VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Analýza a komprese signálů

Reduction Paradox

Vojtěch Prokop

Semestrální práce

Vedoucí práce: Ing. Michal Vašinek, Ph.D.

Ostrava, 2022

Abstrakt

No Czech or Slovak abstract is given

Klíčová slova

No Czech or Slovak keywords are given

Abstract

No English abstract is given

Keywords

No English keywords are given

Obsah

Seznam obrázků

Kapitola 1

Obecný popis

1.1 Komprese dat

Komprese dat je metodika, která má za účel zpracovat počítačová data, tak aby byla snížena jejich velikosti. Počítačová data lze popsat velikostí, kterou bývá jednotka bajt nebo bit. Důvodem, proč se tedy zabýváme zmenšením objemu dat je:

- Snížení prostorových nároků na archivaci.
- Práce se sítí, kde můžeme dosáhnout například nižší doby pro přenos.
- Propustnost sítě.

Komprese samotná je v dnešní době rozdělena do dvou tříd. Ztrátové komprese, která znemožňuje zpětnou rekonstrukci do originálních dat. Ačkoliv tento fakt tolerujeme, jelikož většinou bývá zmenšení mnohem větší než je tomu u druhé třídy a to bezeztrátové komprese. Tato třída má vlastnost, že komprimovaný soubor lze dekompresí rekonstruovat do původní podoby.

1.2 Popis problému

Existují algoritmy, které umožnují kompresi dat s pomocí definované gramatiky. Originální zpráva, pak bývá pomocí definovaných pravidel transformována do nové zprávy, která reprezentuje komprimované množství dat. Jeden z takových algoritmů je algoritmus Re-Pair (viz kapitola 1.3), který v každém iteračním kroku nahrazuje nejčetnější dvojici znaků za nový znak, a tímto vytváří pravidlo, dle kterého dochází k transformaci zprávy. V každém kroku je definováno pravidlo, které na levé straně obsahuje neterminální symbol a na pravé kombinaci terminálu a neterminálu. Příklad můžeme vidět na obrázku č. 1.1.

Dále víme, že lze vypočíst entropii nultého řádu, která nám říká kolik bitů je potřeba k zakódování jednoho symbolu ze souboru. Předpokládejme, že máme tedy dvě zprávy m0 a m1, kde

$$
\begin{aligned}
S &\rightarrow R_0 R_1 R_2 e \\
R_0 &\rightarrow ab \\
R_1 &\rightarrow R_0 c \\
R_2 &\rightarrow R_1 d
\end{aligned}
$$

Obrázek 1.1: Bezkontextová gramatika

m0 reprezentuje originální zprávu a m1 zprávu po prvním kroku Re-Pair algoritmu. Lze definovat problém, kterému říkáme paradox redukce, který si ukážeme na zprávě m0 a m1. Víme že $|m0| > |m1|$ neboli slovně počet znaků v originální zprávě je větší než počet znaků v transformované zprávě, ačkoliv entropie nultého řádu originální zprávy je menší než entropie nultého řádu transformované zprávy $Hm0 < Hm1$.

S tímto vědomím můžeme tedy vynaložit snahu provést transformaci zprávy n krát. Tento pokus reprezentuje nalezení extrému popisujícího kolikrát lze transformovat zprávu s dodržením snížení počtu symbolů, a však zvýšení velikosti entropie oproti předchozí zprávě.

1.3 Re-Pair

Jedná se o kompresní algoritmus založený na gramatice, který na základě vstupního textu vytvoří bezkontextovou gramatiku. V každém kroku je nahrazena nejčastější dvojice znaků vyskytující se v textu. Z provedených pokusů víme, že Re-Pair dosahuje vysokých kompresních poměrů. Nevýhodou bohužel je spotřeba paměti, která je přibližně 5krát větší než velikost vstupu.

1.4 Textové soubory

Byly využity textové soubory o velikost 50 MB z veřejně dostupné textové kolekce obsahující soubory se zdrojovými kódy, DNA, anglickými texty, XML texty a proteiny. Více informací lze vidět na obrázku č. 1.2.

Collection	Size (bytes)	Alphabet size	Inv match prob
SOURCES	210,866,607	230	24.77
PITCHES	55,832,855	133	39.75
PROTEINS	1,184,051,855	27	17.02
DNA	403,927,746	16	3.91
ENGLISH	2,210,395,553	239	15.25
XML	294,724,056	97	28.73

Obrázek 1.2: Textové soubory

V mém případě jsem pro experimenty využil zdrojové kódy, DNA, proteiny a anglické texty. Níže textové soubory budou více popsány.

1.4.1 Zdrojové kódy

Soubor obsahuje sjednocení zdrojových kódu z několika .c, .h, .C a .java souborů. Jedná se tedy o zdrojové soubory s programovacími jazyky Java, C možná C++.

1.4.2 DNA

Soubor obsahující sekvence DNA bez metadat. Můžeme zde očekávat vysokou četnout nukleotidů A, C, G, T. Mimo tyhle znaky se v souboru vyskytují i některé další speciální znaky, ale v mnohonásobně nižší četnosti než zmíněné znaky popisující první písmeno nukleotidů.

1.4.3 Proteiny

Soubor obsahující sekvence proteinů.

1.4.4 Anglické texty

Sjednocení textů v anglickém jazyce z vybraných souborů (knih), které jsou ke stažení v Gutenberg Projektu. Je výhodou, že texty neobsahují hlavičku, která se vyskytuje v některých souborech stažených přímo v zmíněném projektu. Hlavička obsahuje metadata k knížce.

Kapitola 2

Řešení

Celé řešení problému lze popsat následujícími několika dílčími kroky, které pak budou blíže rozebrány v kapitolách níže.

- Nalezení všech k gramů.
- Výpočet změny entropie po nahrazení každého k gramu.
- Výběr k gramu, která bude využit pro vytvoření pravidla.
- Vygenerování nového znaku.
- Vytvoření pravidla.
- Vytvoření nové zprávy.

2.1 Nalezení k-gramů

Textový soubor je vytvořen z posloupnosti znaků. V prvním kroku průběhu algoritmu je potřeba nalézt všechny k-tice, které budeme moci v sekvenci nahradit. Z těchto nalezených dvojic, pak bude vytvořeno pravidlo, pokud splní dvě podmínky. První podmínkou je zvyšující se entropie po nahrazení každého výskytu k-gramu. Druhou podmínkou je výběr daného k gramu námi definovanou metodou. Tento výběr je popsán v kapitole níže (viz 2.4).

Metoda, která nalezne všechny tyto k gramy může vypadat v programovacím jazyce Python následovně:

```
def find_k_grams_freq(data, max_size_k=2):
    kgrams_dic = {}
    for k in range(2, max_size_k+1):
        for i in range(len(data) - k):
            n_gram = data[i:i+k]
            kgrams_dic[n_gram] = kgrams_dic.get(n_gram, 0) + 1
    return kgrams_dic
```

Vstupními parametry jsou textová data a velikost k-gramu, které chceme v textových datech nalézt. Postupně pak iterujeme přes textový soubor a ukládáme do datové struktury slovníku kolikrát jsme daný k gram viděli.

2.2 Výpočet entropie

Z předchozí kroku víme, které k gramy se nám v souboru vyskytují. Této informace využijeme a pro každý k gram vypočteme novou entropii. Entropie bude vycházet z akce vytvoření nového pravidla a nahrazení k-gramu novým symbolem, což zapříčiní rozšíření aktuální abecedy o jeden symbol.

Výpočet entropie lze provést dvěma způsoby a to:

- Vytvořením nové zprávy a výpočtu pomocí shannonského vzorce.
- Využití definovaného vzorce, který vypočte posun vzhledem k entropii.

2.2.1 Shannon vzorec

Tento způsob pracuje s přístupem hrubé síly, kdy je vypočtena entropie pro dvě zprávy. A to zprávu, která byla v předchozím kroku a zprávu, která byla vytvořena po aplikace nově vytvořeného pravidla.

```
def calc_H(p):
    H = 0
    for k, v in p.items():
        #Shannon equation!
        H += p[k] * math.log2(p[k])
    return -H

def calc_entropy_for_message(message):
    counter = calc_freq(message)
    n = get_n(counter)
    p = calc_p(counter, n)
    H = calc_H(p)
    return H

def diff_entropy(message1, message2):
    message1_entropy = calc_entropy_for_message(message1)
    message2_entropy = calc_entropy_for_message(message2)
    message_1_entropy_size = message1_entropy * len(message1)
    message_2_entropy_size = message2_entropy * len(message2)
    diff = message_1_entropy_size - message_2_entropy_size
    return diff
```

2.3 Vytvoření nové zprávy

Abychom mohli vypočíst rozdíl entropie je potřeba transformovat zprávu do podoby, která reprezentuje zprávu v následujícím kroku. Takováto akce lze provést v programovacím jazyce Python velice jednoduchým způsobem.

```
def transform_message(message, ngram_for_replace):
    current_alphabet_size = np.unique(list(message))
    replace_character = find_not_existing_character(current_alphabet_size)
    return message.replace(ngram_for_replace, replace_character),
        replace_character
```

2.4 Výběr k-gramu

Po provedení výpočtu změny entropie pro každý nalezený k gram dostaneme několik možností, jak provést samotnou transformaci. Aktuální stav si můžeme představit pomocí obrázku č. 2.1. Diff pak znázorňuje výpočet $|m0| * Hm0 - |m1| * Hm1$.

	Counter	Diff
AA	862	-711.287863
AC	612	-700.944705
CG	397	-526.484088
GT	585	-642.013983
TG	609	-624.132715
TT	797	-725.647370
TA	856	-863.100574
GC	457	-493.377886
CT	648	-656.895096
TC	616	-682.016142
CC	476	-450.401292
CA	640	-680.422527
AG	604	-644.015856
GG	424	-397.457487
GA	567	-669.955161
AT	848	-870.225504

Obrázek 2.1: Stav před výběrem k gramu

Z těchto možností je potřeba vybrat jednu transformaci, které lze provést pomocí dvou možností s kterými bylo experimentováno:

- Výběr největší změny.
- Výběr náhodné změny.

2.5 Vytvoření pravidla

Jak bylo zmíněno transformace zprávy vychází z nahrazení vybraného k-gramu za nový neexistující znak (viz. kapitola č. 2.6). Tohle pravidlo je následně uloženo do datové struktury slovníku, tak aby bylo možné zpětně rekonstruovat originální zprávu.

2.6 Vygenerování nového znaku

Abychom mohli vytvořit nové pravidlo je potřeba použít nový neexistující znak vzhledem k aktuálnímu textovému řetězci. Tento znak lze vygenerovat pomocí dvou způsobů, které byly vyzkoušeny.

- Využití ASCII.
- Využití UNICODE.

Abychom neměli problém s limitovaným prostorem je potřeba využít UNICODE. Ten nabízí větší množství použitelných znaků.

```
all_chars_uni = tuple(chr(i) for i in range(32, 0x110000) if chr(i).isprintable())

all_chars_ascii = list(range(0, 256))
all_chars_ascii = [chr(ascii_char) for ascii_char in all_chars_ascii]
```

Kapitola 3

Shrnutí a výsledky

Z důvodů zohlednění běhu algoritmu byly experimenty provedeny s velikostí 10 000 znaků, které byly náhodně vybrány z každého datového zdroje. Pro tyto textové řetězce byl proveden běh algoritmu s limitovaným počtem kroků a to číslem 15. Vizualizace a krátké slovní zhodnocení lze nalézt v kapitole č. 3.1.

Zároveň bylo provedeno porovnání vzhledem k výběrové funkci (největší, náhodný), kde více se můžeme rozvědět v kapitole č. 3.2.

Pro každý zdroj byl proveden experiment a s tím změřeno jak dlouho výpočet běžel. Zároveň bylo provedeno pár optimalizací, které zrychlovali samotný výpočet. Více o tomto se lze dozvědět v kapitole č. 3.3.

Jak dopadlo nalezení samotných extrémů bude rozebráno v kapitole č. 3.4. A finální zhodnocení v kapitole č. 3.5.

3.1 Výsledky souborů

3.1.1 DNA

3.1.2 Proteins

3.1.3 Sources

3.1.4 English

3.2 Výběr k gramu

Porovnání největší s náhodným

3.3 Časová náročnost

Porovnání souborů

Porovnání po optimalizaci

3.4 Nalezené extrémy

3.5 Finální zhodnocení