Bezztrátová komprese dat

INF 3D/3E

29. března 2025

Motivace

Problém

Chci poslat básničku o délce **X** znaků,
ale můj message systém umožňuje maximálně **Y** znaků.

Básnička:

Komprese je síla, co chaos zkrátí,
komprese je řád, co data vrátí,
komprese je tanec bitů v toku,
komprese je zázrak v každém kroku.

Archiv je místem, kde se vše zmenší,
archiv je místem, kde je svět tenčí,
archiv je tajemství skryté v datech,
archiv je obsažen v našich tématech.

Obsah prezentace

Co je komprese?

Definice

Komprese je proces zmenšení objemu dat pomocí odstranění redundantních informací.

Dělení kompresí

- **Bezztrátová komprese** – data lze přesně rekonstruovat.
- **Ztrátová komprese** – při rekonstrukci dochází ke ztrátě některých informací.

Základní dělení bezztrátové komprese

Hlavní oblasti

- **Principy:** Entropie, redundance, Kraftova nerovnost.
- **Metody kódování:** Statistické metody (např. Huffmanovo a aritmetické kódování) a slovníkové metody (např. LZW, LZ77, LZ78, RLE).
- **Implementační algoritmy:** Např. DEFLATE.

Huffmanovo kódování – Teorie

Definice

Huffmanův kód je algoritmus pro tvorbu prefixových kódů, který minimalizuje průměrnou délku kódu na základě pravděpodobností výskytu symbolů.

Klíčové informace

- Využívá stromovou strukturu, kde listy reprezentují symboly.
- Častějším symbolům přiřazuje kratší kódy.
- Zaručuje optimální průměrnou délku kódu.

Huffmanovo kódování – Detailní princip

Postup algoritmu

1. Spočítáme frekvenci výskytu jednotlivých symbolů.
2. Seřadíme symboly vzestupně podle frekvence.
3. Postupně spojíme dva nejmenší uzly do nového uzlu, jehož frekvence je součtem.
4. Tento proces opakujeme, dokud nevznikne jeden strom.
5. Při přiřazování bitů se větvička přiřadí 0 a druhá 1 (či opačně).

Huffmanovo kódování – Ukázka výpočtu (zjednodušený příklad)

Příklad se čtyřmi symboly

Symboly a pravděpodobnosti:
A: 0.4, B: 0.3, C: 0.2, D: 0.1.

Huffmanovo kódování – Ukázka výpočtu (zjednodušený příklad)

Příklad se čtyřmi symboly

Symboly a pravděpodobnosti:
A: 0.4, B: 0.3, C: 0.2, D: 0.1.

Postup

1. Seřadíme: D (0.1), C (0.2), B (0.3), A (0.4).
2. Spojíme D a C $\rightarrow$ uzel (0.3).
3. Seřadíme: uzel (0.3), B (0.3), A (0.4); spojíme uzel a B $\rightarrow$ nový uzel (0.6).
4. Nakonec spojíme tento uzel (0.6) s A (0.4) $\rightarrow$ kořen stromu (1.0).

Huffmanovo kódování – Aplikace na básničku: Frekvenční analýza

Krok 1: Frekvenční analýza

Pro zjednodušenou demonstraci zanedbáme interpunkci a velká/malá písmena.

Ukázka:
Analyzujeme text:
`Komprese je síla, co chaos zkrátí, ...`
(Předpokládejme, že získáme frekvence např.:
K: 2, o: 7, m: 4, p: 2, r: 4, e: 9, s: 6, ...)

Huffmanovo kódování – Aplikace na básničku: Sestavení stromu

Krok 2: Sestavení Huffmanova stromu

- Na základě frekvencí se seřadí symboly.
- Postupným spojením nejmenších frekvencí se vytvoří strom.
- Každému listu je přiřazen binární kód (např. symbol s nejvyšší četností získá nejkratší kód).

Poznámka: Pro úplnou demonstraci by se zobrazil celý strom, zde ukazujeme pouze princip.

Huffmanovo kódování – Aplikace na básničku: Výsledný kód

Krok 3: Výsledný kód

- Po přiřazení kódů ke všem znakům se text nahradí odpovídajícími binárními řetězci.
- Výsledný bitový řetězec je výrazně kratší než původní text.

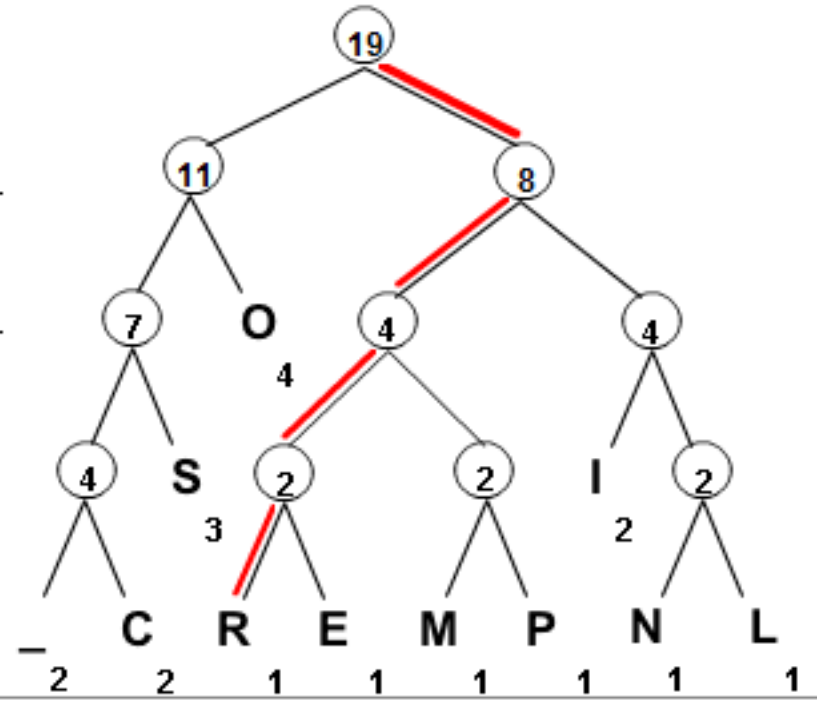
Huffmanovo kódování -Praktický příklad

Huffman Coding

Example: "COMPRESSION_IS_COOL"

To compute the bit pattern for each datum, we go up the tree and note down a "1" (true) if we take the branch to the *left* and a "0" (false) if we take the branch to the *right*, respectively.

-- Step 3 --
compute bit patterns



_ : 0000
C : 0001
S : 001
O : 01
R : 1000

Aritmetické kódování – Teorie

Definice

Aritmetické kódování převádí celou zprávu na jedno reálné číslo v intervalu [0,1] pomocí postupného dělení intervalu dle pravděpodobností symbolů.

Klíčové informace

- Využívá interval, který se při čtení každého symbolu zužuje.
- Výsledný interval reprezentuje zakódovanou zprávu.
- Je efektivní při práci s velmi různorodými pravděpodobnostmi.

Aritmetické kódování – Detailní princip

Postup algoritmu

1. Začneme s intervalem [0,1].
2. Rozdělíme interval podle pravděpodobností symbolů (např. A: [0,0.5], B: [0.5,0.8], C: [0.8,1]).
3. Každým dalším symbolem se zvolený podinterval zužuje.
4. Po zpracování celé zprávy získáme konečný interval, který reprezentuje kód.

Aritmetické kódování – Ukázka výpočtu (zjednodušený příklad)

Příklad se třemi symboly

Symboly a pravděpodobnosti:
A: 0.5, B: 0.3, C: 0.2.

Aritmetické kódování – Ukázka výpočtu (zjednodušený příklad)

Příklad se třemi symboly

Symboly a pravděpodobnosti:
A: 0.5, B: 0.3, C: 0.2.

Postup

1. Rozdělení intervalu [0,1]: A: [0,0.5], B: [0.5,0.8], C: [0.8,1].
2. Zpracování řady symbolů vede k postupnému zužování intervalu.

Aritmetické kódování – Aplikace na básničku: Rozdělení intervalu

Krok 1: Přiřazení pravděpodobností

Pro demonstraci přiřaďme zjednodušené pravděpodobnosti vybraným znakům z básničky.

Příklad:
Předpokládejme, že `K` má 0.1, o 0.2, `m` 0.15, ...
(Celkově rozdělení tak, aby součet byl 1.)

Aritmetické kódování – Aplikace na básničku: Zužování intervalu

Krok 2: Postupné zužování

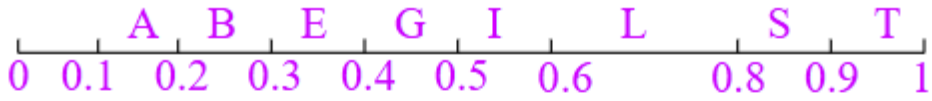
1. Začneme s intervalem [0,1].
2. První symbol (např. `K`) určí podinterval dle své pravděpodobnosti.
3. Následující symboly dále zužují aktuální interval.

Po zpracování celé básničky získáme konečný interval, např. [0.345, 0.346].

Aritmetické kódování - Praktický příklad

znak	pst	podinterval
mezera	1/10	$\langle 0 , 0.1)$
A	1/10	$\langle 0.1 , 0.2)$
B	1/10	$\langle 0.2 , 0.3)$
E	1/10	$\langle 0.3 , 0.4)$
G	1/10	$\langle 0.4 , 0.5)$
I	1/10	$\langle 0.5 , 0.6)$
L	2/10	$\langle 0.6 , 0.8)$
S	1/10	$\langle 0.8 , 0.9)$
T	1/10	$\langle 0.9 , 1.0)$

Aritmetické kódování -Praktický příklad



znak	dolní mez	horní mez
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
mezera	0.2572	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

LZW kódování – Teorie

Definice

LZW kódování je slovníková metoda, kde se dynamicky vytváří slovník opakujících se vzorů a ty se nahrazují indexy slovníku.

Klíčové informace

- Inicializace se základním slovníkem obsahujícím jednotlivé znaky.
- Během čtení textu se do slovníku přidávají nové sekvence.
- Výsledkem je posloupnost indexů, které značně zkracují původní text.

LZW kódování – Detailní princip

Postup algoritmu

1. Inicializace základního slovníku (např. pro všechny znaky ASCII).
2. Procházení textu a hledání nejdelší sekvence, která je již ve slovníku.
3. Pokud nalezená sekvence existuje, zapíšeme její index a přidáme do slovníku novou sekvenci (původní sekvence + následující znak).
4. Postup opakujeme až do konce textu.

LZW kódování – Ukázka výpočtu (zjednodušený příklad)

Příklad:

Vstupní řetězec: `ABABABA`

LZW kódování – Ukázka výpočtu (zjednodušený příklad)

Příklad:

Vstupní řetězec: `ABABABA`

Postup

1. Inicializace slovníku: {A, B}.
2. Čtením řetězce se přidávají nové kombinace (např. `AB`, `BA`, `ABA`, . . .).
3. Výstup je posloupnost indexů slovníku.

Vstupní text

Komprese je síla, co chaos zkrátí,
komprese je řád, co data vrátí,
komprese je tanec bitů v toku,
komprese je zázrak v každém kroku.

Archiv je místem, kde se vše zmenší,
archiv je místem, kde je svět tenčí,
archiv je tajemství skryté v datech,
archiv je obsažen v našich tématech.

(Předpokládáme, že před zpracováním odstraníme interpunkci a převedeme vše na malá písmena. Zároveň zde pro usnadnění pracujeme s předpokladem, že 1 slovo = 1 znak.)

Inicializace slovníku

Počáteční slovník

Inicializujeme slovník se všemi unikátními slovy v básničce (v pořadí prvního výskytu):

1. **komprese**
2. **je**
3. **síla**
4. **co**
5. **chaos**
6. **zkrátí**
7. **řád**
8. **data** ...

Nové sekvence budou přiřazovány s kódy počínaje číslem 31.

První řádka – tokenizovaný vstup

Řádek 1 (po předzpracování)

```
komprese   je   síla   co   chaos   zkrátí
```

Krok 1: Zpracování prvního páru

Postup

- Nastavíme **w = "komprese"**.
- Další token je **"je"**.
- Spojíme: **"komprese je"** – tato sekvence není v počátečním slovníku.
- **Výstup:** Vypíšeme kód pro "komprese"(kód 1).
- **Aktualizace:** Přidáme novou sekvenci "komprese je"s kódem **31**.
- Nastavíme **w = "je"**.

Krok 2: Další token

Postup

- Nyní **w = "je"**.
- Další token je **"síla"**.
- Spojení: **"je síla"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "je"(kód 2).
- **Aktualizace:** Přidáme "je síla"s kódem **32**.
- Nastavíme **w = "síla"**.

Krok 3: Pokračování řádky 1

Postup

- **w** = **"síla"**, další token: **"co"**.
- Spojení: **"síla co"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "síla"(kód 3).
- **Aktualizace:** Přidáme "síla co"s kódem **33**.
- Nastavíme **w** = **"co"**.

Krok 4: Další token řádky 1

Postup

- **w = "co"**, další token: **"chaos"**.
- Spojení: **"co chaos"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "co"(kód 4).
- **Aktualizace:** Přidáme "co chaos"s kódem **34**.
- Nastavíme **w = "chaos"**.

Krok 5: Poslední token řádky 1

Postup

- **w = "chaos"**, další token: **"zkrátí"**.
- Spojení: **"chaos zkrátí"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "chaos"(kód 5).
- **Aktualizace:** Přidáme "chaos zkrátí"s kódem **35**.
- Nastavíme **w = "zkrátí"**.
- Řádek skončil, tudíž vypíšeme kód pro "zkrátí"(kód 6).

Druhá řádka – tokenizovaný vstup

Řádek 2 (po předzpracování)

```
komprese   je   řád   co   data   vrátí
```

Krok 6: Začátek druhé řádky

Postup

- Nastavíme **w** = **"komprese"**.
- Další token je **"je"**.
- Spojení: **"komprese je"** již je v slovníku (kód 31).
- Aktualizujeme **w**: nyní **w** = **"komprese je"**.

Krok 7: Pokračování druhé řádky

Postup

- **w** = **"komprese je"**, další token: **"řád"**.
- Spojení: **"komprese je řád"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "komprese je"(kód 31).
- **Aktualizace:** Přidáme "komprese je řád"s kódem **36**.
- Nastavíme **w** = **"řád"** (původní "řád"má kód 7).

Krok 8: Pokračování druhé řádky

Postup

- **w = "řád"**, další token: **"co"**.
- Spojení: **"řád co"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "řád"(kód 7).
- **Aktualizace:** Přidáme "řád co"s kódem **37**.
- Nastavíme **w = "co"**.

Krok 9: Dokončení druhé řádky

Postup

- **w** = **"co"**, další token: **"data"**.
- Spojení: **"co data"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "co"(kód 4).
- **Aktualizace:** Přidáme "co data"s kódem **38**.
- Nastavíme **w** = **"data"** (kód 8).
- Další token: **"vrátí"**.
- Spojení: **"data vrátí"** není v slovníku.
- **Výstup:** Vypíšeme kód pro "data"(kód 8).
- **Aktualizace:** Přidáme "data vrátí"s kódem **39**.
- Řádek skončil, tudíž vypíšeme kód pro "vrátí"(kód 9).

Shrnutí dosavadního průběhu

Výstupní posloupnost kódů

- **Řádek 1:** 1, 2, 3, 4, 5, 6
- **Řádek 2:** 31, 7, 4, 8, 9

Celková posloupnost: `1, 2, 3, 4, 5, 6, 31, 7, 4, 8, 9`

Nové položky ve slovníku

- 31: "komprese je"
- 32: "je síla"
- 33: "síla co"
- 34: "co chaos"
- 35: "chaos zkrátí"
- 36: "komprese je řád"
- 37: "řád co"
- 38: "co data"
- 39: "data vrátí"

Poznámky

Další postup

Stejný algoritmus se aplikuje postupně na celý vstup (básničku). Díky opakujícím se sekvencím – například opakování slov **komprese** a **archiv** – dochází k nárůstu slovníku a ke zkrácení výsledné posloupnosti kódů oproti původnímu textu.

Shrnutí bezztrátové komprese

Hlavní poznatky

- Bezztrátová komprese využívá odstranění redundance pro přesnou rekonstrukci dat.
- Klíčové principy zahrnují práci s entropií, dodržení podmínek (např. Kraftova nerovnost) a efektivní kódování.
- Ukázané metody: Huffmanovo kódování, aritmetické kódování a LZW.

Děkuji za pozornost

Otázky?