VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Kontextové mapy a jejich aplikace
Contextual Maps and its Application

2013

Michal Vašinek

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2013 .....

Na tomto místě bych rád poděkoval svému vedoucímu práce panu Ing. Janu Platošovi, PhD. za to, že byl ochoten mi tuto práci vést a pomáhat při její tvorbě. Rád bych poděkoval svým rodičům a přátelům za trpělivost a vyjádřenou podporu.

Abstrakt

V dnešní době existuje celá řada kompresních algoritmů, některé se více zaměřují na efektivitu z hlediska času stráveného kompresí, jiné upřednostňují schopnost dosáhnout, co nejlepšího kompresního poměru. Mezi algoritmy jsou značné rozdíly v principu, na jakém pracují. V této diplomové práci se zabývám možnostmi využití kontextové mapy při předzpracování souborů pro kompresi převodem na Huffmanův kód a pro nalézání podobností mezi daty.

Klíčová slova: kompresní algoritmus, kontextová mapa, podobnost, kontextové transformace, Huffmanovo kódování, entropie

Abstract

Nowadays many compression algorithms exist, some of them focus on effectivity from the point of time spend with compression, others prefer ability to reach the best compression ratio. Among algorithms there are big differences in the principle they work on. In my masters work I deal with possibilites of using context map at preprocessing of files for compression using translation to the Huffman code and for finding similarities between data.

Keywords: compression algorithm, context map, similarity, context transformation, Huffman coding, entropy

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

> "Jev, který nazýváme význam, vytváří celou posloupnost vztahů mezi fyzikální informací a její mentální interpretací. Jedním extrémem je kniha napsaná v jazyce, který je nám přirozený a odpovídá úrovni našeho chápání tématu. Taková kniha nejen, že obsahuje velké množství informace, ale také velké množství informace sděluje. Tato kniha je schopna sdělit spoustu informací, protože tyto informace pro nás mají význam. Důvodem, proč pro nás mají význam, je to, že jsme schopni zařadit tuto informaci do osobního kontextu. Takový kontext se skládá ze znalostních struktur v našem mozku, které mohou působit jako informační prostředí pro novou informaci."
>
> Tom Stonier - Informace a vnitřní struktura vesmíru [14].

Tato práce navazuje na bakalářskou práci [15], ve které jsem se zabýval kompresí dat pomocí kontextové mapy. V této práci rozšířím formalizaci kontextové mapy a budu dále definovat některé vlastnosti, které jsem ve svých postupech využíval.

V práci představím několik algoritmů a jejich modifikací, které využívám ve dvou oblastech. V prvním případě se budu zabývat transformací souborů, tak aby tyto soubory lépe vyhovovaly zavedeným kompresním algoritmům. Druhý případ vychází z pozorování, která jsem učinil v práci [15] a na základě kontextových informací se pokusím definovat podobnost mezi uzly kontextové mapy.

V úvodu práce jsou představeny současné trendy ve využívání kontextů, především se jedná o oblasti logické analýzy a shlukování dat. V další části práce potom formalizuji kontextové mapy a popisuji vlastnosti, na které jsem se při jejich analýze soustředil a využíval jich.

Třetí část práce slouží, jako teoretický základ pro transformace dat pomocí kontextových map pro účely jejich komprese. V této části jsou představeny základy Shannonovy teorie informací a některé kompresní algoritmy, které buď v práci využívám, nebo mají s navrženými algoritmy nějakou podobnost. Na tuto část bude navazovat popis transformací, popis algoritmů pro jejich nalezení a jsou zde zpracovány také vlastnosti těchto algoritmů.

Ve čtvrté části této práce představím koncept nalezení podobnosti dat pomocí kontextové mapy.

1.1 Kontext jako pojem

Slovo kontext je původen z Latinského jazyka [16] a je odvozeninou ze dvou slov "con"a "texere", což v překladu znamená být provázán dohromady. V literatuře se vyskytuje několik různých definic slova kontext. Anglická Wikipedie definuje pojem kontext následovně: "kontext je relevantní omezení v komunikačních situacích, která ovlivňují použitý jazyk, rozmanitost jazyka a obsahový význam řeči". V práci [15] jsem definoval pojem kontext následovně:

Definice 1.1 *Za kontext se považuje okolí zkoumaných dat, kontextem ve smyslu metody PPM je myšleno několik předcházejích znaků, ve smyslu metody JBIG jsou kontextem myšleny pixely v okolí jednoho konkrétního pixelu. Kontextem ve smyslu kontextového kompresního algoritmu je množina slov a asociací sestavená algoritmem na základě konkrétní tématické oblasti.*

Kaiyu Wan ve své práci [16] využil definice kontextu podle Oxfordského slovníku anglických výrazů: "kontext znamená okolnosti, které formují pozadí pro událost"a zdůrazňuje, že kontext je vyšší pojem, kterým dáváme slovním spojením v daném konkrétním případu význam. Z hlediska této práce budu za kontext považovat definici 1.1.

Wan dále definuje tři základní oblasti do kterých je kontext v počítačových vědách zasazen:

- Kontext v logice
- Kontext v jazycích
- Kontext v systémech

Wan do svého přehledu oblastí nezahrnul oblast, kterou se zabýval Claude Shannon, proto zde rozšířím oblasti zasažené kontextem o oblast:

- Kontext v teorii informací

1.2 Kontextualismu

V roce 1884 Německý matematik Gottlob Frege v práci Základy Aritmetiky, poprvé definoval pojem kontextualismu a stanovil tři základní principy filosofické analýzy, zde především druhý princip je důležitý z hlediska této práce:

- Vždy striktně oddělujme psychologické od logického, subjektivní od objektivního.
- Nikdy se neptejme na význam daného slova v izolaci, ale vždy pouze v kontextu propozice.
- Vždy mějme na zřeteli rozdíl mezi konceptem a objektem.

Podle Fregeho se každé slovo podílí svým dílem na formování významu celé věty. Později na práci Fregeho navázali další význační logikové 20 století, jako Bertrand Russel, nebo Ludwig Wittgenstein. Konkrétně Wittgenstein inspirován Fregeho principy, zavedl rozdělení jazyka do propozicí a propozičních proměnných, propozicí je zde myšlena věta a v jeho interpretaci je sice věta tvořena slovy, nicméně právě věta je tím, co udává význam slov v ní obsažených.

Kontext vstupuju také do moderní logické analýzy a například intensionální logické systémy(např. TIL) schovávají kontext do konceptu interpretace v dané intensi. Zjednodušeně řečeno interpretace je dána tzv. světočasem, do kterého je konkrétní propozice zasazena. V tomto duchu například hlavní město Německa bude mít různý denotát z hlediska interpretace před rokem 1989 a po roce 1989.

1.3 Algoritmy založené na kontextových informacích

V následujících podkapitolách jsou popsány některé novější algoritmy, ve kterých se využívá kontextových informací.

1.3.1 Sekupování slov na základě statistického kontextu

Christopher C. Huckle se zabýval v práci [5] shrnutím a vyhodnocením kontextových metod k určení významu a sekupování slov s podobným významem obsažených v testovaných korpusech. Hodnotí zde především dva přístupy, první, který ke své činnosti využívá shlukovacích algoritmů, druhý, který je založen především na aplikaci neuronových sítích. Zde si popíšeme první přístup, protože ten vychází čistě z kontextových informací. Popisované metody jsou založeny na práci [2] , kde se k určování významu slov využívá metoda založená na tzv. pohyblivém kontextovém oknu, tato technika je založena na tom, že každé analyzované slovo, které je obsaženo někde v textu, tzv. cílené slovo, je zasazeno mezi jiná slova, která předurčují jeho význam, metoda je velmi podobná definici, kterou zavedl Wittgenstein. Každé slovo w_i je popsáno vektorem, ve kterém každá jeho složka j popisuje pravděpodobnost, že se na pozici w_j v kontextovém okně, vyskytne jiné kontextové slovo. Slovo w_i je umístěno uprostřed kontextového okna a na základě experimentálně nastavované délky okna, se potom do příslušného vektoru udávají hodnoty pravděpodobnosti slov před a za daným cílovým slovem. Po přiřazení vektorů napříč všemi slovy testovaného korpusu, se poté aplikuje metoda hierarchického shlukování pro seskupení slov podle významu. Výsledkem tohoto procesu jsou potom skupiny slov s podobným významem, více viz. [5].

1.3.2 PARAFAC - kontextové shlukování

Metodu kontextového shlukování pomocí dekompozice tenzoru metodou Parafac [10] navrhli Andri Mirzal a Masashi Furukawa a slouží k určování podobnosti mezi vstupními texty. Metoda se zaměřuje na modifikaci matice sousednosti do struktury vyššího typu, kterou autoři označují termínem tří-cestný tenzor sousednosti. Jejich práce je zaměřena na sestavení popisu grafů, které nepopisují pouze návaznosti slov v textech, ale zároveň provazují jednotlivé texty mezi sebou v podobě tenzoru.

V jejich práci sestavují grafy, jejichž uzly reprezentují samotné texty a pomocí rozšíření matice sousednosti na tenzor sousednosti, hledají provázání mezi výskyty jednotlivých slov, tato slova pak reprezentují hrany mezi jednotlivými uzly. Proces sestavení tenzoru sousednosti je následující:

1. Sestavení charakteristické matice, kde řádky matice reprezentují jednotlivé texty a sloupce matice reprezentují slova, pak každý prvek matice určuje počet výskytů daného slova v daném textu.

2. Sestavení bipartitního grafu, kde první partita reprezentuje texty a druhá partita pak reprezentuje slova.

3. Z bipartitního grafu, poté konstruují síť s pojmenovanými hranami, kde uzly této sítě reprezentují texty, a hrany mezi jednotlivými uzly reprezentují slova, doplněná o součet jeho výskytů ve dvou textech, touto hranou spojených.

4. Z takto sestavené sítě potom konstruují tenzor sousednosti, který obsahuje celkovou informaci o vazbách mezi jednotlivými texty.

Na základě tenzoru podobnosti a poté aplikováním kosinovy vzdálenosti, shlukují texty do skupin sobě tématicky podobných textů.

2 Kontextová mapa

Tato část textu bude sloužit k přiblížení konceptu kontextové mapy, k zadefinování pojmů a použitého značení. Na kontextovou mapu lze nahlížet několika způsoby.

1. Kontextová mapa může být zadána, jako orientovaný multigraf, tedy graf, kde je povoleno více hran mezi dvěma uzly.

2. Kontextová mapa, jako orientovaný ohodnocený graf, kde ohodnocení hran je dáno počtem průchodů danou hranou.

3. Kontextová mapa může být reprezentována, jako matice cen.

Z hlediska grafové reprezentace je kontextová mapa jednoznačně dána zápisem $K = (V, H)$, kde V je množina vrcholů a H je množina hran.

V teorii Markovovských procesů se využívá matice přechodů, která obsahuje pravděpodobnosti, že systém přejde z jednoho stavu do druhého. Pro tuto práci je takový popis nedostačující. Transformace popsané v dalších kapitolách pro své správné fungování potřebují uchovávat informace o počtu přechodů, jak z jednoho stavu do druhého, tak také informaci o tom, ze kterých stavů lze vstoupit do jednoho určitého stavu a kolikrát tak bylo učiněno. Tohoto mechanismu se poté využívá pro stanovení míry vzájmené provázanosti uzlů kontextové mapy. Pokud bychom používali matici přechodů, pak bychom část této informace ztratili.

Algoritmus pro sestavení kontextové mapy je uveden v práci [15] a z tohoto algoritmu vycházíme také v této práci. Pro popis algoritmů a pro reprezentování složitosti těchto algoritmů, budeme používat především reprezentaci ve formě matice cen, v této práci ozn. jako kontextová matice:

Definice 2.1 *Kontextová matice C je maticí, která reprezentuje vazby mezi jednotlivými uzly kontextové mapy a každá tato vazba má přiřazenu hodnotu rovnu počtu průchodů touto vazbou skrze zpracovávaná data.*

V definici jsem použil termínu uzel, při grafové reprezentaci bychom použili termínu vrchol grafu, nicméně zde se budeme držet pojmu uzel, který bude reprezentovat nejenom samotný vrchol grafu, ale také sebou ponese své jednoznačné pojmenování, které je identické s hodnotou slova, definice 2.2, které reprezentuje.

Definice 2.2 *Za slovo w je v této práci považována posloupnost symbolů abecedy Σ, kde délka každé posloupnosti(řetězce) je $|w| \geq 1$.*

Pokud budeme hovořit o kontextové matici specifické pro jeden konkrétní soubor, bude tento soubor označen v dolním indexu u symbolu reprezentujícím kontextovou matici, např. C_{text1}.

Dimenze matice C je dána počtem různých slov obsažených v textu. Různé typy souborů jsou charakteristické různou velikostí dimenze kontextové matice. Pro část týkající se kontextových transformací budeme uvažovat délku slova $|w| = 1$. Intuitivně cítíme,

Soubor	Počet uzlů	Počet vazeb
bib	81	1531
book1	82	1862
book2	96	3099
geo	256	13908
news	98	4310
obj1	256	4913
obj2	256	12040
paper1	95	1556
paper2	91	1340
paper3	84	1234
paper4	80	875
paper5	91	1035
paper6	93	1446
pic	159	3009
progc	92	1746
progl	87	1199
progp	89	1454
trans	99	1989

Tabulka 1: Počty uzlů a různých vazeb mezi uzly kontextové mapy v souborech Calgary corpusu pro $|w| = 1$.

že soubory obsahující anglický text mají nižší dimenzi, než-li soubory obsahující binární data, obrázky, hudbu, nebo třeba také český text z důvodu jeho odlišného kódování. Přehled je uveden v tabulce 2, kde jsou uvedeny dimenze kontextové matice pro soubory obsažené v korpusu Calgary.

Sloupce a řádky kontextové matice mají různé významy, i přesto, že mohou reprezentovat stejné uzly. Řádky matice nám dávají informaci o orientovaných vazbách, které z daného uzlu vystupují, naopak sloupce kontextové matice nás informují o vazbách, které do daného uzlu vstupují.

Pro každý uzel tak můžeme definovat dva vektory popisující vstupní a výstupní vazby. Pro další použití zavedeme značení, které bude jednoznačně rozlišovat vstupní a výstupní vektory.

Definice 2.3 *Vektor c_α označuje řádkový vektor matice C, takový, že reprezentuje vazby vystupující z uzlu α.*

Obdobně potom budeme značit sloupcový vektor:

Definice 2.4 *Vektor c^αoznačuje sloupcový vektor matice C, takový, že reprezentuje vazby vstupující do uzlu α.*

Pro reprezentaci jedné konkrétní vazby, budeme používat značení:

Definice 2.5 *Prvek označený c_α^β značí hodnotu v matici C, takovou, že reprezentuje počet průchodů orientovanou vazbou vycházející z uzlu α a směřující do uzlu β.*

Indexy α a β reprezentují prvky z množiny použitých slov a mohou jimi být znaky, číselní reprezentanti znaků, a nebo slova.

Definice 2.6 *Množinu $C_{out,\alpha}$, resp. $C_{in,\alpha}$ nenulových prvků vektoru c_α resp. c^α budeme nazývat množinou výstupních vazeb, resp. množinou vstupních vazeb uzlu α.*

2.1 Kontextové provázání uzlů

Kontextové provázání uzlů je vlastnost, kterou sdílejí dva uzly kontextové mapy, této vlastnosti budeme dále využívat při popisu transformací kontextové mapy, které budou sloužit pro předzpracování zdrojových souborů, pro následnou kompresi pomocí převodu na Huffmanův kód. Transformace využívá provázanosti po sobě jdoucích stavů a snaží se částečně napodobit chování mezi provázánými[1]kvantovými bity více např. [11] , toto provázání se označuje jak tzv. "entanglement"a pochází z článku o EPR paradoxu [3], publikovaném Einsteinem, Podolskim a Rosenem v roce 1935, jako protiargument vůči Koďaňské interpretaci kvantové mechaniky a říká, že dva systémy pokud jsou vzájemně korelovány, mohou být následně přeneseny libovolně daleko a přesto, pokud změříme stav jednoho systému, poznáme tak zároveň stav druhého systému. Původní Einsteinova argumentace byla namířena proti faktu, že takový způsob získání informace by porušoval princip limitní rychlosti šíření informací, tento argument pro účel této práce není důležitý, pro tuto kapitolu je důležité, že se také v informatických systémech vyskytuje určitá forma provázání mezi stavy a tohoto provázání lze využít pro transformace, a určování podobnosti zpráv.

Dva uzly v kontextové mapě se mohou vůči sobě nacházet v různých stavech. Pro potřeby transformací jsem tyto stavy rozdělil do čtyř skupin:

- globálně provázané stavy,
- lokálně provázané stavy,
- částečně provázané stavy,
- neprovázané stavy.

Provázanost budeme definovat, jako relace nad množinou slov. Po zadefinování relací si uvedeme jejich vlastnosti spolu s důkazem těchto vlastností, pro provedení důkazu je třeba si uvědomit, že všechny prvky matice C jsou nezáporná celá čísla.

Definice 2.7 *Dva uzly α, β kontextové mapy C jsou vzájemně v relaci globálního provázání $R_g(\alpha, \beta)$, pokud platí, že $c^\alpha c^\beta = 0$, a zároveň platí, že existuje alespoň jeden prvek $c_\gamma^\alpha \neq 0$.*

[1]Z angl. Quantum Entanglement

Podmínka o existenci alespoň jednoho nenulového prvku je implicitně dána povahou sestavení kontextové mapy. Jediný uzel kontextové mapy, který potenciálně nemusí mít žádného následníka je posledním slovem zpracovávané zprávy, například ve slově "ahoj"by takovým prvkem bylo slovo(viz. definice 2.2) "j".

Věta 2.1 *Relace globálního provázání je i-reflexivní a symetrická relace.*

Důkaz. Nejdříve si dokážeme vlastnost i-reflexivity pomocí sporu. Předpokládejme, že relace $R_g(\alpha, \alpha)$ je reflexivní, pak musí platit, že pro všechna slova γ z množiny slov obsažených v kontextové mapě C, je $c_\gamma^\alpha c_\gamma^\alpha = 0$, zároveň, však musí platit, že existuje alespoň jeden prvek $c_\gamma^\alpha \neq 0$, tak aby byla splněna podmínka z definice 2.7, z čehož vyplývá, že alespoň pro jeden prvek γ platí $c_\gamma^\alpha c_\gamma^\alpha \neq 0$, což je ve sporu s původním tvrzením, že $c_\gamma^\alpha c_\gamma^\alpha = 0$.

Symetričnost relace globálního provázání vychází z faktu, že násobení je komutativní operací, pak pro všechny uzly γ platí, že $c_\gamma^\alpha c_\gamma^\beta = c_\gamma^\beta c_\gamma^\alpha = 0$, z čehož přímo vyplývá, že $\sum_\gamma c_\gamma^\alpha c_\gamma^\beta = \sum_\gamma c_\gamma^\beta c_\gamma^\alpha = 0$. ∎

Definice 2.8 *Dva uzly α, β kontextové mapy C jsou vzájemně lokálně provázané přes uzel γ, ozn. $R_\gamma(\alpha, \beta)$ pokud platí, že $c_\gamma^\alpha c_\gamma^\beta = 0$ a zároveň platí, že $c_\gamma^\alpha \neq c_\gamma^\beta$.*

Věta 2.2 *Relace lokálního provázání přes třetí uzel je i-reflexivní, symetrická relace.*

Zde vlastnost i-reflexivity je dána přímo v definici 2.8. Pro důkaz vlastností symetrie lze použít stejného postupu, jako v případě globálního provázání, s tím, že zobecnění platí ne pro všechny zprávy γ z množiny zpráv kontextové mapy, ale pouze pro jednu konkrétní zprávu γ.

Definice 2.9 *Dva uzly α, β kontextové mapy C jsou vzájemně částečně provázané $R_p(\alpha, \beta)$, pokud platí, že $c^\alpha c^\beta \neq 0$ a zároveň mezi těmito uzly existuje relace lokálního provázání $R_\gamma(\alpha, \beta)$ přes libovolný uzel γ.*

Věta 2.3 *Relace částečného provázání je i-reflexivní, symetrickou relací.*

Důkaz. Předpokládejme vlatnost reflexivity $R_p(\alpha, \alpha)$, protože vyžadujeme existenci relace lokálního provázání, pak by muselo platit, podle definice 2.8, že $c_\gamma^\alpha \neq c_\gamma^\alpha$, což je spor, relace je tedy i-reflexivní.

Symetričnost plyne opět z komutativity násobení vektorů, viz důkaz věty 2.1 .

∎

Definice 2.10 *Dva uzly α, β kontextové mapy C jsou vzájemně neprovázané $R_n(\alpha, \beta)$, pokud platí, že pro všechny uzly γ, nejsou α, β v relaci lokálního provázání $R_\gamma(\alpha, \beta)$.*

Věta 2.4 *Relace neprovázanosti je reflexivní, symetrickou a transitivní relací.*

Důkaz. Abychom ukázali, že je relace $R_n(\alpha, \alpha)$ reflexivní, stačí nám si uvědomit, že pro všechny uzly γ není $R_\gamma(\alpha, \alpha)$ relací lokálního provázání. Symetričnost opět vyplývá z komutativity násobení vektorů, pak pokud (α, β) nejsou v relaci $R_\gamma(\alpha, \beta)$, pak ani (β, α) nejsou v relaci $R_\gamma(\beta, \alpha)$, viz. důkaz věty 2.2 .

Vlastnost transpozice si dokážeme přímo. Pokud jsou uzly (α, β) lokálně provázány přes uzel γ, pak to znamená, že buď je $c_\gamma^\alpha = 0$, pak $c_\gamma^\beta \neq 0$, a nebo je $c_\gamma^\alpha \neq 0$, pak $c_\gamma^\beta = 0$, z toho vyplývá, že pro relaci neprovázanosti nám zbývají případy, kdy jsou buď prvky $c_\gamma^\alpha = c_\gamma^\beta = 0$, a nebo je $c_\gamma^\alpha \neq 0$ a zároveň $c_\gamma^\beta \neq 0$.

Pokud platí, že $c_\gamma^\alpha = c_\gamma^\beta = 0$ a uzly (β, θ) nejsou v relaci $R_\gamma(\beta, \theta)$, pak musí být také $c_\gamma^\theta = 0$, to znamená, že (α, θ) nejsou lokálně provázány.

Ve druhém případě platí, že $c_\gamma^\alpha \neq 0$ a zároveň $c_\gamma^\beta \neq 0$, aby uzly (β, θ) nebyly v relaci $R_\gamma(\beta, \theta)$, pak musí být také $c_\gamma^\theta \neq 0$, avšak, pak musí platit, že uzly (α, θ) nemohou být v relaci lokálního provázání. ∎

Relace neprovázanosti je podle dokázaných vlastností, relací ekvivalence. A je ekvivalentní Markovovské definici ergodicity pro N=1. Počty výskytů jednotlivých relací v testovaných souborech jsou uvedeny v tabulce 2.

Relace provázanosti byly definovány pro sloupcové vektory kontextové matice, zbývá doplnit, že stejné relace existují také mezi řádky kontextové matice. Důkazy jsou identické s jediným rozdílem, že místo kontextové matice C bychom pracovali s transponovanou kontextovou maticí C^T.

Soubor	$R_g(\alpha, \beta)$	$R_p(\alpha, \beta)$	$R_n(\alpha, \beta)$
bib	433	2804	3
book1	206	3099	16
book2	59	4501	0
geo	2	32638	0
news	8	4745	0
obj1	1135	31505	0
obj2	131	32509	0
paper1	514	3949	2
paper2	753	3342	0
paper3	518	2968	0
paper4	661	2483	16
paper5	1057	3038	0
paper6	568	3709	1
pic	3495	9045	21
progc	29	4157	0
progl	344	3396	1
progp	535	3380	1
trans	302	4546	3

Tabulka 2: Výskyt relací provázanosti v souborech Calgary Corpusu. $R_G(\alpha, \beta)$ - počet relací globálního provázání, $R_p(\alpha, \beta)$ - počet relací částečného provázání, $R_n(\alpha, \beta)$ - počet relací neprovázání.

3 Teorie informací

Pro mou práci je Teorie informací klíčovou teorií, proto se jí budu věnovat hned v úvodu práce a později se k jejím důsledkům budu vracet a využívat je v některých oblastech návrhu kontextových transformací. Tvůrcem teorie informací je americký matematik Claude Elwood Shannon[13]. Pojďme se nyní zaměřit na nejdůležitější body této teorie.

3.1 Entropie

Entropie je pojem přejatý ze statistické fyziky, sloužící k popisu množství informace obsažené v jednotlivých zprávách a k vyjádření pravděpodobnosti těchto zpráv.

Definice 3.1 *Zpráva je datový objekt určený ke kompresi*

Matematicky se entropie zapisuje následující rovnicí:

$$H(S) = \sum_{s \in S} p(s) log_2 \frac{1}{p(s)} \tag{1}$$

- $H(S)$ je entropie zpráv
- $p(s)$ je pravděpodobnost jedné určité zprávy s

Entropie je vážený průměr informací obsažených v každé zprávě a tudíž průměrný počet bitů informace v množině zpráv.

Poznámka 3.1 Zařízení se dvěmi stabilními stavy, jako je třeba flip-flop obvod, může uchovat jeden bit informace. N takových zařízení tak uchová 2^N stavů a zároveň platí, že $log_2 2^n = n$.

3.2 Vlastní informace

Uvažujme jednotlivé zprávy $s \in S$. Shannon definoval zápis vlastní informace nesené zprávou jako:

$$i(s) = log_2 \frac{1}{p(s)} \tag{2}$$

Takto definovaná vlastní informace reprezentuje počet bitů informace obsažené ve zprávě a zároveň počet bitů, který bychom měli použít, chceme li zprávu odeslat. Rovnice říká, že zprávy s vyšší pravděpodobností budou obsahovat méně informací.

3.3 Podmíněná entropie

Pravděpodobnost jednotlivých zpráv je založena na kontextu ve kterém se daná zpráva vyskytuje. Kontext obecně snižuje entropii systému. V současnosti se používají především dvě kompresní metody založené na kontextu, JBIG a PPM.

Podmíněná pravděpodobnost události e založené na kontextu c se značí jako $p(e|c)$. Celková pravděpodobnost události e je dána:

$$p(e) = \sum_{c \in C} p(c)p(e|c) \tag{3}$$

- C je množina všech dostupných kontextů

Zápis vlastní informace události e v kontextu c je v tomto případě dán vztahem:

$$i(e|c) = log_2 \frac{1}{p(e|c)} \tag{4}$$

Průměrná podmíněná vlastní informace je nazývána, jako takzvaná podmíněná entropie zdrojových zpráv. Pro množinu zpráv S a kontext c je podmíněná entropie definována jako:

$$H(S|C) = \sum_{c \in C} p(c) \sum_{s \in S} p(s|c) log_2 \frac{1}{p(s|c)} \tag{5}$$

Z tohoto vztahu můžeme ukázat, že pokud je rozdělení pravděpodobnosti množiny S nezávislé na kontextu c , pak $H(S|C) = H(S)$, tudíž pomocí použití kontextu můžeme entropii systému pouze snížit.

4 Kompresní algoritmy

Podle Shannonovy teorie, každý soubor obsahuje určité přesně dané množství informace. Toto množství informace však nemusí vždy být shodné s velikostí souboru. Účelem kompresních algoritmů je, abychom redukovali počet použitých bitů pro uložení zprávy na minimum. Na základě této snahy můžeme stanovit definici:

Definice 4.1 *Nechť $L(N)$ je funkce vracející počet bitů potřebných pro uložení zprávy N, pak kompresní algoritmus je posloupnost kroků $S_1, ..., S_n$, zkráceně S^n, při kterých se snažíme dosáhnout, aby $L(S^n(N)) < L(N)$.*

Z hlediska toho, zda je kompresní algoritmus při dekompresi schopen zprávu rekonstruovat do původní podoby, nebo původní podobu pouze aproximovat, dělíme kompresní algoritmy do dvou skupin:

1. Ztrátové algoritmy - zprávu pouze aproximujeme.
2. Bezeztrátové algoritmy - rekonstruujeme zprávu do původní podoby.

Protože hlavním tématem této práce jsou kontextové transformace souborů a jejich následné zpracování Huffmanovým kompresním algoritmem, přiblížíme si tyto oblasti více do hloubky. V následujících podkapitolách se budeme věnovat třem skupinám existujících algoritmů, na pravděpodobnostech a na gramatikách založených algoritmech a na závěr také současným algoritmům provádějícím transformace zpráv tak, aby lépe vyhohovovaly kompresním algoritmům.

4.1 Kritéria pro vyhodnocení úspěšnosti kompresních algoritmů

Pro vyhodnocování kvality kompresních algoritmů se používá několika vlastností. Nelze přitom porovnávat ztrátové a bezeztrátové algoritmy, každý nachází své uplatnění v jiné oblasti. Zatímco bezeztrátové algoritmy jsou důležité např. při kompresi binárních souborů, textů, zdrojových kódů..., ztrátové algoritmy nacházejí uplatnění v oblastech, kde se využívá nedokonalosti lidských orgánů a jejich neschopnosti rozpoznávat snížení informačního obsahu zpráv, jako příklad třeba hudební nebo obrazové soubory.

4.1.1 Vyhodnocování bezeztrátových algoritmů

1. Čas potřebný pro kompresi.
2. Čas potřebný pro dekompresi.
3. Velikost zkomprimovaného souboru vůči původní velikosti souboru, kompresní poměr.
4. Obecnost - jak je algoritmus úspěšný při kompresi různých typů souborů.

4.1.2 Vyhodnocení ztrátových algoritmů

Vyhodnocení ztrátových algoritmů je o něco obtížnější, protože musíme brát v úvahu, kvalitu aproximace, tato vlastnost je obzvláště důležitá při kompresi obrázků.

Jednou z běžně používaných metod porovnání algoritmů je od Jeffa Gilchrista nazvaná Archive Comparison Test. Tato metoda je zaměřena na porovnání času a výsledného kompresního poměru.

4.1.3 Calgary korpus

Calgary korpus je standardním nástrojem pro hodnocení(benchmark) kompresního poměru, převážně se skládá z anglického textu. Obsahuje dvě knihy, pět odborných článků, jednu bibliografii, kolekci novinových článků, tři programy, jeden výpis terminálového sezení, dva objektové soubory a jeden bitmapový obrázek.

Existuje však celá řada dalších korpusů např. Canterburry corpus, jeden z předních odborníků na kompresní algoritmy Matt Mahoney má na svých internetových stránkách[2]několik takových testovacích soustav. Algoritmy, prezentované v této práci, jsou však testovány pouze na corpusu Calgary.

4.2 Pravděpodobnostní kódování

Rozlišujeme mezi algoritmy, které přiřazují každé zprávě unikátní posloupnost bitů, a takové které spojují kód dohromady z více než jen jedné zprávy. Prvními se budu zabývat Huffmanovými kódy, které jsou označovány jako tzv. "Prefix codes", krátce se budu zabývat také aritmetickými kódy. Aritmetické kódy mohou dosáhnout lepšího kompresního poměru, výsledky jsou blíže entropickému optimu.

4.2.1 Prefixové kódy(směrové kódy)

V oblasti počítačů obvykle pracujeme s kódy pevné délky, jako příklad může posloužit ASCII kód, ten přiřazuje každému tisknutelnému znaku a několika dalším řídícím znakům sekvenci sedmi bitů. Pro účely komprese by však bylo vhodnější mít kódovací slova, která mohou mít proměnnou délku v závislosti na pravděpodobnosti dané zprávy.

Variabilní délka slova však přináší potenciální problémy, pokud dekomprimujeme nějakou zprávu, potřebujeme znát kde tato zpráva začíná a kde končí. Tyto dvě informace budou vyžadovat určitý paměťový prostor.

Efektivnějším řešením je navrhnout kódy, ze kterých můžeme vždy jednozančně dekódovat sekvenci bitů na svá kódová slova. Takové kódy se označují jako "jednoznačně dekódovatelné kódy".

Na směrový kód se nahlíží jako na binární strom, kde:

- Každá zpráva je listem stromu.

[2]http://www.mattmahoney.net/dc/

- Kód pro každou zprávu je dán následováním cesty od kořene stromu k listu a přidáním nuly pokaždé, když přecházíme do levé větve, respektive přídáním jedničky pokaždé, když přecházíme do pravé větve.

4.2.2 Huffmanovo kódování

Protože transformace kontextové mapy cílí na zlepšení kompresních schopností Huffmanova kódování, budeme se jim v této kapitole zabývat více do hloubky, budeme přitom vycházet z původní práce Davida Huffmana [6], kterou publikoval v roce 1952 během svého doktorského studia na MIT. Algoritmus je dodnes používanou součástí mnoha algoritmů, slouží jako základ pro komprese GZIP, JPEG, využívá se také při kódování MP3 souborů a několika dalších.

Mějme soustavu N zpráv a nechť $P(i)$ značí pravděpodobnost i-té zprávy, pak:

$$\sum_{i=1}^{N} P(i) = 1 \tag{6}$$

Nechť $L(i)$ vrací délku zprávy i, jako počet použitých kódovacích číslic, pak průměrná délka zprávy je:

$$\overline{L} = \sum_{i=1}^{N} P(i)L(i) \tag{7}$$

Huffmanův kód je optimální prefixový kód, který vede k nejmenší průměrné délce zpráv, samozřejmě zde myslíme pomocí celočíselného kódování(aritmetické kódování je schopno kódovat také neceločíselné hodnoty).

Huffman stanovil dvě základní vlastnosti, které vedou k vytvoření optimálního prefixového kódu.

- Žádné dvě různé zprávy se nesmí skládat z identického uspořádání číslic.
- Kód zpráv bude vytvářen tak, že nebude zapotřebí žádné další informace pro určení, kde zpráva začíná a kde končí.

Druhá vlastnost uvozuje fakt, že u takového kódu jsme schopni po přečtení určitého počtu symbolů rozhodnout, jaký symbol kóduje, tedy, že žádný prefixový kód nemůže být součástí jiného prefixového kódu. Pro otimální kód platí, že délka určité kódované zprávy nemůže být kratší, než délka zprávy s vyšší pravděpodobností, pak mějme následující seřazenou posloupnost pravděpodobností $(1..N)$ zpráv:

$$P(1) \geq P(2) \geq ... \geq P(N-1) \geq P(N) \tag{8}$$

a pro délku zpráv, pak platí třetí vlasnost pro optimální kód:

$$L(1) \leq L(2) \leq ... \leq L(N-1) \leq L(N) \tag{9}$$

Zároveň platí další dvě vlastnosti:

- Nejméně dvě, ale ne více, než D zpráv s kódem délky $L(N)$ má stejné kódy až na poslední číslice.
- Každá přípustná posloupnost $L(N) - 1$ číslic musí být kódem zprávy, nebo jeden z jeho prefixů musí být kódem zprávy.

Na základě popsaných vlastností, lze přistoupit k sestavení Huffmanova kódovacího stromu. Postup je následující:

1. Sestupné seřazení zpráv podle pravděpodobnosti do seznamu uzlů, jedna zpráva reprezentuje jeden uzel.
2. Vyjmutí dvou nejníže položených zpráv a jejich spojení do jednoho uzlu.
3. Zařazení nového uzlu do seznamu, tak aby seznam zůstal seřazený.
4. Opakování bodů 1-3, dokud v seznamu není pouze jeden uzel.
5. Ohodnocení zpráv pak probíhá stejně, jako bylo popsáno v sekci 4.2.1 o prefixových kódech, s tím, že postupujeme od posledního(kořenového) uzlu k listům(uzly, jež nejsou složené z více zpráv).

4.3 Na gramatikách založené kódování

Skupina algoritmů, která se snaží nad původní zprávou konstruovat bezkontextovou gramatiku se označuje anglickým termínem "Grammar-based codes", více např. [9] nebo [8].

Bezkontextová gramatika je definována čtveřicí $G(\Pi, \Sigma, P, S)$, kde

- Π je konečná množina neterminálních symbolů,
- Σ je konečná množina terminálních symbolů,
- S je počáteční neterminální symbol,
- P je množina přepisovacích pravidel typu $A \rightarrow \beta$, kde A je neterminální symbol a β je řětězec složený z terminálních a neterminálních symbolů.

Příkladem takového algoritmu je byte-pair kódování popsané v následující podkapitole. Grammar-based codes jsou zřejmě nejblíže konceptu kontextových transformací, které jsou navrženy v této práci.

4.3.1 Byte-Pair kódování

Je algoritmus navržený Philipem Gagem [4] , je založen na nalezení nejčastěji se vyskytujících dvojic znaků a ty nahrazuje pomocí znaků, které se v původní zprávě nevyskytují, každé takové nahrazení si poznamenává na zásobník. Dekomprese je pak obrácený postup, z vrcholu zásobníku odebere nahrazení, podle kterého nové znaky ve zprávě nahrazuje příslušnou dvojicí, takto postupuje až do vyprázdnění zásobníku, což znamená, že zpráva je rekontruovaná do původní podoby.

4.4 Transformace souborů

Ne všechny algoritmy pracující v oblasti kompresních algoritmů se snaží vykonávat kompresní funkci. Existují algoritmy, označované jako transformace, které převádějí původní zprávu na zprávu, která je vhodnější pro aplikaci kompresní funkce. Cílem této práce je vytvořit transformační algoritmus, který by se stal konkurentem těchto transformací. Proto si stručně přiblížíme dva nejznámější zástupce transformačních algoritmů, jedná se o Burrows-Wheelerovu transformaci a MoveToFront transformaci.

4.4.1 Burrows-Wheelerova transformace

Burrows-Wheelerova transformace má za cíl transformovat bloky zprávy, tak aby se stejné symboly vyskytovaly vedle sebe. Tohoto cíle dosahuje následujícím postupem:

1. Vytvoří matici rotací bloku zprávy.
2. Řádky rotační matice lexikograficky seřadí.
3. Výstupem transformace je poslední sloupec a index řádku, na kterém se vyskytuje původní blok.

Burrows-Wheelerova transformace se používá v kombinaci s transformací MoveToFront a entropickým kompresním algoritmem.

4.4.2 MoveToFront transformace

V doslovném překladu znamená ”move to front””přesuň na začátek”, což přesně vystihuje podstatu transformace. Transformace si na začátku zaindexuje znaky použité abecedy od 1 do n, potom předpokládá, že často se vyskytující znaky se nacházejí ve zpracovávaných datech blízko u sebe, proto aktuálně čtený znak přeindexuje na začátek v seznamu symbolů. Výsledkem je soubor v němž se zvýší výskyt nížších čísel. Transformace se obvykle používá až po aplikování Burrows-Wheelerovy transformace a je součástí například algoritmu bzip2.

5 Transformace kontextové mapy

Transformace kontextové mapy vycházejí z relací představených v kapitole 2., jejich cílem není provádět nad vstupní zprávou její kompresi, ale její transformaci, tak aby lépe vyhovovala kompresi převodem na Huffmanovo kódování, příp. aritmetické kódování.

Definice 5.1 *Transformace kontextové mapy je funkcí $T : \Sigma^* \rightarrow \Sigma^*$, kde Σ^* je iterací nad množinou symbolů abecedy vystupující v původní zprávě.*

Nyní si ujasníme základní vlastnosti transformace a požadavky, které na ni budeme klást. Zůstaneme u značení délky zprávy z kapitoly 4. o Huffmanově kódování.

Definice 5.2 *Nechť N_t značí transformovanou zprávu.*

Pak platí, že délka původní zprávy $L(N)$ je rovna délce zprávy transformované:

$$L(N) = L(N_t) \tag{10}$$

Definice 5.3 *Nechť $L_H(N)$ značí velikost zprávy po převodu na Huffmanův kód.*

Našim cílem je zajistit, aby platilo následující:

$$L_H(N) > L_H(N_t) \tag{11}$$

Tento vztah je základním požadavkem na transformaci, nikoliv však dostačujícím požadavkem, který by nám zajistil, že takto definovaná transformace bude mít menší požadavek na délku výsledného kódu i po započtení informace potřebné pro uložení metadat o transformacích.

Definice 5.4 *Nechť $L_T(N)$ je délka zprávy potřebná pro uložení metadat popisujících transformace.*

Obecný požadavek, který pak budeme klást na transformaci bude v následující podobě:

$$L_H(N) > L_H(N_t) + L_T(N) \tag{12}$$

Požadavek na inverzní transformaci:

Definice 5.5 *Inverzní transformace je funkce $T^{-1} : \Sigma^* \rightarrow \Sigma^*$ vracející původní zprávu N ze zprávy N_T, která byla vytvořena transformací T.*

Musí tedy platit:

$$T^{-1}(T(N)) = N \tag{13}$$

Důkaz výrazu (13) bude uveden později, až si popíšeme kontextové transformace.
Než přistoupíme k samotnému popisu transformace, uveďme si demonstrační příklad.

Příklad 5.1
Uvažujme slovo “abeceda” a podrobněji prozkoumejme návaznosti znaků, které se v tomto slově vyskytují. Postupnou analýzou zjistíme následující návaznosti, tabulka 3:

	a	b	e	c	d
a	0	1	0	0	0
b	0	0	1	0	0
e	0	0	0	1	1
c	0	0	1	0	0
d	1	0	0	0	0

Tabulka 3: Kontextová matice pro slovo ”abeceda”.

Interpretovat tyto návaznosti můžeme například následovně: čteme-li znak “b”, pak víme, že jeho předchůdcem mohl být pouze znak “a”, obdobně pro znak “b”, víme, že pokud jej aktuálně čteme, pak jej musel následovat znak “e”. Ve slově “abeceda” se vyskytuje pět různých znaků a šest různých návazností.

Můžeme si položit otázku, zda je možné nějakým způsobem snížit počet znaků, nebo přeneseně, zda je počet znaků ve slově se vyskytujících oprávněný, zda při zachování stejné informace, nelze počet znaků snížit.

Nejčastěji se vyskytujícím vrcholem návazností je znak “e”, vyberme si jej jako kandidáta, pomocí něhož se pokusíme snížit počet různých znaků ve slově “abeceda”. Z přehledu návazností vidíme, že znak “e” nemá s žádným jiným znakem společného předchůdce, proto můžeme zvolit libovolný znak a pomocí něj zvolit transformaci. Vybereme např. znak “b”, u kterého víme, že jej předchází znak “a”, poznačíme si na zásobník transformaci “abe0”(0 znamená, že jsme na jednom řádku provedli změnu ve dvou sloupcích, analogicky pro 1 bude znamenat, že jsme v jednom sloupci provedli záměnu dvou řádků), kterou čteme následovně: pokud čteme dvojici “ab”, pak ji nahraď za dvojici “ae”. Tímto krokem jsme z množiny použitých znaků odstranili znak “b” a získali jsme výsledný tvar slova “aeecede”. Nová kontextová matice je znázorněna v tabulce 4.

	a	e	c	d
a	0	1	0	0
e	0	1	1	1
c	0	1	0	0
d	1	0	0	0

Tabulka 4: Kontextová matice pro slovo ”aeeceda”.

Pokusme se opět snížit počet znaků, znaky ”c”a ”d”se ve slově ”aeeceda”vyskytují pouze jednou. Vyberme si např. znak ”c”, abychom jej mohli odstranit, museli bychom přesunout vazbu ”ec”, jediný prvek, jež je nulový v řádku označeném ”e”, je vazba ”ea”, proveďme transformaci ”eca0”a poznačme si ji na zásobník. Nahradíme dvojici ”ec”za ”ea”. Vznikne nám slovo ”aeeaeda”, na zásobníku máme $Z = [\text{"}eca0\text{"}, \text{"}abe0\text{"}]$ a kontextová matice je nyní ve stavu popsaném v tabulce 5.

	a	e	d
a	0	2	0
e	1	1	1
d	1	0	0

Tabulka 5: Kontextová matice pro slovo ”aeeaeda”.

Opět se pokusíme snížit počet znaků, prvkem s nejméně výskyty je znak ”d”. Zbývá jediná možnost, nahradit ”da”za ”aa”. Na zásobník si poznačíme transformaci ”ada1”. Nově vzniklé slovo je ”aeeaeaa”a jeho kontextová matice je zapsána v tabulce 6.

	a	e
a	1	2
e	2	1

Tabulka 6: Kontextová matice pro slovo ”aeeaeaa”.

Kontextová matice $C_{aeeaeaa}$ již neobsahuje žádný prvek, který by byl roven nule, nemůžeme tak dále provádět transformace. Na zásobníku máme tři transformace: $Z = [”ada1”, ”eca0”, ”abe0”]$. Výsledné slovo bylo redukováno na použití pouhých dvou znaků při zachování délky slova. Nyní je nasnadě další otázka a to zda jsme schopni zpětně slovo ”abeceda” ze slova ”aeeaeaa” rekonstruovat.

1. Ze zásobníku vybereme nejvyšší prvek ”ada1”. Čteme slovo ”aeeaeaa”po znacích, pokud narazíme na znak ”a”následovaný znakem ”a”, pak tuto dvojici ”aa”nahradíme za dvojici ”da”. Stav zásobníku je $Z = [”eca0”, ”abe0”]$ a slovo má tvar ”aeeaeda”.

2. Ze zásobníku vybereme nejvyšší prvek ”eca0”. Čteme slovo ”aeeaeda”po znacích, pokud narazíme na znak ”e”následovaný znakem ”a”, pak tuto dvojici ”ea”nahradíme za dvojici ”ec”. Stav zásobníku je $Z = [”abe0”]$ a slovo má tvar ”aeeceda”.

3. Ze zásobníku vybereme poslední prvek ”abe0”. Čteme slovo ”aeeceda”po znacích, pokud narazíme na znak ”a”následovaný znakem ”e”, pak tuto dvojici ”ae”nahradíme za dvojici ”ab”. Stav zásobníku je $Z = []$ a slovo má tvar ”abeceda”.

Postup naznačený na tomto příkladu budeme dále rozebírat v dalších podkapitolách.

■

5.1 Popis transformace

Základním principem kontextových transformací je posílení pravděpodobnosti výskytu uzlů, které jsou v původní zprávě nejčetnější. Dosahuje toho naplněním principu "chudým brát, bohatým dávat". Pokud transformace narazí na dvojici uzlů (α, β) a (α, γ), kde počet výskytů β je větší, než počet výskytů γ a zároveň platí, že počet výskytů $\alpha\beta$ je menší, než počet výskytů $\alpha\gamma$, pak se pokusí tuto dvojici zaměnit. V následujících větách a jejich důkazech si ukážeme, že zaměnit lze pouze dvojici (β, γ), takovou, že se nachází v relaci lokálního provázání přes nějaký uzel α.

5.1.1 Průběh transformace

Transformace probíhá v těchto krocích:

1. Vyhledáme transformaci a poznačíme si ji na zásobník.
2. Provedeme transformaci na aktuálně zpracovávaném řetězci.
3. Aktualizujeme kontextovou mapu.
4. Pokud existuje další připustná transformace, pak pokračujeme bodem 1.

Definice 5.6 *Stav systému S provádející transformace je určen dvojicí $S = (w, Z)$, kde w je aktuální hodnota zpracovávaného řetězce a Z je množinou transformací umístěných na zásobníků.*

Z hodnoty w, lze kdykoli sestavit kontextovou mapu. Můžeme tak říci, že se systém vyvíjí od jedné konfigurace k další.

Definice 5.7 *Konfigurace systému S je dána přirozeným číslem i, takovým, že stav systému po provedení transformace i je $S_i = (w_i, Z_i)$.*

Stav S_0 je výchozím stavem systému, kde w je vstupní řetězec a zásobník je prázdný. Na zásobník se s každou transformací ukládá trojice (α, β, γ), kde α, β, γ jsou slovy vyskytujícími se v původním řetězci w.

Pro lepší pochopení chování transformací si ukážeme některé vlastnosti transformací ve formě vět a jejich důkazů.

Věta 5.1 *Existují takové různé transformace T_1,T_2 a zprávy N, pro které platí, že $T_1(T_2(N)) \neq T_2(T_1(N))$.*

Důkaz. Pro důkaz věty 5.1 nám stačí ukázat zástupce tohoto tvrzení. Předpokládejme slovo "abc"a množinu transformací $T = \{aba0, bca0\}$. Pak platí, že $T_{bca0}(T_{aba0}(abc)) = aac$ a $T_{aba0}(T_{bca0}(abc)) = aaa$, z čehož vyplývá, že $T_1(T_2(N)) \neq T_2(T_1(N))$. ∎

Z věty 5.2 vyplývá, že při aplikování transformací záleží na pořadí, ve kterém jsou transformace provedeny.

Věta 5.2 *Pokud slova α, β, γ, tvoří relaci lokálního provázání $R_\alpha(\beta, \gamma)$, pak pro každou kontextovou transformaci $T_{\alpha,\beta,\gamma}$ existuje inverzní transformace $T^{-1}_{\alpha,\gamma,\beta}$, taková, že platí vztah(13).*

Pro důkaz věty 5.2 použijeme další pomocné věty.

Definice 5.8 *Označme si $T^i_{\rightarrow}(N)$ transformaci, kde $i \in \{0,1\}$, 0 značí transformaci dvou sloupců v jednom řádku a 1 značí transformaci dvou řádků v jednom sloupci, kterou provádíme, tak že zprávu N čteme postupně od prvního znaku po poslední znak zprávy N.*

Definice 5.9 *Označme si $T^i_{\leftarrow}(N)$ transformaci, kterou provádíme, tak že zprávu N čteme postupně od posledního znaku k prvnímu znaku zprávy N.*

Věta 5.3 *Pro transformaci $T^0_{\rightarrow(\alpha\beta\gamma)}(N)$ není transformace $T^0_{\rightarrow(\alpha\gamma\beta)}(N)$ inverzní transformací.*

Důkaz. Uvažujme slovo "babb", pro něj existuje relace lokálního provázání $R_a(b,a)$, na základě níž můžeme provést transformaci $T^0_{\rightarrow(a,b,a)}(\text{"}babb\text{"}) = \text{"}baaa\text{"}$. Transformace $T^0_{\rightarrow(a,a,b)}(\text{"}baaa\text{"}) = \text{"}baba\text{"}$, z čehož vyplývá, že $T^0_{\rightarrow(a,b,a)}(T^0_{\rightarrow(a,a,b)}(\text{"}babb\text{"})) \neq \text{"}babb\text{"}$. ∎

Důvod, proč platí věta 5.3 je v tom, že při prvotní transformaci dochází ke zřetězenému nahrazování, kdy jedno nahrazení "b(ab)b", dá vzniknout dalšímu nahrazení "ba(ab)", toto řetězené nahrazení neumožní zpětnou rekonstrukci. Až si budeme dokazovat větu 5.2, pak si ukážeme posloupnost konfigurací pro jednu transformaci, které využijeme pro její důkaz. Zřetězenému nahrazení lze zabránit čtením zprávy pozpátku.

Věta 5.4 *Pro transformaci $T^0_{\leftarrow(\alpha\beta\gamma)}(N)$ není transformace $T^0_{\leftarrow(\alpha\gamma\beta)}(N)$ inverzní transformací.*

Důkaz. Uvažujme slovo "abbb", pro něj existuje relace lokálního provázání $R_b(b,a)$, na základě níž můžeme provést transformaci $T^0_{\leftarrow(b,b,a)}(\text{"}abbb\text{"}) = \text{"}abaa\text{"}$. Transformace $T^0_{\leftarrow(b,a,b)}(\text{"}abaa\text{"}) = \text{"}abba\text{"}$, z čehož vyplývá, že $T^0_{\leftarrow(b,a,b)}(T^0_{\leftarrow(b,b,a)}(\text{"}abbb\text{"})) \neq \text{"}abbb\text{"}$. ∎

Důvod platnosti věty 5.4 je obdobný, jako u věty 5.3, s rozdílem, že zpráva obsahuje zřetězenou informaci, při zpětné transformaci dojde k provedení pouze jedné záměny, namísto dvou: "ab(bb)", zatímco zpětně: "a(ba)a".

Věta 5.5 *Pro transformaci $T^0_{\rightarrow(\alpha\beta\gamma)}(N)$ není transformace $T^0_{\leftarrow(\alpha\gamma\beta)}(N)$ inverzní transformací.*

Důkaz. Uvažujme slovo "abab", pro něj existuje relace lokálního provázání $R_a(b,a)$, na základě níž můžeme provést transformaci $T^0_{\rightarrow(a,b,a)}(\text{"}abab\text{"}) = \text{"}aaaa\text{"}$. Transformace $T^0_{\leftarrow(a,a,b)}(\text{"}aaaa\text{"}) = \text{"}abbb\text{"}$, z čehož vyplývá, že $T^0_{\leftarrow(a,a,b)}(T^0_{\rightarrow(a,b,a)}(\text{"}abab\text{"})) \neq \text{"}abab\text{"}$. ∎

Nyní můžeme přistoupit k důkazu věty 5.2, z vět 5.3, 5.4 a 5.5 je patrné, že jedinou zbývající přípustnou kombinací transformace a její inverze je $T = T^0_{\leftarrow}$ a $T^{-1} = T^0_{\rightarrow}$.

Důkaz. Uvažujme zprávu N a v ní relaci lokálního provázání $R_\alpha(\beta,\gamma)$. Nejdříve si dokážeme, že počet výskytů dvojice $(\alpha\beta)$ je roven počtu záměn β za γ. Protože zprávou N procházíme od konce, transformovaný znak nemůže ovlivnit čtení dalších částí zprávy a nemůže tak generovat jiné výskyty dvojic $(\alpha\beta)$, než těch, co už ve zprávě jsou.

Dále musíme ukázat, jaké dvojice (α,γ) mohou záměnami vzniknout či zaniknout. Na počátku je počet dvojic (α,γ) nulový, což je podmínka lokálního provázání. Víme, že

se provede tolik záměn, kolik je počet výskytů $(\alpha\beta)$ v původní zprávě. Jsou tři způsoby, jak může vzniknout nová dvojice (α, γ), buďto přímým nahrazením $\alpha\beta$ v řetězci "$y\alpha\beta x$", kde $x, y \neq \alpha$, v případě, kdy $x, \gamma = \alpha$ a nácházejí se v posloupnosti znaků "$\alpha\beta x$", pak vznikají dvě dvojice (α, γ) a posledním případem je posloupnost "$y\alpha\beta$", kde $y, \beta = \alpha$, pak jedna vznikne a druhá zanikne: "$\alpha\alpha\alpha$" $\rightarrow$ "$\alpha\alpha\gamma$" $\rightarrow \alpha\gamma\gamma$".

Pro tyto tři případy musíme ukázat, že transformace $T_{\rightarrow(\alpha,\gamma,\beta)}$ je inverzní transformací. První případ je triviální: "$y\alpha\beta x$" $\rightarrow$ "$y\alpha\gamma x$" $\rightarrow$ "$y\alpha\beta x$". Ve druhém případě nastává: "$\alpha\beta\alpha$" $\rightarrow$ "$\alpha\alpha\alpha$" $\rightarrow$ "$\alpha\beta\alpha$". Poslední případ nástává při: "$\alpha\alpha\alpha$" $\rightarrow$ "$\alpha\alpha\gamma \rightarrow$ "$\alpha\gamma\gamma$" $\rightarrow$ "$\alpha\alpha\gamma$" $\rightarrow$ "$\alpha\alpha\alpha$". V posledním případě bylo korektně použito řetězené nahrazování.

Ukázali jsme tak, že k transformaci $T^0_{\leftarrow(\alpha,\beta\gamma)}$, existuje inverzní transformace $T^0_{\rightarrow(\alpha,\gamma\beta)}$ ■

Věta 5.6 *Pro transformaci $T^1_{\rightarrow(\alpha\beta\gamma)}(N)$ je transformace $T^1_{\leftarrow(\alpha\gamma\beta)}(N)$ inverzní transformací.*

Důkaz. Nejdříve si připomeňme, že transformace $T^1(N)$ je záměna dvou řádků v jednom sloupci a provádí nahrazení $\beta\alpha$ za $\gamma\alpha$. Transformace $T^1(N)$, tak bude vykazovat stejné chování, jako transformace $T^0(N)$, pokud zprávu N budeme číst od konce resp. četli bychom její zrcadlový obraz N^R a na místo nahrazení $\beta\alpha$ za $\gamma\alpha$, bychom prováděli nahrazení $\alpha\beta$ za $\alpha\gamma$. Podle důkazů vět 5.2, 5.3, 5.4 a 5.5 pak zjistíme, že $T(N) = T^1_{\rightarrow(\alpha\beta\gamma)}(N)$ a k ní inverzní transformace je $T^{-1} = T^1_{\leftarrow(\alpha\gamma\beta)}(N)$. ■

5.2 Algoritmy pro nalezení transformací

Algoritmy pro nalezení transformací budou pracovat se slovem délky jednoho bytu. V následujících kapitolách je popsáno několik algoritmů a k nim přiloženy tabulky s výsledky. Na vstupu algoritmu budeme předpokládat kontextovou matici v následujícím tvaru.

Definice 5.10 *Nechť $f_c(w, N) : \Sigma^* \times \Sigma^* \to \mathbb{N}$ je funkce vracející pro slovo w počet jeho výskytů ve zprávě N.*

Seřadíme sestupně uzly kontextové mapy podle hodnoty f_c a sestavíme kontextovou matici, tak aby pro řádky i a sloupce j, které reprezentují slova v kontextové mapě platilo:

$$f_{c,i}(w_i, N) \geq f_{c,i+1}(w_{i+1}, N) \tag{14}$$

a zároveň také:

$$f_{c,j}(w_j, N) \geq f_{c,j+1}(w_{j+1}, N) \tag{15}$$

Účelem algoritmů je nalézt transformace a pomocí nich vytvořit zprávu, jejíž entropie bude menší, než entropie původní zprávy. Jedná se tak o formu optimalizačního problému. Díky tomu můžeme definovat optimalizační problém, který algoritmy řeší, označme si jej, jako "problém nalezení kontextových transformací":

Definice 5.11 *Vstupem problému nalezení kontextových transformací je řetězec w. Výstupem je posloupnost transformací P_T, taková, že výsledná entropie transformovaného řetězce spojeného s metadaty o transformacích je minimální.*

5.2.1 Greedy algoritmus

Základním algoritmem pro vyhledání transformace je greedy přístup. Budeme procházet vždy dvojice sloupců(sloupcové indexy i a j) a pro každou takovou dvojici vyhledáme, zda se nenechází v relaci lokálního provázání přes nějaký uzel γ (řádkový index k).

Protože sloupce a řádky kontextové matice jsou seřazeny sestupně podle počtu výskytů ve zdrojové zprávě, budeme se tak zajímat o situace kdy $c_k^i = 0$ a zároveň $c_k^j \neq 0$. Pak můžeme ve zdrojové zprávě nahradit dvojici (kj) za (ki). Tím posílíme výskyt uzlu i a naopak oslabíme výskyt uzlu j.

Transformace vyhledáváme tak dlouho, dokud se v kontextové mapě vyskytují nějaké relace lokálního provázání. V následujícím pseudokódu je celá transformace zachycena. Složitost greedy algoritmu je v $O(n^3)$, kde n je dimenze kontextové matice. Proces nalezení transformace je demonstrován v následujícím pseudokódu.

```
function greedy(cm) is
    input: kontextova matice cm
    max := 0
    transformace := 0
    for i = 1 ... dim(cm):
        for j = i+1 ... dim(cm):
            for k = 1 ... dim(cm):
                if (cm[k,i].hodnota == 0 and cm[k,j].hodnota != 0) and
                   (cm[k,j].hodnota > max):
                        max := cm[k, j].hodnota
                        transformace := [k, j, i]

    return transformace
```

Výpis 1: Greedy algoritmus pro nalezení transformace.

Greedy algoritmus připouští zhoršující se řešení. Zlepšení komprese se pohybuje mezi 0 - 8%, v závislosti na typu souboru. Výsledky jsou uvedeny v tabulce 7.

Soubor	$L(N)$	$L_H(N)$	$L_H(N_t)$	T	Optimum	T_{opt}
bib	111261	72760	67006	1741	67639	332
book1	768771	438373	-	-	-	-
book2	610856	368299	-	-	-	-
geo	102400	72555	-	-	-	-
news	377109	246393	-	-	-	-
obj1	21504	16051	-	-	-	-
obj2	246814	194095	-	-	-	-
paper1	53161	33336	31802	1465	32850	208
paper2	82199	47614	47781	1218	46829	183
paper3	46526	27274	26832	1061	27067	97
paper4	13286	7859	7404	877	7847	35
paper5	11954	7430	-	-	-	-
paper6	38105	24022	22733	1796	23571	146
pic	513216	106550	-	-	-	-
progc	39611	25913	24166	2066	24922	290
progl	71646	42981	40568	1090	41174	209
progp	49379	30213	27814	1545	28822	218
trans	93695	65217	65181	1519	65217	3

Tabulka 7: Velikosti výsledných souborů pro Greedy algoritmus ve srovnání s výsledky pro samostatnou Huffmanovu kompresi. Vysvětlivky: $L(N)$ - velikost původního souboru, $L_H(N)$ - velikost souboru po Huffmanově kompresi, $L_H(N_t)$ - velikost transformovaného souboru po Huffmanově kompresi, T - počet transformací, Optimum - nejlepší výsledek algoritmu, T_{opt} - počet transformací při dosažení optima. Pokud u souboru není uvedena hodnota, pak algoritmus nedosáhl vylepšení. Hodnoty v tabulce vyjadřují velikost v bytech.

5.2.2 Modifikovaný greedy algoritmus

Modifikovaný greedy algoritmus vychází z greedy algoritmu z předchozí podkapitoly. Algoritmus nevyhledává maximum v celé kontextové matici, ale zpracovává postupně jednotlivé uzly, s tím, že začíná od uzlu s nejčastějším výskytem a pokračuje přes uzly s nižším výskytem(index k). Index k zde vystupuje, jako řádkový index.

Pro aktuálně zpracovávaný uzel k postupně vyhledává relace lokálního provázání a na jejich základě provádí transformaci. Složitost algoritmu pro nalezení transformace je v $O(n^2)$, ve skutečnosti je však třeba provést $O(n^3)$ operací, protože vyhledání provádíme přes všechny uzly k. Průběh algoritmu je demonstrován v následujícím pseudokódu.

```
function modified_greedy(cm, k) is
    input: kontextová matice cm, zpracovávaný řádek k
    max := 0
    transformace := 0
    for i = 1 ... dim(cm):
        for j = i+1 ... dim(cm):
            if (cm[k,i].hodnota == 0 and cm[k,j].hodnota != 0) and
                    (cm[k,j].hodnota > max):
                        max := cm[k, j].hodnota
                        transformace := [k, j, i]

    return transformace
```

Výpis 2: Modifikovaný greedy algoritmus pro nalezení transformace.

Soubor	$L(N)$	$L_H(N)$	Optimum	T_{opt}
bib	111261	72760	67639	332
book1	768771	438373	-	-
book2	610856	368299	-	-
geo	102400	72555	-	-
news	377109	246393	246118	48
obj1	21504	16051	-	-
obj2	246814	194095	-	-
paper1	53161	33336	33194	19
paper2	82199	47614	47444	34
paper3	46526	27274	27201	10
paper4	13286	7859	-	-
paper5	11954	7430	7389	18
paper6	38105	24022	23691	62
pic	513216	106550	-	-
progc	39611	25913	25714	44
progl	71646	42981	39763	111
progp	49379	30213	29912	152
trans	93695	65217	65214	2

Tabulka 8: Velikosti výsledných souborů pro Modifikovaný greedy algoritmus ve srovnání s výsledky pro samostatnou Huffmanovu kompresi. Vysvětlivky: $L(N)$ - velikost původního souboru, $L_H(N)$ - velikost souboru po Huffmanově kompresi, Optimum - nejlepší výsledek algoritmu, T_{opt} - počet transformací při dosažení optima. Pokud u souboru není uvedena hodnota, pak algoritmus nedosáhl vylepšení. Hodnoty v tabulce vyjadřují velikost v bytech.

5.2.3 Redukční algoritmus

Redukční algoritmus má za cíl zmenšit počet uzlů, které kontextová mapa obsahuje. Provádí to tak, že vybírá postupně uzly(index k) od nejmenšího počtu výskytů a pokouší se pomocí transformací odstranit jeho vazby na ostatní uzly a tím tak odstranit daný uzel z kontextové mapy. Naší snahou bude převést vazbu z uzlu, s co nejmenším počtem výskytů, k uzlu, s co největším počtem výskytů.

Redukční algoritmus má časovou složitost v $O(n^2)$. Redukční algoritmus sám o sobě neslouží ke zmenšení entropie, ale spíše slouží pro předzpracování kontextové matice, na kterou je poté aplikován některý z dalších algoritmů pro nalezení transformací.

```
function reduction(cm,k) is
    input: kontextová mapa cm, řádkový index k
    transformace := 0
    for i = dim(cm) ... 1:
        for j = i−1 ... 1:
            if ((cm[k,i] != 0) and (cm[k,j] == 0)):
                transformace := [k, i, j]

    return transformace
```

Výpis 3: Redukční algoritmus pro nalezení transformace.

Soubor	Odstraněné uzly
bib	12
book1	1
book2	5
geo	0
news	1
obj1	0
obj2	0
paper1	14
paper2	20
paper3	17
paper4	15
paper5	19
paper6	13
pic	0
progc	3
progl	9
progp	16
trans	1

Tabulka 9: Počty odstraněných uzlů z kontextové mapy po aplikování Redukčního algoritmu.

5.2.4 Nulová transformace

Uvažujme kontextovou matici C, při hledání nulových transformací se budeme snažit o setřídění prvků matice $c_{i,j}$, takovým způsobem, aby platilo:

$$c_{i,j} > c_{i,j+1} \tag{16}$$

Abychom toho docíli, budeme postupovat v těchto krocích:

1. Vybíráme sloupcový index j sestupně od uzlu s největším počtem výskytů.
2. Vybíráme postupně uzly sestupně podle počtu jejich výskytů(řádkový index i).
3. Pro aktuální uzel nalezneme vazbu (i, k), která má nulový počet výskytů(může to být také vazba (i, j)).
4. Provedeme záměnu dvojice (i, k) za dvojici (i, j).
5. Nalezneme dvojici (i, m) s nejvyšší hodnotou v řádku i.
6. Provedeme záměnu (i, m) za (i, j).

Pro tento algoritmus si nebudeme uvádět výpis v pseudokódu, ale raději si důkladně ozřejmíme, co v jednotlivých krocích algoritmus provádí. Snahou algoritmu je umístit na pozice dané, co nejnižšími indexy i, j, vazby s nejvyšším počtem výskytů. Musíme si uvědomit, že jsme limitováni použitím záměn, jen u takových dvojic uzlů, které se vyskytují v relaci lokálního provázání přes řádkový index uzlu. Pokud na nejnižší pozici, řekněme $(1, 1)$ nebude hodnota 0, pak abychom mohli nějakou dvojici $(1, m)$(maximální dvojice na řádku) na ni umístit, musíme nejdříve pozici $(1, 1)$ vynulovat, proto je nezbytné nejdříve nalézt dvojici $(1, k)$, která má nulovou hodnotu.

Pokud takovou nulovou dvojici nenalezneme, pak musíme pokračovat dalším indexem i. Pokud však nalezneme, pak se nám otevře možnost, podle relace lokálního provázání zaměnit vazbu $(1, 1)$ za vazbu $(1, k)$, tím jsme získali hodnotu 0 na pozici $(1, 1)$. Protože hodnota $(1, 1)$ reprezentuje ve zpracovávané zprávě posloupnost dvou nejčastěji se vyskytujících uzlů, je našim zájmem do této pozice umístit maximální hodnotu, která se na řádku vyskytuje. Proto vyhledáme maximum(index m) a provedeme záměnu.

Pro případ, kdy se v řádku nevyskytuje žádná dvojice s nulovou hodnotou, je možné algoritmu nastavit, aby mu byl přidán prázdný uzel, což je nějaký prvek, který se v kontextové mapě nevyskytuje, nemá tak žádné vazby na ostatní uzly a díky tomu, do každého řádku i umístí jednu nulovou dvojici. Složitost algoritmu je opět v $O(n^3)$. Výsledky, jak pro samotnou nulovou transformaci, tak pro vybrané soubory s přidaným nulovým uzlem jsou shrnuty v tabulce 10. Přidání nulového uzlu bylo zkoumáno později především u algoritmu nulové entropické transformace.

Soubor	$L(N)$	$L_H(N)$	Optimum	T_{opt}	$Optimum_0$	$T_{opt,0}$
bib	111261	72760	64401	718	64349	633
book1	768771	438373	434121	322	433130	273
book2	610856	368299	339932	2850	-	-
geo	102400	72555	-	-	-	-
news	377109	246393	230713	1678	-	-
obj1	21504	16051	-	-	-	-
obj2	246814	194095	-	-	-	-
paper1	53161	33336	31929	282	-	-
paper2	82199	47614	45361	254	-	-
paper3	46526	27274	26312	92	-	-
paper4	13286	7859	7808	62	-	-
paper5	11954	7430	7346	66	-	-
paper6	38105	24022	23054	284	-	-
pic	513216	106550	-	-	-	-
progc	39611	25913	24519	430	24507	434
progl	71646	42981	39473	730	40362	581
progp	49379	30213	27015	432	27022	436
trans	93695	65217	-	-	-	-

Tabulka 10: Velikosti výsledných souborů pro Nulový algoritmus ve srovnání s výsledky pro samostatnou Huffmanovu kompresi. Vysvětlivky: $L(N)$ - velikost původního souboru, $L_H(N)$ - velikost souboru po Huffmanově kompresi, Optimum - nejlepší výsledek algoritmu, T_{opt} - počet transformací při dosažení optima, $Optimum_0$ - nejlepší výsledek algoritmu - přidán prázdný uzel, $T_{opt,0}$ - počet transformací při dosažení optima - přidán prázdný uzel. Pokud u souboru není uvedena hodnota, pak algoritmus nedosáhl vylepšení. Hodnoty v tabulce vyjadřují velikost v bytech.

5.2.5 Redukce + nulová transformace

V této části jsou uvedeny výsledky pro vybrané soubory z Calgary corpusu, na které byl nejdříve aplikován redukční algoritmus pro redukci abecedy a teprve následně na něj byla aplikována nulová transformace. Vybrány byly především soubory, u kterých došlo k výrazné redukci abecedy. V tabulce 11 je uvedeno srovnání algoritmu se samotnou nulovou transformací. U zkoumaných souborů došlo k mírnému zlepšení vlastností, oproti samotnému nulovému algoritmu.

Soubor	$L(N)$	$L_H(N)$	$L_H(N_{t,r,n})$	$L_H(N_{t,n})$
bib	111261	72760	63720	64401
paper1	53161	33336	31838	31929
paper2	82199	47614	45223	45361
paper3	46526	27274	26118	26312
paper4	13286	7859	7717	7808
paper5	11954	7430	7291	7346
paper6	38105	24022	22979	23054

Tabulka 11: Velikosti výsledných souborů, pro kombinaci Redukčního a Nulového algoritmu, ve srovnání s výsledky pro samostatnou Huffmanovu kompresi pro vybrané soubory Calgary corpusu. Vysvětlivky: $L(N)$ - velikost původního souboru, $L_H(N)$ - velikost souboru po Huffmanově kompresi, $L_H(N_{t,r,n})$ - nejlepší výsledek kombinace algoritmů, $L_H(N_{t,n})$ - výsledek po nulové transformaci. Hodnoty v tabulce vyjadřují velikost v bytech.

5.2.6 Vliv kontextových transformací na entropii

Uzavřený fyzikální termodynamický systém je charakteristický tím, že v rovnovážném stavu je zároveň ve stavu s maximální entropií [7]. Pokud tento pohled aplikujeme na entropii založenou na teorii informací, pak se můžeme soustředit na vývoj míry entropie zpráv. Uvažujme nyní případ kontextových transformací. Rovnovážný stav takového systému daného zprávou N, pak nastane, když nelze jeho entropii E žádnou transformací T z množiny dostupných transformací $\mathcal{T}$ zvýšit:

$$E_{max} = \{E : E(N) \geq E(T(N)), T \in \mathcal{T}\}$$

Stejným způsobem budeme definovat minimální entropii systému daného zprávou N:

$$E_{min} = \{E : E(N) \leq E(T(N)), T \in \mathcal{T}\}$$

Lokální transformace kontextové mapy je vratnou změnou systému a nás v případě komprese dat nás bude zajímat případ, kdy se vratnými změnami snažíme entropii systému snižovat. Obdobně bychom mohli nahlížet na fyzikální systém, řekněme, že se daný systém nachází v čase t ve stavu Ψ_t s entropií E_t. Druhá termodynamická věta říká, že

entropii uzavřeného systému nelze snížit, nicméně z pohledu informatického ji změnit lze a to takovým způsobem, že pokud je změna takového systému jasně dána nějakým pravidlem, například chemickou reakcí, pak jsme schopni takový systém modelovat před provedením dané reakce, kdy je ve stavu $\Psi_{t-\delta t}$ s entropií $E_{t-\delta t} < E_t$ a provedenou reakci si poznamenat.

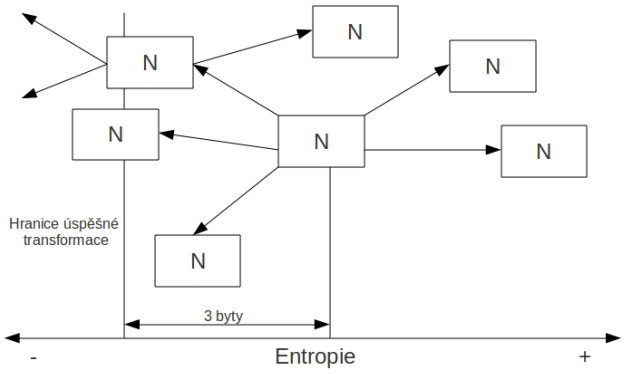
Kontextové transformace jsou příkladem takového informatického systému, který popisem změny stavu z $\Psi_{t-\delta t} \rightarrow \Psi_t$, snižuje entropii systému, kde stav Ψ_t považujeme za stav popisující výchozí zprávu. Samozřejmě máme možnost provádět vratné operace oběma směry, tedy můžeme systém modelovat ve stavech s nižší i vyšší entropiií, za přepokladu, že se systém nenalézá ve svém globálním minimu, resp. maximu.

Definice 5.12 *Entropická transformace je takovou transformací T zprávy N, pro kterou platí, že $E(T(N)) < E(N)$.*

Další analogií s fyzikálním systémem je energetická náročnost reakce, každá reakce potřebuje ke svému spuštění dodat tzv. aktivační energii. Aktivační energií reprezentuje v kontextových transformacích paměťový nárok na uložení transformace. Abychom zajistili, že transformace T systému bude výhodná z hlediska nároků na paměť, musí platit vztah:

$$E(N)L(N) > E(T(N))L(N) + L(M) \tag{17}$$

Změny jakými se transformace mohou ubírat jsou zobrazeny na obrázku 1. Poslední analogií s fyzikálními systémy je přidání nulového uzlu do kontextové mapy. Přidání nulového prvku, může být vnímáno, jako přidání katalyzátoru do transformačního prostředí. Účelem katalyzátoru je pozměnit průběh vývoje systému k dosažení požadovaných vlastností. Například v (odkaz na tabulku nulové transformace) u souboru book1 můžeme vidět, že přidání nulového prvku způsobilo zlepšení komprese.



Obrázek 1: Dostupné transformace v prostoru entropie.

5.2.7 Nulová entropická transformace

Poslední představený algoritmus je založen na předcházející nulové transformaci. Jediný rozdíl v algoritmu je, že nepřipouští zhoršující se řešení. Po každé transformaci zkontroluje, zda nedošlo ke zhoršení entropie podle rovnice (17), kde $L(M)$ je v tomto případě dáno počtem transformací vynásobenou 3 byty, pokud dojde ke zhoršení entropie, pak je aplikována inverzní transformace a pokračuje se další dostupnou transformací.

Výsledky jsou shrnuty v tabulce 12 a jak je v tabulce vidět, algoritmus dosahuje dosud nejlepších výsledků, ze všech zkoumaných transformačních algoritmů.

Soubor	$E(N)$	$E(N_t)$	$L(M + H(N_t)$	T	$L_0(M + H(N_t)$	T_0
bib	5,20	4,45	63641	466	63285	480
book1	4,53	4,43	429493	224	427296	322
book2	4,79	4,30	334366	1167	335512	1134
geo	5,64	-	-	-	-	-
news	5,18	4,75	229082	1004	230269	1006
obj1	5,95	-	-	-	-	-
obj2	6,26	-	-	-	-	-
paper1	4,98	4,52	30974	266	30988	268
paper2	4,60	4,18	43974	290	43972	290
paper3	4,66	4,27	25595	206	25642	196
paper4	4,70	4,41	7587	76	7581	70
paper5	4,94	4,57	7114	82	7131	82
paper6	5,01	4,53	22527	278	22535	274
pic	1,21	-	-	-	-	-
progc	5,19	4,57	23714	321	23726	313
progl	4,77	4,10	37983	328	38237	300
progp	4,87	4,14	26898	306	26895	288
trans	5,53	-	-	-	-	-

Tabulka 12: Velikosti výsledných souborů, pro Nulový entropický algoritmus, ve srovnání s výsledky pro samostatnou Huffmanovu kompresi. Vysvětlivky: $E(N)$ - entropie původního souboru, $E(N_t)$ - entropie po transformacii, $L(M + H(N_t)$ - velikost souboru po transformaci, T - počet transformací. $L_0(M + H(N_t)$ - velikost souboru po transformacích - přidání prázdného uzlu, T_0 - počet transformací - přidání prázdného uzlu. Pokud u souboru není uvedena hodnota, pak algoritmus nedosáhl vylepšení. Hodnoty v tabulce vyjadřují velikost v bytech.

5.3 Provedení transformace

V přechozím kroku algoritmus nalezl transformaci, nyní je zapotřebí přepracovat kontextovou mapu, tak aby odpovídala provedené transformaci. Je třeba si uvědomit, že se budou měnit vazby mezi jednotlivými uzly v kontextové mapě. Mějme dánu transformaci (α, β, γ), pak pokud v transformované zprávě dojde k nalezení sekvence $\alpha\beta\epsilon$ a my tuto sekvenci zaměňujeme za $\alpha\gamma\epsilon$, tak to pro nás znamená následující:

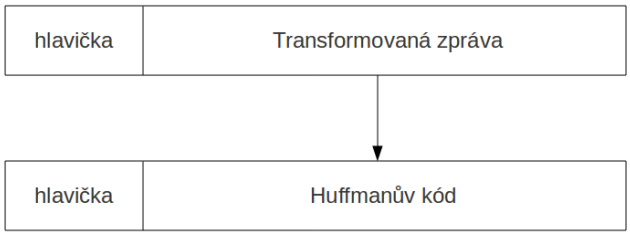
1. V kontextové mapě musíme snížit hodnotu vazeb $\alpha\beta$ a $\beta\epsilon$ o jedničku.
2. V kontextové mapě musíme zvýšit hodnotu vazeb $\alpha\gamma$ a $\gamma\epsilon$ o jedničku.

Složitost nahrazení textu transformacemi je v $O(kL(N))$, kde k je počet provedených transformací a $L(N)$ je velikost zprávy v bytech, protože průchod zprávou musíme udělat tolikrát, kolik je transformací. V případech testovaných v této práci platí, že $k << L(N)$. Obdobně, pokud se podíváme na složitosti algoritmů pro nalezení transformací, pak jejich složitost je sice v $O(n^3)$, nicméně dimenze n kontextové matice je vždy $n \leq L(N)$ a v testovaných případech platilo, že $L(N) \in < n^2, n^3 >$. Proto je časová náročnost nalezení transformace, velmi blízká časové náročnosti nahrazení textu.

5.4 Struktura výsledného souboru

Strukturu výsledného souboru si popíšeme ve dvou provedeních, na obrázku 2 je struktura souboru po transformacích a zároveň struktura souboru po transformacích a Huffmanově kódování.

V hlavičce souboru je vepsán počet transformací, jako 2-bytová hodnota(až 65536 transformací), transformace se ukládají, jako 3-bytová slova a jsou umístěny za počtem transformací. Za hlavičkou následuje transformovaná zpráva. V případě, po aplikaci Huffmanova kódování je, namísto transformované zprávy, umístěn Huffmanův strom a samotný Huffmanův kód.



Obrázek 2: Struktura výsledného souboru.

5.5 Zpětná transformace

Zpětná transformace souborů je fakticky totožnou operací, jako transformace pro kompresi. Jediný rozdíl spočívá ve změně směru čtení souboru(věta 5.2) a v záměně trojice

(α, β, γ) za (α, γ, β), s tím, že při kompresní transformaci jsme ukládali jednotlivé transformace na zásobník, při zpětné transformaci je postupně vybíráme z vrcholu zásobníku. Algoritmus je při zpětné transformaci poměrně rychlý a jeho časová složitost je v $O(kL(N))$. Časová složitost vyplývá z potřeby projít soubor, opět tolikrát, kolik máme na zásobníku transformací.

5.6 Shrnutí výsledků

V následující tabulce se nachází přehled zlepšení výsledků Huffmanovy komprese pro jednotlivé algoritmy, společně s experimentálním minimem, pokud nebudeme započítávat informační náklady na uložení transformací:

Soubor	Greedy	Mod.greedy	Redukce	Nulový	Nulový ent.	Minimum
bib	7,04	7,04	3,03	11,44	13,02	17,18
book1	-	-	-	1,20	2,53	2,98
book2	-	-	-	7,70	9,21	10,95
geo	-	-	-	-	-	-
news	-	0,11	-	6,36	7,02	9,25
obj1	-	-	-	-	-	-
obj2	-	-	-	-	-	-
paper1	1,46	0,43	0,35	4,22	7,09	13,70
paper2	1,65	0,36	0,35	4,73	7,65	10,94
paper3	0,76	0,27	0,99	3,53	6,16	10,88
paper4	0,65	-	0,81	0,65	3,54	13,16
paper5	-	0,55	0,10	1,13	4,25	14,70
paper6	1,88	1,38	0,28	4,03	6,22	14,02
pic	-	-	-	-	-	-
progc	3,82	0,77	-	5,43	8,49	15,18
progl	4,20	7,49	-	8,16	11,63	14,18
progp	4,60	1,00	-	10,58	10,98	18,08
trans	-	-	-	-	-	0,07

Tabulka 13: Shrnutí výsledků pro všechny algoritmy. Hodnoty jsou uvedeny, jako procentuální zlepšení komprese oproti použití samotného Huffmanova kódování. Minimum značí experimentálně naměřené minimum, ve kterém není započítán prostor pro uložení transformací.

Jak je vidět v tabulce 13, nulový entropický algoritmus dosahoval na všech testovaných souborech nejlepších výsledků, avšak žádný z navrhovaných algoritmů nebyl úspěšný při transformování pěti souborů: geo, obj1, obj2, pic a trans. Je to dáno především jejich povahou, u všech se vyskytuje jeden dominantní symbol, který má hlavní vliv a popsanými algoritmy nebylo možné zvýšit pravděpodobnost jeho výskytu.

Při transformacích se vyskytuje ještě další problémová vlastnost, kterou označuji pojmem "negativní zpětná vazba". To znamená, že ve fázi, kdy dochází k přeměně znaků,

dochází ke změnám hodnot vazeb nejenom mezi nahrazovanými uzly a jejich společnou vstupní vazbou, ale také mezi nahrazovanými uzly a jejich následovníky, tato změna negativně ovlivňuje rozdělení pravděpodobnosti výskytů vazeb. Negativní zpětná vazba bude muset být při rozvoji algoritmu brána, jako důležitý faktor.

Obecně se však ukazuje, že kontextové transformace mají schopnost narušovat rovnoměrnost rozložení znaků ve zpracovávaných souborech a tím působí, jako přiznivý faktor pro Huffmanovo kódování.

6 Podobnost na základě relací provázanosti

Definice 2.7 a věta 2.4 nám říkají, že relace globálního provázání je relací ekvivalence. Každá relace ekvivalence definuje třídy rozkladu X_i na množině uzlů. Nicméně relace globálního provázání by nebyla schopna vzhledem k počtu všech vazeb mezi uzly vytvořit smysluplné třídy rozkladu. Proto stanovíme jinou míru, která bude vyjadřovat podobnost dvou uzlů a to ve dvou směrech:

1. Podobnost $P^{\alpha,\beta}$ mezi vazbami vstupujícími do uzlu.
2. Podobnost $P_{\alpha,\beta}$ mezi vazbami vystupujícími z uzlu.

Tato podobnost P bude dána počtem relací lokálního provázání mezi dvěma uzly α, β přes všechny uzly γ kontextové mapy.

Definice 6.1 *Nechť U značí množinu uzlů vyskytujících se v kontextové mapě C.*

Pak podobnost budeme definovat následovně:

$$P(\alpha,\beta) = \frac{1}{|U|}\sum_{\gamma\in U} p_{\alpha,\beta}, \begin{cases} p_{\alpha,\beta} = 0 & (\alpha,\beta,\gamma) \notin R_\gamma(\alpha,\beta) \\ p_{\alpha,\beta} = 1 & (\alpha,\beta,\gamma) \in R_\gamma(\alpha,\beta) \end{cases} \tag{18}$$

Čím více se bude $P(\alpha,\beta)$ blížit 0, tím podobnější si uzly α, β jsou. Podobnost $P(\alpha,\beta)$ jinými slovy vyjadřuje míru ortogonality mezi vektory c_α, c_β a c^α, c^β. Celkovou podobnost $\bar{P}$ mezi dvěma uzly, pak budeme definovat následovně:

$$\bar{P}(\alpha,\beta) = \frac{1}{2}(P_{\alpha,\beta} + P^{\alpha,\beta}) \tag{19}$$

Schopnosti této míry porovnáme s mírou založenou na vzdálenosti dvou pravděpodobnostních rozdělení, někdy též označovanou, jako Kolmogorova vzdálenost:

$$D_{\alpha,\beta} = \frac{1}{2}\sum_{\gamma\in U} |p_\alpha^\gamma - p_\beta^\gamma| \tag{20}$$

$$D^{\alpha,\beta} = \frac{1}{2}\sum_{\gamma\in U} |p_\gamma^\alpha - p_\gamma^\beta| \tag{21}$$

,kde p_α^β vyjařuje pravděpodobnost přechodu z uzlu α do uzlu β.

Srovnání jsem provedl pro slova w o velikosti $|w| = 1$ a výsledky jsou uvedeny v tabulkách 14, 15 16, pro charakteristické zástupce skupin znaků. Mezi zástupce byla vybrána číslovka, souhláska a samohláska. V tabulkách jsou vybráni vždy čtyři nejpodobnější prvky a čtyři nejodlišnější znaky, 1) podle podobnosti na základě relací lokálního provázání a 2) podle Kolmogorovy vzdálenosti.

V tabulce 14 je vidět, že pododnost na základě relací provázanosti přesně identifikuje číslovky, zatímco pomocí Kolmogorovy vzdálenosti se na třetí pozici objeví závorka. V tabulce 15 pro znak "a"vidíme, že už na třetí a čtvrté pozici jsou podle obou mír umístěny

Znak β	4	6	7	5	...	O	s	e	\n
$P(0, \beta)$	0.037	0.037	0.049	0.06	...	0.54	0.54	0.55	0.66
Znak β	6	4	)	5	...	i	e	%	\n
$D(0, \beta)$	0.24	0.25	0.26	0.27	...	1.64	2.52	3.40	10.54

Tabulka 14: Přehled hodnot podobností na základě relací provázanosti a na základě Kolmogorovy vzdálenost pro znak ”0”.

Znak β	o	u	s	r	...	/	%	mezera	\n
$P(a, \beta)$	0.02	0.09	0.09	0.09		0.48	0.53	0.54	0.64
Znak β	u	o	z	k	...	e	.	%	\n
$D(a, \beta)$	0.72	0.90	0.93	0.94		1.99	2.01	4.09	10.72

Tabulka 15: Přehled hodnot podobností na základě relací provázanosti a na základě Kolmogorovy vzdálenost pro znak ”a”.

Znak β	r	n	y	d	...	3	mezera	%	\n
$P(s, \beta)$	0.05	0.05	0.09	0.09		0.49	0.55	0.57	0.65
Znak β	g	y	d	x	...	.	e	%	\n
$D(s, \beta)$	0.66	0.69	0.71	0.78		1.90	2.27	3.99	10.98

Tabulka 16: Přehled hodnot podobností na základě relací provázanosti a na základě Kolmogorovy vzdálenost pro znak ”s”.

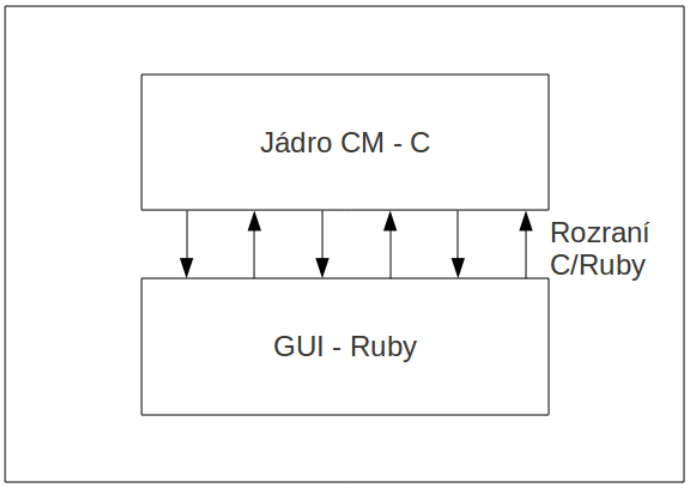
souhlásky, nicméně v případě Kolmogorovy vzdálenosti byla samohláska ”e”určena až mezi nejméně podobnými znaky.

Na příkladech je demonstrováno, že podobnost na základě relací provázanosti má v konkrétních případech vypovídací schopnost, i přesto, že je mírou založenou fakticky pouze na přičítání jedničky, může tak být použita například, jako vstup pro shlukovací algoritmy, které by na základě této míry stanovovaly skupiny sobě podobných znaků, případně také slov, nebo textů.

7 Popis implementace

Zdrojové kódy programu CMA(Context Map Analyser) jsou napsány ve dvou jazycích:

1. C - knihovna pro práci s kontextovou mapou.
2. Ruby - grafické rozhraní v Ruby GTK.



Obrázek 3: Struktura knihovny.

Jazyk Ruby a grafické knihovny Gtk jsem pro svou práci zvolil především z důvodů osobní zkušenosti s prací s nimi. Nicméně jazyk Ruby je zástupce jazyků skriptovacích, s tím, že je to jazyk čistě objektový. Čistá objektovost znamená, že v jazyce Ruby se nevyskytují datové typy, naopak každá proměnná reprezentuje objekt definovaný třídou. Problémem jazyka Ruby je jeho rychlost, zkušenost z tvorby této práce mi ukázala, že i jednoduchý algoritmus pro nahrazení znaků v řetězci, může být až sto-násobně déle trvající, než-li identický algoritmus implementovaný v jazyce C.

Právě rychlost byla důvodem, proč jsem zvolil pro implementaci algoritmů a reprezentaci kontextové mapy jazyk C. Mezi kódy jazyka C a Ruby jsem implementoval rozhraní, které umožňuje volání funkcí kódu v jazyce C z Ruby a naopak při volání z Ruby umožní dotazování na kód v jazyce C. Tyto převody jsou realizovány pomocí volání z knihovny ruby.h. Knihovna v C má navíc tu výhodu, že se pro ni dá snadno vytvořit rozhraní také do Pythonu skrze PythonExtensions a nebo do Javy skrze Java Native Interface.

Knihovna ContextMap.so je ve dvou verzích, první verze podporuje pouze slova o délce $|w| = 1$, zatímco druhá verze knihovny umožňuje práci se slovy libovolné délky. Knihovna byla připravena a zkompilována pouze pro operační systém Linux.

Kód v jazyce C je umístěn v kořenovém adresáři programu ve složce src. Pro některé funkce jsem napsal testy jejich správnosti a jsou uloženy ve složce tests. Rozhraní mezi

jazyky C a Ruby je ve složce interface. V kořenovém adresáři je pak umístěn bash script deploy.sh, který se stará o kompilaci knihovny.

Kód v jazyce Ruby je umístěn v adresáři src/Ruby. Tento adresář je dále rozdělen do adresářů algorithms, config, lib, models a views. V adresáři algorithms jsou umístěny skripty, které obsluhují transformařní algoritmy. V adresáři config je umístěn skript pro načtení všech skriptů programu. Adresář lib obsahuje knihovnu ContextMap.so. Adresář models obsahuje skripty reprezentující datové objekty, jak pro práci s algoritmy, tak pro práci s gui. Adresář views obsahuje třídy pro obsluhu gui.

Program CMA se spouští příkazem: ruby program.rb z adresáře src/Ruby. Popis funkčnosti a obsluhy programu je umístěn v Dodatku A této práce.

8 Závěr

V této práci jsem zpracoval teorii ke kontextovým transformacím, na základě níž jsem poté mohl vytvářet kontextové transformace nad vstupními daty a stanovit vztahy pro podobnost mezi dvěma uzly kontextové mapy. Teorie je založena na čtyřech relacích a jejich vlastnostech. Transformace jsou podobné přepisovacím pravidlům v bezkontextových gramatikách, ale podstata jejich užití je odlišná, jejich cílem je snižovat entropii zprávy, při zachování její délky(bez metadat). Algoritmy dosahovaly na souborech Calgary corpusu zlepšení Huffmanova statického kódování v rozmezí 0-13%.

V této práci jsem se nezabýval limity kontextových transformací, tak nemohu zhodnotit, do jaké míry se například nulový entropický algoritmus přiblížil minimální, dosažitelné entropii. Algoritmy v této práci pracovaly pouze s jedním typem relací lokálního provázání a to transformací dvou sloupců v jednom řádku, potenciál dalšího rozvoje algoritmů, tak vidím především v zapracování obousměrné transformace, která by mohla dále vylepšit vlastnosti transformací, případně otestovat napojení na Burrows-Wheelerovou a MoveToFront transformace.

Jedním ze zajímavých důsledků práce je, že přidáním malého množství informace ke zprávě, lze snížit celkovou entropii této zprávy, tato informace, zde vystupuje spíše jako instrukce programu provádějící nahrazení dvou znaků zprávy. Kontextovými transformacemi můžeme původní zprávou manipulovat v entropickém prostoru, díky tomu, dalším potenciálním využitím transformací může být aplikace v symetrické kryptografii. Pomocí transformací lze konstruovat zprávy, jejichž entropie je maximální a tak získat zprávy odolné proti útokům na základě frekvenční analýzy, klíčem by pak byla posloupnost transformací. Zpětná transformace hrubou silou, bez znalosti transformací, by vyžadovala vyzkoušení všech dostupných transformačních cest, přičemž chyba v jedné transformaci by se propagovala celou zprávou.

Michal Vašinek

9 Reference

[1] Blelloch, Guy E., *Introduction to Data Compression,* Carnegie Mellon University, 2001.

[2] Brown, F. Peter, Della Pietra, Vincent. J., deSouza, Peter V., Lai, Jenifer C., Mercer, Robert L., *Class-base n-gram models of natural language. Computational Liguistics*, Vol. 18 (4), pp. 467-479, 1992.

[3] Einstein, A., Podolsky B., Rosen N., *Can Quantum-Mechanical Description of Physical Reality be Considered Complete, Physical Review*, Vol. 47 (10), pp. 777-780, 1935.

[4] Gage, Philip, *Philip Gage, A New Algorithm for Data Compression. "Dr Dobbs Journal"*, Dostupné z: http://www.drdobbs.com/a-new-algorithm-for-data-compression/184402829, 1994

[5] Huckle, C. Christopher, *Grouping Words Using Statistical Context. CoRR*, Vol. cmp-lg/9502034, 1995.

[6] Huffman, D.A, *A Method for the Construction of Minimum-Redundancy Codes, Proceedings of the IRE*, Vol. 40 (9), pp.1098-1101, 1952.

[7] Jaynes, E. T., *Information Theory and Statistical Mechanics, Phys. Rev.*,American Physical Society, Vol. 106 (4), pp 620-630, 1957.

[8] Kieffer, J. C., Yang, E.-H., *Grammar-based codes: a new class of universal lossless source codes, IEEE Trans. Inform. Theory 46*, no. 3, 737-754, 2000.

[9] Larsson, N. Jesper, Moffat, Alistair, *Offline dictionary-based compression. Proc. IEEE*,IEEE Computer Society, pp. 296-305, 1999.

[10] Mirzal, A., Furukawa, M., *Node-Context Network Clustering using PARAFAC Tensor Decomposition. CoRR* Vol. abs/1005.0268, 2010

[11] Nielsen, Michael A., Chuang, Isaac L., *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[12] Salomon, David, *Data Compression: The Complete Reference*, Springer, NewYork, 2004.

[13] Shannon, Claude E., *A mathematical theory of communication* Bell System Technical Journal, vol. 27, pp. 379–423 and 623–656, 1948.

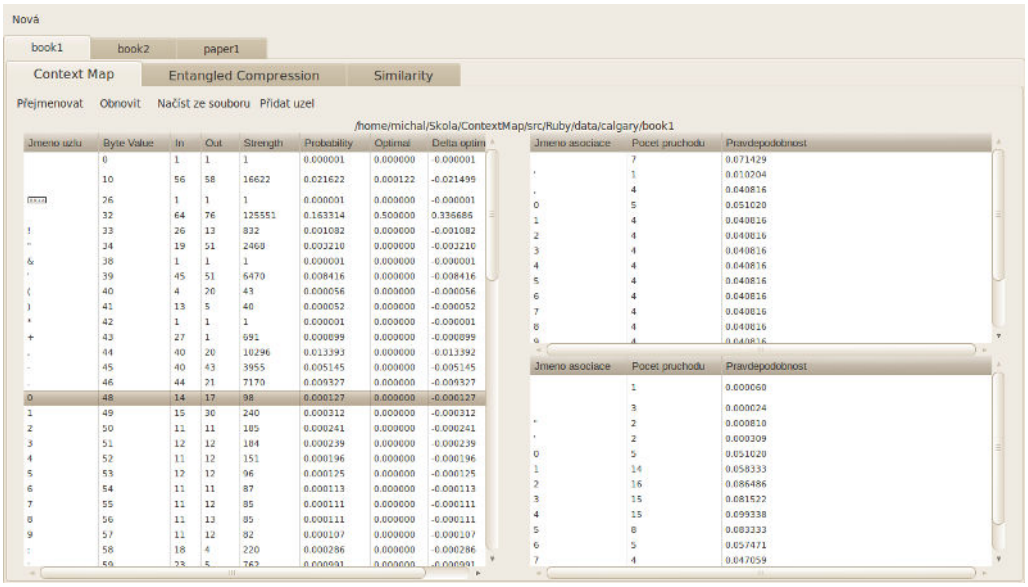
[14] Stonier, Tom, *Informace a vnitřní struktura vesmíru*, BEN-Technická literatura, 2002.

[15] Vašinek, Michal, *Komprese dat pomocí kontextové mapy*, VŠB-TU Ostrava, 2011.

[16] Wan, Kaiyu, *A brief history of context. International Journal of Computer Science Issues*, IJCSI, Vol. 6 (2), pp. 33-43, 2009.

A Context Map Analyser

Program Context Map Analyser slouží pro získávání informací o kontextové mapě a provádění transformačních algoritmů. CMA umožňuje vytvářet pro různé soubory oddělené kontextové mapy. Na obrázku 4 je přehled jednotlivých prvků hlavní obrazovky.



Obrázek 4: Uživatelské rozhraní CMA - kontextová mapa.

V levém horním rohu je tlačítko ”Nová”, po jehož stisku dojde k vytvoření nové prázdné kontextové mapy. V liště záložek pod tlačítkem ”Nová”se nachází přehled všech otevřených kontextových map. Každá záložka je pojmenována názvem souboru, ze kterého byla kontextová mapa vytvořena.

Obsluha kontextové mapy je rozdělena do tří funkčních záložek:

- Context Map - stránka pro zobrazení uzlů a vazeb vedoucích z označeného uzlu.
- Entangled Compression - stránka pro práci s transformačními algoritmy.
- Similarity - stránka pro práci s algoritmy pro podobnost mezi uzly kontextové mapy.

A.1 Stránka Context Map

Na stránce Context Map jsou čtyři funkční tlačítka:

- Přejmenovat - slouží pro vyvolání nabídky pro přejmenování kontextové mapy.
- Obnovit - slouží pro obnovení seznamu uzlů z kontextové mapy.
- Načíst ze souboru - slouží pro výběr souboru, který bude načten do kontextové mapy.

- Přidat uzel - slouží pro vyvolání nabídky pro přidání prázdného uzlu do kontextové mapy.

Samotný seznam uzlů(levá část), obsahuje několik sloupců:

- Jméno uzlu - ASCII znak reprezentující uzel.
- Byte Value - hodnota od 0 do 256, pro případy netisknutelných znaků.
- In - počet vazeb vstupujících do uzlu.
- Out - počet vazeb vystupujících z uzlu.
- Strength - počet výskytů uzlu v textu.
- Probability - pravděpodbnost výskytu uzlu.
- Optimal - momentálně není využíváno, mělo by určovat optimální rozdělení dat.
- Delta optimal - odchylka od optimálního rozdělení.

Na pravo od seznamu uzlů se nachází dva seznamy vazeb, seznam výše je seznamem vazeb z uzlu vystupujících, zatímco níže je uveden seznam vazeb do uzlu vstupujících. Každé vazbě je přiřazen ASCII kód, je zde uveden počet průchodů danou vazbou a pravděpodobnost průchodu vazbou.

A.2 Stránka Entangled Compression

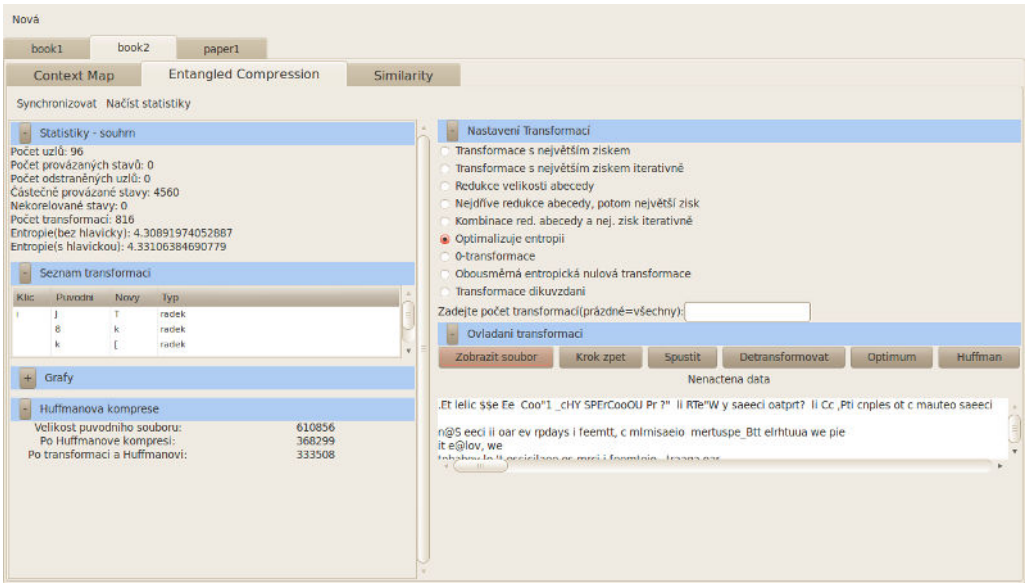
Vzhled stránky Entangled Compression je uveden na obrázku **??**. Účelem stránky je zpřístupnit volbu a ovládání algoritmů, poskytování informací o relacích provázanosti, zpřístupnit provedené transformace, poskytnout informace o výsledcích Huffmanovy komprese a případně zpřístupnit podobu transformovaného souboru.

Stránka obsahuje pouze v hlavní liště dvě funkční tlačítka:

- Synchronizovat - není momentálně používáno.
- Načíst statistiky - načte statistiky do panelu ”Statistiky - souhrn”

Panel ”Statistiky - souhrn”nabízí informace o počtu relací provázanosti v kontextové mapě.

- Počet uzlů - počet uzlů v kontextové mapě.
- Počet provázaných stavů - počet výskytů relace globálního provázání.
- Počet odstraněných uzlů - počet uzlů, které neobsahují žádné vstupní, ani výstupní vazby.
- Částečně korelované stavy - počet částečně korelovaných uzlů.



Obrázek 5: Uživatelské rozhraní CMA - správá algoritmů.

- Nekorelované stavy - počet výskytu relace neprovázanosti.
- Počet transformací - udává počet provedených transformací.
- Entropie(bez hlavičky) - udává entropii transformovaného souboru bez započítání hlavičky s transformacemi.
- Entropie(s hlavičkou) - udává entropii transformovaného souboru se započítánou hlavičkou s transformacemi.

Panel ”Seznam transformací”ukazuje přehled provedených transformací, na vrchu seznamu je vždy poslední provedená transformace. V seznamu jsou obsaženy ASCII hodnoty zpracovávané trojice, plus typ transformace, zda se jedná o řádkovou transformaci, nebo o sloupcovou.

Panel ”Huffmanova komprese”informuje o velikosti původního souboru, o velikosti souboru po převodu na Huffmanův kód a velikost souboru po aplikování tranformačního algoritmu a následném převodu na Huffmanův kód.

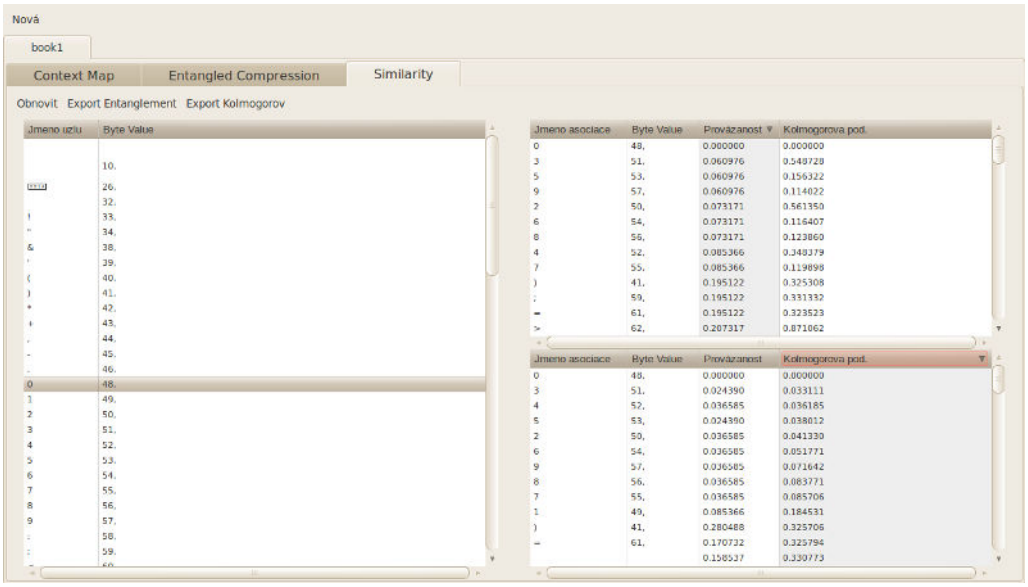
Panel nastavení transformací nabízí možnost zvolit algoritmus pro vyhledání a umožňuje také zadat limitní počet nalezených transformací. Oproti algoritmům představeným v diplomové práci umožňuje volbu ještě dalších dvou algoritmů, které již provádějí obousměrné transformace(řádkové i sloupcové), jedná se algoritmy ”Obousměrná entropická nulová transformace”a ”Transformace díkuvzdání”.

Panel ”Ovládání transformací”umožňuje nechat si zobrazit transformovaný text v komponentě v pravém dolním rohu stránky, umožňuje vracet transformace po kroku, spuštění transformace podle zvoleného algoritmu, zpětnou transformaci, nalezení optimál-

ního počtu transformací(podporuje pouze některé algoritmy) a tlačítko ”Huffman”slouží pro zobrazení hodnot do panelu ”Huffmanova komprese”.

A.3 Stránka Similarity

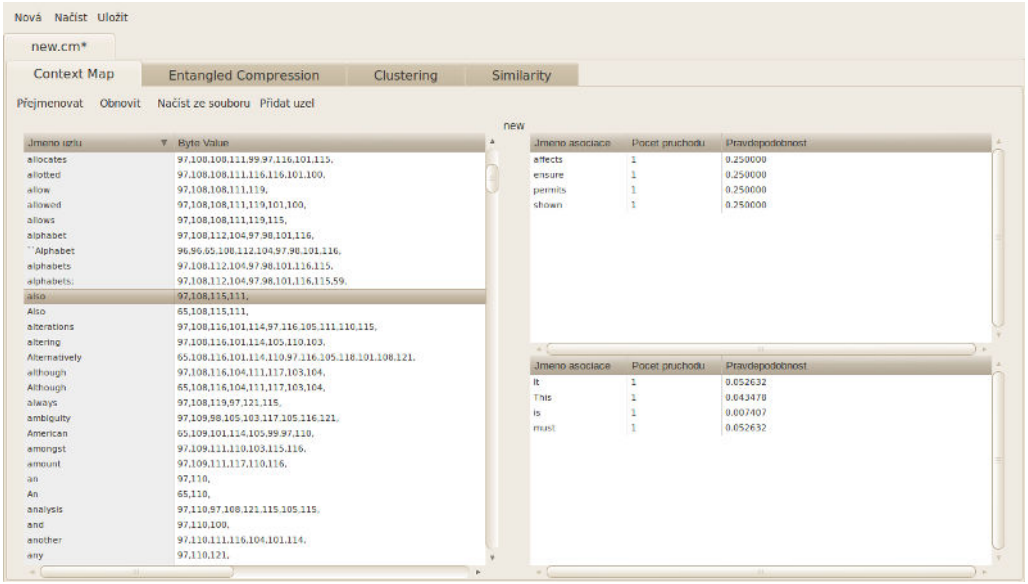
Stránka similarity slouží pro získání informací o podobnostech mezi dvěma uzly. Podoba stránky je uvedena na obrázku 6. Zobrazované seznamy obsahují totožné uzly a vazby z uzlů, jako na stránce ”Context Map”, jedinou modifikací jsou hodnoty uvedené ve sloupcích seznamů. V seznamu asociací je tak dostupná informace o podobnosti mezi uzly na základě relací lokálního provázání a na základě Kolmogorovy vzdálenosti.



Obrázek 6: Uživatelské rozhraní CMA - podobnosti uzlů.

A.4 Verze 2. knihovny ContextMap

Ve verzi dvě jsem provedl úpravu knihoven, tak aby bylo možné načítat slova různých délek, tato verze je určena pouze pro posouzení podobnosti mezi uzly kontextové mapy a stránka algoritmy v ní nefunguje. Podoba druhé verze knihovny je uvedena na obrázku 7.



Obrázek 7: Uživatelské rozhraní CMA - kontextová mapa verze 2.

B CD s implementací algoritmů