۲(ادامه...)

abracadabra		**a**,b,r,c,d
a**b**racadabra	0	a,**b**,r,c,d
ab**r**acadabra	0,1	b,a,**r**,c,d
abr**a**cadabra	0,1,2	r,b,**a**,c,d
abra**c**adabra	0,1,2,2	a,r,b,**c**,d
abrac**a**dabra	0,1,2,2,3	c,**a**,r,b,d
abraca**d**abra	0,1,2,2,3,1	a,c,r,b,**d**

مثال ۲(ادامه...)

abracadabra		**a**,b,r,c,d
a**b**racadabra	0	a,**b**,r,c,d
ab**r**acadabra	0,1	b,a,**r**,c,d
abr**a**cadabra	0,1,2	r,b,**a**,c,d
abra**c**adabra	0,1,2,2	a,r,b,**c**,d
abrac**a**dabra	0,1,2,2,3	c,**a**,r,b,d
abracad**d**abra	0,1,2,2,3,1	a,c,r,b,**d**
abracadabra	0,1,2,2,3,1,4,1,4,4,2	



بعد از این مراحل دنباله دارای تعداد زیادی صفر خواهد بود و بدین‌ترتیب آنتروپی کاهش می‌یابد.

Run Length Encoding(RLE)

- در این شیوه از تعداد تکرار نمادهای متوالی برای کدگذاری استفاده می‌شود:

abbbaacccca => (a,1),(b,3),(a,2),(c,4),(a,1)

– در فرمت فشرده‌سازی تصویر jpeg از این شیوه بهره گرفته شده است.

- بدین‌ترتیب شمای کلی سیستم فشرده‌سازی به صورت زیر خواهد بود:

Associative Coder of Buyanovsky (ACB)

- در این شیوه، یک واژه‌نامه از محتوای رشته‌ی دیده شده تهیه می‌شود.
  - از یک پنجره‌ی لغزنده شبیه LZ77 استفاده می‌کند.
  - بخش چپ پنجره **Context** و بخش راست آن **Content** نامیده می‌شود.
- واژه‌نامه نیز دارای دو بخش مشابه است:
  - محتوای زمینه (دیده شده)
  - رشته‌ی تالی محتوای زمینه

مقایسه از راست به چپ

- جستجو بهترین تطبیق (بر اساس ترتیب الفبایی)
- سپس content مدخل‌های همسایه نیز مورد بررسی قرار می‌گیرند (طولانی‌ترین).

مقایسه از چپ به راست

Associative Coder of Buyanovsky (ACB)

- یک سه‌تایی به عنوان خروجی ارسال می‌شود:

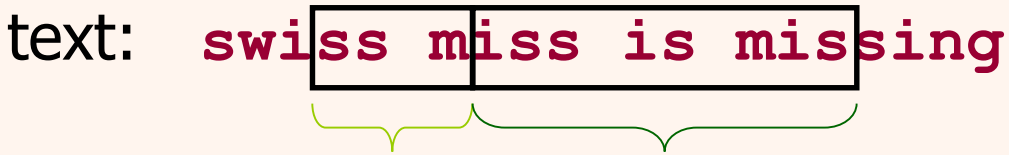
  (***d***, ***count***, ***ch***)

  - «d»: فاصله‌ی بین بهترین context و بهترین content
  - «count»: تعداد نمادهای منطبق
  - «ch»: اولین نماد منطبق نشده در پنجره‌ی look-ahead
- واژه‌نامه به روز می‌شود.
- context یافت شده یکی از مدخل‌های مرتبط است و ممکن است مدخل‌های همسایه‌ی آن مربوط به contentهای مرتبط‌تر باشد.
  - همیشه طولانی‌ترین content منجر به بهترین فشرده‌سازی نمی‌شود، در برخی موارد انتخاب عبارات کوتاه نزدیک به context ممکن است منجر به کدگذاری بهتری شوند.

در مورد جزییات پیاده‌سازی ACB اطلاعات چندانی در دست نیست!

محتوا و رشته‌ی تالی

پنجره‌ی لغزان مانند LZ77



با هر بار شیفت یک context-content جدید به دست می‌آید:

```
ss m|iss is mis
```

Salomon, D. (2012). Data Compression: The Complete Reference, Springer Berlin Heidelberg.

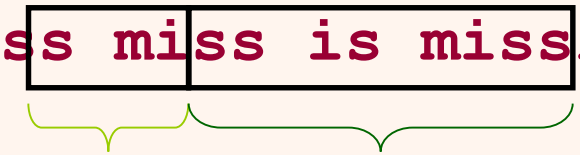
připravil Tomáš Skopal

محتوا و رشته‌ی تالی

پنجره‌ی لغزان مانند LZ77

۲

text: `swiss miss is missing`



context content

با هر بار شیفت یک context-content جدید به دست می‌آید:

```
ss m|iss is mis
s mi|ss is miss
```

محتوا و رشته‌ی تالی

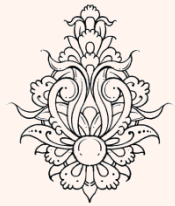
۳ پنجره‌ی لغزان مانند LZ77

text: swiss miss is missing

context content

با هر بار شیفت یک context-content جدید به دست می‌آید:

```
ss m|iss is mis
s mi|ss is miss
 mis|s is missi
```



واژه‌نامه

بدین‌ترتیب واژه‌نامه به دست می‌آید(برای هفت کاراکتر).

```
...s|wiss m
...sw|iss m
...swi|ss m
...swis|s m
...swiss|m
...swiss |m
```

context content

```
...swiss |m
...swi|ss m
...s|wiss m
...swis|s m
...swiss|m
...sw|iss m
```

context content

بخش context از راست به چپ مرتب شده‌است.
بعد از پردازش هر کاراکتر یک مدخل به واژه‌نامه افزوده می‌شود.
برای این که در کدگشا، واژه‌نامه به همین شیوه قابل ساخت باشد،
تنها رشته‌ی موجود در search buffer در نظر گرفته می‌شود.

مثال-کدگذاری

تا اینجا پردازش شده است

```
swiss m|iss is missing
```

current context

وضعیت جاری واژه‌نامه:

```
1   ...swiss |m
2      ...swi|ss m
3        ...s|wiss m
4      ...swis|s m
5    ...swiss| m
6       ...sw|iss m
```

4

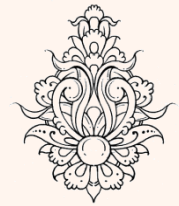
۱ نزدیک‌ترین context در واژه‌نامه جستجو شود:

بین مدخل دوم و سوم

۲ فاصله‌ی نزدیک‌ترین content به context یافته شده در واژه‌نامه به دست آید.

مدخل ششم، فاصله: چهار

۶-۲



مثال-کدگذاری(ادامه...)

swiss m|iss is missing

current context

۳ یافتن طول content یکسان

چهار

۴ یافتن اولین نماد ناهمسان با content

۵ تولید سه تایی مرتب

وضعیت جاری واژه‌نامه:

1 ...swiss |m

2 ...swi|ss m

3 ...s|wiss m

4 ...swis|s m

5 ...swiss| m

6 ...sw|iss m

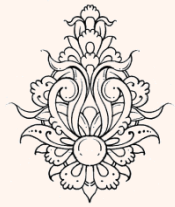
4

'i'

(6-2, 4, 'i')

معادل رشته‌ی پنج‌تایی زیر

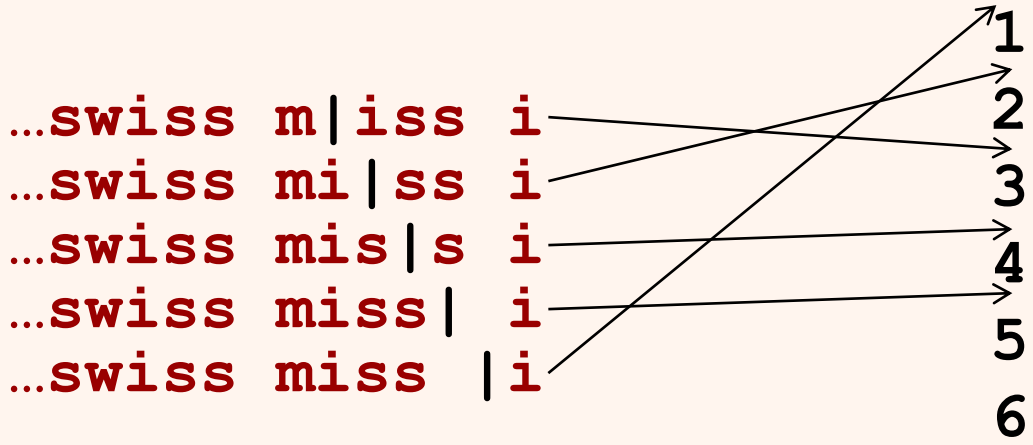
iss i



مثال-کدگذاری(ادامه...)

۶ واژه‌نامه به‌روز شود:

context: swiss m
content: iss i



واژه‌نامه به‌روزرسانی‌شده

```
1   …swiss miss |i
2        …swiss |miss i
3      …swiss mi|ss i
4           …swi|ss miss i
5       …swiss m|iss i
6             …s|wiss miss i
7     …swiss mis|s i
8         …swis|s miss i
9    …swiss miss| i
10        …swiss| miss i
11          …sw|iss miss i
```

پایان یک مرحله✦

واژه‌نامه به‌روزرسانی‌شده

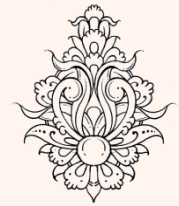
```
1   …swiss miss |i
2        …swiss |miss i
3      …swiss mi|ss i
4           …swi|ss miss i
5       …swiss m|iss i
6             …s|wiss miss i
7     …swiss mis|s i
8         …swis|s miss i
9    …swiss miss| i  6
10        …swiss| miss i
11          …sw|iss miss i
```

تا اینجا پردازش شده است

```
swiss miss i|s missing
```

current context

(8-2, 6, 'i')



مثال-کدگذاری(ادامه...)

- همین مراحل ادامه می‌یابد:

...s|wiss␣miss␣is␣missi	1	...swiss␣miss␣is␣|missi
...sw|iss␣miss␣is␣missi	2	...swiss␣miss␣|is␣missi
...swi|ss␣miss␣is␣missi	3	...swiss␣|miss␣is␣missi
...swis|s␣miss␣is␣missi	4	...swiss␣miss␣i|s␣missi
...swiss|␣miss␣is␣missi	5	...swiss␣miss␣is␣mi|ssi
...swiss␣|miss␣is␣missi	6	...swiss␣mi|ss␣is␣missi
...swiss␣m|iss␣is␣missi	7	...swi|ss␣miss␣is␣missi
...swiss␣mi|ss␣is␣missi	8	...swiss␣miss␣is␣m|issi
...swiss␣mis|s␣is␣missi	9	...swiss␣m|iss␣is␣missi
...swiss␣miss|␣is␣missi	10	...s|wiss␣miss␣is␣missi
...swiss␣miss␣|is␣missi	11	...swiss␣miss␣is|␣missi
...swiss␣miss␣i|s␣missi	12	...swiss␣miss␣is␣mis|si
...swiss␣miss␣is|␣missi	13	...swiss␣mis|s␣is␣missi
...swiss␣miss␣is␣|missi	14	...swis|s␣miss␣is␣missi
...swiss␣miss␣is␣m|issi	15	..swiss␣miss␣is␣miss|i
...swiss␣miss␣is␣mi|ssi	16	...swiss␣miss|␣is␣missi
...swiss␣miss␣is␣mis|si	17	...swiss|␣miss␣is␣missi
..swiss␣miss␣is␣miss|i	18	...sw|iss␣miss␣is␣missi

swiss miss is missi|ng

current context

(0, 0, 'n')

و به همین ترتیب

مثال-کدگشایی

۱ نزدیک‌ترین context در واژه‌نامه جستجو شود.

۲ بر اساس کد، content به دست آید.

۳ رشته‌ی مشابه بر اساس کد مشخص شود.

۴ کاراکتر سوم به رشته‌ی کدگشایی شده اضافه شود.

۵ واژه‌نامه به روز شود.

context: swiss m

content: unkown

(4,4,´i´)

d=4 →“iss ”

i=´i´ →“ iss i ”

وضعیت کنونی واژه‌نامه:

1	…swiss	|m
2	…swi	|ss m
3	…s	|wiss m
4	…swis	|s m
5	…swiss	| m
6	…sw	|iss m