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 form of end-user programming software. This paper describes a dataset, called FUSE, containing information about 2,127,284 spreadsheet-related HTTP responses, 719,223 of which are linked to 249,376 unique binary spreadsheets, obtained from a public web crawl of over 26.83 billion pages through 60,000 hours of computation. The resulting dataset offers several useful properties over prior spreadsheet corpora, including reproducibility and extendability. 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 data and the spreadsheet extraction process, describe the data schema, and discuss the trade-offs of FUSE with other corpora.**

I. INTRODUCTION

End-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 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 computation, such as functions and macros [4]. This 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], and debugging in spreadsheets [8].

To better understand end-user activities and design tools to assist end-users, researchers have responded by curating spreadsheet corpora to support spreadsheet studies [9], [10], [11]. This paper presents one such spreadsheet corpus, called FUSE, extracted from the over 26.83 billion web pages in the Common Crawl index. We believe that FUSE offers several useful traits not found in prior corpora; for example, FUSE is obtained in such a way that researchers can independently reproduce an identical corpus from source materials.

The contributions of this paper are:

- A corpus of metadata, augmented with our analysis, and binary spreadsheets extracted from public web sites through the Common Crawl archive, made accessible to the research community.[1]
- A modular, open-source tool pipeline, implemented using MapReduce. Our tool is scalable and can process over 1 million spreadsheets in under an hour.

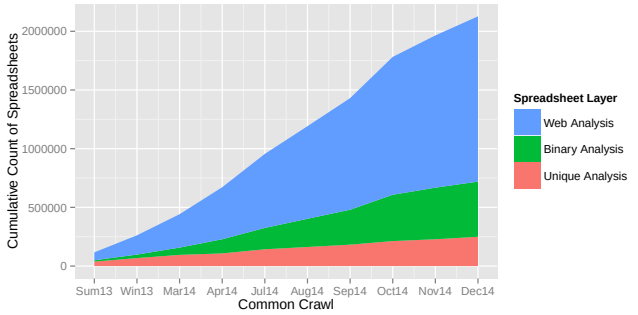

Fig. 2. Cumulative count of spreadsheets obtained with each additional crawl. Spreadsheet records are faceted into three layers, with each layer containing more specific information about the spreadsheet than the one before.

II. DESCRIPTION OF DATA

The Common Crawl[2] non-profit organization is 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 HTTP responses of web pages, but also the raw content of each of these resource, including binaries. It is from these updated-monthly Web crawls that we extract binary spreadsheets, and make available corresponding metadata tailored for researchers.

The result, FUSE, is characterized through three, hierarchical layers (Figure 2): 1) as a corpus that contains the target URL and HTTP response headers of spreadsheets on the Web, 2) as a corpus that captures the associated binary spreadsheets for a subset of those URLs, described through metadata using our analysis tools, and 3) as a corpus of unique spreadsheets extracted from the Web, where the binary content itself is primarily of interest.

[1]The corpus metadata, binary spreadsheets, tools, and other documentation can be obtained at http://go.barik.net/fuse.

[2]http://commoncrawl.org

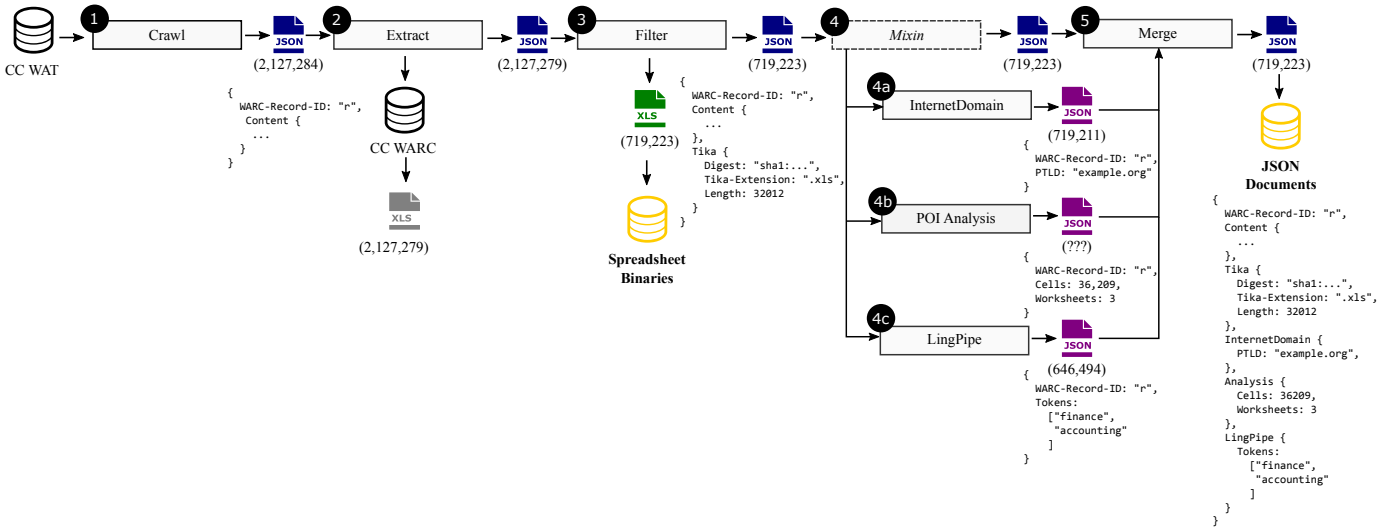

Fig. 1. The MapReduce pipeline for extracting spreadsheets and associated spreadsheet metadata from Common Crawl.

Web Analysis Layer: Contains 2,127,284 spreadsheet-related URLs and HTTP responses, across 51,380 distinct domain names. Primarily, this layer is intended for tasks that automatically crawl the live Internet, and as a result, binary contents from this stage are not reproducible.

Binary Analysis Layer: 719,223 of the records in the URL layer have valid, associated binary content stored alongside the record Web Analysis layer, this layer is stable because it represents a fixed snapshot in time.

Unique Binary Analysis Layer: Of the 719,223 records in the binary analysis tier, the a unique spreadsheet may appear multiple times. The simplest case where this occurs is when multiple Web sites host the same spreadsheet. For researchers who are only interested in content analysis of the spreadsheets, these unwanted duplicates will bias the analysis. This layer contains 249,336 binary spreadsheets. Through this layer, we learned that `SUM` is the most frequently used function, and that it is found in 8.7% of all formula cells. We also discover that `=SUM(R[-3]C:R[-1]C)` is the most common formula, in which a cell is the sum of the three cells to its left, and that it appears in 1,322 spreadsheets. In contrast with domain specific corpora, such as Enron, our general spreadsheet corpora has fewer formulas. We identified that only 6.95% (147282) of our spreadsheets as containing any formula, where as 59.4% of Enron spreadsheets contain formulas, which is consistent with anecdotal findings of the Microsoft Excel team.[3]

A consequence of this layering technique is that FUSE has several properties desirable to researchers. First, the corpus is reproducible, at several layers. At the URL layer, an independent research should always obtain the same set of spreadsheet-related URLs, when using our same heuristics. Because the binary analysis layers and unique analysis layers are obtained from content embedded in the Common Crawl corpus, these too are static, reproducible resources. A second property of our corpus is that it is open to extension, without sacrificing reproducibility. Common Crawl releases a new data set a frequency, these crawls can be incorporated into FUSE using our toolchain. A third useful property of the dataset is related not to the dataset itself, but to its broader ecosystem: FUSE is unencumbered by any licensing requirements, available to all, and includes an open source toolchain.

[3]Joel Spolsky writes, "Everybody thought of Excel as a financial modeling application, [but] we visited dozens of Excel customers, and did not see anyone using Excel to actually perform what you would call 'calculations.' Almost all of them were using Excel because it was a convenient way to create a table." — http://www.joelonsoftware.com/items/2012/01/06.html

III. METHODOLOGY

This section describes our approach to extracting spreadsheets and their associated data from Common Crawl.

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. S3 supports downloading segments of files in this way.

We considered spreadsheets from the period of Summer 2013 through 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. Extracting such a dataset from a single desktop machine is computationally intractable, and thus we extracted the spreadsheets using the Amazon Elastic MapReduce service.

A. Hadoop MapReduce Pipeline

We now describe our MapReduce architecture. 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.~~ For each stage in the architecture, we compute the cost in terms of normalized instance hours, that is, the approximate number of compute hours for the stage based on a 1 vCPU, 1.7 GiB machine — in order worth, roughly comparable to a single end-user desktop machine.

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 MIME types as supported by Microsoft Excel.[4] 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 the 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 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 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 1,000 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.

B. Local Operations

Using the SHA-1 hash of the records, a local (non-MapReduce), deterministic de-duplication operation is performed. The result of this operation is 249,376 unique spreadsheets. Locally, the JSON metadata documents for the three layers of FUSE are also converted to a form readable by MongoDB.

IV. Data Schema

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
  }
}
```

The `InternetDomain` element is useful for analysis relating to the origin of a spreadsheet. It uses the `WARC-Target-URI` and extracts the host, the top private domain, and a public suffix[5]:

```
{
  "WARC-Target-URI":
      "http://www.example.org/results/test.xls",
  "InternetDomainName": {
```

[4] https://technet.microsoft.com/en-us/library/ee309278.aspx

[5] https://publicsuffix.org/

```
    "Host": "www.example.org",
    "Top-Private-Domain": "example.org",
    "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 cells, the number of formulas used more than 50 times, the most common formula used, and so on.

V. Trade-offs

In this section, we articulate the trade-offs of FUSE in the context of other corpora that provide binary spreadsheets. The EUSES corpus of 4,498 unique spreadsheets is obtained predominately through parsing the top-ranked Google search results for simple keywords, such as "finance" [9]. In contrast, Fuse has no explicit classification for each spreadsheet, though it may be possible to infer a classification using the `LingPipe` tokens. However, unlike FUSE, EUSES is not reproducible. First, it provides no URL information to obtain the origin for each spreadsheet. Second, the methodology is fundamentally non-deterministic, because Google search results are non-deterministic.

The Enron corpus contains 15,770 spreadsheets extracted from e-mails obtained as legal evidence [10]. Unlike FUSE, Enron is a domain-specific corpus, accounting, and consequently each spreadsheet contains significantly more formulas than a general corpus such as ours. In the same vein, FUSE can only offer spreadsheets that are intentionally (or inadvertently) made publicly accessible, and a result, may contain fewer errors than spreadsheets not for public dissemination. On the other hand, FUSE results suggest that formula-heavy accounting spreadsheets are not representative of general spreadsheet users. Finally, the Enron corpus is forever fixed.

Irrespective of other corpora, FUSE has other challenges and limitations. A significant limitation is that Common Crawl restricts its storage of binary files to 1 MB. As a result, large spreadsheets are not available in FUSE. However, if one is willing to give up reproducibility, they may use the `WARC-Target-URI` from the Web Analysis Layer to download them using a similar technique as WEB [11], which provided researchers with a list of spreadsheet URLs from a 2009 web crawl. Yet another limitation is that the methodology for Common Crawl primarily geared towards text-based HTML pages, not binary files. Consequently, any spreadsheets within Common Crawl are only incidental, and not by design. Finally, our analysis tools do not support very old, so-called `BIFF5` format spreadsheets.

Researchers should consider these trade-offs and limitations before using choosing FUSE.

VI. Conclusion

This paper contributes a spreadsheet corpus, FUSE, derived from the Common Crawl. FUSE offers properties not available in existing corpora, including reproducibility and extensibility. Mining software repositories is an inherently cyclic activity: mining data informs insights that require further mining. Our JSON metadata bootstraps this process with a broad set of binaries and metadata, but it is only through custom mixins developed by other researchers that the full potential of FUSE can be realized.

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] 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.

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

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

[11] 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.