دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر و فناوری اطلاعات

گزارش سمینار کارشناسی ارشد رشته مهندسی کامپیوتر (M.S.c)

گرایش نرم‌افزار

عنوان سمینار

بهبود نسبت فشرده‌سازی متن برای متن اسکی (بررسی و مرور)

استاد راهنما

دکتر سید علی رضوی ابراهیمی

نگارنده

مریم سادات موردگر

شهریور ۱۴۰۰

فهرست مطالب

فهرست جداول

فهرست شکل‌ها

فهرست علائم اختصاری

ASCII	American Standard Code for Information	استاندارد آمریکایی برای تبادل اطلاعات
TCS	Tool-less Compression System	یک روش فشرده‌سازی ترکیبی
ACM	ASCII Compression Module	ماژول فشرده‌سازی اسکی
PNG	Portable network graphics	گرافیک‌های قابل‌حمل در شبکه
Re-pair	Recursive Pairing	زوج بازگشتی
UTF-8	eight bit Unicode Transformation Format	فرمت تبدیل یونیکد ۸ بیتی
XML	Extensible Markup Language	زبان نشانه‌گذاری گسترش‌پذیر
JSON	JavaScript Object Notation	نشانه‌گذاری شی جاوا اسکریپت

چکیده

فشرده‌سازی داده‌ها زمینه‌ای است که به‌طور گسترده موردتحقیق قرارگرفته است. بیشتر الگوریتم‌های فشرده‌سازی که امروزه مورداستفاده قرار می‌گیرند، از چندین دهه پیش است، مانند *کدگذاری هافمن*[1] و *کدگذاری دیکشنری*[2] که همه‌منظوره هستند. الگوریتم‌های فشرده‌سازی می‌تواند در هر چیزی از داده‌های متنی گرفته تا تصاویر و ویدئو استفاده شود. بااین‌حال، الگوریتم‌های بسیار کمی برای فشرده‌سازی انواع خاصی از داده‌ها، مانند متن اسکی[3] وجود دارد که بدون اتلاف داده باشند. این پروژه در مورد ایجاد یک‌راه‌حل فشرده‌سازی متن با استفاده از ترکیبی از سه مورد است الگوریتم فشرده‌سازی کدگذاری دیکشنری، الگوریتم فشرده‌سازی اسکی و الگوریتم فشرده‌سازی کدگذاری هافمن.

این راه‌حل مخصوص فشرده‌سازی متن اسکی است، اما می‌تواند در هر شکلی از متن استفاده گردد. الگوریتم‌ها برای ایجاد یک نمونه اولیه ترکیب می‌شوند که بر اساس برنامه‌های فشرده‌سازی عمومی مورد ارزیابی قرار می‌گیرد. یک ارزیابی در برابر فشرده‌سازی اسکی از برنامه شوکو[4] انجام خواهد شد. نتایج حاصل از ارزیابی نشان می‌دهد که ترکیب کدگذاری دیکشنری، فشرده‌سازی اسکی و کدینگ هافمن از نسبت فشرده‌سازی به‌دست‌آمده از برنامه‌های فشرده‌سازی عمومی بهتر نمی‌باشد. (Haldar,2020)

واژگان کلیدی

فشرده‌سازی متن، کدگذاری دیکشنری، اسکی، کدینگ هافمن

[1] Huffman coding
[2] Dictionary coding
[3] Ascii
[4] Program Shoco

فصل اول

مقدمه

فشرده‌سازی داده‌ها فرایند کاهش تعداد بیت‌های موردنیاز برای نمایش رسانه است (Sayood,2006). به‌عبارت‌دیگر، کاهش حجم داده‌ها در حفظ تمامیت آن. فشرده‌سازی داده‌ها می‌تواند در هنگام پخش ویدئو از فضای ذخیره‌سازی، ذخیره کرده یا پهنای باند را کاهش دهد (Salomon, 2004). زیرمجموعه فشرده‌سازی داده‌ها فشرده‌سازی متن است. فشرده‌سازی متن فشرده‌سازی فایل‌های متنی است که فایل‌هایی هستند که شامل کاراکترها و نمادها می‌باشد. یک تفاوت مهم بین فشرده‌سازی متن و داده‌های باینری، مانند تصاویر و ویدئو، این است که فشرده‌سازی متن باید بدون اتلاف[1]باشد. فشرده‌سازی بدون اتلاف به این معنی است که یک فایل را می تون فشرده کرد به‌طوری‌که فایل اصلی بدون از دست دادن اطلاعات بازیابی شود (Sayood, 2006). الگوریتم‌های فشرده‌سازی که قادر به فشرده‌سازی داده‌ها نیستند از فشرده‌سازی اتلافی استفاده می‌کنند. فشرده‌سازی با اتلاف[2]بیشتر در داده‌های باینری استفاده می‌شود که در آن برخی از داده‌ها از بین می‌روند (Shukla and Prasad, 2011).

۱-۱ تاریخچه

فشرده‌سازی داده‌ها توسط الگوریتم‌های فشرده‌سازی انجام می‌شود. چندین الگوریتم متداول در برنامه‌های فشرده‌سازی استفاده می‌شود، و بسیاری از آن‌ها چندین دهه است که وجود دارند. الگوریتم متداولی که هنوز مورداستفاده قرار می‌گیرد، کدگذاری هافمن است که در سال ۱۹۵۲ اختراع شد (Huffman, 1952). کدگذاری هافمن کاراکترهای پرکاربردتر را با رشته‌های بیتی کوتاه‌تری نسبت به آن‌هایی که کاربردشان کمتر است، نشان داده می‌شوند. بنابراین یک متن انگلیسی ممکن است فشرده‌سازی خوبی به دست آورد، به‌عنوان‌مثال، حرف"e" بیشتر از حرف "z" استفاده می‌شود (Trost, 2020). یکی دیگر از الگوریتم‌ها، کدگذاری دیکشنری است که تکنیک جایگزینی زیررشته‌ها و کلمات با ارجاع به دیکشنری است. دیکشنری و منابع، فایل فشرده را تشکیل می‌دهند که می‌توان برای بازیابی فایل اصلی آن‌ها را از حالت فشرده خارج کرد. LZ77 و LZW از پرکاربردترین الگوریتم‌های

[1] Lossless data compression
[2] Lossy compression

کدگذاری دیکشنری هستند . هر دو الگوریتم مبتنی بر تکنیک جایگزینی زیررشته‌های تکراری با منابع هستند، اما از رویکردهای متفاوتی استفاده می‌کنند.

الگوریتم DEFLATE ترکیبی از کدگذاری دیکشنری (LZ77) (Ziv and Lempel, 1977) و کدگذاری هافمن است. DEFLATE در چندین برنامه فشرده‌سازی مانند Gzip، zip و WinRar استفاده می‌شود . باوجوداینکه درزمینه فشرده‌سازی داده‌ها تحقیقات گسترده‌ای انجام‌شده است، فناوری‌های جدید همچنان در حال ظهور هستند (Farina, et.al., 2012)، مانند شوکو ، که در سال ۲۰۱۴ منتشر شد. شوکو یک برنامه فشرده‌سازی متن است که به‌ویژه‌بر روی متن ASCII مؤثر است. ASCII یک کدگذاری برای متن ساده است؛ یعنی متنی که از تنوع زیادی از کاراکترها و نمادها استفاده نمی‌کند. رمزگذاری کاراکتر یک متن به کامپیوتر می‌گوید چگونه داده‌ها باید خوانده شوند تا انسان بتواند آن را درک کند (Indurkhya, et.al., 2010).

۲-۱ تعریف مسئله و بیان سؤالات اصلی تحقیق

انتقال حجم زیادی از داده‌ها از طریق اینترنت کار زمان بری است. حتی در شرایطی که یک کاربر دارای پهنای باند فوق‌العاده باشد، انتقال مگابایت یا گیگابایت داده بازهم چند دقیقه طول می‌کشد. این مشکل اغلب مربوط به فایل‌های باینری مانند فیلم‌ها است، اما ممکن است در مورد داده‌های مبتنی برمتن نیز صادق باشد. یک مثال می‌تواند شبیه‌سازی مخزن گیت باشد، هسته لینوکس مخزن گیت هاب نزدیک به سه گیگابایت حجم دارد. داده‌های متنی را می‌توان با استفاده از الگوریتم‌هایی مانند DEFLATE فشرده کرد، که توسط چندین برنامه فشرده‌سازی محبوب استفاده می‌شود. فشرده‌سازی داده‌ها حجم فایل را کاهش می‌دهد، پس انتقال آن از طریق اینترنت سریع‌تر خواهد بود الگوریتم DEFLATE برای کاربر روی انواع متن‌ها و داده‌های باینری طراحی‌شده است (Sayood, 2006). بااین‌حال یک الگوریتم فشرده‌سازی عمومی است که کار با آن می‌تواند بهینه شود. به‌عنوان‌مثال متن ASCII اکثر برنامه‌های فشرده‌سازی عمومی هستند و از الگوریتم‌های بدون اتلاف استفاده می‌کنند که به فشرده‌سازی هر نوع داده مانند متن، تصویر فایل‌های ویدئویی منجر می‌شود. چندین الگوریتم فشرده‌سازی اتلافی وجود دارد که به‌طور خاص برای هر نوع داده استفاده می‌شوند، اما تعداد آن‌ها کمتر از الگوریتم‌های بدون اتلاف است. این امکان وجود دارد که از الگوریتم‌های بدون اتلاف سفارشی برای فشرده‌سازی انواع داده‌های خاص، مانند متنASCII ، نسبت به الگوریتم‌های فشرده‌سازی عمومی، بیشتر استفاده شود.

پایان‌نامه به سؤالات تحقیق زیر پاسخ می‌دهد:

- چه ترکیبی از تکنیک‌ها می‌تواند نسبت فشرده‌سازی را بهبود بخشد؟
- آیا استفاده از ترکیب کدگذاری دیکشنری و کدگذاری هافمن برای فشرده‌سازی ASCII، نسبت به فشرده‌سازی DEFLATE فراتر می‌رود؟
- آیا این ترکیب فشرده‌سازی متن، فشرده‌سازی بالاتری نسبت به برنامه‌های فشرده‌سازی عمومی برای متن ASCII به دست می‌آورد؟
- آیا این ترکیب فشرده‌سازی متن، فشرده‌سازی بالاتری نسبت به برنامه‌های فشرده‌سازی که مخصوص متن ASCII هستند، به دست می‌آورد؟

۱-۳ انگیزه و ضرورت تحقیق

راه‌حل فشرده‌سازی متن پیشنهادی (که در اینجا به آن اشاره می‌شود TCSیا "ترکیبی") می‌تواند برای هرکسی که با مقدار زیادی (عمدتاً) متن اسکی سروکار دارد مفید باشد. این راه‌حل به‌گونه‌ای طراحی‌شده است که روی هر فرم متن کار کند، اما برای متن اسکی سفارشی‌شده است. این راه‌حل حجم فایل را کاهش می‌دهد تا فضای زیادی را اشغال نکند و همچنین هنگام بارگیری یا انتقال یک فایل فشرده از طریق اینترنت، پهنای باند را کاهش می‌دهد. کدگذاری دیکشنری (Langiu, 2013) و کدگذاری هافمن از قبل به‌طور گسترده‌ای در الگوریتم‌های فشرده‌سازی استفاده می‌شود (هر دو در DEFLATE)، اما الگوریتم‌های فشرده‌سازی کمی از روش فشرده‌سازی اسکی استفاده می‌کنند. بنابراین این ترکیب ممکن است فشرده‌سازی بهتری برای متن اسکی نسبت به الگوریتم‌های فشرده‌سازی عمومی باشد.

۱-۴ محدودیت‌ها

این پروژه به‌طور خاص برای فشرده‌سازی متن، و نه داده‌های باینری، مانند فایل‌های تصویری یا ویدیویی در نظر گرفته‌شده است. بنابراین آزمایش‌های انجام‌شده در این پروژه تنها روی‌داده‌های متنی می‌باشد. انتظار می‌رود که از ترکیب برای فشرده‌سازی مقادیر زیادی از متن استفاده شود، زیرا داده‌های متنی معمولاً فضای ذخیره‌سازی زیادی را در مقایسه با داده‌های ویدئویی یا یک فایل متنی کوچک که کاربرد چندانی ندارد؛ اشغال نمی‌کند. بنابراین فقط روی فایل‌های متنی با اندازه قابل‌توجه آزمایش می‌شود. اثربخشی راه‌حل (تأخیر) عاملی در ارزیابی نیست، زیرا

ترکیب فقط بر اساس نسبت فشرده‌سازی آن ارزیابی می‌شود، نه زمان اجرا. پایتون به‌عنوان زبان برنامه‌نویسی برای ماژول فشرده‌سازی ASCII (ACM) انتخاب‌شده است تا کد منبع را ساده‌تر کند. پیاده‌سازی برنامه در C می‌توانست راه‌حل را سریع‌تر کند (Li Jun and Li Ling, 2010)، اما قابلیت فشرده‌سازی برای ACM اولویت بالاتری نسبت به تأخیر دارد (Salib, 2004). این پروژه پیاده‌سازی‌های مختلفی را برای استفاده در TCS ارزیابی کرد. پیاده‌سازی‌ها بر اساس میانگین نتایج حاصل از آزمایشات ارزیابی می‌شوند. این بدان معناست که موارد استفاده در آزمایش اهمیت برابر دارند. این پروژه با هیچ معضل اخلاقی مواجه نمی‌شود. این پروژه همکاری با یک شرکت نیست، بنابراین هیچ مشکلی با کپی‌رایت (حق چاپ) آن وجود نخواهد داشت. این پروژه تنها از کد منبع با مجوز باز استفاده می‌کند. آزمایشات روی فایل‌های ویکی‌پدیا فقط از یک زیرمجموعه از متن برای آزمایش فشرده‌سازی آن‌ها استفاده می‌کند، زیرا استفاده از چنین مجموعه داده بزرگی برای آزمایش غیرممکن است. زیرمجموعه به‌اندازه کافی بزرگ است که نتایج واضحی را ارائه می‌دهد.

۵-۱ پیشنهادها و هدف

اکثر برنامه‌های فشرده‌سازی از الگوریتم DEFLATE استفاده می‌کنند، که شامل کدگذاری دیکشنری و کدگذاری هافمن است. در این پروژه هم از این موارد استفاده خواهد شد زیرا مؤثر بودن این الگوریتم‌های استاندارد عملاً اثبات‌شده است (Fowers, et.al., 2015). الگوریتم‌های زیر در این پروژه استفاده می‌شود:

- کدگذاری دیکشنری؛ به‌ویژه در متن نوشتاری[۱] مؤثر است، مانند دایرةالمعارف (Shi and Sun, 1999).
- فشرده‌سازی ASCII؛ بر روی متن نوشتاری، به‌ویژه جایی که از کاراکترهای خاص بیشتر از بقیه استفاده می‌شود، مؤثر است (Trost, 2020).
- کدگذاری هافمن؛ تکنیکی که به‌طور گسترده در فشرده‌سازی داده‌ها استفاده‌شده است (Aggarwal, Narayan, 2000) و (Klein, 2003).

این پروژه با استفاده از این سه روش، راه‌حلی را ایجاد می‌کند که برای فشرده‌سازی متن اسکی مطلوب است و سعی می‌کند نسبت فشرده‌سازی بیشتری برای متن اسکی نسبت به یک الگوریتم فشرده‌سازی عمومی داشته باشد.

[۱] " متن نوشتاری" در پایان نامه مورد بررسی به عنوان متنی که به برخی از زبان ها مانند یک زبان گفتاری یا یک زبان برنامه نویسی نوشته شده است تعریف شده است.

۱-۶ روش و مراحل انجام تحقیق

تحقیقات آماری که با داده‌های عددی سروکار دارند، کم هستند، درحالی‌که تحقیقات کیفی، تحقیقاتی و معمولاً غیر آماری است. آزمایشات برای سیستم هنگام مقایسه TCS با داده‌های عددی استفاده می‌شود، بنابراین تحقیقات کمی اندازه‌گیری‌های بهتری را در مقایسه با تحقیقات کیفی ارائه می‌دهد (Merriam, 2020). در مقابل آن برای نتیجه‌گیری از آزمایشات در تحقیقات کیفی به استدلال منطقی نیاز است که این استدلال می‌تواند استدلال قیاسی یا استدلال استقرایی باشد. استدلال قیاسی تشکیل یک نتیجه‌گیری بر اساس اظهارات یا حقایق عمومی پذیرفته‌شده است. استدلال استقرایی از منطق زیر پیروی می‌کند: اگر A = B و B = C سپس A = C . استدلال استقرایی شامل یک عنصر احتمال است و ممکن است به این نتیجه برسد که A احتمالاً برابر با B است (Merriam, 2020).

در پروژه موردبررسی تعدادی از تکنیک‌های فشرده‌سازی بر روی متون مشابه آزمایش می‌شوند، و نسبت فشرده‌سازی برای پیدا کردن تکنیک بیشترین فشرده‌سازی را به دست می‌آورد. آزمایشات داده‌های قطعی در مورد این‌که کدام راه‌حل بهترین نسبت فشرده‌سازی را به دست می‌آورد، ارائه می‌دهد و بنابراین نتیجه‌گیری با استدلال استقرایی یا قیاسی صورت می‌گیرد.

۱-۷ طرح کلی پایان‌نامه موردبررسی

فصل دوم این پایان‌نامه توضیح می‌دهد که فشرده‌سازی متن چیست و سابقه و کار مرتبط برای سه ماژول TCS را ارائه می‌دهد. کدگذاری دیکشنری، فشرده‌سازی اسکی و کدگذاری هافمن. فصل سوم روش‌های مختلفی که در تحقیقات دانشگاهی برای این پروژه استفاده می‌شود، و دلیل انتخاب این روش‌ها را، ارائه می‌دهد. فصل چهارم طراحی TCS و الزامات راه‌حل را ارائه می‌دهد. فصل پنجم فرایند ارزیابی TCS و ماژول‌های آن را شرح می‌دهد و نتایج حاصل از ارزیابی‌ها را ارائه می‌دهد. فصل ششم بحث نتایج حاصل از ارزیابی‌ها را پوشش می‌دهد و فصل هفتم نتیجه‌گیری و کار آینده پایان‌نامه را ارائه می‌دهد.

۱-۸ ساختار گزارش تحقیق

فصل اول به تعریف و مقدمه و دلایل نیاز به طرح ارائه‌شده پرداخته می‌شود.

فصل دوم به مفاهیم عمومی فشرده‌سازی پرداخته می‌شود.

فصل سوم مروری است بر کارهای انجام‌شده طرح پیشنهادی پایان‌نامه

فصل چهارم به ارزیابی و پیاده‌سازی الگوریتم‌ها و بررسی نتایج پرداخته می‌شود.

فصل پنجم به جمع‌بندی و نتیجه‌گیری پرداخته می‌شود.

فصل دوم

مفهوم فشرده‌سازی داده‌ها و برخی از انواع آن

۲-۱ مقدمه

فشرده‌سازی داده‌ها فرایند کاهش تعداد بیت‌های موردنیاز برای نمایش رسانه است (Sayood, 2006). این بدان معناست که یک فایل فشرده کوچک‌تر از فایل اصلی و می‌تواند به‌عنوان‌مثال در مدت‌زمان کوتاه‌تری از طریق اینترنت منتقل شود. دو نوع تکنیک فشرده‌سازی داده‌ها وجود دارد: اتلافی و بدون اتلاف. فشرده‌سازی بدون اتلاف به این معنی است که می‌توان تمام داده‌های هنگام فشرده‌سازی را، بازیابی کرد. این بدان معناست که پس از بازیابی فایل فشرده‌شده با فایل اولیه کاملاً یکسان خواهد بود و بدون اتلاف می‌باشد. از طرف دیگر، فشرده‌سازی ازدست‌رفته (اتلافی)، برخی از داده‌ها را در حین فرایند فشرده‌سازی حذف می‌کند (Shukla and Prasad, 2011). هنگامی‌که فایل از حالت فشرده خارج می‌شود، داده‌های ازدست‌رفته قابل بازیابی نیستند. فشرده‌سازی اتلافی بیشتر در مورد داده‌های دودویی استفاده می‌شود. فشرده‌سازی اتلافی برای داده‌های دودویی در ترکیب با فشرده‌سازی بدون اتلاف به کار می‌رود (Shukla and Prasad, 2011). به‌عنوان‌مثال فایل‌های صوتی، تصویری و ویدئویی (Sayood, 2006). مزیت فشرده‌سازی اتلافی این است که معمولاً نسبت فشرده‌سازی بسیار بالاتری نسبت به فشرده‌سازی بدون اتلاف به دست می‌آورد، درحالی‌که کاربران هنوز می‌توانند اطلاعات داده‌های فشرده را درک کنند (Shukla and Prasad, 2011). اغلب، داده‌های ازدست‌رفته در فشرده‌سازی برای کاربر قابل‌توجه نیست (Kleiman and Red, 2007). کدگذاری هافمن و کدگذاری دیکشنری نمونه‌هایی از فشرده‌سازی بدون اتلاف هستند. این الگوریتم‌ها در بسیاری از برنامه‌های فشرده‌سازی مانند Gzip، Zip یا Winrar استفاده می‌شوند، آنچه برای این الگوریتم‌ها متداول است این است که آن‌ها بالاترین نسبت فشرده‌سازی را از فایل‌هایی که داده‌ها را تکرار می‌کنند یا توزیع ناهموار کاراکترها / نمادها رادارند، دریافت می‌کنند. این بدان معناست که داده‌های تصادفی یا یک متن "گیج‌کننده" ممکن است نسبت فشرده‌سازی کمی داشته باشد. نسبت فشرده‌سازی اندازه‌گیری میزان حجم فایل فشرده‌شده در مقایسه با فایل اصلی است (Togneri and Desilva, 2002). به‌عنوان‌مثال، این الگوریتم‌ها هنگام فشرده‌سازی متن نوشته‌شده به برخی از زبان‌ها مانند انگلیسی، آلمانی یا زبان برنامه‌نویسی مؤثر هستند؛ به این دلیل که برخی از کلمات، عبارات و کاراکترها بیشتر از بقیه استفاده می‌شوند (Togneri and Desilva, 2002). این

الگوریتم‌ها فقط در متن استفاده نمی‌شوند. فرمت PNG از این تکنیک‌ها برای یافتن مطابقت و الگوهای موجود در تصاویر استفاده می‌کند.

۲-۲ کدگذاری دیکشنری

کدگذاری یک تکنیک فشرده‌سازی است که رشته‌های تکراری متن را با منابع جایگزین می‌کند. درحالی‌که کدگذاری هافمن به کاراکترهای فردی و میزان استفاده آن‌ها در متن نگاه می‌کند، کدگذاری دیکشنری از رشته‌های تکراری استفاده می‌کند (Shi and Sun, 1999). به‌عنوان‌مثال، اگر متنی چندین بار حاوی کلمه "فشرده‌سازی" است، موارد تکراری کلمه را می‌توان با اشاره به کلمه در دیکشنری جایگزین کرد. هنگامی‌که متن از حالت فشرده خارج می‌شود، الگوریتم از مرجع برای جستجوی کلمه استفاده می‌کند و مرجع را با کلمه اصلی جایگزین می‌کند. دیکشنری لیستی از رشته‌های متنی است که اغلب مورداستفاده قرار می‌گیرد (Huffman, 1997). برخی از رمزگذارهای دیکشنری، مانندLZ77 از یک دیکشنری صریح استفاده نمی‌کنند (Ziv and Lempel, 1977)، اما در عوض از ارجاعات انجام‌شده قبلی در متن استفاده می‌کند (Hankerson, et. Al., 2003). رمزگذاری‌های دیکشنری نیز می‌توانند در داده‌های غیر متنی مانند تصاویر استفاده شوند. فرمت تصویر PNG از کدگذاری دیکشنری LZ77 استفاده می‌کند. در مورد تصاویر، الگوریتم باید دنباله‌های پیکسلی را به‌جای رشته‌های متنی تکرار کند.

مزیت استفاده از کدگذار دیکشنری این است که می‌تواند نسبت فشرده‌سازی بسیار بالایی را برای فایل‌های خاص به دست آورد. کدگذاری‌های دیکشنری به‌ویژه‌بر روی‌داده‌های تکراری، مانند متن نوشتاری مؤثر هستند (Shi and Sun, 1999). متن، که به زبان نوشته‌شده است، معمولاً برخی از کلمات یا ترکیب حروف را بیشتر از بقیه تکرار می‌کند که برنامه نویسان دیکشنری از این مزیت استفاده می‌کنند. عیب استفاده از برخی کدگذارهای دیکشنری، مانندLZW ، این است که فایل فشرده ممکن است بزرگ‌تر از فایل فشرده نشده باشد (Senthil and Robert, ۲۰۱۱) و (Taleb, et.al., 2010). این اتفاق در صورتی می‌افتد که داده‌هایی که باید فشرده شوند هیچ اطلاعات تکراری نداشته باشند. اگر فشرده‌سازی یک فایل باعث کاهش حجم آن نشود، پس هدف شکست‌خورده است.

۲-۲-۱ الگوریتم‌های رایج کدگذاری دیکشنری

این بخش ویژگی‌ها و تفاوت‌های اصلی بین الگوریتم‌های کدگذاری دیکشنری را توصیف می‌کند. ویژگی‌های یک الگوریتم می‌تواند راه‌حل فنی آن یا چیزی باشد که آن را از بقیه متمایز کند. یک الگوریتم همچنین می‌تواند به این معنا باشد که کمابیش برای کارهای فشرده‌سازی مناسب است.

الگوریتم LZ77

LZ77 یک کدگذار دیکشنری محبوب است که در بین سایر موارد ،در فرمت تصویرPNG و الگوریتم DEFLATE استفاده می‌شود. الگوریتم LZ77 هنگام فشرده‌سازی داده‌ها از پنجره *لغزان*[1] (Ziv and Lempel, 1977) استفاده می‌کند (به‌عنوان‌مثال یک فایل تصویری یا یک فایل متنی). sliding window یک بافر است که میزان متن (یا داده) را که الگوریتم در هر نقطه تجزیه‌وتحلیل می‌کند تا زیررشته‌های منطبق را پیدا کند را تعیین می‌کند. پنجره لغزان شامل یک بافر *پیش‌بینی* و یک بافر *جستجو است*. بافر پیش‌بینی، متنی را که هنوز فشرده یا کدگذاری نشده است، تجزیه‌وتحلیل می‌کند، به‌عنوان‌مثال ۲۰ کاراکتر بعدی، درحالی‌که بافر جستجو متنی است که اخیراً کدگذاری شده است و معمولاً بسیار بزرگ‌تر از بافر پیش‌بینی است. اگر یک زیررشته در بافر جستجو باشد، نیز در بافر پیش رو ظاهر شود، سپس زیررشته در بافر پیش رو با اشاره به بافر قبلی جایگزین می‌شود (Ziv and Lempel, 1977).

الگوریتم LZ78

درحالی‌که الگوریتم LZ77 با اشاره به یک نقطه قبلی در متن، زیررشته‌های مکرر ایجاد می‌کند، LZ78 از یک دیکشنری صریح استفاده می‌کند .الگوریتم LZ78 یک دیکشنری از زیررشته‌ها ایجاد می‌کند و زیررشته‌ها را در یک متن با اشاره به دیکشنری جایگزین می‌کند (Ziv and Lempel, 1978). الگوریتم LZ78 تجدیدنظر در الگوریتم LZ77 است، و به‌جز دیکشنری صریح در بقیه موارد مشابه‌اند (Atwal and Bansal, 2015).

الگوریتم LZW

الگوریتم LZW یک نوع LZ78 است و همچنین از یک دیکشنری صریح استفاده می‌کند (Hankerson, et.al., ۲۰۰۳). فرض اساسی LZW این است که فرایند کدگذاری را با ۲۵۶ ورودی‌ها/کاراکتر در دیکشنری شروع می‌کند (Taleb, 2010). این معمولاً مجموعه کاراکتر اسکی است، که از یک بایت در هر کاراکتر استفاده می‌کند. این الگوریتم دائماً زیررشته‌هایی را به دیکشنری با مرجع بالاتر از ۲۵۶ اضافه می‌کند. این ورودی‌ها ممکن است در متن تکرار شوند یا اینکه تکرار نشوند. اگر یک زیررشته تکرار شود، از همان اشاره به زیررشته استفاده می‌شود (Hankerson, et.al., 2003).

الگوریتم Re-pair

الگوریتم Re-pair از یک دیکشنری صریح استفاده می‌کند و شبیه الگوریتم LZ78 است. تفاوت Re-pair با LZ78 این است که Re-pair فقط زیررشته‌های دو رشته‌ای را ذخیره می‌کند (Ziv and Lempel, 1978). Re-pair همه

[1] sliding window

جفت‌های کاراکتر را که بیش از یک‌بار در یک متن وجود دارد، می‌یابد و با اشاره به دیکشنری آن‌ها را جایگزین می‌کند. این امر تا زمانی ادامه می‌یابد که هیچ جفت کاراکتری که بیش از یک‌بار در متن ظاهر شود، وجود نداشته باشد (Larsson and Moffat, 2020).

۲-۳ فشرده‌سازی ASCII

فشرده‌سازی ASCII در پایان‌نامه موردبررسی به‌عنوان الگوریتم‌های فشرده‌سازی تعریف‌شده است. به‌ویژه‌بر روی متن اسکی مؤثر است. کدگذاری اسکی دارای تغییرات زیادی برای زبان‌های مختلف است، اما پایان‌نامه موردبحث بر US-ASCII تمرکز می‌کند، مگر این‌که خلاف آن بیان شود. برای ارتباطات اینترنت US-ASCII به کدگذاری اسکی ارجحیت دارد. امروزه US-ASCII با ASCII کاربرد یکسانی دارد (Arregoces and Portolani, 2004) و (Carlsson and Miller, 2012). رمزگذاری ASCII یک کدگذاری متنی است که به‌طور گسترده برای متنی که کاراکترها و نمادهای خیلی متنوع ندارد، استفاده می‌شود (Arregoces and Portolani, 2004) و (Carlsson and Miller, 2012). این محدودیت در تعداد نمادها از نحوه ذخیره متن بر روی کامپیوتر نشأت می‌گیرد. یک نماد دقیقاً یک بایت (۸ بیت) برای ذخیره‌سازی استفاده می‌کند، که در آن ۷ بیت کد نماد و ۱ بیت "مهم‌ترین بیت" است. هدف از مهم‌ترین بیت می‌تواند به‌عنوان یک بررسی برابری، برای تشخیص خطاها در داده‌ها استفاده شود (Korpela, ۲۰۰۶). همچنین گروهی از رمزگذاری‌ها معروف به "اسکی توسعه‌یافته" از ۸ بیت برای نماد استفاده می‌کنند و درنتیجه مهم‌ترین بیت را کنار می‌گذارند. رمزگذاری‌هایی که از همه ۸ بیت استفاده می‌کنند، به‌جای ۱۲۸ نماد استاندارد از ۲۵۶ نماد استفاده می‌کنند (Ediger, 2008).

۲-۳-۱ الگوریتم شوکو (Shoco)

شوکو یک الگوریتم فشرده‌سازی متن است که توسط Christian Schramm توسعه‌یافته است، که به‌عنوان "کمپرسور سریع برای رشته‌های کوتاه" توصیف می‌شود و به‌ویژه‌بر روی متن اسکی مؤثر است. راه‌حل فشرده‌سازی شوکو از این واقعیت استفاده می‌کند که در هر زبان، برخی از کاراکترها بیشتر از بقیه استفاده می‌شود .همان‌طور که قبلاً ذکر شد، اولین بیت در یک کاراکتر اسکی اضافی است، مگر اینکه برای اهدافی مانند تشخیص خطا استفاده شود. بنابراین

الگوریتم از اولین بیت در یک بایت استفاده می‌کند تا نشان دهد که آیا بیت‌های زیر به یک کاراکتر مشترک اشاره می‌کنند یا خیر. اگر بیت اول روی ۰ تنظیم شود، بدین معنی که ۷ بیت زیر نشان‌دهنده یک کاراکتر غیرمعمول است؛ پس از فشرده شدن کاراکتر اسکی تغییر نخواهد کرد. اگر بیت اول روی ۱ تنظیم شود، بدین معنی که بیت‌های زیر دو کاراکتر مشترک را نشان می‌دهند. سپس دو کاراکتر با ۳ یا ۴ بیت نشان داده می‌شوند. همه باهم از ۸ بیت استفاده می‌کنند که به‌عنوان یک بایت ذخیره می‌شوند.

شوکو همچنین می‌تواند کاراکترهای غیر اسکی را فشرده کند، اما هزینه دارد. "اگر رشته ورودی شما به‌طور کامل (یا بیشتر) اسکی نباشد، خروجی ممکن است افزایش یابد". وقتی الگوریتم با کاراکتری مواجه می‌شود که بیش از ۷ بیت استفاده می‌کند (مانند یک کاراکتر UTF-8) استفاده می‌کند، درست قبل از کاراکتر "نشانگر" را وارد می‌کند. نشانگر یک کاراکتر خاص است که یک بایت را اشغال می‌کند. هدف آن این است که نشان دهد که کاراکتر بعدی اسکی نیست و بنابراین بیش از ۷ بیت استفاده خواهد کرد. نشانگر همچنین می‌گوید که کاراکتر بعدی از چند بایت استفاده می‌کند (به‌عنوان‌مثال ۱ یا ۲ بایت).

تکنیکی که شوکو از آن استفاده می‌کند شمارش تکرار (فراوانی) بیگرام در یک متن است. بیگرام ها، درزمینه شوکو، جفت‌های منحصربه‌فرد از دو کاراکتر متوالی در یک متن است. یک بیگرام رایج در انگلیسی "qu" است، زیرا بعد از "q" تقریباً همیشه "u" می‌آید. بعدازاینکه شوکو لیستی از متداول‌ترین کاراکترهای یک متن تهیه کرد، لیستی دیگر از کاراکترهایی که به‌احتمال‌زیاد به دنبال کاراکترهای رایج می‌آیند، تهیه می‌کند. اگر شوکو متوجه شد که به‌عنوان‌مثال "he" یک بیگرام رایج است، آنگاه می‌تواند تمام کلمات حاوی این بیگرام، مانند "the"، "she" و "then" را فشرده کند.

مزیت استفاده از شوکو این است که تا زمانی که ورودی صد درصد اسکی باشد، فایل فشرده هرگز بزرگ‌تر از فایل فشرده نشده نخواهد بود. علاوه بر این، شوکو را می‌توان به‌راحتی در ترکیب با سایر الگوریتم‌های فشرده‌سازی استفاده کرد، مانند الگوریتم‌های کدگذاری دیکشنری، یا کدگذاری هافمن برای دستیابی به فشرده‌سازی بیشتر. همچنین یک الگوریتم شوکو بسیار سریع است. به گزارش وب‌سایت، شوکو هنگام فشرده‌سازی یک فایل ۴,۹ مگابایتی تقریباً ۷ برابر سریع‌تر از Gzip است. نقطه‌ضعف شوکو این است که استفاده از شوکو به‌تنهایی نسبت فشرده‌سازی بسیار بدتری از برنامه‌های فشرده‌سازی استاندارد دارد. علاوه بر این، کاربر باید بداند که چه نوع متنی را فشرده می‌کند، اگر متن شامل مقدار زیادی از کاراکترهای چند بایتی، مانند UTF, 8 باشد، ممکن است فایل فشرده بزرگ‌تر از فایل فشرده نشده باشد، که هنگام استفاده از ابزار فشرده‌سازی بسیار نامطلوب است.

۲-۴ کدگذاری هافمن

کدگذاری هافمن به‌گونه‌ای طراحی‌شده است که پرکاربردترین کاراکترها را در یک متن تا حد امکان بیت، نشان دهد. اول، الگوریتم ورودی را با پیمایش متن کاراکتر به کاراکتر، تجزیه‌وتحلیل می‌کند. سپس الگوریتم لیستی از هر کاراکتر که در متن استفاده می‌شود، ایجاد کرده و لیست را بر اساس تکرارها مرتب می‌کند. پس از ایجاد لیست، الگوریتم یک درخت دودویی بر اساس جدول می‌سازد. درخت دودویی یک ساختار داده است که از "گره" و "شاخه" تشکیل‌شده است (Pai, 2008). هر گره می‌تواند حداکثر دو "گره فرزند" داشته باشد. در شکل ۱، گره‌ها مستطیل‌های آبی‌رنگ هستند. "گره‌های برگ" آخرین گره‌های یک شاخه هستند. یک درخت دودویی را به‌عنوان یک درخت در نظر بگیرید که وارونه است. همه شاخه‌های یک درخت از یک ریشه شروع می‌شوند، و برگ‌ها در انتهای شاخه‌ها هستند. هر گره برگ ارزش یک کاراکتر مورداستفاده در متن را دارد.

هنگامی‌که یک درخت هافمن ساخته می‌شود، فشرده‌سازی متن شروع می‌شود. این الگوریتم درخت را برای هر کاراکتر در یک متن پیمایش می‌کند. برای رسیدن به یک گره برگ، الگوریتم به سمت گره فرزند چپ یا راست حرکت می‌کند تا کاراکتر مشخص‌شده، پیدا می‌شود. این پیمایش به‌صورت دوتایی با یک و صفر نشان داده‌شده است، جایی که ۱ است، به معنای "رفتن به راست" و ۰ به معنای "رفتن به سمت چپ" است. به‌منظور فشرده‌سازی یک متن برای بازیابی داده‌های اصلی، درخت هافمن نیز باید ذخیره شود. به همین دلیل است که کدگذاری هافمن در متون بزرگ‌تر مؤثرتر است. به‌عنوان‌مثال وقتی یک متن بزرگ وجود دارد که فقط شامل ۲۶ حرف الفبای انگلیسی و چند علامت نقطه‌گذاری است، درخت صرف‌نظر از اندازه متن، بیش از ۲۶ گره برگ ندارد.

کدگذاری هافمن یک رمزگذار آنتروپی است، به این معنی که داده‌ها را بر اساس تکرار نماد، فشرده می‌کند؛ که این برخلاف کدگذاری دیکشنری است که به تکرارهای زیررشته‌ها یا توالی نمادها نگاه می‌کند. برنامه‌های آنتروپی مانند کدگذاری هافمن هنگامی سودمند است که از داده‌های خاص بیشتر از بقیه نمادها استفاده شود. کدگذاری هافمن می‌تواند به‌آسانی هم درداده متنی و هم در داده‌های دودویی مورداستفاده قرار گیرد و می‌تواند در ترکیب با سایر الگوریتم‌های فشرده‌سازی مانند کدگذاری دیکشنری استفاده شود (Desoky and Gregory, 1988). کدگذاری هافمن اغلب با کدگذاری حسابی مقایسه می‌شود، که نوع دیگری از رمزگذار آنتروپی است. نتایج چنین مقایسه‌هایی نشان می‌دهد که کدگذاری هافمن سریع‌تر از کدگذاری حسابی است، اما کدگذاری حسابی به‌طورکلی نسبت فشرده‌سازی بهتری دارد (Bookstein and Klein, 1993) و (Duda, 2014).

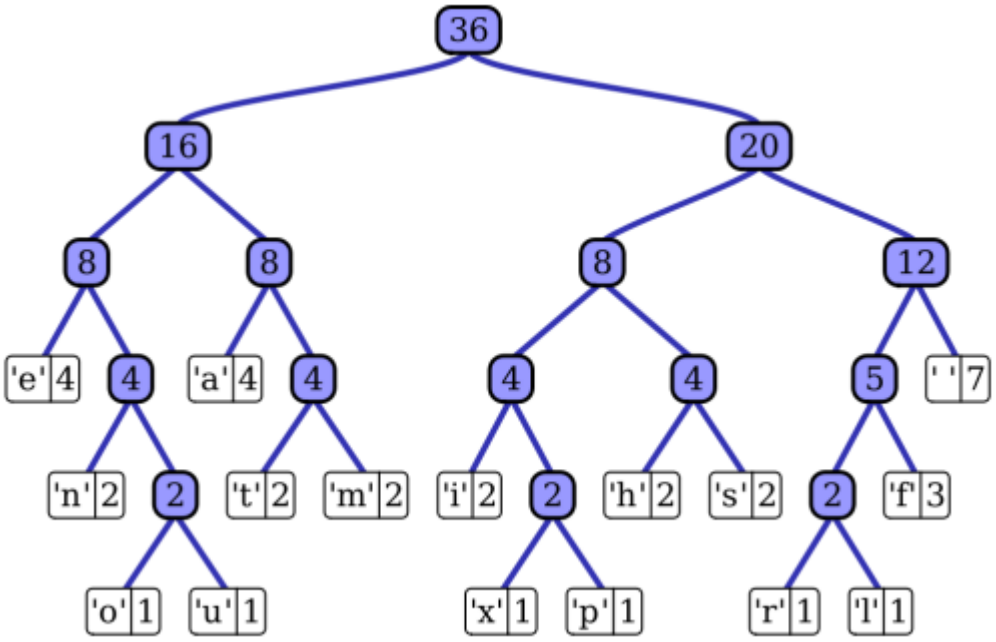
۲-۴-۱ فرایند فشردهe سازی

این بخش فرایند فشرده‌سازی برای کدگذاری هافمن را توضیح می‌دهد. یک مثال ورودی برای الگوریتم می‌تواند "this is an example of a huffman tree" جدول ۱-۲ فراوانی از هر کاراکتر در جمله مثال است.

جدول ۱-۲. مثال فراوانی کاراکتر

کاراکتر	Space	A	E	F	H	I	M	N	S	T	L	P	O	R	U	X
فراوانی	۷	۴	۴	۳	۲	۲	۲	۲	۲	۲	۱	۱	۱	۱	۱	۱

از این جدول می‌بینیم که کاراکتر space بیشترین فراوانی را دارد، پس‌ازآن "a" و"e" با چهار مورد. شکل ۱-۲ نشان می‌دهد که چگونه جمله مثال شبیه درخت دودویی هافمن است.



شکل ۱-۲. درخت دودویی هافمن

گره‌های این شکل همه دارای اعدادی هستند که در گره‌های فرزند نشان می‌دهد چند بار از هر کاراکتر استفاده‌شده است. این عدد هنگام ساختن درختان هافمن مهم است زیرا کاراکترهایی که بیشتر تکرار می‌شوند باید تا حد ممکن بالای درخت باشند (نزدیک به گره ریشه). جدول ۱-۲ نشان می‌دهد که فروانی (تکرار) کاراکتر "a" و "e" بیشترین هستند و بنابراین به گره ریشه نزدیک هستند. این سه کاراکتر را می‌توان پس از سه عبور از گره ریشه به دست آورد.

جمله مثال "this is an example of a huffman tree "، که با کدگذاری اسکی ذخیره می‌شود، از ۳۶ بایت ذخیره‌سازی استفاده می‌کند. این اندازه فایل قبل از فشرده‌سازی است، جایی که هر کاراکتر از یک بایت (یا ۸ بیت) ذخیره‌سازی استفاده می‌کند. هنگامی‌که فشرده‌سازی شروع می‌شود، با اولین کاراکتر "t" شروع می‌شود، برای رسیدن به گره برگ که مقدار "t" را از گره ریشه نگه می‌دارد، الگوریتم ابتدا به گره فرزند چپ، سپس دو بار به گره فرزند راست و درنهایت به چپ برود. بنابراین نمایش "t" فشرده دودویی (باینری) "۰۱۱۰" خواهد بود، که فقط ۴ بیت است. الگوریتم این کار را برای هر کاراکتر در متن انجام می‌دهد و پس از اتمام، با یک جریان دودویی ۱۳۵ بیت به پایان می‌رسد. این جریان فقط از ۱۷ بایت حافظه در کامپیوتر استفاده می‌کند که کمتر از نیمی از متن فشرده نشده است. البته، درخت هافمن نیز باید برای فشرده‌سازی متن در آن ذخیره شود.

۲-۵ کارهای مرتبط و تاریخچه تحقیق

این بخش مقاله‌ها و راه‌حل‌های مربوط به فشرده‌سازی متن یا به‌طور خاص، فشرده‌سازی اسکی، را موردبحث قرار می‌دهد، زیرا راه‌حل ارائه‌شده در پایان‌نامه موردبررسی یک فشرده‌ساز متن است که برای فشرده‌سازی متن اسکی سفارشی‌شده است.

۲-۵-۱ تقویت فشرده‌سازی متن با رمزگذاری آماری مبتنی بر کلمه

این مقاله از نسخه ۲۰۱۲ مجله کامپیوتر("The Computer Journal", 2020) است که توسط آنتونیو فاریا، گونزالو ناوارو و خوزه آر پاراما نوشته‌شده است. این مقاله پیشرفت احتمالی در برنامه‌های فشرده‌سازی را ارائه می‌دهد. هدف از راه‌حل پیشنهادی این است که نسبت فشرده‌سازی متن را افزایش داده و زمان فشرده‌سازی را برای فشرده‌سازهای متن کاهش دهد. این راه‌حل از تکنیک‌های فشرده‌سازی مبتنی بر کلمه و بایت جهت پیش‌پردازش برنامه‌های فشرده‌سازی عمومی استفاده می‌کند (Farina, 2012).

فشرده‌سازی مبتنی بر کلمه نوعی کدگذاری دیکشنری است که در آن الگوریتم کل کلمات را جستجو می‌کند (Moffat, 1989). اکثر برنامه نویسان دیکشنری، مانند الگوریتم‌های Lempel, Ziv، زیررشته‌های مکرر را جستجو می‌کنند (Ziv, Lempel, 1977). زیررشته‌ها می‌توانند هر طولی داشته باشند و می‌توانند بخشی از یک کلمه باشند. فشرده‌سازی مبتنی بر کلمه فقط کلمات کامل را به دیکشنری خود اضافه می‌کند. این مقاله بیان می‌کند که استفاده از فشرده‌سازی مبتنی بر کلمه، برخلاف کدگذارهای استاندارد دیکشنری، زمان فشرده‌سازی و فشرده‌سازی بهتری

را ارائه می‌دهد. علاوه بر این، امکان جستجوی کلمات و عبارات روی فایل فشرده بدون نیاز به فشرده‌سازی آن وجود دارد (Farina, 2012). فشرده‌سازی بایت گرا دومین پیش‌پردازشی است که راه‌حل پیشنهادی از آن استفاده می‌کند. فشرده‌سازی بایت گرا تکنیکی است که هنگام نوشتن کدهای فشرده‌سازی فقط از بایت‌های کامل استفاده می‌کند و نه از بیت‌ها (De Moura, et. Al., 1998). برعکس، کدگذاری استاندارد هافمن یک تکنیک فشرده‌سازی است که از توالی بیت‌ها به‌عنوان کد برای حرکت در درخت هافمن استفاده می‌کند. بنابراین کدگذاری هافمن باید به‌عنوان دنباله‌ای از بیت‌ها خوانده شود تا بایت. که می‌تواند باعث افزایش تأخیر شود، زیرا داده‌ها به‌صورت بایت در کامپیوتر ذخیره و خوانده می‌شوند. راه‌حل پیشنهادی از یک الگوریتم بایت گرا به نام Tagged Huffman استفاده می‌کند (Silva de Moura, 2000). برچسب‌گذاری هافمن (Tagged Huffman) از تکنیک فشرده‌سازی مشابه کدگذاری استاندارد هافمن استفاده می‌کند، اما هر کد برای جستجوی درخت هافمن از تعداد مشخصی بایت استفاده می‌کند. بر اساس این مقاله، استفاده از Tagged Huffman به‌جای کدگذاری استاندارد هافمن، تأخیر را کاهش می‌دهد، اما نسبت فشرده‌سازی را در حدود ۰/۶ کاهش می‌دهد، که آن‌ها به‌عنوان یک معامله پذیرفتند (Farina, ۲۰۱۲).

نتایج آزمایشات نشان می‌دهد که یک برنامه فشرده‌سازی در فشرده‌سازی یک فایل از پیش‌پردازش شده، ۵ برابر سریع‌تر از فایل اصلی است. بااین‌حال، زمان فشرده‌سازی پیش پردازنده را شامل نمی‌شود. مقایسه‌ها نشان می‌دهد استفاده از پیش پردازنده قبل از یک فشرده‌ساز عمومی (ازجمله Gzip, Bzip2, 7-Zip, Re-pair)، فایل‌های فشرده نهایی بین ۵,۵ تا ۱۰ درصد کوچک‌تر از فایل فشرده‌شده با فشرده‌ساز عمومی بوده‌اند (Farina, 2012).

مقایسه باTCS

فاریا (Fariña) و همکاران: راه‌حل (FS) و TCS هر دو فشرده‌ساز متنی هستند که نسبت فشرده‌سازی بالاتری را برای متن‌های نوشته‌شده به برخی از زبان‌ها به دست می‌آورد. روش فشرده‌سازی مبتنی بر کلمه به فشرده‌سازی FS می‌انجامد زیرا در زبان‌ها بیشتر از کلمات خاص استفاده می‌شود تا کلمات دیگر. همچنین TCS از ویژگی‌های زبان‌ها برای افزایش نسبت فشرده‌سازی استفاده می‌کند؛ ماژول فشرده‌سازی اسکی (ASCII compression module (ACM)) از یک مدل فشرده‌سازی (توضیح داده‌شده در بخش ۱-۱۰-۳) بر اساس فراوانی حروف در زبان انگلیسی استفاده می‌کند، و ماژول کدگذاری دیکشنری زمانی فشرده‌سازی می‌کند که یک متن دارای ترکیبی از کاراکترها باشد. هدف FS افزایش نسبت فشرده‌سازی و کاهش تأخیر در برنامه‌های فشرده‌سازی است. هر دو روش مبتنی

بر کلمه و بایت جهت کاهش زمان فشرده‌سازی است (Farina, 2012). بااین‌حال، TCS فقط نسبت فشرده‌سازی را در اولویت قرار می‌دهد، نه سرعت را.

۲-۵-۲ فشرده‌سازی آنلاین فایل‌های اسکی

این مقاله از کنفرانس بین‌المللی ۲۰۰۴ درزمینهٔ فناوری اطلاعات: کدگذاری و محاسبه است که توسط جان ایستل، پاملا مندلباوم و اما رجنتوا (Istle, et. Al., 2004) نوشته‌شده است. درواقع راه‌حل پیشنهادی این است که در یک متن انگلیسی کدهای کوتاه‌تری به بیگرام های مکرر بدهید و درنتیجه به فشرده‌سازی برسید. این مقاله ادعا می‌کند که فشرده‌سازی باید رخ دهد زیرا ترکیبات خاصی از حروف بیشتر به زبان انگلیسی هستند نسبت به سایر زبان‌ها. این راه‌حل فقط بر روی متون دارای کد اسکی کار می‌کند و نه هیچ کدگذاری دیگری (Istle, et. Al., 2004). این الگوریتم با استفاده از ۲۸ دیکشنری استاتیک کار می‌کند. دیکشنری‌های استاتیک به این معنی است که با تجزیه‌وتحلیل یک متن خاص، برای یک مورداستفاده، ایجاد نشده‌اند، بلکه مجموعه‌ای از واژه‌نامه های عمومی هستند که باید فشرده شوند – مخصوصاً برای متون انگلیسی. یکی از دیکشنری ها برای اعداد، دیگری برای علائم نگارشی و کاراکترهای خاص و ۲۶ دیکشنری باقی مانده برای هر حرف الفبا استفاده می‌شود. اگر متنی که فشرده می‌شود حاوی حرف "a" باشد، الگوریتم، دیکشنری آن حرف خاص را جستجو می‌کند و پیدا می‌کند که حرف بعدی در چه شاخصی قرار دارد. به‌عنوان‌مثال، دیکشنری حرف "a" حرف "t" را به‌عنوان اولین فهرست دارد، زیرا "t" حرفی است که بیشتر از بقیه حروف به دنبال "a" می‌آید. اگر متن حاوی حرف "t" بود با شماره ۱ جایگزین می‌شد (Istle, et. Al., 2004).

برای هر کلمه جدید در یک متن، الگوریتم با جستجوی جایی که اولین حرف یا نماد در "دیکشنری پیش‌فرض" قرار دارد شروع می‌شود. دیکشنری پیش‌فرض بعد از علائم نگارشی یا کاراکترهای خاص استفاده می‌شود. اگر کلمه ای که قرار است فشرده شود "hat" باشد، "h" در دیکشنری پیش‌فرض فهرست ۱۳ خواهد بود، "a" در دیکشنری حرف "h" فهرست ۲ خواهد بود و حرف "t" در دیکشنری "a" فهرست ۱ خواهد بود. سپس اعداد ۱۳، ۲ و ۱ به دنباله‌های بیتی ترجمه می‌شوند (Istle, et. Al., 2004). دنباله‌های بیتی کوتاه تر برای هر کاراکتر کمتر از یک بایت استفاده می‌کنند، در مقابل کاراکترهای فشرده نشده اسکی که از بایت برای هر کاراکتر استفاده می‌کنند. جایگزینی کاراکترهای اسکی با دنباله‌های بیتی باعث فشرده‌سازی می‌شود.

آزمایش‌های انجام‌شده در مقاله نتایج حاصل از راه‌حل پیشنهادی را با برنامه فشرده‌سازی Winzip مبتنی بر DEFLATE مقایسه کرده است. یک فایل "کمتر از ۱۰۰ کیلوبایت " و " تا ۱ مگابایت " برای مقایسه استفاده شد

(Istle, et. Al., 2004). راه‌حل پیشنهادی نسبت فشرده‌سازی ۱,۶۷ را برای هر دو فایل به دست آورد، درحالی‌که WinZip نسبت فایل بزرگ‌تر را ۲,۵ و نسبت فایل کوچک‌تر را کمتر از ۱ به دست آورد. از این نتایج، آن‌ها به این نتیجه رسیدند که "تکنیک دیکشنری چندگانه در فشرده‌سازی فایل‌ها در هر اندازه ای سازگار است" (Istle, et. Al., 2004).

مقایسه با ماژول فشرده‌سازیASCII

شباهت های زیادی بین روش دیکشنری چندگانه و ACM از TCS وجود دارد. هر دو فقط می‌تواند فایل‌های متنی را فشرده کنند، هر دو برای متن اسکی در نظر گرفته‌شده‌اند، و هر دو از دیکشنری‌های استاتیک که برای متون انگلیسی سفارشی‌شده‌اند استفاده می‌کنند. راه‌حل دیکشنری چندگانه از تکنیکی هم تراز با کدگذاری هافمن استفاده می‌کند. کدگذاری هافمن کدهای کوتاه‌تری به کاراکترهای مکرر می‌دهد. راه‌حل دیکشنری چندگانه، کدهای کوتاه‌تری را به بیگرام های تکراری می‌دهد، ACM کاراکترها را به کدها تبدیل نمی‌کند، بلکه از تکنیکی به نام ادغام استفاده می‌کند که در بخش ۱-۱۰-۳ توضیح داده می‌شود. تفاوت دیگر این است که ACM می‌تواند متن کدگذاری‌های مختلف را فشرده کند، درحالی‌که راه‌حل دیکشنری چندگانه نمی‌تواند. در مقاله آن‌ها این را به‌عنوان یک کار آینده بیان کردند (Istle, et. Al., 2004).

فصل سوم

مروری بر کارهای انجام‌شده

۳-۱ مقدمه

در فصل ۳ پایان‌نامه موردبررسی، روش‌ها و متدولوژی های مختلف را تشریح و مقایسه می‌کند و توضیح می‌دهد که کدام روش‌ها برای این پروژه انتخاب‌شده‌اند و چرا. همچنین روش‌های مختلف دانشگاهی را که برای پروژه‌های تحقیقاتی استفاده می‌شود و روش‌ها و مدل‌هایی را که بطور خاص برای توسعه نرم‌افزار استفاده می‌شود، ارائه می‌دهد و در فصل۴ پایان‌نامه موردبررسی، الزامات کلی سیستم و نیز الزامات مجموعه داده و پیاده‌سازی‌های مورداستفاده در TCS را شرح می‌دهد. سپس طراحی کلی TCS و راه‌حل فنی ماژول فشرده‌سازی اسکی (ACM) را توضیح می‌دهد. در ادامه این مطالب را به تفکیک توضیح خواهم داد.

۳-۲ روش‌ها و استراتژی‌های تحقیق

چندین روش تحقیق وجود دارد که می‌تواند برای پروژه‌های تحقیقاتی مورداستفاده قرار گیرد. روش‌های تحقیق روش‌هایی برای انجام وظایف تحقیق هستند و نحوه انجام تحقیق را توضیح می‌دهند. روش‌های رایج تحقیق شامل تجربی، توصیفی و بنیادی است. رویکرد تجربی علل، آثار و متغیرها را بررسی می‌کند. آزمایشات انجام‌شده و نتایج تجزیه‌وتحلیل می‌شوند. روش تحقیق توصیفی ویژگی‌های یک موقعیت را توصیف می‌کند، اما علل آن را توضیح نمی‌دهد. روش توصیفی اغلب ازنظرسنجی یا مطالعات موردی استفاده می‌کند. روش تحقیق بنیادی ابتکاری است و ایده ها، اصول و نظریه های جدیدی را ایجاد می‌کند. تحقیقات بنیادی بر اساس کنجکاوی انجام می‌شود و در مورد مشاهده یک پدیده است (Hakansson, 2013). این پروژه از روش تحقیق تجربی استفاده می‌کند، زیرا آزمایش بخش مهمی از پروژه است. آزمایش روابط بین متغیرها و نحوه تغییر متغیرها بر نتایج را بررسی می‌کند.

استراتژی تحقیق، رویکرد مشخص‌تری برای نحوه انجام تحقیق است، درحالی‌که روش تحقیق چارچوبی برای تحقیق است. یک استراتژی تحقیق یک‌راهنما یا یک روش است. یک استراتژی تحقیق می‌تواند تجربی باشد، درست مانند روش تحقیق تجربی. رویکرد تجربی باهدف کنترل همه عواملی که ممکن است بر نتیجه یک آزمایش تأثیر بگذارد، کنترل می‌شود. این استراتژی بر اساس نتایج یک آزمایش، یک فرضیه را تأیید یا رد می‌کند. یکی دیگر از

استراتژی‌های رایج تحقیق استفاده ازنظرسنجی یا پرسشنامه است. این استراتژی شامل جمع‌آوری اطلاعات از افراد است. این نظرسنجی‌ها می‌تواند عمیق باشد و ممکن است فقط شامل چند نفر باشد که روش کیفی است یا می‌توان نظرسنجی‌ها را برای تجزیه‌وتحلیل داده‌های تعداد بیشتری از افراد طراحی کرد که روش کمی است (Hakansson, ۲۰۱۳). این پروژه به همان دلیلی که از روش تحقیق تجربی استفاده می‌کند، از استراتژی تحقیق تجربی استفاده می‌کند.

۳-۳ جمع‌آوری داده‌ها

پایان‌نامه موردبررسی از تحقیقات کمی استفاده می‌کند؛ مناسب‌ترین روش جمع‌آوری داده‌ها برای پروژه‌های تحقیقاتی کمی، آزمایش، پرسشنامه، مطالعات موردی و مشاهدات است. آزمایش‌ها مجموعه داده بزرگی را که برای متغیرها استفاده می‌شود جمع‌آوری می‌کند. پرسشنامه‌ها داده‌ها را از طریق سؤالات کمی یا کیفی جمع‌آوری می‌کند، مطالعات موردی تجزیه‌وتحلیل عمیق یک یا چند شرکت‌کننده است. مشاهدات رفتار را با تمرکز بر موقعیت‌ها و فرهنگ بررسی می‌کند (Hakansson, 2013). این پروژه از آزمایشات برای جمع‌آوری داده‌ها استفاده می‌کند، همان‌طور که از استراتژی تحقیق تجربی استفاده می‌کند.

۳-۴ تجزیه‌وتحلیل داده‌ها

برای تحقیقات کمی، متداول‌ترین روش‌های تجزیه‌وتحلیل داده‌ها آمار و ریاضیات محاسباتی است. آمار استنباطی است و شامل نتایج تجزیه‌وتحلیل برای نمونه‌های آماری و همچنین ارزیابی اهمیت نتایج است. ریاضیات محاسباتی شامل محاسبه روش‌های عددی، مدل‌سازی و استفاده از الگوریتم‌ها است (Hakansson, 2013). ریاضیات محاسباتی از کد کامپیوتری برای تجزیه‌وتحلیل داده‌ها استفاده می‌کند، برخلاف آمار، که توسط افراد قابل تجزیه‌وتحلیل است. نتایج حاصل از آزمایشات پایان‌نامه موردنظر، داده‌های عددی را ارائه می‌دهد. تجزیه‌وتحلیل این داده‌ها شامل مقایسه داده‌ها است. ریاضیات محاسباتی برای تجزیه‌وتحلیل داده‌ها موردنیاز نیست. بنابراین در پروژه برای تجزیه‌وتحلیل داده‌ها از روش آمار استفاده‌شده است.

۳-۵ تضمین کیفیت

تضمین کیفیت اعتبار و تأیید تحقیق است. داده‌ها باید قابل‌اعتماد، معتبر، قابل تکرار و اخلاقی باشند تا در تحقیق مورداستفاده قرار گیرند. پایایی به ثبات و ثبات اندازه‌گیری‌ها اشاره دارد. این بدان معناست که هنگام در نظر گرفتن همه متغیرهای آزمایش، اندازه‌گیری‌های مختلف باید نتایج قابل قبولی را ارائه دهند. قابلیت اطمینان در هر آزمایشی که در این پروژه صورت گرفته ارزیابی‌شده است.

تصدیق (روایی) اطمینان می‌دهد که ابزارهای یک آزمایش آن‌گونه که انتظار می‌رود اندازه‌گیری شود، انجام شود. اعتبار این پروژه با ارزیابی مراحل و ابزارهای موردنیاز برای انجام آزمایشات حفظ‌شده است.

تکرارپذیری اطمینان حاصل می‌کند که یک آزمایش با تکرار دقیق متغیرها نتایج یکسانی را به همراه خواهد داشت. در این پروژه، همه آزمایش‌ها تکرار شده است تا بررسی شود آیا نتایج یکسان هستند یا خیر.

تحقیق همچنین باید اخلاقی باشد. اخلاق اصول اخلاقی برنامه‌ریزی، انجام و گزارش نتایج حاصل از تحقیقات و همچنین حفظ حریم خصوصی و برخورد محرمانه با مطالب را پوشش می‌دهد (Hakansson, 2013). این جنبه‌های اخلاقی در تحقیق پروژه موردبررسی نیز ارزیابی‌شده است.

۳-۶ روش‌های توسعه نرم‌افزار

یکی از روش‌های توسعه نرم‌افزار، یک برنامه (یا مجموعه‌ای از دستورالعمل‌ها) برای توسعه یک سیستم از ابتدا تا پیاده‌سازی است. مزایای استفاده از چنین روشی عبارتند از: این به درک چرخه حیات فرایند کمک می‌کند، یک رویکرد ساختاریافته برای توسعه را اجرا می‌کند و برنامه‌ریزی منابع را که در پروژه مورداستفاده قرار می‌گیرد، امکان‌پذیر می‌سازد (Agarwal, et. al., 2010).

مدل آبشار از فازهای متوالی و خطی استفاده می‌کند، که در آن هر مرحله ادامه فاز قبلی است. مراحل در مدل آبشار عبارتند از: امکان‌سنجی (درک مشکل)، الزامات و مشخصات، طراحی راه‌حل، کدگذاری و تست ماژول، آزمایش سیستم، تحویل و نگهداری (Agarwal, et. al., 2010).

روش نمونه اولیه، نسخه‌های ناقص از یک محصول را که نمونه اولیه نامیده می‌شود، در حین توسعه ایجاد می‌کند. مزیت استفاده از نمونه‌های اولیه این است که توسعه‌دهندگان می‌توانند بازخورد کاربران را در حین توسعه محصول دریافت کنند و تغییرات احتمالی که باید انجام شود به‌راحتی انجام می‌شود. نمونه‌های اولیه همچنین می‌توانند توسعه‌دهندگان را درمورد قطعات ناتمام قسمت بهتری به توسعه‌دهندگان ارائه دهند، که می‌تواند برآورد بهتری در مورد زمان به پایان رساندن محصول ارائه دهد (Guide, et. al., 1999).

مدل چابک (سریع) یک رویکرد سازگار و انعطاف‌پذیر است که بر تحویل زودهنگام و مداوم نرم‌افزار به مشتری تمرکز دارد. درحالی‌که مدل آبشار خطی و سازگار است، مدل چابک انعطاف‌پذیر و قابل‌تغییر است. مدل چابک ۱۲ اصل را تعریف می‌کند که "بیانیه" توسعه چابک را تشکیل می‌دهند (Highsmith, 2002).

همه این روش‌ها معمولاً برای تیم توسعه‌دهندگان سازمان‌دهی می‌شود که در آن محصول برای مشتری اجرا می‌شود. پایان‌نامه موردبررسی یک پروژه انفرادی است و محصولی را برای مشتری ایجاد نمی‌کند، بنابراین پیروی از تمام اصول مرتبط با این روش‌های توسعه غیرضروری و یا غیرممکن است. در عوض، ایده کلی پشت روش‌ها موردتوجه قرارگرفته است. روش نمونه اولیه نمونه‌هایی را ایجاد می‌کند که برای آزمایش در حین توسعه محصول در نظر گرفته‌شده است. این پروژه شامل سه ماژول است که به‌قدری ساده‌اند که ایجاد نمونه‌های اولیه ضروری نخواهد بود. در عوض نرم‌افزار در حین توسعه، آزمایش‌شده است. بنابراین این پروژه از روش نمونه اولیه استفاده نکرده است. انعطاف‌پذیری مدل چابک برای این پروژه مناسب‌تر از مدل آبشار است زیرا این پروژه فازهایی ندارد که به مراحل قبلی وابسته باشد تا تکمیل شود. بنابراین این پروژه از ایده کلی پشتیبان مدل چابک استفاده کرده است.

۳-۷ الزامات سیستم

الزامات سیستم الزامات عملکردی و غیر عملکردی هستند که باید مورداستفاده قرار گیرند تا TCS قابل‌استفاده و قوی باشد. الزامات بر اساس موارد استفاده برای TCS تعریف می‌شود. الزامات عملکردی خدماتی را که راه‌حل باید ارائه دهد و نحوه عملکرد راه‌حل در شرایط خاص را توصیف می‌کند. الزامات غیرکاربردی محدودیت‌هایی در خدمات ارائه‌شده توسط یک‌راه‌حل است. الزامات غیرکاربردی ممکن است بر عملکرد، قابلیت اطمینان و قابلیت استفاده متمرکز شوند (Sommerville, 2007).

۱- الزامات عملکردی TCS عبارتند از:

- در طول فشرده‌سازی هیچ داده‌ای از بین نمی‌رود.

TCS از فشرده‌سازی بدون اتلاف استفاده می‌کند، بدین معنی که داده‌های اصلی باید پس از فشرده‌سازی قابل بازیابی باشد. اگر داده‌ها در طول فشرده‌سازی از بین بروند، راه‌حل فایده‌ای ندارد.

- TCS می‌تواند متن را با رمزگذاری‌های معمول فشرده کند.

رمزگذاری‌های رایج به کدگذاری UTF-8 و ASCII اشاره می‌کند. TCS باید بتواند رایج‌ترین رمزگذاری‌های متنی را مدیریت کند تا یک‌راه‌حل مفید باشد. چهار فایل با رمزگذاری UTF-8 و یک فایل با رمزگذاری ASCII برای آزمایش این کار استفاده می‌شود.

- **TCS می‌تواند متن را با رمزگذاری‌های غیرمعمول فشرده کند.**

درصورتی‌که TCS قادر به فشرده‌سازی متن با رمزگذاری‌هایی باشد که اغلب استفاده نمی‌شوند، علاوه بر رمزگذاری‌های رایج، این راه‌حل محکم است و می‌تواند انواع بیشتری از موارد استفاده را اداره کند. برای آزمایش این مورد از فایلی با کدگذاری cpo37 استفاده می‌شود.

- ماژول‌های TCS به‌راحتی متصل می‌شوند.

سه واحد ماژول TCS برنامه‌های مستقل هستند که می‌توانند در ترکیب با یکدیگر استفاده شوند. ماژول‌ها باید به‌طور جداگانه به هم متصل شوند تا به‌صورت جداگانه ارزیابی شوند.

۲- الزامات غیرکاربردی برای TCS عبارتند از:

- TCS می‌تواند فایل‌های بالای ۲۰ مگابایت را فشرده کند.

کاهش حجم فایل بیشتر از فایل‌های کوچک‌تر روی فایل‌های بزرگ‌تر تأثیر می‌گذارد. برای ارزیابی این نیاز، یک فایل XML با حجم ۲۱,۶ مگابایت استفاده می‌شود.

- TCS می‌تواند فایل‌ها را حداقل به نصف اندازه اصلی خود فشرده کند.

چندین تست روی هر یک از متون انجام می‌شود و اگر حداقل یکی از تست‌ها دارای ضریب فشرده‌سازی حداقل ۲ باشد، شرایط موردنیاز برآورده می‌شود.

- **تمام کدهای شخص ثالثی که در TCS استفاده می‌شود باید منبع باز باشد.**

کد شخص ثالثی که در TCS استفاده می‌شود باید دارای مجوز باز باشد که اجازه اصلاحات را می‌دهد. به‌منظور ادغام ماژول‌ها در یک‌راه‌حل، ممکن است تغییرات در کد لازم باشد.

۸-۳ ویژگی مجموعه داده‌ها

ویژگی مجموعه داده‌ها در پایان‌نامه موردبررسی به‌عنوان مجموعه‌ای از متون مختلف که TCS هنگام ارزیابی قابلیت فشرده‌سازی از آن‌ها استفاده می‌کند، تعریف‌شده است. این متون از رمزگذاری‌های مختلف استفاده می‌کنند، به‌عنوان‌مثال UTF-8 و ASCII، کدگذاری‌های معمولی هستند که ممکن است کاربر از آن‌ها استفاده کند. UTF-8 یک رمزگذاری کاراکتر بسیار رایج است که شامل انواع زیادی از نمادها است. ASCII برای متنی استفاده می‌شود که از تنوع زیادی از کاراکترها و نمادها استفاده نمی‌کند و معمولاً به زبان انگلیسی نوشته می‌شود (McEnery and Xiao, 2005). از کدگذاری‌های مختلف برای ارزیابی متفاوت در فشرده‌سازی استفاده می‌شود و اینکه چه نوع کدگذاری برای راه‌حل بهینه است. متون از کدگذاری ASCII، cpo37 و UTF-8 استفاده خواهند کرد.

۱-۸-۳ تعریف مجموعه داده‌ها

مجموعه داده‌های مورداستفاده در ارزیابی پایان‌نامه موردبررسی، نمونه‌هایی از فایل‌های متنی بزرگ‌تر هستند که کاربر ممکن است TCS را روی آن‌ها اعمال کند. مجموعه داده‌ها شامل ۶ متن با ویژگی‌های مختلف است؛

۱- یک فایل XML ویکی‌پدیا. رمزگذاری UTF-8 ؛ ۲۱,۶ مگابایت

(https://dumps.wikimedia.org/backupindex.html)

۲- یک فایل XML ویکی‌پدیا. کدگذاری cpo37 ؛ ۱۸,۱ مگابایت

(https://dumps.wikimedia.org/backupindex.html)

۳- یک فایل کد نوشته‌شده با C. کدگذاری ASCII؛ ۶۴ کیلوبایت

(https://github.com/torvalds/linux/blob/master/kernel/sys.c)

۴- کتابی که به زبان انگلیسی نوشته‌شده است. رمزگذاری UTF-8 ؛ ۶۸۰ کیلوبایت (Thus spake Zarathustra)

۵- کتابی که به زبان ایتالیایی نوشته‌شده است. رمزگذاری UTF-8 ؛ ۶۲۶ کیلوبایت (Magugliani, La divina commedia)

۶- کتابی که به زبان چینی نوشته‌شده است. رمزگذاری UTF-8 ؛ ۲۸۵ کیلوبایت (Salt Lake City)

فایل XML ویکی‌پدیا یک نسخه محلی و فقط متنی از مجموعه‌ای از مقالات (صفحات جداگانه) از ویکی‌پدیا است. فایل XML بیشتر شامل کاراکترهای ASCII است، بدین معنا که اکثر کاراکترهای متن را می‌توان در اسکی کدگذاری کرد مجموعه داده‌ها نیز شامل همان فایل XML است که با cpo37 رمزگذاری شده است. Cpo37 رمزگذاری یک فایل پس از فشرده‌سازی آن با ماژول فشرده‌سازی ASCII است. این متن برای تست وجود دو ماژول دیگر راه‌حل (کدگذاری هافمن و کدگذاری دیکشنری) با ACM سازگار است. فایل کد C از مخزن

سیستم‌عامل لینوکس GitHub گرفته‌شده است و فقط شامل کاراکترهای ASCII است. این سه کتاب برای آزمایش وجود تفاوت بین زبان کد ساختاریافته و زبان طبیعی گنجانده‌شده است. کتاب انگلیسی بیشتر شامل کاراکترهای ASCII است، کتاب ایتالیایی نیز شامل بیشتر کاراکترهای ASCII است، اما از چندین کاراکتر استفاده می‌کند که در طرح کدگذاری ASCII گنجانده نشده است، و کتاب چینی شامل هیچ کاراکتر اسکی نمی‌شود. متون مختلف دارای اندازه‌های بسیار متنوعی هستند که برخی از آن‌ها چند کیلوبایت بزرگ و برخی دیگر چند مگابایت هستند. صرف‌نظر از این، متن‌ها اندازه کافی برای ارزیابی صحیح TCS دارند.

۳-۹ الزامات پیاده‌سازی

TCS شامل سه ماژول است. ACM، کدگذاری دیکشنری و کدگذاری هافمن. ACM به‌طور خاص برای پروژه موردبررسی ایجادشده است، اما کدگذاری دیکشنری و کدگذاری هافمن پیاده‌سازی‌هایی است که توسط افراد دیگر انجام‌شده و در راه‌حل یکپارچه‌شده است. پیاده‌سازی برنامه‌هایی است که دیگران برای پیاده‌سازی الگوریتم‌ها ساخته‌اند، پیاده‌سازی‌ها باید مجموعه‌ای از الزامات را برآورده کنند، اما همچنین باید بتوانند با سایر ماژول‌های TCS ادغام شوند تا راه‌حل به‌طورکلی بتواند الزامات سیستم را برآورده کند. چندین پیاده‌سازی برای الگوریتم‌های کدگذاری دیکشنری و کدگذاری هافمن یافت می‌شود. در پایان‌نامه موردبررسی شرایط زیر را برای پیاده‌سازی‌ها در نظر گرفته است:

پیاده‌سازی‌ها باید دارای مجوز باز باشند.

پیاده‌سازی‌ها باید درست عمل کنند، حتی زمانی که داده‌های ورودی می‌توانند چند مگابایت باشند.

پیاده‌سازی‌ها باید با متن کدگذاری‌های مختلف سازگار باشند.

پیاده‌سازی‌ها همچنین باید بتوانند با متن کدگذاری‌های مختلف کار کنند. پیاده‌سازی‌هایی که الزامات را برآورده می‌کنند در برابر یکدیگر آزمایش می‌شوند و پیاده‌سازی‌هایی که بهترین نتیجه را می‌گیرند (بالاترین نسبت فشرده‌سازی) به‌عنوان کدگذاری دیکشنری و ماژول کدگذاری هافمن برای TCS انتخاب می‌شوند.

۳-۱۰ طراحی TCS

TCS یک‌راه‌حل فشرده‌سازی است که از سه ماژول تشکیل‌شده است: ماژول فشرده‌سازی ASII (ACM)، ماژول کدگذاری دیکشنری و ماژول کدگذاری هافمن. این ماژول‌ها به‌هم‌پیوسته هستند، به این معنی که آن‌ها برنامه‌های مستقل هستند که می‌توانند در یک‌راه‌حل ادغام شوند، یا می‌توانند به‌عنوان برنامه‌های مستقل استفاده شوند (Kaye, ۲۰۰۳). داشتن ماژول‌های به‌هم‌پیوسته، چابکی را در جستجوی بالاترین نسبت فشرده‌سازی ایجاد می‌کند و برای ارزیابی ماژول‌ها به‌صورت جداگانه ضروری است. ماژول‌ها در C یا Python نوشته‌شده‌اند و می‌توانند به‌صورت متوالی با استفاده از یک برنامه پایتون یا یک اسکریپت پوسته یونیکس اجرا شوند.

TCS ابتدا برای ACM در نظر گرفته‌شده است، سپس رمزگذار دیکشنری و درنهایت ماژول کدگذاری هافمن استفاده کند. ACM ابتدا استفاده می‌شود زیرا فقط قادر به فشرده‌سازی کاراکترهای اسکی است. خروجی رمزگذار دیکشنری یا ماژول‌های کدگذاری هافمن، داده‌های دودویی هستند که شامل کاراکترهای اسکی نیستند، بنابراین این ماژول‌ها نباید قبل از ACM استفاده شوند. خروجی ACM یک فایل متنی در کدگذاری cpo37 است. این خروجی به کدگذار دیکشنری ارسال می‌شود که زیررشته‌های تکراری را در متن جستجو می‌کند. سپس خروجی دودویی از رمزگذار دیکشنری به ماژول کدگذاری هافمن ارسال می‌شود که مقادیر بایت را که بیشتر از بقیه استفاده می‌شود، جستجو می‌کند. فایل خروجی از ماژول کدگذاری هافمن آخرین فایل فشرده‌شده از TCS است.

۳-۱۰-۱ طراحی ACM

ACM برنامه‌ای است که در Python 3.6.9 نوشته‌شده است و شامل دو فایل اسکریپت است: ascii-compression.py و charprocessing.py. ascii-compression.py مسئول خواندن و نوشتن فایل‌هاست، درحالی‌که charprocessing.py فشرده‌سازی متن واقعی را کنترل می‌کند.

Ascii-compression.py الگوریتم فشرده‌سازی است که ACM از آن استفاده می‌کند مشابه الگوریتم شوکو است و با ادغام کاراکترها با یکدیگر کار می‌کند. ادغام کاراکتر به این معنی است که دو کاراکتر از یک فایل متنی، به‌عنوان‌مثال "a" و "b"، می‌توانند باهم در فضای یک کاراکتر واحد ذخیره شوند. این الگوریتم تنها قادر به ادغام کاراکترهای اسکی است. همان‌طور که در بخش ۳-۲ ذکر شد، کاراکترهای اسکی فقط از ۷ بیت آخر برای کد نماد استفاده می‌کنند، درحالی‌که از اولین بیت می‌توان برای تشخیص خطاها در داده‌ها استفاده کرد. خروجی ACM (فایل فشرده) یک فایل متنی با کدگذاری cpo37 است. برخلاف ASCII، کدگذاری cpo37 از همه ۸ بیت برای کد نماد استفاده می‌کند. در مورد ACM، اولین بیت از یک کاراکتر cpo37 برای نشان دادن ادغام یا نبودن کاراکتر استفاده می‌شود و ۷ بیت آخر کد نماد یک یا دو کاراکتر است. رمزگذاری cpo37 به‌عنوان فرمت خروجی انتخاب

می‌شود زیرا از همه ۸ بیت به‌جای ۷ بیت ASCII استفاده می‌کند. cpo37 همچنین یک کدگذاری استاندارد در پایتون است و درست مانند US-ASCII برای زبان انگلیسی در نظر گرفته‌شده است.

Asci-compression.py

اسکریپت ascii-compression.py یک فایل متنی با کدگذاری ASCII، UTF-8 یا cpo37 به‌عنوان ورودی می‌گیرد. فایل‌های باینری یا دیگر کدگذاری‌های متنی را نمی‌پذیرد. سپس هر خط از فایل متنی را به اسکریپت charprocessing.py ارسال می‌کند که فشرده‌سازی را کنترل می‌کند. پس از نوشتن فایل خروجی، یک آزمایش ساده را انجام می‌دهد تا بررسی کند که آیا فایل خروجی بزرگ‌تر از فایل اصلی است یا خیر. این می‌تواند در صورتی اتفاق بیفتد که فایل ورودی دارای تعداد زیادی کاراکتر باشد که نمی‌توان آن‌ها را در اسکی کدگذاری کرد، اگر فایل خروجی بزرگ‌تر از فایل اصلی باشد، داده‌های فشرده‌شده حذف می‌شوند، زیرا هدف برنامه فشرده‌سازی ایجاد فایل‌های کوچک‌تر است. در عوض، فایل خروجی با داده‌های اصلی بازنویسی می‌شود و فایل فشرده‌شده را به‌اندازه فایل اصلی می‌سازد. درنهایت، فایل فشرده یک خط جدید اضافه می‌شود که شامل o یا ۱ است. این عدد نشان می‌دهد که آیا فایل فشرده تغییر کرده است (o) یا کپی از فایل اصلی است (۱). این اطلاعات برای فشرده‌ساز مهم است، که الگوریتم فشرده‌سازی فایل را کاهش می‌دهد یا همان‌طور که هست کپی می‌کند. هنگامی‌که یک فایل فشرده از حالت فشرده خارج می‌شود، عدد حذف‌شده و فایل متنی به حالت اصلی خود بازگردانده می‌شود.

charprocessing-py

اسکریپت charprocessing.py وظیفه فشرده‌سازی و رفع فشار را بر عهده دارد. الگوریتم فشرده‌سازی هر کاراکتر یک متن را تکرار کرده و آن‌ها را در یک فایل فشرده با کدگذاری cpo37 ذخیره می‌کند. نحوه ذخیره هر کاراکتر به دو عامل بستگی دارد. آیا می‌توان کاراکتر را در ASCII رمزگذاری کرد و آیا bigram زیر فقط شامل کاراکترهای معمولی است. اگر کاراکتر در برنامه کدگذاری ASCII گنجانده نشده باشد، به این معنی است که کاراکتر ممکن است بیش از یک بایت ذخیره‌سازی را اشغال کند. مثال UTF-8 کاراکتر "e" است که از دو بایت استفاده می‌کند. وقتی این کاراکتر فشرده می‌شود، الگوریتم باید علامت دهد که از بیش از یک بایت استفاده می‌کند. در غیر این صورت، هنگام خروج از فشرده‌سازی الگوریتم تصور می‌کند که دو کاراکتر وجود دارد که هرکدام از یک بایت استفاده کرده‌اند. قبل از نوشتن کاراکتر، دو علامت بک اسلش "\\" به کاراکتر اضافه می‌شود، که نشان می‌دهد بایت بعدی کاراکتر اسکی نیست. سپس کاراکتر ویژه به عددی که نشان‌دهنده کاراکتر است تبدیل می‌شود. این امر ضروری است زیرا اگر کاراکتر خاص بیشتر از یک بایت باشد، نمی‌تواند در cpo37 کدگذاری شود. سرانجام، یک بک اسلش "\" پس از کاراکتر اضافه می‌شود، به این معنی که کاراکتر ویژه به پایان رسیده است. داشتن علامت قبل

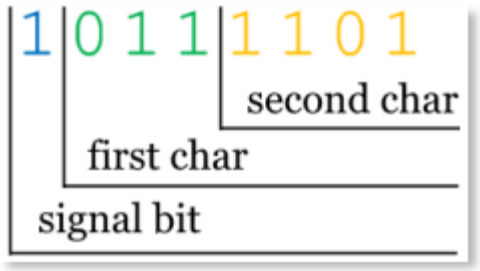
و بعد از کاراکتر ضروری است زیرا طول کاراکتر می‌تواند بین ۲ تا ۴ بایت باشد. نسخه فشرده‌شده کاراکتر "e" به‌صورت \\۳۳۲۱ نشان داده می‌شود.

اگر فایل اصلی حاوی کاراکترهای بک اسلش "\" است، باید آن‌ها را علامت دهید تا هنگام خروج از فشرده‌سازی الگوریتم برای یک کاراکتر غیر ASCII اشتباه نگیرد. به‌عنوان‌مثال اگر دو بک اسلش "\\" در یک سطر توسط کمپرسور به‌عنوان علامت نباشند، دی کمپرسور انتظار دارد کاراکتر بعدی یک کاراکتر ویژه باشد. روشی که سیگنال کمپرسور به بک اسلش نشان می‌دهد این است که قبل از آن دو بک اسلش دیگر اضافه می‌کند. این بدان معناست که هر بک اسلش در فایل اصلی با سه بک اسلش جایگزین می‌شود. هنگامی‌که دستگاه فشرده‌ساز در یک فایل فشرده یا بک اسلش مواجه می‌شود، می‌داند که کاراکتر بعدی نیز یک بک اسلش خواهد بود، اما یکی پس‌ازآن می‌تواند آغاز یک کاراکتر ویژه یا سومین بک اسلش باشد. اگر کاراکتر خاصی باشد، به‌درستی از حالت فشرده خارج می‌شود، و اگر سومین بک اسلش باشد، دو بک اسلش برداشته می‌شود. اگر یک کاراکتر ASCII و کاراکتر بعدی آن هر دو در لیست کاراکترهای مشترک گنجانده شوند، آن‌ها باهم ادغام می‌شوند. ادغام تنها در صورتی کار می‌کند که دو کاراکتر متوالی مشترک باشند. اگر هیچ‌یک، یا فقط یک مورد، از کاراکترهای متوالی در لیست کاراکترهای معمولی نباشند، ادغام نمی‌شوند. در عوض، آن‌ها همانند فایل اصلی در فایل فشرده ذخیره می‌شوند، به‌عنوان‌مثال، کاراکتر "a" دارای یک بیت ارزش است ۰۱۱۰۰۰۰۱ . این نشان‌دهنده باینری عدد ۹۷ است و به این صورت است که ACM کاراکترها را می‌خواند. اولین بیت ۰ است که به الگوریتم می‌گوید که ۷ بیت بعدی نشان‌دهنده یک کاراکتر ادغام نشده است.

```
2
3  firstCommonChars = ["e","t","a","o","n","i","h"," "]
4  secondCommonChars = ["e","t","a","o","n","i","h"," ","s","r","l","d","u","c","m","w"]
5
```

شکل ۱–۳. فهرست کاراکترهای متداول

اگر یک کاراکتر در لیست firstCommonChars گنجانده‌شده باشد، همان‌طور که در شکل ۱–۳ نشان داده‌شده است، و کاراکتر بعدی در لیست secondCommonChars گنجانده‌شده باشد، باهم ادغام می‌شوند. این لیستی از رایج‌ترین حروف (به‌علاوه نماد فاصله) در زبان انگلیسی است (Trost, 2020) و مدل فشرده‌سازی ACM است. یک مدل فشرده‌سازی می‌تواند لیستی از تکرار کاراکترها، نمادها یا بیگرام ها باشد. ACM برای همه انواع متون از یک مدل استفاده می‌کند، به این معنی که متون انگلیسی باید نسبت فشرده‌سازی بهتری نسبت به متون سایر زبان‌ها داشته باشند.



شکل ۲-۳ نمایش باینری از کاراکترهای ادغام‌شده

یک کاراکتر ادغام‌شده از ۳ قسمت تشکیل‌شده است: بیت سیگنال، کاراکتر اول و کاراکتر دوم. برای یک کاراکتر ادغام‌شده، اولین بیت روی ۱ تنظیم می‌شود. سه بیت بعدی نشان‌دهنده موقعیت اولین کاراکتر در لیست firstCommonChars است. در مثال شکل ۲-۴، اولین بیت‌های کاراکتر ۰۱۱ هستند، که مربوط به عددی است که اولین کاراکتر دارای کاراکتر با شاخص ۳ در firstCommonchars است که "o" است. کاراکتر دوم از ۴ بیت استفاده می‌کند، به این معنی که می‌تواند دو برابر کاراکتر اول را نشان دهد. در مثال، بیت‌های کاراکتر دوم ۱۱۰۱ هستند که با عدد ۱۳ مطابقت دارد. کاراکتر با شاخص ۱۳ در secondCommonChars کاراکتر "c" است. نتایج حاصل از فشرده‌سازی نشان می‌دهد که کاراکتر ادغام‌شده در شکل ۲-۴ نشان‌دهنده بیگرام "oc" است.

فصل چهارم

ارزیابی و پیاده‌سازی الگوریتم‌ها و بررسی نتایج

در فصل پنجم پایان‌نامه موردبررسی روند ارزیابی واحدهای مختلف واحد TCS و ارزیابی سیستم را به‌طورکلی شرح می‌دهد. ارزیابی ماژول‌های کدگذاری دیکشنری و کدگذاری هافمن شامل انتخاب پیاده‌سازی‌هایی با بیشترین نسبت فشرده‌سازی برای استفاده در TCS است. ارزیابی ACM شامل اندازه‌گیری نسبت فشرده‌سازی آن و اندازه‌گیری نسبت فشرده‌سازی Shoco و مقایسه نتایج است. ارزیابی کل سیستم (TCS) شامل اندازه‌گیری نسبت فشرده‌سازی آن و مقایسه عملکرد آن با سایر برنامه‌های فشرده‌سازی عمومی است.

۴-۱ ارزیابی ماژول‌های کدگذاری دیکشنری و کدگذاری هافمن

مجموعه‌ای از پیاده‌سازی‌های احتمالی برای کدگذاری دیکشنری و کدگذاری هافمن باهم مقایسه شده است. مجموعه داده‌ها روی هر یک از پیاده‌سازی‌ها آزمایش‌شده است. مجموعه داده‌ها به‌صورت محلی در کامپیوتر ذخیره‌شده و نسبت فشرده‌سازی اندازه‌گیری شده است. نسبت فشرده‌سازی اندازه‌گیری میزان کوچک‌تر بودن فایل فشرده در مقایسه با فایل فشرده نشده است و با تقسیم اندازه فشرده نشده بر اندازه فشرده‌شده یک فایل محاسبه‌شده است. به‌عنوان‌مثال، اگر یک فایل فشرده نشده ۱۰ مگابایت و فایل فشرده آن ۸ مگابایت است، نسبت پیاده‌سازی برای آن فایل خاص ۱,۲۵ است. اگر نسبت فشرده‌سازی ۱ باشد، فایل فشرده‌شده دارای اندازه یکسانی با فایل فشرده نشده است، بدین معنی که الگوریتم قادر به فشرده‌سازی فایل نیست. پیاده‌سازی‌هایی که بالاترین نسبت فشرده‌سازی را به‌طور متوسط برای هر متن در مجموعه داده دارند، به‌عنوان ماژول‌های کدگذاری دیکشنری و کدگذاری هافمن برای TCS انتخاب‌شده‌اند. میانگین نسبت فشرده‌سازی به‌عنوان معیار انتخاب شد.

۴-۲ انتخاب پیاده‌سازی کدگذاری دیکشنری

دریایان نامه بررسی‌شده چندین کدگذاری دیکشنری برای TCS در نظر گرفته‌شده است. این پیاده‌سازی‌ها برخی از الگوریتم‌های کدگذاری دیکشنری را نشان می‌دهند. الگوریتم دیکشنری LZW ، LZ77، LZ78 و Re-Pair است.

این الگوریتم‌ها توسط برنامه‌های فشرده‌سازی محبوب استفاده می‌شوند بنابراین موثربودن آن‌ها ثابت‌شده است. الگوریتم‌های کدگذاری دیکشنری بیشتری وجود دارد، اما توسط برنامه‌های فشرده‌سازی محبوب، استفاده نمی‌شود.

بسیاری از پیاده‌سازی‌های موجود در وب‌سایت کنترل نسخه GitHub مشخص نمی‌کند که راه‌حل آن‌ها دارای چه مجوزی است و بنابراین نمی‌توان آن‌ها را در نظر گرفت.

ماژول‌های استانداردی در پایتون برای برنامه‌های فشرده‌سازی محبوب مانند Gzip ، Bz2 یا Zipfile وجود دارد، اما این ماژول‌ها همه از چند الگوریتم در ترکیب با یکدیگر استفاده می‌کنند. به‌عنوان‌مثال، همه این ماژول‌ها از کدگذاری هافمن استفاده می‌کنند و برخی از کدگذاری دیکشنری استفاده می‌کنند، اما نمی‌توان فقط از الگوریتم کدگذاری دیکشنری و کدگذاری هافمن استفاده کرد. پیاده‌سازی‌هایی که مورداستفاده قرار می‌گیرند نمی‌توانند بسته‌های چند الگوریتم باشند.

۱-۲-۴ مقایسه پیاده‌سازی‌ها

در پایان‌نامه ذکرشده یک پیاده‌سازی پایتون و دو پیاده‌سازی C شرایط موردنیاز را برآورده می‌کند. آن‌ها دارای مجوز باز هستند، حتی برای فایل‌های بزرگ نیز به‌درستی کار می‌کنند و با متن کدگذاری‌های مختلف سازگار هستند. سپس پیاده‌سازی‌ها با یکدیگر مقایسه شده و نسبت فشرده‌سازی برای هر متن در مجموعه داده‌ها اندازه‌گیری شده است. پیاده‌سازی پایتون با پایتون نسخه ۲,۷,۱۷ اجرا می‌شود و پیاده‌سازی C با GNU Compiler Collection نسخه ۷,۵,۰ کامپایل شده است.

یک پیاده‌سازی الگوریتم LZ77 است و در پایتون نوشته‌شده است. پیاده‌سازی ۱۵ ژوئن ۲۰۲۰ بارگیری شده است. این پیاده‌سازی به کاربر این امکان را می‌دهد که اندازه پنجره الگوریتم (پنجره لغزان) (Ziv, Lempel, 1977) را تغییر دهد. اندازه پنجره تعیین می‌کند که چقدر الگوریتم متن در هر نقطه برای پیدا کردن زیررشته‌های منطبق تجزیه‌وتحلیل می‌کند. اندازه پنجره بزرگ‌تر نسبت فشرده‌سازی بیشتری را ارائه می‌دهد، اما الگوریتم را کندتر می‌کند. اندازه پنجره ۴۰۰ کاراکتر برای همه آزمایش‌ها استفاده‌شده، زیرابه گفته نویسنده پایان‌نامه این حداکثر اندازه ممکن برای پنجره بوده است.

پیاده‌سازی دیگر، پیاده‌سازی C الگوریتم LZW است. پیاده‌سازی در ۹ ژوئن ۲۰۲۰ بارگیری شد. الگوریتم LZW مانند LZ77 پنجره‌های لغزان استفاده نمی‌کند، اما در عوض زیررشته‌ها را به یک دیکشنری اضافه می‌کند که ممکن

است در متن تکرار شود یا نشود. اگر یک زیررشته تکرار شود، آن را با اشاره به ورودی در دیکشنری جایگزین می‌کند.

آخرین پیاده‌سازی C الگوریتم LZ77 است. پیاده‌سازی ۱۵ ژوئن ۲۰۲۰ بارگیری شد. درست مانند اجرای پایتون، امکان تغییر اندازه پنجره نیز وجود دارد. حداکثر اندازه پنجره ممکن برای این پیاده‌سازی ۱,۰۴۸,۵۷۶ کاراکتر است و بنابراین از این برای همه آزمایش‌ها استفاده‌شده است.

جدول ۱-۴. نتایج حاصل از پیاده‌سازی کدگذاری دیکشنری

Data set	مجموعه داده‌ها	LZ77 (Python)	LZW (C)	Lz77 (C)
XML file (21.6 MB)	فایل XML	۱.۴۷	۱.۹۲	۱.۸۵
cpo37 encoded XML file (18.1 MB)	فایل XML کدگذاری cpo37	۱.۴۵	۱.۶۱	۱.۷۵
C code file (64 KB)	فایل کد C	۱.۷۵	۲.۱۳	۲.۵
English book (680 KB)	کتاب انگلیسی	۱.۳	۱.۹۶	۱.۶۱
Italian book (626 KB)	کتاب ایتالیایی	۱.۳۳	۲.۱۳	۱.۵۶
Chinese book (285 KB)	کتاب چینی	۱.۱۶	۱.۵۹	۱.۳۴
Average	میانگین	۱.۴۱	۱.۸۹	۱.۷۷

جدول ۱-۴ نشان‌دهنده نسبت فشرده‌سازی سه کدگذاری دیکشنری برای هر متن در مجموعه داده است. اجرای پایتون از الگوریتم LZ77 بدترین نسبت فشرده‌سازی را دارد. پیاده‌سازی C برای الگوریتم LZ77 دارای نسبت فشرده‌سازی بسیار بهتری است، حتی اگر از الگوریتم LZ77 نیز استفاده شود. پیاده‌سازی LZW نسبت به پیاده‌سازی C در LZ77 برای برخی از متون دارای نسبت فشرده‌سازی بالاتری است، اما برای متون دیگر نسبت کمتری دارد. پیاده‌سازی که بهترین نتایج متوسط را از آزمایش داشته باشد، به‌عنوان ماژول کدگذاری دیکشنری برای TCS انتخاب‌شده بنابراین پیاده‌سازی C الگوریتم LZW به‌عنوان ماژول کدگذاری دیکشنری استفاده‌شده است.

۴-۳ انتخاب پیاده‌سازی کدگذاری هافمن

تعدادی پیاده‌سازی منبع باز از الگوریتم کدگذاری هافمن پیداشده و مورد ارزیابی قرارگرفته، اکثر پیاده‌سازی‌هایی که پیدا می‌شوند مشخص نمی‌کنند که راه‌حل آن‌ها دارای چه مجوزی است و بنابراین نمی‌توان آن‌ها را در نظر

گرفت. ماژول‌های استاندارد پایتون برای برنامه‌های فشرده‌سازی محبوب که از الگوریتم هافمن استفاده می‌کنند وجود دارد، اما با الگوریتم‌های دیگر ترکیب‌شده است. دو پیاده‌سازی پایتون یافت می‌شود که قادر به فشرده‌سازی متن ASCII و UTF-8 هستند، اما هیچ‌یک از آن‌ها قادر به فشرده‌سازی متن کد شده cpo37 نیستند. پیاده‌سازی در C یافت شد، اما قادر به ساخت کد منبع نیست، بنابراین نمی‌توان آن را ارزیابی کرد.

۴-۳-۱ مقایسه پیاده‌سازی‌ها

در پایان‌نامه موردبررسی سه پیاده‌سازی الزامات را برآورده می‌کنند و با یکدیگر مقایسه شده‌اند. یک پیاده‌سازی پایتون و دو پیاده‌سازی C. پیاده‌سازی پایتون با پایتون نسخه ۳,۶,۹ و پیاده‌سازی C با GNU Compiler Collection نسخه ۷,۵,۰ کمپایل شده است.

پیاده‌سازی پایتون Dahuffman نام دارد و توسط کاربر GitHub، Stefaan Lippens ساخته‌شده است. پیاده‌سازی ۲۶ ژوئیه ۲۰۲۰ بارگیری شده است. پیاده‌سازی این قابلیت را دارد که بر اساس فهرستی از تکرارهای نماد، یک جدول کد بسازد. فراوانی نماد می‌تواند توسط کاربر ارائه شود یا پیاده‌سازی می‌تواند آن را از یک متن ورودی محاسبه کند. اگر فراوانی نماد از یک متن ورودی محاسبه شود، پیاده‌سازی ابتدا یک جدول کد ایجاد می‌کند، و سپس متن را با استفاده از جدول کد ذکرشده فشرده می‌کند.

دو پیاده‌سازی C، توسط کاربران GitHub، Gagarine Yaikhom و Doug Richardson، جداول کد را به‌طور خودکار در طول فشرده‌سازی ایجاد می‌کنند. پیاده‌سازی Yaikhom در ۲۶ ژوئیه ۲۰۲۰، و اجرای Richardson در ۲۸ ژوئیه ۲۰۲۰ بارگیری شده است.

جدول ۲-۴. نتایج حاصل از پیاده‌سازی کدگذاری هافمن

Data set	مجموعه داده‌ها	Dahuffman	Yaikhom C impl.	Richardson C impl.
XML file (21.6 MB)	فایل XML	۱.۵۱	۱.۵	۱.۵
cpo37 encoded XML file (18.1 MB)	فایل XML کدگذاری cpo37	۱.۲	۱.۲	۱.۲
C code file (64 KB)	فایل کد C	۱.۴۸	۱.۴۷	۱.۴۷
English book (680 KB)	کتاب انگلیسی	۱.۸	۱.۷	۱.۷
Italian book (626 KB)	کتاب ایتالیایی	۱.۹۵	۱.۳۷	۱.۳۷
Chinese book (285 KB)	کتاب چینی	۲.۳۲	۱.۳۱	۱.۳۱
Average	میانگین	۱.۷۱	۱.۴۹	۱.۴۹

جدول ۲-۴ نشان‌دهنده نسبت فشرده‌سازی سه برنامه‌نویسی هافمن برای هر متن در مجموعه داده است که نشان می‌دهد، برای اکثر متون تفاوت چندانی بین پیاده‌سازی‌ها وجود ندارد. درواقع، دو پیاده‌سازی C نتایج دقیقاً یکسانی را به دست آوردند، اگرچه دو پیاده‌سازی متفاوت توسط افراد مختلف است. پیاده‌سازی پایتون، Dahuffman، نسبت فشرده‌سازی کمی بهتر از دو پیاده‌سازی C برای کتاب انگلیسی و ایتالیایی، اما نسبت بهتری برای کتاب چینی دارد.

در مورد سه متن دیگر، Dahuffman کم‌وبیش نتایج مشابهی با سایر پیاده‌سازی‌ها دارد. این می‌تواند به این دلیل باشد که این کتاب‌ها بیشتر از سایر متون دارای کاراکتر UTF-8 هستند، و Dahuffman در فشرده‌سازی کاراکترهای غیر اسکی کار آیی بهتری دارد. کتاب چینی به‌طور کامل از کاراکترهای UTF-8 تشکیل‌شده است، بنابراین این منطقی‌ترین نظریه به نظر می‌رسد. Dahuffman، دارای بالاترین نسبت فشرده‌سازی متوسط پیاده‌سازی‌ها است و بنابراین به‌عنوان ماژول کدگذاری هافمن برای TCS استفاده‌شده است.

۴-۴ ارزیابی ACM

برنامه فشرده‌سازی TCS شبیه به Shoco است. هر دو TCS و Shoco برنامه‌های فشرده‌سازی متن بدون اتلاف هستند که در فشرده‌سازی متن ASCII تخصص دارند. یک تفاوت اصلی این است که Shoco یک الگوریتم است درحالی‌که TCS یک‌راه‌حل فشرده‌سازی است که از ۳ الگوریتم کدگذاری LZW، ACM و Huffman تشکیل‌شده است. ACM عملکرد و راه‌حل بسیار مشابهی با Shoco دارد. پایان‌نامه موردبررسی به‌منظور مقایسه عادلانه، فقط ACM در این قسمت با Shoco مقایسه کرده است.

مقاله موردبحث در ۲-۵-۲ یکی دیگر از برنامه‌های فشرده‌سازی ASCII است. بااین‌حال، این راه‌حل در مقایسه با ACM به دلیل محدودیت‌های موجود نیست. راه‌حل پیشنهادی پایان‌نامه موردنظر فقط قادر به فشرده‌سازی متن ASCII است و هیچ کدگذاری دیگری ندارد. از متون مورداستفاده در مجموعه داده‌ها، فقط فایل کد C با راه‌حل سازگار است، بنابراین نمی‌توان مقایسه عادلانه‌ای با ACM داشت.

ACM از یک مدل فشرده‌سازی پیش‌فرض استفاده می‌کند که برای داده‌هایی که قرار است فشرده شود آموزش ندیده است. یک مدل فشرده‌سازی لیستی از کاراکترها، نمادها یا بیگرام ها است که اغلب در یک متن استفاده می‌شود. Shoco این قابلیت را دارد که یک مدل فشرده‌سازی را برای یک فایل متنی خاص آموزش دهد، به این معنی که نمادها و بیگرام های موجود در مدل اغلب در یک متن استفاده می‌شود. ACM این قابلیت را ندارد و از

یک مدل فشرده‌سازی عمومی استفاده می‌کند که برای کلمات انگلیسی بهینه‌شده است. در مقایسه، شوکو هم با مدل پیش‌فرض خود و هم با یک مدل آموزش‌دیده آزمایش می‌شود.

۱-۴-۴ نتایج حاصل از مقایسه ACM وShoco

جدول ۳-۴. نتایج حاصل از مقایسه ACM و Shoco

Data set	مجموعه داده‌ها	ACM	Shoco پیش‌فرض	Shoco آموزش‌دیده
XML file (21.6 MB)	فایل XML	۱.۱۹	۱.۲۳	۱.۴
cpo37 encoded XML file (18.1 MB)	فایل XML کدگذاری cpo37	۱	۰.۵۹	۰.۷۵
C code file (64 KB)	فایل کد C	۱.۲۱	۱.۱۷	۱.۳۵
English book (680 KB)	کتاب انگلیسی	۱.۴۶	۱.۳۲	۱.۵۶
Italian book (626 KB)	کتاب ایتالیایی	۱.۲۹	۱.۱۵	۱.۵
Chinese book (285 KB)	کتاب چینی	۱	۰.۵۳	۰.۷۵
Average	میانگین	۱.۱۹	۱	۱.۲۲

فایل XML با کد cpo37 و کتاب چینی دو متنی هستند که نسبت فشرده‌سازی ۱ برای هر دو الگوریتم دارند، به این معنی که حجم فایل فشرده کوچک‌تر نمی‌شود. این متون به ترتیب از نویسه cpo37 و UTF-8 تشکیل‌شده‌اند. ACM و Shoco هر دو فقط قادر به فشرده‌سازی کاراکترهای ASCII هستند، و بنابراین رمزگذاری‌های دیگر می‌توانند خروجی را بزرگ‌تر کنند. برای هر دو Shoco آموزش‌دیده و پیش‌فرض، نسبت فشرده‌سازی کمتر از ۱ است، به این معنی که فایل فشرده بزرگ‌تر از فایل اصلی است.

ACM دارای نسبت ۱ است، به این معنی که فایل فشرده دارای اندازه یکسان با فایل اصلی است. دلیل این امر این است که ACM آزمایشی را برای بررسی بزرگ‌تر بودن فایل فشرده از نسخه اصلی انجام می‌دهد. اگر چنین باشد، فشرده‌سازی کنار گذاشته می‌شود و فایل فشرده بدون تغییر باقی می‌ماند. Shoco این آزمایش را ندارد و بنابراین یک فایل فشرده بزرگ‌تر دریافت می‌کند.

کتاب انگلیسی بالاترین نسبت فشرده‌سازی را برای هر دو الگوریتم دارد. مدل‌های پیش‌فرض این دو الگوریتم برای کلمات انگلیسی بهینه‌شده‌اند، بنابراین نسبت بالایی برای کتاب انگلیسی به دست می‌آورند. این نسبت هنگام استفاده از یک مدل آموزش‌دیده برای شوکو حتی بیشتر است زیرا مدل برای آن متن خاص سفارشی‌شده است.

۴-۵ ارزیابی TCS

ارزیابی TCS اندازه‌گیری نسبت فشرده‌سازی در مجموعه داده‌ها و سپس مقایسه نتایج با سایر برنامه‌های فشرده‌سازی عمومی است. وقتی TCS ارزیابی‌شده است، از هر سه ماژول استفاده کرده است یعنی خروجی از یک ماژول به ماژول بعدی ارسال می‌شود، نسبت فشرده‌سازی برای خروجی ماژول قبلی اندازه‌گیری شده است. درنهایت، نسبت تراکم کل اندازه‌گیری می‌شود. این نسبت اندازه فایل اصلی در مقایسه با اندازه نهایی فایل فشرده است. ابتدا ACM، سپس کدگذاری دیکشنری (LZW) و درنهایت کدگذاری هافمن استفاده‌شده است.

۴-۵-۱ نتایج حاصل از ارزیابی TCS

جدول ۴-۴. نتایج حاصل از ارزیابی TCS

Data set	مجموعه داده‌ها	ACM	LZW	Huffman coding	Total ratio
XML file (21.6 MB)	فایل XML	۱.۱۹	۱.۶	۱.۰۱	۱.۹۳
cpo37 encoded XML file (18.1 MB)	فایل XML کدگذاری cpo37	۱	۱.۶	۱.۰۱	۱.۶۲
C code file (64 KB)	فایل کد C	۱.۲۱	۱.۸۳	۱.۰۱	۲.۲۳
English book (680 KB)	کتاب انگلیسی	۱.۴۶	۱.۳۶	۱	۱.۹۹
Italian book (626 KB)	کتاب ایتالیایی	۱.۲۹	۱.۵۷	۱	۲.۰۲
Chinese book (285 KB)	کتاب چینی	۱	۱.۵۹	۱	۱.۵۹
Average	میانگین	۱.۱۹	۱.۵۹	۱.۰۱	۱.۹

همان‌طور که گفته شد، نسبت فشرده‌سازی اندازه‌گیری شده برای هر ماژول بر اساس خروجی ماژول قبلی است. "نسبت کل" نسبت فشرده‌سازی است که از TCS (هر سه ماژول) برای متون موجود در مجموعه داده به دست می‌آید. نسبت کل نتیجه‌ای است که در ارزیابی با سایر برنامه‌های فشرده‌سازی در بخش بعدی مورداستفاده قرار می‌گیرد.

همان‌طور که جدول ۴-۴ نشان می‌دهد، برخی از فایل‌ها بیشتر از ACM و برخی دیگر از LZW سود بیشتری می‌برند. به‌عنوان‌مثال فایل کد C کتاب انگلیسی از ACM نسبت کوچک‌تری از LZW دارد. به این دلیل است که کتاب انگلیسی از کاراکترهای مکرر زبان انگلیسی بیشتری استفاده می‌کند و بنابراین نسبت بهتری از ACM دریافت می‌کنند، درحالی‌که فایل کد C دارای کلمات و زیررشته‌های تکراری‌تر است و بنابراین نسبت بهتری از LZW دریافت می‌کند.

جدول ۴-۴ همچنین نشان می‌دهد که کدگذاری هافمن تقریباً هیچ فشرده‌سازی را برای هیچ‌یک از فایل‌ها به دست نمی‌آورد. این می‌تواند به این دلیل باشد که برنامه‌نویسی هافمن برای فشرده‌سازی داده‌های باینری طراحی نشده است. همچنین ممکن است خروجی دودویی از LZW بسیار تصادفی و نامنظم باشد تا کدگذاری هافمن نتواند به فشرده‌سازی برسد. خروجی LZW قبلاً توسط ACM فشرده‌شده است و ممکن است ترکیب این دو ماژول نیاز به کدگذاری هافمن را برطرف کند. نتایج حاصل از این ارزیابی نیاز به ارزیابی جدیدی دارد که فقط از کدگذاری LZW و هافمن استفاده می‌شود، تا به پاسخ این سؤال برسیم که آیا ACM قابلیت‌های ماژول کدگذاری هافمن را مختل می‌کند یا این ماژول قادر به فشرده‌سازی داده‌های باینری نیست.

جدول ۵-۴. نتایج حاصل از ارزیابی ماژول کدگذاری LZW و هافمن

Data set	مجموعه داده‌ها	LZW	Huffman coding	Total ratio
XML file (21.6 MB)	فایل XML	۱.۹۲	۱	۱.۹۲
C code file (64 KB)	فایل کد C	۲.۱۳	۱.۰۱	۲.۱۴
English book (680 KB)	کتاب انگلیسی	۱.۹۶	۱	۱.۹۶
Italian book (626 KB)	کتاب ایتالیایی	۲.۱۳	۱	۲.۱۳
Average	میانگین	۲.۰۴	۱	۲.۰۴

فایل XML کد شده cpo37 و کتاب چینی در این ارزیابی گنجانده نشده‌اند زیرا ACM هیچ تأثیری بر روی این فایل‌ها نخواهد داشت و هدف از این ارزیابی مقایسه با نتایجی است که ACM روی آن تأثیر گذاشته است. جدول ۵-۴ نشان می‌دهد که حتی وقتی از ACM استفاده نمی‌شود، ماژول کدگذاری هافمن عملاً قادر به فشرده‌سازی داده‌های باینری نیست.

۲-۵-۴ نتایج حاصل از مقایسه TCS

جدول ۶-۴ نشان می‌دهد که TCS نسبت به سایر برنامه‌ها برای هر فایل در مجموعه داده نسبت فشرده‌سازی کمتری دارد. Bzip2 وZip-۷ بالاترین میانگین نسبت فشرده‌سازی رادارند زیرا از الگوریتم‌های متفاوتی نسبت به Zip و Gzip استفاده می‌کنند که بر اساس الگوریتم DEFLATE است (Korpela, 2006). TCS از راه‌حلی مشابه DEFLATE، اما با ACM به‌عنوان پیش پردازنده استفاده می‌کند. الگوریتم DEFLATE ترکیبی از رمزگذار دیکشنری LZ77 و کدگذاری هافمن است، درحالی‌که TCS از رمزگذار دیکشنری LZW و کدگذاری هافمن استفاده می‌کند.

جدول ۶-۴. نتایج حاصل از مقایسه TCS

Data set	مجموعه داده‌ها	TCS	Bzip2	Zip	۷-Zip	Gzip
XML file (21.6 MB)	فایل XML	۱.۹۳	۳.۷۹	۲.۹۶	۴.۲۴	۲.۹۶
cpo37 encoded XML file (18.1 MB)	فایل XML کدگذاری cpo37	۱.۶۲	۳.۱۲	۲.۴۵	۳.۴۸	۲.۴۵
C code file (64 KB)	فایل کد C	۲.۲۳	۴.۲۲	۳.۸۸	۴.۲۵	۳.۹۱
English book (680 KB)	کتاب انگلیسی	۱.۹۹	۳.۵۲	۲.۶۵	۳.۱۸	۲.۶۵
Italian book (626 KB)	کتاب ایتالیایی	۲.۰۲	۳.۵۴	۲.۶۶	۳.۱۴	۲.۶۶
Chinese book (285 KB)	کتاب چینی	۱.۵۹	۲.۵۹	۲.۰۱	۲.۳۵	۲.۰۱
Average	میانگین	۱.۹	۳.۴۶	۲.۷۷	۳.۴۴	۲.۷۷

جدول ۵-۴ نشان می‌دهد که ماژول کدگذاری هافمن مورداستفاده برای TCS با ماژول LZW سازگار نیست و بنابراین TCS جایگزینی برای الگوریتم DEFLATE ندارد. برای بررسی اینکه آیا TCS در صورت جایگزینی ماژول کدگذاری LZW و Huffman که در TCS با یک الگوریتم DEFLATE خالص جایگزین شده است، نتایج بهتری خواهد گرفت یا خیر، باید ارزیابی جدیدی انجام شود. در این ارزیابی، از برنامه‌های Zip و Gzip با ACM به‌عنوان پیش پردازنده استفاده‌شده است. این ارزیابی به این سؤال پاسخ می‌دهد که آیا استفاده از ACM به‌عنوان پیش پردازنده باعث افزایش نسبت فشرده‌سازی الگوریتم DEFLATE می‌شود یا خیر.

جدول ۷-۴. نتایج حاصل از مقایسه ACM + DEFLATE

Data set	مجموعه داده‌ها	Zip	ACM + Zip	Gzip	ACM + Gzip
XML file (21.6 MB)	فایل XML	۲.۹۶	۲.۹۲	۲.۹۶	۲.۹۲
C code file (64 KB)	فایل کد C	۳.۸۸	۳.۷۷	۳.۹۱	۳.۸۲
English book (680 KB)	کتاب انگلیسی	۲.۶۵	۲.۶۶	۲.۶۵	۲.۶۶
Italian book (626 KB)	کتاب ایتالیایی	۲.۶۶	۲.۵۸	۲.۶۶	۲.۵۸
Average	میانگین	۳.۰۴	۲.۹۸	۳.۰۵	۳

مانند جدول ۵-۴، فایل XML کد شده cpo37 و کتاب چینی در این ارزیابی گنجانده نشده‌اند زیرا ACM هیچ تأثیری بر روی این فایل‌ها نخواهد داشت. همان‌طور که جدول ۷-۴ نشان می‌دهد، هنگامی‌که ACM به‌عنوان پیش پردازنده Zip یا Gzip استفاده می‌شود، نسبت فشرده‌سازی کمی پایین‌تر است. ACM فقط نسبت به کتاب انگلیسی

نسبت فشرده‌سازی را تا حدی بیشتر نشان می‌دهد، اما افزایش ۰,۰۱ نسبت فشرده‌سازی ناچیز است. استفاده از ACM به‌علاوه الگوریتم DEFLATE که Zip و Gzip استفاده می‌کند نسبت فشرده‌سازی بسیار بالاتری نسبت به TCSدارد.

۴-۶ بررسی نتایج پایان‌نامه

فصل ۶ پایان‌نامه موردبحث به بررسی نتایج حاصل از ارزیابی‌ها را پوشش می‌دهد. همچنین به کاستی‌های راه‌حل، مشارکت این پایان‌نامه و این‌که در چه شرایطی می‌تواند مورداستفاده قرار گیرد، می‌پردازد.

۴-۶-۱ کار آیی TCS

TCS مدعی الگوریتم DEFLATE بود، اما با تمرکز بر فشرده‌سازی متن ASCII. ازنظر مفهومی، TCS از تکنیک‌های مشابه DEFLATE (کدگذاری دیکشنری و کدگذاری هافمن) استفاده می‌کند اما ACM را به‌عنوان یک گام جدید در کانال فشرده‌سازی معرفی می‌کند. تفاوت بین DEFLATE و TCS در این است که DEFLATE از رمزگذار دیکشنری LZ77 استفاده می‌کند درحالی‌که TCS از رمزگذار دیکشنری LZW استفاده می‌کند. ارزیابی TCS نشان می‌دهد که جایگزین DEFLATE که TCS از آن استفاده می‌کند ناقص است. بنابراین، استفاده از TCS (در حالت فعلی آن) به‌عنوان جایگزینی برای الگوریتم DEFLATE گزینه مناسبی نیست. مقایسه برنامه‌های فشرده‌سازی در جدول ۶-۴ همچنین نشان داد که برنامه‌هایی که از راه‌حل‌های جایگزین برای الگوریتم DEFLATE استفاده می‌کردند نسبت فشرده‌سازی بالاتری داشتند. بنابراین، حتی اگر TCS دارای یک الگوریتم کارکرد DEFLATE باشد، بازهم نمی‌تواند با نسبت‌های فشرده‌سازی Bzip2 وZip-۷ رقابت کند.

۴-۶-۲ مقایسه ACM وShoco

شوکو مدعی نزدیک ACM است و کار مرتبطی است که بیشتر در پایان‌نامه بررسی‌شده، موردبحث قرارگرفته است. هر دو الگوریتم دارای پیش‌فرض یکسان و راه‌حل‌های فنی مشابه هستند، اما یکسان نیستند. یک تفاوت پیچیدگی مدل‌های فشرده‌سازی است که دو الگوریتم از آن استفاده می‌کنند. تفاوت دیگر کار آیی الگوریتم‌ها است. الگوریتم شوکو به‌عنوان "کمپرسور سریع برای رشته‌های کوتاه" توصیف می‌شود و به نظر می‌رسد نقطه اصلی فروش شوکو سرعت آن باشد. ACM برای تأخیر اندازه‌گیری نشده است، اما آزمایش نشان داده است که Shoco (با مدل

فشرده‌سازی پیش‌فرض) سریع‌تر از ACM است. دلیل اینکه ACM برای تأخیر اندازه‌گیری نشده این است که تمرکز ACM بر قابلیت فشرده‌سازی آن است و نه بر کار آیی آن. ACM به‌عنوان اثبات این مفهوم بود که الگوریتم فشرده‌سازی ASCII ازلحاظ نظری امکان‌پذیر است. این قبل از اینکه شوکو برای نویسنده شناخته شود، ثابت شد که فشرده‌سازی ASCII امکان‌پذیر است.

کاستی‌های ACM

تکنیک فشرده‌سازی که ACM برای کاراکترهای غیر ASCII استفاده می‌کند بهینه نیست. برخی از کاراکترهای UTF-8 مانند ایموجی ها می‌توانند تا ۴ بایت در یک فایل فشرده نشده استفاده کنند. این کاراکترها در صورت فشرده‌سازی از ۹ بایت استفاده می‌کنند. فشرده‌سازی غیر بهینه با تبدیل کاراکتر به عددی که آن را نشان می‌دهد و نوشتن رقم عدد برای رقم ایجاد می‌شود. این می‌تواند بهینه‌سازی شود، اما محدودیت زمانی منجر به این مسئله می‌شود که هنوز باید برطرف شود. فایلی که با ACM فشرده می‌شود دارای یک خط اضافه‌شده یا ۱ یا o می‌باشد. این به این معنی است که آیا فایل تغییر کرده است یا نه، و هنگامی‌که فایل از حالت فشرده خارج می‌شود، حذف می‌شود. مشکلی که ممکن است رخ دهد این است که اگر یک فایل دارای خط ۱ یا o به‌عنوان آخرین خط باشد، و مشخص نیست که فایل فشرده است یا نه. اگر کاربر این فایل فشرده نشده را با یک فایل فشرده اشتباه کند، پس از فشرده‌سازی تغییر می‌کند. یک فایل فشرده به‌عنوان یک فایل متنی (txt.) ذخیره می‌شود و بنابراین احتمالاً دارای همان نوع فایل با یک فایل فشرده نشده است، بنابراین نمی‌توان گفت آیا فایل فشرده است یا نه.

ACM به زبان برنامه‌نویسی پایتون نوشته‌شده است، که در زمان اجرا به‌طور قابل‌توجهی برای مثال کندتر از زبان C است. زبان برنامه‌نویسی C به‌طورکلی سریع‌تر از پایتون است، به همین دلیل برنامه‌های پایتون اغلب به دلایل کار آیی ماژول‌های C را ادغام می‌کنند (Beazley, 2009). اگر ACM به زبان C نوشته می‌شد، به‌طور بالقوه می‌توانست سریع‌تر باشد.

فصل پنجم

جمع‌بندی و پیشنهادها

۵-۱ مقدمه

پایان‌نامه موردبررسی راه‌حلی با TCS برای فشرده‌سازی متن با تمرکز ویژه‌بر فشرده‌سازی متن ASCII ارائه می‌دهد. TCS مشکلات ذخیره و انتقال زمان زیادی از داده‌های متنی را برطرف می‌کند. TCS حجم فایل‌ها را کاهش می‌دهد بنابراین به فضای ذخیره‌سازی کمتری نیاز دارند. علاوه بر این، TCS هنگام بارگیری یا انتقال فایل فشرده از طریق اینترنت، پهنای باند را کاهش می‌دهد. TCS برای متن ASCII سفارشی‌شده است بااین‌حال برای کاربر روی هر شکل از متن طراحی‌شده است. جاه‌طلبی TCS فشرده‌سازی فایل‌هایی برای کاربرانی است که با مقادیر زیادی متون ASCII یا ASCII سنگین سروکار دارند. TCS از سه ماژول ACM، ماژول کدگذاری دیکشنری و ماژول کدگذاری هافمن تشکیل‌شده است. TCS مبتنی بر الگوریتم DEFLATE است که همچنین کدگذاری دیکشنری و کدگذاری هافمن را ترکیب می‌کند. TCS از ACM به‌عنوان پیش پردازنده جایگزین DEFLATE برای افزایش فشرده‌سازی برای آزمایش ASCII استفاده می‌کند. ACM توسط نویسنده به‌طور خاص برای این پروژه توسعه‌یافته است، درحالی‌که دیکشنری و ماژول‌های کدگذاری هافمن پیاده‌سازی شده توسط افراد دیگر است. ماژول‌های خاص مورد ارزیابی قرارگرفته‌اند، و TCS ارزیابی‌شده و با برنامه‌های فشرده‌سازی عمومی مورد مقایسه قرارگرفته است.

۵-۲ نتایج حاصل از تحقیق

نتایج حاصل از ارزیابی TCS نشان می‌دهد که در حالت فعلی، مدعی سایر برنامه‌های فشرده‌سازی عمومی نیست. مقایسه فشرده‌سازی اسکی با الگوریتم Shoco نشان می‌دهد که هنگام استفاده از مدل‌های فشرده‌سازی عمومی، ACM نسبت فشرده‌سازی متوسط بهتری دارد. بنابراین ACM می‌تواند مدعی رشته تخصصی‌تر فشرده‌سازی ASCII باشد. ارزیابی ACM (جدول ۳-۴) نشان می‌دهد که می‌توان آن را در هر شکلی از متن بدون افزایش اندازه فشرده استفاده کرد، اما نسبت فشرده‌سازی قابل‌توجهی فقط در متون سنگین ASCII به دست می‌آید. بااین‌وجود، نسبت فشرده‌سازی ACM در مقایسه با برنامه‌های فشرده‌سازی عمومی بسیار کوچک است (جدول ۶-۴). ترکیب ACM با برنامه‌های فشرده‌سازی مبتنی بر DEFLATE (جدول ۷-۴) نسبت فشرده‌سازی متوسط را فقط تحت DEFLATE

نشان می‌دهد، با یک مثال از مزیت جزئی ACM. ممکن است بهبود الگوریتم فشرده‌سازی ACM نسبت فشرده‌سازی آن را افزایش دهد و حتی ممکن است از DEFLATE فراتر رود (درصورتی‌که ACM به‌عنوان پیش پردازنده DEFLATE استفاده شود).

۱-۲-۵ پاسخ به سؤالات تحقیق

"چه ترکیبی از تکنیک‌ها می‌تواند نسبت فشرده‌سازی متن ASCII را بهبود بخشد؟"

بدیهی است که ACM برای فایل‌های ASCII به فشرده‌سازی دست می‌یابد، اگرچه برخی از فایل‌های متنی نسبت فشرده‌سازی بیشتری نسبت به سایرین دارند (جدول۳-۴). علیرغم شایستگی ACM، استفاده از آن در ترکیب با کدگذاری دیکشنری و ماژول کدگذاری هافمن (و همچنین برنامه‌های فشرده‌سازی مبتنی بر DEFLATE) نسبت فشرده‌سازی متن ASCII را بهبود نداده است. دلیل این امر این است که خروجی ACM کمتر از متن ساده قابل فشرده‌سازی است. ACM فقط به‌عنوان پیش پردازنده برنامه‌های فشرده‌سازی مبتنی بر DEFLATE آزمایش‌شده است، زیرا TCS جایگزین DEFLATE بود. ممکن است ترکیب الگوریتم‌های دیگر (مانند الگوریتم‌های مورداستفاده در Bzip2 و Zip-۷) با ACM به‌عنوان پیش پردازنده بتواند نسبت فشرده‌سازی متن ASCII را افزایش دهد. این کار آینده TCS است.

"آیا استفاده از فشرده‌سازی ASCII در ترکیب با کدگذاری دیکشنری و کدگذاری هافمن از نسبت فشرده‌سازی DEFLATE فراتر می‌رود؟"

در حالت فعلی، TCS (ACM، ماژول کدگذاری دیکشنری و ماژول کدگذاری هافمن) از نسبت فشرده‌سازی DEFLATE فراتر نمی‌رود. همان‌طور که گفته شد، ماژول برنامه‌نویسی هافمن TCS با ماژول برنامه‌نویسی دیکشنری سازگار نیست و بنابراین TCS جایگزین DEFLATE نیست. در عوض، ACM در ترکیب با برنامه‌نویسی دیکشنری و برنامه‌نویسی هافمن برنامه‌های فشرده‌سازی مبتنی بر DEFLATE در جدول ۷-۴ آزمایش‌شده است. بر اساس میانگین نسبت فشرده‌سازی از ۴ متن مجموعه داده، با استفاده از ACM در ترکیب با رمزگذاری دیکشنری و هافمن کدگذاری از نسبت فشرده‌سازی DEFLATE فراتر نمی‌رود.

"آیا این ترکیب فشرده‌سازی متن نسبت فشرده‌سازی بالاتری نسبت به برنامه‌های فشرده‌سازی عمومی برای متن ASCII به دست می‌آورد؟"

در مقایسه بین TCS و ۴ برنامه فشرده‌سازی عمومی در جدول ۶-۴، ۴ متون بیشتر شامل کاراکترهای ASCII هستند. همان‌طور که گفته شد، نه ترکیب TCS و نه ACM + DEFLATE نسبت فشرده‌سازی بالاتری نسبت به برنامه‌های فشرده‌سازی مبتنی بر DEFLATE برای متون ASCII ندارند. علاوه بر این، Bzip2 و Zip-۷ از هر دو برنامه فشرده‌سازی مبتنی بر DEFLATE و TCS برای هر متنی در مورداستفاده بهتر هستند. درنتیجه، TCS نسبت فشرده‌سازی بیشتر از برنامه‌های فشرده‌سازی عمومی برای متن ASCII به دست نمی‌آورد.

"آیا این ترکیب فشرده‌سازی متن نسبت فشرده‌سازی بالاتری نسبت به راه‌حل‌های فشرده‌سازی که روی متن ASCII تخصص دارند، به دست می‌آورد؟"

در پایان‌نامه موردبررسی، ACM تنها با فشرده‌سازی اسکی توسط الگوریتم Shoco مقایسه شده است. تا آنجا که نویسنده مطلع است، Shoco تنها الگوریتم فشرده‌سازی اسکی دیگری است که برای فشرده‌سازی متن اسکی سفارشی‌شده است و همچنین قادر به فشرده‌سازی کدگذاری‌های دیگر است. راه‌حل‌های فشرده‌سازی ASCII که قادر به فشرده‌سازی سایر کدگذاری‌ها نیستند، مانند مقاله موردبحث در قسمت ۲-۵-۲، برای مقایسه در نظر گرفته نشده است، زیرا تعداد بسیار کمی فایل‌های متنی امروزه فقط از کاراکترهای ASCII استفاده می‌کنند در مقایسه ACM و Shoco، ACM نسبت فشرده‌سازی بالاتری نسبت به Shoco دارد – با استفاده از مدل فشرده‌سازی پیش‌فرض خود، اما ACM نسبت به Shocoنسبت کمی کمتر دارد – هنگام استفاده از یک مدل آموزش‌دیده. در بخش ۱-۴-۴ ذکرشده است، مدل‌های پیش‌فرض و آموزش‌دیده شوکو دارای مزایا و معایبی هستند. هر دو مدل قابل‌اجرا هستند و بسته به شرایط می‌توان از آن‌ها استفاده کرد. بنابراین، این سؤال تحقیق را نمی‌توان با بله یا خیر قطعی پاسخ داد.

۳-۵ پیشنهادها و کارهای آینده

کارهای آینده در TCS شامل یافتن یک ماژول کدگذاری هافمن است که می‌تواند خروجی دودویی را از ماژول کدگذاری دیکشنری فشرده کند. دستیابی به این امر می‌تواند قابلیت فشرده‌سازی TCS را با الگوریتم DEFLATE برابر کند. آزمایش الگوریتم‌هایی که توسط DEFLATE برای بهبود نسبت فشرده‌سازی استفاده‌نشده است، نیز کار آینده است.

برای افزایش نسبت فشرده‌سازی می‌توان پیشرفت‌هایی در ACM انجام داد. ACM می‌تواند فشرده‌سازی کاراکترهای غیر ASCII را بهبود بخشد، زیرا تکنیک فعلی بهینه نیست. افزودن قابلیت ایجاد مدل‌های آموزش‌دیده ممکن است

نسبت فشرده‌سازی آن را نیز به میزان قابل‌توجهی افزایش دهد. با این پیشرفت‌ها، ممکن است ACM در صورت استفاده از پیش پردازنده، نسبت فشرده‌سازی برنامه‌های فشرده‌سازی عمومی را افزایش دهد.

همچنین می‌توان افزایش تأخیر ACM را بهبود داد. کد را می‌توان در C بازنویسی کرد و بهینه‌سازی را برای فرآیند فشرده‌سازی و خروج از فشرده‌سازی انجام داد. اگر ACM بیشتر بر زمان تأخیر تمرکز داشت تا نسبت فشرده‌سازی، استفاده از ACM در شرایطی که سرعت در اولویت بیشتری نسبت به ضریب فشرده‌سازی است، مفید خواهد بود. به‌عنوان‌مثال، می‌توان آن را در ترکیب با کمپرسور اسنپی مبتنی بر LZ77 درزمینه و پس‌زمینه سرور استفاده کرد. Snappy نسبت فشرده‌سازی را ازنظر سرعت به خطر می‌اندازد و در فایل‌های متنی نسبت به فایل‌های باینری به نسبت بهتری دست می‌یابد. نتایج جدول ۴-۴ نشان می‌دهد که رمزگذار دیکشنری (LZW) می‌تواند نسبت فشرده‌سازی مناسبی را با ACM به‌عنوان پیش پردازنده به دست آورد، بنابراین استفاده از ACM به‌عنوان پیش پردازنده Snappy می‌تواند نسبت فشرده‌سازی را افزایش دهد.

۴-۵ جمع‌بندی

در آغاز مطالعه این پایان‌نامه به‌عنوان تحقیق سمینار دانشجویی تصور متفاوتی نسبت به فشرده‌سازی داده‌ها و روش‌های تحقیق در این زمینه داشتم. با انجام مطالعه پایان‌نامه با موضوع "بهبود نسبت فشرده‌سازی متن برای متن ASCII" اطلاعات مفیدی در این زمینه به دست آوردم. با مطالعات مرتبط بیشتری سعی کردم در این گزارش تحقیق سمینار مطالب درک شده را گردآوری کنم.

این گزارش شامل ۵ فصل است. فصل اول به تعریف و مقدمه، فصل دوم مفاهیم فشرده‌سازی داده‌ها، فصل سوم مروری است بر کارهای انجام‌شده طرح پیشنهادی پایان‌نامه، فصل چهارم ارزیابی و پیاده‌سازی الگوریتم‌ها و بررسی نتایج و فصل پنجم نیز جمع‌بندی و نتیجه‌گیری نهایی است.

پس از درک موضوع و تجزیه‌وتحلیل آن‌ها در راستای ادامه کار موضوعات پیشنهادی مطرح شد.

مراجع

- K. Sayood, Introduction to data compression, 3rd ed. Amsterdam ; Boston: Elsevier, 2006.
- D. Salomon, Data compression the complete reference. New York: Springer, 2004.
- K. K. Shukla and M. V. Prasad, Lossy image compression: domain decomposition-based algorithms. London ; New York: Springer, 2011.
- «GNU Gzip». https://www.gnu.org/software/gzip/manual/gzip.html#Overview (opened Sep. 13th, 2020).
- «WinRAR - WinRAR Documentation». https://documentation.help/WinRAR/ (opened Sep. 14th, 2020).
- D. Huffman, «A Method for the Construction of Minimum-Redundancy Codes», Proc. IRE, vol. 40, nr. 9, s. 1098–1101, sep. 1952, doi: 10.1109/JRPROC.1952.273898.
- J. Ziv and A. Lempel, «A universal algorithm for sequential data compression», IEEE Trans. Inform. Theory, vol. 23, nr. 3, pp. 337–343, May 1977, doi: 10.1109/TIT.1977.1055714.
- Farina, G. Navarro, and J. R. Parama, "Boosting Text Compression with Word-Based Statistical Encoding", The Computer Journal, vol. 55, no. 1, pp. 111–131, Jan. 2012, doi: 10.1093/comjnl/bxr096.
- «Shoco - a fast compressor for short strings». https://ed-vonschleck.github.io/shoco/ (opened Sep. 14th, 2020).
- «torvalds/linux: Linux kernel source tree». https://github.com/torvalds/linux (opened Sep. 14th, 2020).
- Y. Q. Shi and H. Sun, Image and Video Compression for Multimedia Engineering, Boca Raton, FL: CRC Press, 1999, pp. 140–141.
- M. Aggarwal and A. Narayan, «Efficient Huffman decoding», in Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101), Vancouver, BC, Canada, 2000, vol. 1, pp. 936–939, doi: 10.1109/ICIP.2000.901114.
- Langiu, «On parsing optimality for dictionary-based text compression—the Zip case», Journal of Discrete Algorithms, vol. 20, pp. 65– 70, May 2013, doi: 10.1016/j.jda.2013.04.001.
- «bzip2 and libbzip2, version 1.0.8». https://www.sourceware.org/bzip2/manual/manual.html (opened Sep. 14th, 2020).
- R. Togneri and C. J. S. DeSilva, Fundamentals of information theory and coding design. Boca Raton: Chapman & Hall/CRC, 2002.
- R. Hankerson, G. A. Harris, and P. D. Johnson, Introduction to information theory and data compression, 2nd ed. Boca Raton, Fla: Chapman & Hall/CRC Press, 2003.
- M. Burrows and D. J. Wheeler, «A block-sorting lossless data compression algorithm», 1994.
- J. K. Korpela, Unicode explained, 1st ed. Sebastopol, CA: O'Reilly, 2006.
- «The Computer Journal». https://academic.oup.com/comjnl (opened Oct. 29th, 2020).

- Silva de Moura, G. Navarro, N. Ziviani, and R. Baeza-Yates, “Fast and flexible word searching on compressed text,” ACM Trans. Inf. Syst., vol. 18, no. 2, pp. 113–139, Apr. 2000, doi: 10.1145/348751.348754.
- «Information Technology Coding and Computing – ITCC». http://www.itcc.info/ (opened Nov. 3rd, 2020).
- Håkansson, «Portal of Research Methods and Methodologies for Research Projects and Degree Projects», 2013, [Online].
- B. Agarwal, S. P. Tayal, and M. Gupta, Software engineering & testing: an introduction. Sudbury, Mass: Jones and Bartlett, 2010.
- M. McEnery and R. Z. Xiao, «Character encoding in corpus construction.», in Developing Linguistic Corpora : A Guide to Good Practice, M. Wynne, Red. Oxford, UK: AHDS, 2005.
- «JSON Syntax». https://www.w3schools.com/js/js_json_syntax.asp (opened Sep. 16th, 2020).
- «HTML Charset». https://www.w3schools.com/html/html_charset.asp (opened Sep. 16th, 2020). [72] «Wikimedia Downloads». https://dumps.wikimedia.org/backupindex.html (opened Sep. 24th, 2020).
- «torvalds/linux». https://github.com/torvalds/linux/blob/master/kernel/sys.c (opened Sep. 24th, 2020).
- F. Nietzsche and T. Common, Thus spake Zarathustra. Ware: Wordsworth Editions, 1997.
- Dante Alighieri, B. Garavelli, and L. Magugliani, La divina commedia. Milano: BUR Rizzoli, 2012.
- Ainajushi, 豆棚閒話. Salt Lake City: Project Gutenberg, 2008.
- «Open license - Creative Commons». https://wiki.creativecommons.org/wiki/Open_license (opened Sep. 16th, 2020).
- Kaye, Loosely coupled: the missing pieces of Web services. Marin County, Calif: RDS Press, 2003.
- «LZW compression». https://rosettacode.org/wiki/LZW_compression#Python (opened Sep. 16th, 2020).
- «gzip — Support for gzip files — Python 3.8.6rc1 documentation». https://docs.python.org/3/library/gzip.html (opened Sep. 16th, 2020).
- «dahuffman». https://github.com/soxofaan/dahuffman (opened Sep. 16th, 2020).
- T. Summers, “Hardware based GZIP compression, benefits and applications,” CORPUS, vol. 3, pp. 2–68, 2008.
- Belanger, K. Church, and A. Hume, “Virtual Data Warehousing, Data Publishing and Call Detail,” in Databases in Telecommunications, vol. 1819, W. Jonker, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 106–117.
- «7z Format». https://www.7-zip.org/7z.html (opened Oct. 18th, 2020).
- M. Salib, “Faster than C: Static type inference with Starkiller” in PyCon Proceedings, Mar. 2004, pp. 2–26.
- M. Beazley, Python essential reference, 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2009.

واژه‌نامه

واژه‌نامه فارسی

پنجره کشویی	Sliding window
داده‌های دودویی	Binary data
فشرده‌سازی با اتلاف	Loss compression
فشرده‌سازی بدون اتلاف	Lossless compression
فشرده‌سازی داده	Data compression
کدگذاری دیکشنری	Coding dictionary
کدگذاری هافمن	Coding Huffman
گره برگ	Leaf node
گره فرزند	Child node
مجموعه داده‌ها	Data set
مخزن گیت هاب	Git hab repository

واژه‌نامه انگلیسی

داده‌های دودویی	Binary data
گره فرزند	Child node
کدگذاری دیکشنری	Coding dictionary
کدگذاری هافمن	Coding Huffman
فشرده‌سازی داده	Data compression
مجموعه داده‌ها	Data set
مخزن گیت هاب	Git hab repository
گره برگ	Leaf node
فشرده‌سازی با اتلاف	Loss compression
فشرده‌سازی بدون اتلاف	Lossless compression
پنجره کشویی	Sliding window

Abstract

Data compression is a field that has been extensively researched. Many compression algorithms in use today have been around for several decades, like Huffman Coding and dictionary coding. These are general-purpose compression algorithms and can be used on anything from text data to images and video. There are, however, much fewer lossless algorithms that are customized for compressing certain types of data, like ASCII text. This project is about creating a text-compression solution using a combination of the three compression algorithms dictionary coding, ASCII compression, and Huffman coding. The solution is customized for compressing ASCII text, but it can be used on any form of text. The algorithms will be combined to create a prototype that will be evaluated against general-purpose compression programs. An evaluation will also be made against the ASCII compression program Shoco. The results from the evaluation show that combining dictionary coding, ASCII compression, and Huffman coding does not surpass the compression ratio achieved from general-purpose compression programs.

Keywords Text compression, Dictionary coding, ASCII, Huffman coding

Payam Noor University

Department of Computer Engineering and Information Technology

Seminar Report (M.S.c)

Title:

Improving the text compression ratio for ASCII text (Review)

Supervisor:

Dr. Seyed Ali Razavi Ebrahimi

By:

Maryam Sadat Mourdgar

September 2021