Reconstructing Corrupt DEFLATEd Files

By

Ralf Brown

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2011 USA

New Orleans, LA (Aug 1st - 3rd)

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Reconstructing corrupt DEFLATEd files

Ralf D. Brown

Carnegie Mellon University Language Technologies Institute, 5000 Forbes Avenue, Pittsburgh PA 15213, USA

Keywords:
Data recovery
File reconstruction
DEFLATE compression
Zip archive
Language modeling

ABSTRACT

We present a method by which to determine a synchronzation point within a DEFLATE-compressed bit stream (as used in Zip and gzip archives) for which the beginning is unknown or damaged. Decompressing from the synchronization point forward yields a mixed stream of literal bytes and co-indexed unknown bytes. Language modeling in the form of byte trigrams and word unigrams is then applied to the resulting stream to infer probable replacements for each co-indexed unknown byte. Unique inferences can be made for approximately 30% of the co-indices, permitting reconstruction of approximately 75% of the unknown bytes recovered from the compressed data with accuracy in excess of 90%. The program implementing these techniques is available as open-source software.

1. Introduction

Since its introduction with PKWare's PKZIP version 2.0 in 1993, the DEFLATE compression algorithm has become ubiquitous. Not only are zip archives still popular for distributing collections of files, many modern file and document types are in fact zip archives. Microsoft Office 2007 documents (.docx, .pptx, .xlsx) are collections of XML files and other embedded files using the zip format as a container; likewise for OpenDocument documents (.odt, .odp, .ods). The ePub electronic book format is a zip container for XML files, while both Java applications (.jar) and Android apps (.apk) are zip archives containing compiled byte codes and optionally source files. Thus, the ability to deal with corrupted zip archives is of great interest in digital forensics.

Various tools are readily available for extracting intact member files from a corrupted zip archive or generating a new zip archive consisting of the intact members (e.g. Zip Recovery v1.6 and Info-ZIP's open-source zip (2008) with the -F commandline argument) by scanning the archive for file header records. Many of these tools can also recover the portions of a compressed file prior to a point of corruption or truncation, as this can be performed simply by running the decompressor until it encounters an invalid bit sequence or runs out of data.

However, generally-available tools do not permit recovering data *after* the point of corruption. In this paper, following a brief description of the DEFLATE algorithm, we will present a method by which to resynchronize with the compressor after the point of corruption and then how to reconstruct a majority of the bytes which remain unknown when decompressing from the synchronization point.

2. The DEFLATE algorithm

The DEFLATE algorithm consists of two phases: redundancy removal and entropy coding.

In redundancy removal, a window of the previous 32 kB (64 for the infrequently-used DEFLATE64 variant) is searched for sequences of three or more bytes which are identical to the bytes at the current location and the current bytes are replaced by a reference to the best match, if any. This results in a mixed stream of literal bytes and length:offset pairs.

The entropy coding phase then represents the output of the first phase in the fewest number of bits using Huffman coding. Two separate Huffman trees are used; one represents literal bytes and lengths, while the other represents the high-order bits of the offset. Both length symbols and offsets may

E-mail address: ralf@cs.cmu.edu.
1742-2876/$ – see front matter © 2011 R. Brown. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2011.05.015

be followed by a variable number of literal bits to specify the exact value of the length or offset.[1]

The compressor may make a variety of decisions which affect the size of the compressed bit stream, as can be seen in Figs. 1 and 2. In the example, the compressor can choose to output either three back-references or two back-references and two literals. If tuned for compression speed, the choice may be fixed, while a compressor tuned for maximum compression would check both and select the alternative which produces the smaller final bit stream after entropy coding.

Further, the compressor is allowed to switch Huffman trees at any point. Whenever it determines that the benefit of such a switch would outweigh the attendant overhead, it may output an end-of-data symbol using the current Huffman tree, and then immediately (without padding) a three-bit packet header followed by the information required for the compressor to follow the switch. The new packet may be uncompressed, compressed using pre-defined Huffman trees, or compressed using trees transmitted following the three-bit header. There is no *a priori* limit on the size of packets (other than uncompressed packets, which have an explicit 16-bit length field rather than using an end-of-data symbol), since packet size does not affect the decompressor's memory requirements. In practice, each packet generally represents no more than 100–200 kB of the original input.

It is these packet boundaries which provide the first avenue of attack when the beginning of the compressed stream is missing. Once the correct position is found, the state of the Huffman compression is known, and the bit stream can be expanded back into the symbol stream of mixed literals and back-references. Because back-references may cross packet boundaries (including multiple packets up to the window size of 32 kB or 64 kB), the recovered stream will contain references to unknown bytes, each of which may itself be referenced by a future back-reference. The multiple occurrences of copies of a single unknown byte provide the second avenue of attack, as the context in which each occurrence exists provides constraints on the possible values of the byte. The next section describes both of these approaches in greater detail.

3. Method

To reconstruct the original file given a damaged DEFLATE stream, we first need to determine the location of the first packet following the point of corruption. This is a straightforward but potentially time-consuming search, so it is important to rule out candidates as quickly as possible.

As there is no padding between packets, the packet header could start at any bit boundary. The header itself is a mere three bits, with no signature string. The header consists of one bit indicating whether it is the last packet in the stream, and two bits representing which of three possible packet types is present. Thus, with no other information, six of the eight possible three-bit values are valid, implying an average of six possible packet starting positions for *every byte* of the bit stream.

[1] See PKWare Inc. (2007) for full details of the encoding.

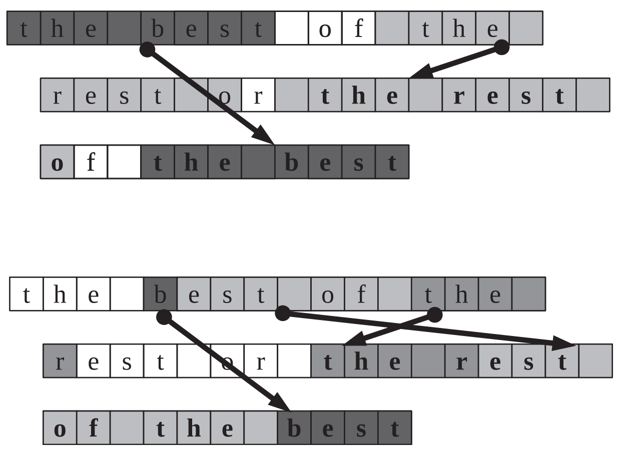


Fig. 1 – Lempel-Ziv redundancy removal. The compressor can select among different sequences to be eliminated.

The first optimization is to work forward from the end of the bit stream. This provides the correct value for the last-packet flag, immediately cutting the search space in half. In addition, because both variants of compressed packets use an end-of-data symbol, we can verify its presence at the known end of the packet without decompressing the entire packet. In the rare case of an uncompressed packet, we can verify that the known end is consistent with the explicit length.

Before we can verify the end-of-data symbol, we first need a valid Huffman tree. While nearly all candidate positions pass the checks for valid tree sizes, over half fail in decoding the Huffman tree with which the bit-length values representing the actual Huffman compression trees are encoded, and most of the remainder fail because the sequence of bit lengths computed using the first Huffman tree is invalid. Fewer than 1 in 1000 possible bit positions yield valid Huffman trees; of those, 95% fail the end-of-data symbol check. Among the candidates which have a valid end-of-data symbol, over 80% are in fact valid, intact packets (see Fig. 3).

A standard decompressor maintains a 32 kB (or 64 kB) circular buffer of bytes, inserting literal bytes into the next position in the buffer. When the decompressed Lempel-Ziv stream includes a back-reference, the decoder computes the proper offset in the buffer and then progressively inserts copies of the given number of bytes following that offset (it is

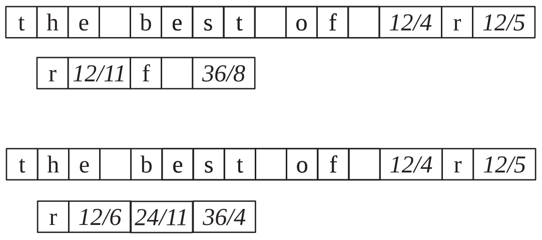


Fig. 2 – Possible symbol streams resulting from redundancy removal. The symbols are encoded with varying numbers of bits to produce the final compressed file.

Little Brother	**apw_spa_200412**	**e-Book Collection**
Candidate packets:	Candidate packets:	Candidate packets:
268418 uncompressed	1679193 uncompressed	510369297 uncompressed
272549 fixed Huffman	1746030 fixed Huffman	515689552 fixed Huffman
273632 dynamic, of which	1743024 dynamic, of which	515699752 dynamic, of which
233670 with valid alphabet sizes	1485977 with valid alphabet	437536205 with valid alphabet
154464 bad bit-length trees	986690 bad bit-length trees	289024897 bad bit-length trees
79061 invalid bit lengths	498299 invalid bit lengths	148207467 invalid bit lengths
130 with valid Huffman tree	869 with valid Huffman tree	255901 with valid Huffman tree
4 with valid EOD marker	34 with valid EOD marker	11948 with valid EOD marker
4 valid	32 valid	9665 valid

Fig. 3 – Search statistics for finding the first uncorrupted packet in the DEFLATE stream for two sample files and a collection of archived electronic books.

valid for the count to be greater than the offset, in which case an already-copied byte will be copied again). Because we need to deal with references to bytes which are unavailable, our buffer consists not of bytes but of integers with a flag which indicates whether the integer represents a literal byte or a position in the window prior to the start of the first recoverable packet. These positions serve as co-indices for the later reconstruction phase. The buffer is initialized with unknowns whose position is the distance from the start of the first available packet. The decompression proceeds as in a standard decompressor, resulting in a sequence of literal bytes and co-indexed unknown bytes.

To reconstruct the values of the unknown bytes, we use two language models, constraint propagation, and a greedy replacement strategy. The two language models are a byte-trigram model (storing joint probabilities) and a word unigram model (a word list with frequencies). During reconstruction, we store a bitmap of the permissible values for each of the potentially 65,535 co-indices, which will constrain the search for possible replacements.

After initializing the set of possible values for each co-index to be all 256 byte values, the trigram model is used to eliminate values which are not possible because the corresponding trigrams are never seen in training text (e.g. “sZt” would not appear in English text, thus “Z” is not a possible value for an unknown byte occurring between “s” and “t”, while a blank would be quite likely and would not be eliminated from consideration). Because the file being reconstructed could contain unusual sequences, all trigrams of literal bytes in the Lempel-Ziv stream are counted and added to the overall trigram counts (a form of self-training).

The trigram constraint propagation proceeds in multiple passes to reduce computation and to perform the most reliable eliminations first. In the first pass, the only unknown bytes which are processed are those with two literal bytes on either side; in the second pass, the immediately adjacent bytes are permitted to have two possible values while the bytes on either side of the central trigram may have any number of possible values; in the third pass, the adjacent bytes may have up to four possible values. Each occurrence is scored for each of the three trigrams in which it occurs (for word w_n, these are $w_{n-2}w_{n-1}w_n$, $w_{n-1}w_nw_{n+1}$, and $w_nw_{n+1}w_{n+2}$), with a large negative weight for any trigram which has a count of zero in the model. At the end of each pass, the 256 byte values for each co-index are examined and any which are sufficiently negative (indicating that a substantial proportion of their occurrences produced unseen trigrams) are removed from the set of possible replacements. After all three passes have been performed, any co-indices which have only one possibility remaining have that sole possibility assigned as their reconstructed replacement, and all occurrences of unknown bytes with the co-index are replaced.

The first phase considerably reduces the search space, even though it only generates a small number of unambiguous replacements. Typically, fewer than 20 co-indices have values assigned.

In the second phase, the updated Lempel-Ziv stream is divided into words. Duplicates are merged to obtain two word lists with frequencies. The first list consists of words containing only literal bytes; these will be used as part of the lookup process to determine possible words, analogously to how trigrams from the file were added to the pre-trained trigram model. The pre-trained word list will be called the global vocabulary, while the word list extracted from the current file will be called the local vocabulary. The second list consists of words containing one or more unknowns, and it is sorted to order the words by the likelihood that they can accurately be reconstructed and the usefulness of that reconstruction in finding additional replacements: fewest unknowns first; if the same number of unknowns, less ambiguous unknowns first; then longer words first (they provide more constraints); finally, most frequent words first. Additionally, an inverted index is created, listing for each co-index all of the words containing an unknown byte with that co-index.

The sorted list of words containing unknowns is now processed one word at a time. For each word, the existing replacements are applied. If the result contains at least one known byte, or has a sufficiently small cross-product of allowable replacement bytes, the set of possible words in either the global or local vocabularies which are consistent with the literal and replaced bytes, as well as the set of allowable replacements for each co-index (determined in the trigrams phase), is retrieved. For each possible replacement word,

- temporarily assign replacements for the unknowns in the word being processed
- retrieve all the words containing any of those unknowns, and apply the temporary set of replacements to each

- if the replacements turn a string with unknowns into multiple words, split the string and process each subword separately
- if the (sub)word does not contain any unknowns, determine whether either the local or global vocabularies contain that word, and assign a bonus to the candidate replacement word if so (a larger bonus for longer words, as well as for words contained in the local vocabulary) and a penalty if not.
- if the (sub)word does contain unknowns, determine whether it is consistent with either the global or local vocabulary; if not, assign a penalty.

The processing of candidate replacements also includes a check whether an unknown byte could be replaced by whitespace, resulting in two valid words.

At the end of this process, the highest-scoring candidate word (if its score is greater than zero) is selected, and the replacements corresponding to the unknown bytes in the original word are permanently added as the values of the corresponding co-indices. Processing then continues with the next word on the sorted list; in many cases, the replacements which have already been assigned result in a word which no longer contains unknowns, and such words are simply skipped.

The reconstruction process may optionally be iterated multiple times, with each iteration using the selected replacements for unknown bytes as if they had been literal bytes. Subsequent iterations proceed far more quickly than the first iteration because of the greatly reduced search space. Not only are there fewer co-indices to be evaluated, many long strings of unknown bytes have been interrupted by non-word characters, resulting in multiple short words with few (or even no) unknown bytes rather than a single long word with many unknowns. However, errors made during previous iterations may cascade or prevent a replacement by producing strings which can not become known words regardless of the values chosen for the unknown bytes they contain. Recognizing that errors may cascade, the confidence scores associated with each replacement are reduced in every subsequent iteration.

It is important to keep in mind that the reconstruction process described above is only suited for files which may be segmented into word-like units; in practice, this means files consisting primarily of text. The recovery of undamaged DEFLATE packets is applicable to any type of file, but the presence of scattered unknown bytes is likely to make reconstruction of non-textual files impractical by any method unless their format is extremely tolerant to corruption.

4. Related work

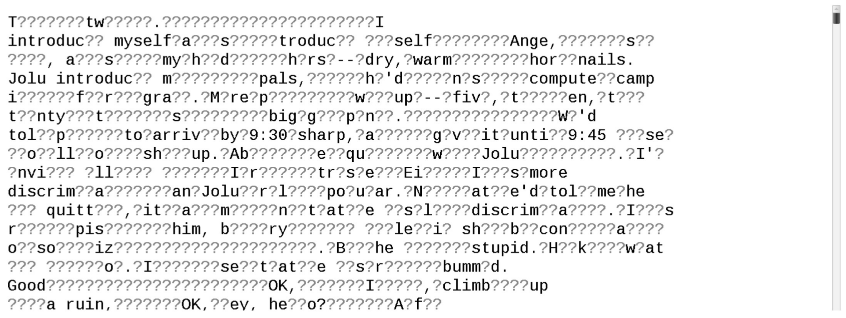
ZipRec takes advantage of the signature strings contained at the beginning of each data record within a ZIP archive, as did Cohen's semantic file carver for PDF and ZIP files (Cohen, 2007). The latter did not attempt recovery of partial archive members, but in contrast to ZipRec did specifically deal with file fragmentation. As ZipRec was originally implemented for use on possibly-corrupted files rather than raw disk images, it currently assumes that archives are not fragmented – fragmentation is treated as archive corruption.

Park et al. (2008) successfully recovered partial files in some cases by advancing one bit at a time within the DEFLATE stream and attempting a decompression. Unlike the current implementation of ZipRec, their Decompression Helper program can recover a partial block if it was compressed using the fixed, pre-specified Huffman table and does not reference any unknown bytes. On the other hand, Decompression Helper is much slower than ZipRec (a reported 7 kB of compressed data per second versus 1.0–1.5 MB/s for simple recovery by ZipRec) and will not decompress blocks which reference unknown bytes. Since it will not generate a decompressed stream with unknown bytes, Decompression Helper also does not perform any reconstruction on recovered data.

Language models have long been ubiquitous in automatic speech recognition (Kuhn and Mori, 1990) and machine translation (Brown & Frederking, 1995), used to guide the search for the best textual transcription of a speech utterance or translation of a piece of text. While unigram models as currently used in ZipRec are extremely weak in the absence of other constraints, the co-indexing of unknown bytes provides such constraints when applying the model.

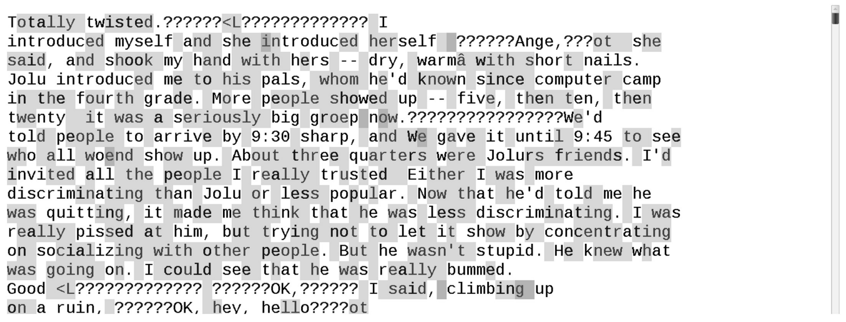
5. Results

Fig. 4 shows a passage of text recovered from the HTML version of the novel "Little Brother" by Cory Doctorow (2008). The file was compressed with Info-ZIP's Zip version 3.0 (2008) and then the first 1024 bytes of the archive were removed.



```
T???????tw?????.??????????????????????I
introduc?? myself?a???s?????troduc?? ???self????????Ange,???????s??
????, a???s?????my?h??d??????h?rs?--?dry,?warm????????hor??nails.
Jolu introduc?? m?????????pals,??????h?'d?????n?s?????compute??camp
i??????f??r???gra??.?M?re?p?????????w???up?--?fiv?,?t?????en,?t???
t??nty???t???????s?????????big?g???p?n??.????????????????W?'d
tol??p??????to?arriv??by?9:30?sharp,?a??????g?v??it?unti??9:45 ???se?
??o??ll??o????sh???up.?Ab???????e??qu???????w????Jolu??????????.?I'?
?nvi??? ?ll???? ???????I?r??????tr?s?e???Ei?????I???s?more
discrim??a???????an?Jolu??r?l????po?u?ar.?N?????at??e'd?tol??me?he
??? quitt???,?it??a???m?????n??t?at??e ??s?l????discrim??a????.?I???s
r??????pis???????him, b????ry??????? ???le??i? sh???b??con?????a????
o??so????iz?????????????????????.?B???he ???????stupid.?H??k????w?at
??? ??????o?.?I???????se??t?at??e ??s?r??????bumm?d.
Good??????????????????????OK,???????I?????,?climb????up
????a ruin,???????OK,??ev, he??o????????A?f??
```

Fig. 4 – Text recovered from corrupted archive. Question marks are propagated unknowns from the missing beginning of the file.



```
Totally twisted.??????<L????????????? I
introduced myself and she introduced herself  ??????Ange,???ot  she
said, and shook my hand with hers -- dry, warmâ with short nails.
Jolu introduced me to his pals, whom he'd known since computer camp
in the fourth grade. More people showed up -- five, then ten, then
twenty  it was a seriously big groep now.????????????????We'd
told people to arrive by 9:30 sharp, and We gave it until 9:45 to see
who all woend show up. About three quarters were Jolurs friends. I'd
invited all the people I really trusted  Either I was more
discriminating than Jolu or less popular. Now that he'd told me he
was quitting, it made me think that he was less discriminating. I was
really pissed at him, but trying not to let it show by concentrating
on socializing with other people. But he wasn't stupid. He knew what
was going on. I could see that he was really bummed.
Good <L????????????? ??????OK,?????? I said, climbing up
on a ruin, ??????OK, hey, hello????ot
```

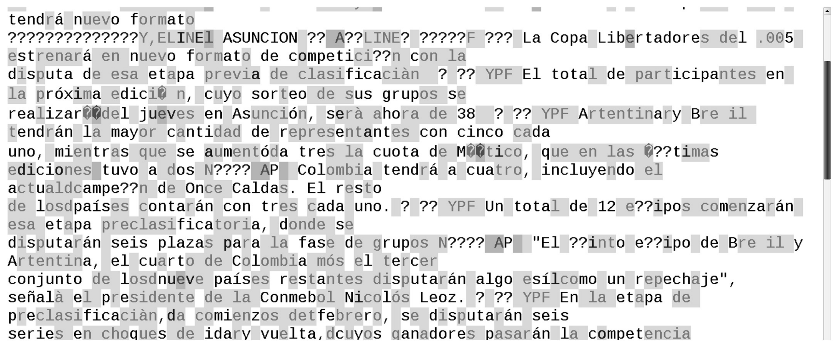
Fig. 5 – Reconstructed Text. Color coding is used to indicate the confidence score for inferred bytes.

Boldfaced characters are literal bytes, while the question marks are unknown bytes preceding the first recovered packet which propagated due to back-references in the Lempel-Ziv symbol sequence. Of the 786,775 bytes of the original file, 632,988 are recovered, 294,021 of which are literal bytes and 338,967 are unknown bytes. On a 3.2 GHz AMD Phenom II processor, Info-ZIP's UnZip version 6.0 (2009) requires 30 ms to decompress an uncorrupted archive, while our prototype program ZipRec requires 295 ms to decompress the corrupted archive. The left column of Fig. 3 shows the search statistics, demonstrating the efficacy of the fail-early checks. For this particular file, there were no false positives requiring full decompression to determine that the candidate start position was in fact incorrect.

Next, we applied the reconstruction algorithm, using a collection of Doctorow works other than "Little Brother", totaling 1.2 MB, as language model training data. Using six iterations and soft byte-trigram constraints, 259,981 bytes (with 2707 co-indices) of the 338,967 unknown bytes (with 8244 co-index positions) were reconstructed. As shown in Fig. 5, the text is now easily readable despite a few errors. The remaining unknown characters in this passage are HTML markup; these frequently-repeated fixed passages tend to be propagated the entire length of the file by the Lempel-Ziv redundancy removal, leaving little opportunity to observe those co-indices in other contexts. Reconstruction is far more computationally expensive than the search for uncorrupted DEFLATE packets, and requires a total of 69 s (58 if only a single iteration is run).

Fig. 6 shows a portion of a reconstructed Spanish file, demonstrating that the technique is language-neutral. The training data for this experiment was the entire collection of newswire articles from 2005 (slightly less than 600 MB) contained within the Spanish Gigaword Corpus, version 1, while the test file was apw_spa_200412.gz from that corpus. A missing initial byte of the DEFLATE stream was simulated, and the reconstruction algorithm run on the resulting partial decompression. A total of six reconstruction iterations were used, with soft trigram constraints applied. Packets representing 5,155,255 of the original file's 5,187,142 bytes were recovered, of which 4,192,309 were literal bytes. Of the 962,946 unknown bytes with 6597 co-indices, 748,161 bytes with 2324 co-indices were reconstructed. On this file, UnZip requires 0.15 s, ZipRec extraction 1.5 s, and reconstruction 247 s (215 if only a single iteration is run).

Because the language model affects the reconstruction, we compared three different models on each of the sample files. The first model is the empty language model, automatically augmented with the recovered text of the file being reconstructed; this serves as a baseline for comparison. The second model is one well-matched to the file, i.e. the Doctorow texts for "Little Brother", while the third is a wildly mismatched model – Spanish text for the English "Little Brother" and the English Doctorow texts for the Spanish newswire file. Fig. 7 shows the effects of varying the language models, as well as the effects of performing multiple reconstruction iterations. As expected, the percentage of reconstructed text rises with increasing iterations, while the percentage of correctly-reconstructed bytes drops due to error propagation. The Spanish text shows little variation among models, which is likely due to three factors: the file is a collection of diverse newswire articles on a wide range of topics by multiple



```
tendrá nuevo formato
??????????????Y,ELINEl ASUNCION ?? A??LINE? ?????F ??? La Copa Libertadores del .005
estrenará en nuevo formato de competici??n con la
disputa de esa etapa previa de clasificaciàn  ? ?? YPF El total de participantes en
la próxima edici� n, cuyo sorteo de sus grupos se
realizar��del jueves en Asunción, serà ahora de 38  ? ?? YPF Artentinary Bre il
tendrán la mayor cantidad de representantes con cinco cada
uno, mientras que se aumentóda tres la cuota de M��tico, que en las �??timas
ediciones tuvo a dos N???? AP  Colombia tendrá a cuatro, incluyendo el
actualdcampe??n de Once Caldas. El resto
de losdpaíses contarán con tres cada uno. ? ?? YPF Un total de 12 e??ipos comenzarán
esa etapa preclasificatoria, donde se
disputarán seis plazas para la fase de grupos N???? AP  "El ??into e??ipo de Bre il y
Artentina, el cuarto de Colombia mós el tercer
conjunto de losdnueve países restantes disputarán algo esílcomo un repechaje",
señalà el presidente de la Conmebol Nicolós Leoz. ? ?? YPF En la etapa de
preclasificaciàn,da comienzos detfebrero, se disputarán seis
series en choques de idary vuelta,dcuyos ganadores pasarán la competencia
```

Fig. 6 – Reconstructed Spanish text. Unknown-character glyphs are due to errors in one byte of a multi-byte character.

Language Model	Iterations	Little Brother	apw_spa_200412
None (self-training only)	1	87.5%c 63.5%r	84.3%c 62.4%r
	2	80.9%c 74.4%r	73.1%c 73.7%r
	4	79.9%c 75.6%r	69.0%c 78.3%r
	6	79.8%c 75.9%r	69.0%c 78.3%r
Doctorow texts	1	95.3%c 71.1%r	84.7%c 62.7%r
	2	92.7%c 76.1%r	73.4%c 73.7%r
	4	92.4%c 76.6%r	69.0%c 78.5%r
	6	92.4%c 76.7%r	69.0%c 78.6%r
Spanish GigaWord	1	80.0%c 67.9%r	81.2%c 62.4%r
	2	75.0%c 75.6%r	73.9%c 73.7%r
	4	74.2%c 76.7%r	70.5%c 77.7%r
	6	74.2%c 76.7%r	70.5%c 77.7%r

Fig. 7 – Proportion of unknown bytes which are reconstructed (*r*) and proportion of reconstructured bytes which are correct (*c*) for two sample texts using varying training data and number of iterations.

authors; it is large enough for self-training to be a major factor in performance (self-training data is at least as much as the total training data for the English language model); and the program does not yet have specific support for multi-byte characters (used for accented vowels), which affects word segmentation.

Finally, we applied ZipRec to an arbitrarily-selected disk image from the Real Data Corpus (Garfinkel et al., 2009), UAE10-009.E01, to explore forensic performance. This image file represents a 20 GB disk from which all files and directories have been deleted. ZipRec detects 10,478 local file header signatures, 11,725 central directory entries, and 550 end of central directory records, and is able to extract 6922 complete files and an additional 446 partial files. A large fraction of the archived files are quite small (CSS files, Java .class files, .ini files, and the like) or fairly small and already compressed (e.g. PNG or GIF icons) and are stored uncompressed; 5309 of the 6922 completely recovered files were not compressed. Among attempted decompressions, 2389 streams contained zero or one packet,[2] 58 contained two packets, and 43 contained more than two packets. Approximately 78 MB of data is recovered, of which 77 MB is literal bytes and approximately 1 MB is unknown bytes.

6. Conclusions

We have shown that it is possible to recover substantial amounts of data from a DEFLATEd file for which the start of the compressed bit stream is missing or corrupted, and to reconstruct a majority of the remaining unknown bytes in the recovered portion. Corrupt or missing bytes in the middle of the bit stream may be handled in the same manner, with the addition of standard decompression from the start of the stream up to the point of corruption. The proportion of a file which can be recovered depends directly on the proportion of the stream's packets which follow the point of corruption, while the proportion of the recovered Lempel-Ziv sequence consisting of literals (and thus also the efficacy of reconstruction) increases with increasing number of packets.

The ZipRec program implementing the algorithms described in this paper is available as open-source software under the terms of the GNU General Public License at http://ziprec.sourceforge.net/.

7. Future work

It should be possible to infer replacements for additional co-indices using stronger language models than the simple unigram word model currently used. A trigram model would allow the context around a word to influence the possible candidate reconstructions of that word in cases where it is not currently possible to narrow the choices to a single unambiguous value. However, the current approach would be considerably slower with a trigram model because each instance of a word needs to be scored individually, whereas currently multiple instances of the same string are scored just once, with the score multiplied by the frequency of occurrence.

The current software also has difficulty with whitespace and punctuation, often reconstructing a punctuation mark as a blank. A stronger, higher-order language model would help avoid these errors as well.

Independent of improvements to the ZipRec program itself, a GUI for manual correction of the reconstructed bytes may be implemented. Taking advantage of co-indexing, each time the user corrects one instance, all other co-indexed instances can be updated as well.

Acknowledgement

The author would like to thank the anonymous reviewers for their helpful suggestions and Simson Garfinkel for his assistance with the Real Data Corpus while revising this paper.

REFERENCES

Cohen MI. Advanced carving techniques. Digital Investigation September–December 2007;4(3–4):119–28.

Doctorow Cory. Little Brother. Tor Books; 2008.

David Graff (Linguistic Data Consortium). Spanish Gigaword First Edition. Catalog number LDC2006T12.

Info-ZIP workgroup. UnZip, version 6.0, http://info-zip.org/UnZip.html; April 2009.

Info-ZIP workgroup. Zip, version 3.0, http://infozip.org/Zip.html; July 2008.

Kuhn Roland, Mori Renato De. A cache-based natural language model for speech recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence; June 1990: 570–83.

Park Bora, Savoldi Antonio, Gubian Paolo, Park Jungheum, Hee Lee Seok, Lee Sangjin. Data extraction from damage compressed file for computer forensic purposes. International Journal of Hybrid Information Technology October 2008;1(4): 89–102.

[2] When the local file header is missing, it is not always possible to distinguish between the case where there is an actual stream consisting of a single (but corrupted) packet and where there is no actual compressed stream.

PKWare, Inc. Application note on the .ZIP file format, version 6.3.2, http://www.pkware.com/documents/casestudies/APPNOTE.TXT; September 2007.

Recoveronix, Ltd. Zip Recovery v1.6. http://www.officerecovery.com/zip/.

Ralf Brown and Robert Frederking. Applying statistical English language modeling to symbolic machine translation. In: Proceedings of the sixth international conference on theoretical and methodological issues in machine translation (TMI-95), pp. 221–239, 1995.

Simson Garfinkel, Paul Farrell, Vassil Roussev, and George Dinolt. Bringing science to digital forensics with standardized forensic corpora. In: Proceedings of the 2009 Digital Forensics Research Workshop (DFRWS 2009), August 2009.