FUSE: A Reproducable, Extendable, Internet-scale Dataset of Spreadsheets

Titus Barik*†, Kevin Lubick†, Justin Smith†, John Slankas†, Emerson Murphy-Hill†
*ABB Corporate Research, Raleigh, North Carolina, USA
†North Carolina State University, Raleigh, North Carolina USA
titus.barik@us.abb.com, {kjlubick, jssmit11, jbslanka}@ncsu.edu, emerson@csc.ncsu.edu

***Abstract*—Spreadsheets are perhaps the most ubiquitous, widely used form of end-user programming software. This paper describes a dataset, called FUSE, consisting of metadata information for 719,223 spreadsheet accesses and their corresponding 249,376 binary files from a public web crawl of over 26.83 billion pages. The resulting dataset offers several useful properties over prior spreadsheet corpora, including reproducibility, extendability, and queryability. The dataset is unencumbered by any license agreements, available to all, and intended for wide usage by end-user software engineering researchers. In this paper, we detail the spreadsheet extraction process, describe the data schema, and discuss the limitations and challenges of FUSE.**

I. INTRODUCTION

End-user programmers today constitute a broad class of users, including teachers, accountants, administrators, managers, research scientists, and even children [1]. Although these users are typically not professional software developers, their roles routinely involve computational tasks that, in many ways, are similar to those of developers — not just in activity, but also in their underlying cognitive demands on users [2].

Perhaps the most ubiquitous and widely used form [3] of end-user programming software are *spreadsheets*, a table-oriented visual interface that serves as the underlying model for the users' applications [4]. *Cells* within these tables are augmented with computational techniques, such as functions and macros, that are expressive and yet simultaneously shield users from the low-level details of traditional programming [4].

This unique interplay between presentation and computation within the spreadsheet environment has, unsurprisingly, garnered significant interest from the software engineering research community [5]. In noticing the similarities and differences with traditional programming environments, researchers have adopted techniques and approaches to studying errors [6], code smells [7], refactoring [8], and debugging in spreadsheets [9]. For example, Abraham and Erwig exploit the spatial arrangements of tables within spreadsheets, a visual property inapplicable to traditional programming languages, to infer templates that help end-users safely edit spreadsheets [10].

To better understand end-user activities and design tools to assist end-users, researchers have responded by curating spreadsheet corpora to support spreadsheet studies: among them, EUSES [11], obtained by simple Google keyword searches and from Oregon State University students and researchers; Enron [12], extracted from e-mails obtained during legal evidence; and SENBAZURU/ClueWeb09 [13], obtained from the ClueWeb Web crawl by Cargenie Mellon University.

TABLE I
COMPARISON OF FUSE AND OTHER SPREADSHEET CORPORA

	FUSE	**EUSES**	**Enron**	**ClueWeb**
Size (n)	249,376	6,000	15,570	410,554
Space (GB)	87	0.64	23.3	110
Access	All	Researchers	All	All
Unique formulas	894,361	84,004	784,380	–
Extendable	Yes	Yes	No	Yes
Framework	Hadoop	Excel/VBA	Scantool	–
Scalable	Yes	No	No	Yes
Collection period	2013-2014	2006	2006	2009
Pri. origin	CC	Google	Enron	ClueWeb
Binaries	Yes	Yes	Yes	No
Metadata	Yes	No	No	No
Domains	4,318	-	-	51,157
Distinct functions	219	209	139	–

This paper presents another spreadsheet corpus, called FUSE, extracted from the over 4.2 billion web pages in the Common Crawl index. We believe that FUSE offers several useful traits not found in previous corpora. First, unlike EUSES or ClueWeb, FUSE is fully reproducible, as it is derived from archived snapshots containing both HTTP response data and the associated binary data. In contrast, EUSES provides binary spreadsheets but no origin information indicates the URL from which these spreadsheets were obtained. Similarly, ClueWeb provides only the URL of the spreadsheet. However, since the Internet is not a static entity, many of these URLs do not point to the same content as it did when the crawl was originally performed, and others are no longer available. Second, unlike EUSES or Enron, our corpus supports systematic updating as new crawls are added to the Common Crawl. Third, and perhaps most importantly, our corpus is the only one to support querying of spreadsheet metadata, facilitated by a JSON document associated with each spreadsheet. A comparison of these and other differences between corpora are summarized in Table I.

The contributions of this paper are:

- A corpus of metadata and binary spreadsheets extracted from public web sites through the Common Crawl

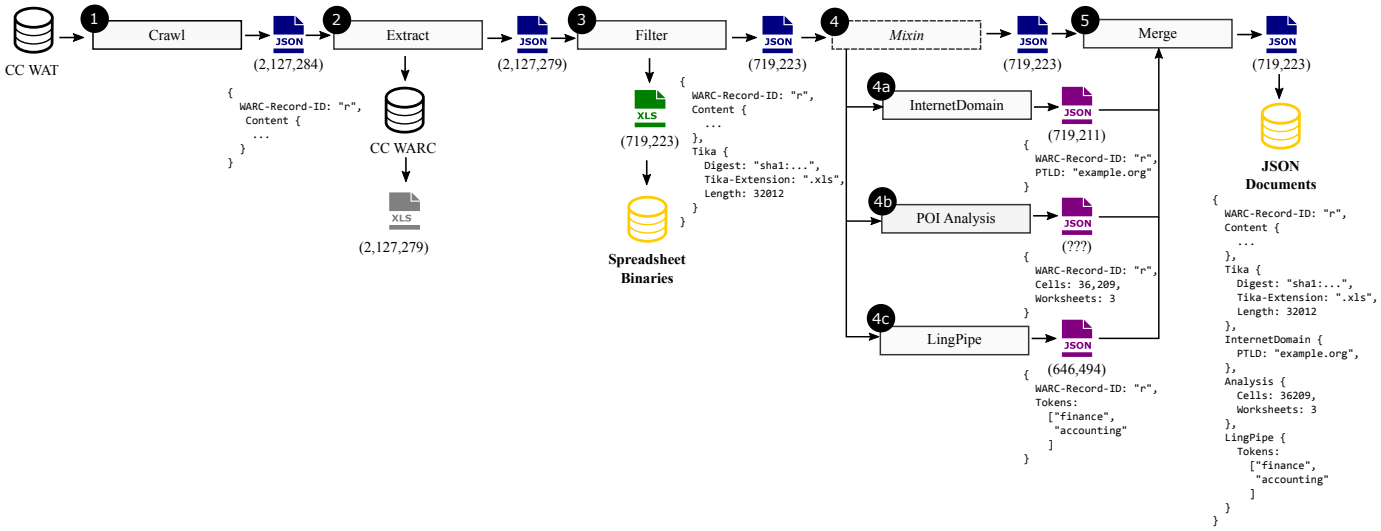

Fig. 1. The MapReduce pipeline for extracting spreadsheets and associated spreadsheet metadata from Common Crawl.

archive, made accessible to the research community.[1]

- A modular, open-source pipeline of tools, implemented using MapReduce. Our tool supports scalability from the ground up, and can cost-effectively process over 1 million spreadsheets in under an hour.
- A mixin system that enables researches to augment our analysis with their own data, using a schema-free, document-centric JSON format, supported by many popular database technologies.

II. Methodology

The Common Crawl[2] organization is 501(c)(3) non-profit dedicated to providing a copy of the Internet, and democratizing the data so that it is accessible to everyone. Of specific interest to us is that the corpus contains not only the metadata of web pages, but also the raw content of each resource, including binaries. Importantly, these crawls occur periodically, at a frequency of approximately once per month.

The Common Crawl is available as a public data set on Amazon, and crawl data is stored on Simple Storage Service (S3) as a set of WARC (Web ARCive) files, which store the raw crawl data, and a corresponding WAT file, which stores the web crawler metadata for the given WARC file. Essentially, each WAT file contains JSON-formatted records that act as an index into the WARC raw data. That is, each record contains a globally unique identifier, which we call a `WARC-Record-ID`, and a reference to a WARC filename, offset, and length. Because S3 supports object ranges, it then becomes possible to extract the raw content of a single record without downloading an entire WARC file.

We considered spreadsheets from the period of Summer 2013–December 2014, which consists of 26.83 billion web pages, compressed as 423.8 TB (1.9 PB uncompressed). To support parallelization, this data is split into 481,427 segments, such that segment requests can be computed independently by a task node in a cluster. We extracted the spreadsheets using the Amazon Elastic MapReduce service, to take advantage of network co-location.

A. Hadoop MapReduce Pipeline

Our overall framework is illustrated in Figure 1, and consists of five MapReduce tasks that comprise a pipeline. In this section, we consider each of the stages in this pipeline.

1) Crawl: The first stage of the pipeline is also the most expensive computationally, because it requires that we traverse every JSON metadata record in the 481,427 WAT segments and heuristically tag spreadsheet-related records, which we call candidate spreadsheets. This is a heuristic process because cannot know for sure that a record is actually a spreadsheet until we inspect the corresponding WARC file. First, we check if the HTTP response payload `Content-Type` field corresponds to one of seven spreadsheet MINE types, as listed in MSDN. However, some records contain a generic binary `Content-Type` of `application/octet-stream`, in which case `Content-Disposition` is checked via a file pattern matching "`.xls*`". If either of these conditions are true, we save the record using the `WARC-Record-ID` as this key. This key is propagated through the pipeline.

After filtering through the some 26.83 billion records, we identified 2,127,284 candidate spreadsheets. This stage requires approximately 55,000 normalized instance hours[3] to process.

2) Extract: In this stage of the pipeline, the extract process loads the 2,127,284 candidate spreadsheets records. Using the Filename, Offset, and Deflate-Length fields of the record, the corresponding WAT record is extracted into memory. The

[1]The corpus metdata, binary spreadsheets, tools, and other documentation can be obtained at http://go.barik.net/fuse. This URL is currently intended only for examination by the review committee.

[2]http://commoncrawl.org

[3]A normalized instance hour is the amount of computation it would require for a `m1.small` compute node to complete the task.

WARC record is then stripped of its header information (e.g., the HTTP response), and the remaining content is saved to S3, again using the `WARC-Record-ID` from the WAT file as the key. Theoretically, this process should yield the same number of records as crawl stage; however, five records had corrupted `gzip` entries, yielding 2,127,279 candidate spreadsheets.

This stage requires approximately 1000 normalized instance hours to complete.

3) Filter: The third stage of the pipeline, filter, checks the extracted file and tag those that are actually spreadsheets. Using Apache Tika, this stage uses Tika's built-in MimeType detector, which returns the actual Content-Type of the file. If this result is one of the spreadsheet content types, the record is retained. During this stage, we also compute the length (in bytes) of the spreadsheet, identify the most appropriate file extension (e.g., ".xlsx"), and generate a SHA-1 digest of the spreadsheet content.

At this stage, 719,223 spreadsheets are retained in the pipeline, although many of these may be duplicates. This stage requires 420 normalized instance hours to complete.

4) Mixin: The fourth stage of the pipeline is actually a meta-stage, in which researchers can augment the framework with their own analysis, which we call *mixins*. For our dataset, we augment the JSON document with three mixins: `InternetDomain`, which uses the Google guava library to extracts domain-related information from the `WARC-Target-URI`. A second Apache POI analysis extracts relevant information on the content of the spreadsheets, such as the use of functions. A third analysis using LingPipe extracts language-related information from the spreadsheet. These JSON records are all saved to S3 by their `WARC-Record-ID`. For various reasons, not all APIs can analyze all spreadsheets, even when they open in Microsoft Excel. This stage requires about 200 normalized instance hours per mixin.

Researchers wishing to build on our approach will be able to insert their own mixins at this stage, without having to recompute the first three stages, saving research time and effort.

5) Merge: The final stage of the pipeline simply takes the resulting JSON files from all previous stages and combines them with the original WAT record to facilitate downstream analysis. The stage requires approximately 130 normalized instance hours for each mixin.

III. Data Schema

Using the SHA-1 hashes of the records, a local de-duplication operation is performed. The result of this operation is 249,376 unique spreadsheets, and can be downloaded from our site, or preferably, directly from S3 using the script provided on our site. The output of our pipeline also generates 719,223 JSON documents, which are intended for import into document-centric databases such as MongoDB.[4] In this section, we describe the important records in this document.

[4]http://www.mongodb.org/.

The most relevant elements from the WARC record are `WARC-Target-URI`, that is, the URL from which the spreadsheet and downloaded, and `Container`, the containing CommonCrawl file and offset used to extract the spreadsheet from the crawl. The `WARC-Date` element may also be of interest, since it contains the time and date of the access. Using the `Content-Disposition` element, one can often extract the original spreadsheet file name.

The Tika JSON element contains four fields, which looks something like this:

```
"Tika": {
    "Tika-Content-Type":
       "application/vnd.ms-excel",
    "Tika-Extension": ".xls",
    "Digest": "sha1:...",
    "Length": 5123
}
```

These fields are self-explanatory. The `InternetDomain` element is also relatively straightforward; we provide it because the individual domain components are useful for analysis, and because this extraction is non-trivial to perform once the data is already in a database.

```
"InternetDomainName": {
    "Host": "www.example.org",
    "Top-Private-Domain": "example.org",
    "WARC-Target-URI":
      "http://www.example.org/results/test.xls",
    "Public-Suffix": "org"
  }
```

Next, we also provide a `LingPipe` element, which extracts the token stream from spreadsheets, lowercases the tokens, removes English stop words (such as 'a' or 'the'), and filters out non-words (such as numbers). Again, the representation of this is simple:

```
"LingPipe": {
    "Tokens": [
      "finance",
      "branch",
      "city",
      ...
    ]
  }
```

Finally, to get a high-level overview of the content of the spreadsheets, as well as to aid other researchers in narrowing their queries, we used Apache POI to analyze the content of the spreadsheets and provide a summary. There are over 450 entries, which include the number of times a given Excel function (such as `SUM` or `VLOOKUP`) is used, the total number of input cells (i.e. cells that are not formulas), the number of numeric input cells, the number of formulas used more than 50 times, the most common formula used, and so on.

While space limitations do not permit a detailed discussion of our analysis, we will briefly highlight the function counts found in Table II. This table demonstrates how the three corpora differ in composition. The Enron set, being mostly financial computation makes extensive use of commands such as VLOOKUP, where as the FUSE spreadsheets focus on string

TABLE II
SELECTED FUNCTIONS WITH COUNTS PER 1000 FORMULA CELLS

	FUSE	Enron	EUSES
IF	178.8	157.3	166.8
+ (operator)	166.4	173.7	167.6
SUM	87.7	63.7	153.6
ISBLANK	57.8	0.6	27.4
VLOOKUP	30.8	55.8	12.2
HLOOKUP	9.5	2.1	1.4
AVERAGE	32.0	8.1	7.9
AND	6.6	16.0	21.7
NOT	5.5	0.1	1.9

manipulation (ISBLANK) and simple calculations (AVERAGE). Surprisingly, the distributions differ between EUSES and FUSE, despite the two corpora being collected in similar ways. No one corpus is sufficient for all possible end-user research, but we believe our extendable, scalable approach will give a wide variety for researchers now, and as it grows over the coming years.

IV. CHALLENGES AND LIMITATIONS

One of the limitations of the FUSE corpus is that in being derived from Common Crawl, the corpus is also constrained by what the Common Crawl considers to be relevant. Since the Common Crawl is a general crawling infrastructure, spreadsheet content arises infrequently. Still, we obtained 249,376 unique spreadsheets from this extraction, a non-trivial number. A second limitation of our crawl is that we cannot infer the quality of the spreadsheets. In contrast with Euses, whose extraction was guided by page ranking, no such ranking function is exposed in Common Crawl.

Yet another challenge is that it is impossible to provide a set of metadata that is satisfactory to all researchers, since the potential features of interest within a binary spreadsheet are innumerable. It is expected that researchers will have to integrate into our mixin stage to obtain specific metrics of interest. However, this process is considerably simpler and less expensive than having to obtain the spreadsheets from the original Common Crawl data.

V. CONCLUSION

This paper contributes a spreadsheet corpus, FUSE, which is derived from the Common Crawl. In comparison with other corpora, FUSE stacks up well. FUSE is a relatively balanced dataset that contains a significant number of spreadsheets, can be queried because of its associated JSON metadata documents, and is easily reproducible and extendable because because it derives from a source that couples both the metadata with the raw crawl data. It is also more up-to-date than existing corpora, and can remain up-to-date as additional crawls can be incorporated into FUSE.

An additional property that separates FUSE from existing corpora is that is both a dataset and an open source tool framework. Consequently, our code can not only be inspected for correctness (and defects resolved when issues are identified), but our programs can also be modified and tailored to support specific research needs, which we have exposed through mixins.

In many cases, we expect that FUSE can replace existing corpora, with stronger guarantees concerning reproducibility. In other cases, we expect that FUSE can offer an excellent complement to existing corpora in order to increase the diversity and representativeness of tool evaluations for end-user programmers.

ACKNOWLEDGMENT

This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

REFERENCES

[1] A. J. Ko, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, S. Wiedenbeck, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, and H. Lieberman, "The state of the art in end-user software engineering," *ACM Computing Surveys*, vol. 43, no. 3, pp. 1–44, Apr. 2011.

[2] A. Blackwell, "First steps in programming: a rationale for attention investment models," in *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, 2002, pp. 2–10.

[3] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *VL/HCC '05*, 2005, pp. 207–214.

[4] B. A. Nardi and J. R. Miller, "The spreadsheet interface: A basis for end user programming," in *Human-Computer Interaction: INTERACT '90*, 1990, pp. 977–983.

[5] M. Burnett, "What Is end-user software engineering and why does it matter?" in *End-User Development SE - 2*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5435, pp. 15–28.

[6] M. Pinzger, F. Hermans, and A. van Deursen, "Detecting code smells in spreadsheet formulas," in *Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM)*, ser. ICSM '12, 2012, pp. 409–418.

[7] S. Badame and D. Dig, "Refactoring meets spreadsheet formulas," in *Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM)*, ser. ICSM '12, 2012, pp. 399–409.

[8] R. Abraham, M. Burnett, and M. Erwig, "Spreadsheet Programming," in *Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc., 2009, pp. 2804–2810.

[9] S. G. Powell, K. R. Baker, and B. Lawson, "A critical review of the literature on spreadsheet errors," *Decis. Support Syst.*, vol. 46, no. 1, pp. 128–138, Dec. 2008.

[10] R. Abraham and M. Erwig, "Inferring templates from spreadsheets," in *Proceeding of the 28th international conference on Software engineering - ICSE '06*, May 2006, p. 182.

[11] M. Fisher and G. Rothermel, "The EUSES spreadsheet corpus," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4. ACM, Jul. 2005, p. 1.

[12] F. Hermans and E. Murphy-Hill, "Enrons spreadsheets and related emails: A dataset and analysis," 2015, to appear.

[13] Z. Chen and M. Cafarella, "Automatic web spreadsheet data extraction," in *Proceedings of the 3rd International Workshop on Semantic Search Over the Web*, Aug. 2013, pp. 1–8.