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 4 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, illustrate example queries to highlight the features of the dataset, 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)	b	0.64	23.3	110
Access	All	Researchers	All	All
Unique formulas	894361	693266	84004	–
Extendable	Yes	Not scalable	No	Yes
Framework	Hadoop	Excel/VBA	Scantool	–
Time Period	2006	2006	2006	2009
Origin	CC	Google	Enron	ClueWeb09
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 archive, made accessible to the research community.[1]
- A modular, open-source pipeline of tools, implemented using MapReduce. Our tool supports scalability from 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.

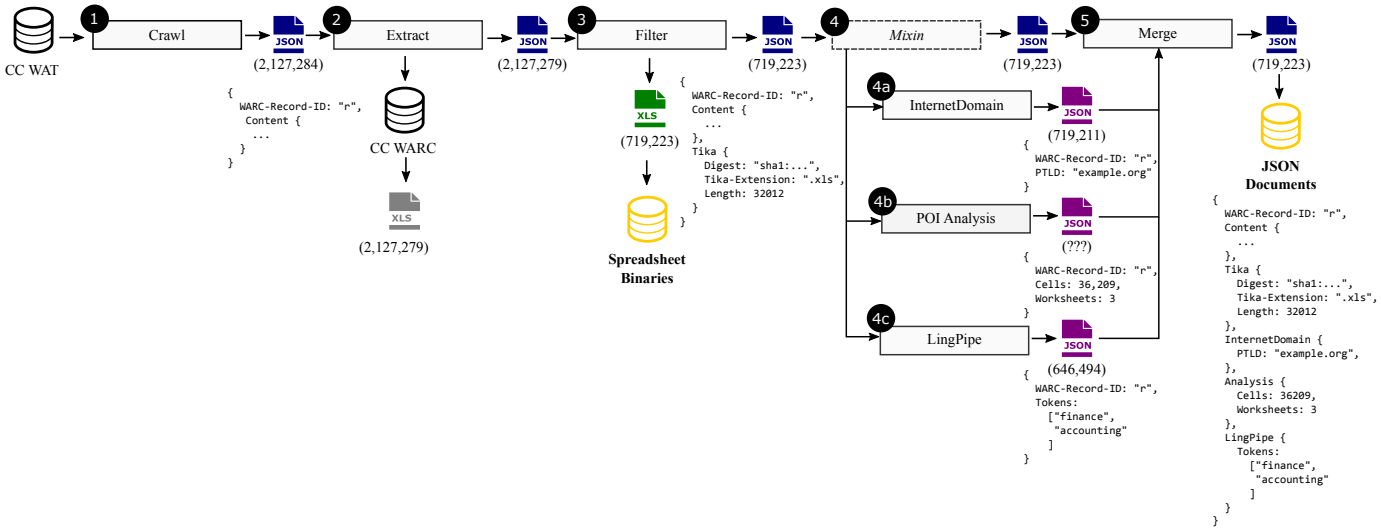

Fig. 1. The MapReduce pipeline for extracting spreadsheets and associated spreadsheet metadata from Common Crawl.

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 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 files, store the raw crawl data, and a corresponding WAT file, which stores the computed metadata for the for the 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 download 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 the data being co-located with the service.

[2]http://commoncrawl.org

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 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.

The crawl filters through the some 26.83 billion records and identifies 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 loads the 2,127,284 candidate spreadsheets records. For each record, the task

3) Filter: Because the WAT files contained some incomplete and incorrect tags, we extracted some invalid spreadsheets in the first phase. We used Tika[4] to identify the valid spreadsheets and then filtered out the invalid files. This stage resulted in ??? valid spreadsheets.

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

[4]http://tika.apache.org

4) Mixin: Next we processed the valid spreadsheets using Apache POI[5] to extract metadata from each spreadsheet (see Section III). By computing a sha512 hash we identified and removed duplicate spreadsheets, resulting in a final total of ??? spreadsheets.

5) Merge: In this step,

III. Data Schema

After MERGE (Stage 5) the data schema of the metadata consists of 5 parts, each of which will be described in more detail. These are 5 parts are all encoded in JSON, merged into one top level object.

A. The WARC Record

The WARC (Web ARChive) contains information about the original file when it was downloaded by the Common-Crawl. The WARC format is well-documented [CITATION NEEDED], so we will only briefly discuss it here. Fields of interest include `WARC-Target-URI`, the original url containing the spreadsheet, and `Container`, the containing Common-Crawl file and offset used to extract the spreadsheet from the crawl.

B. Tika Content-Type

Our pipeline leverages Apache Tika[6] to determine the content type of the candidate spreadsheet. The Tika JSON object contains 4 fields: Tika-Content-Type, Tika-Extension (file extension), a SHA1 hash digest and the size of the file.

C. InternetDomain

The JSON object produced by the InternetDomain mixin contains the original URL of the file, as well as three fields produced by parsing this URL. These are the host (e.g. www.example.com), the domain (example.com) and the suffix (.com).

D. LingPipe Tokens

The LingPipe[7] mixin analyses the web page containing the spreadsheet and produces a list of keywords that may ease querying. These keywords are an array embedded in the LingPipe JSON object. *Stop words*, that is, frequently appearing but unhelpful words like "as", "the", have been removed.

E. Spreadsheet Content Analysis

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[8] 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, etc.

[5]http://poi.apache.org

[6]http://tika.apache.org/

[7]http://alias-i.com/lingpipe/

[8]http://poi.apache.org/

TABLE II
COMMON CRAWL ARCHIVE

Description	TB	Yield
Summer 2013	30.6	42.14%
Winter 2013	35.1	33.46%
March 2014	36.4	
April 2014	41.2	
July 2014	59.2	
August 2014	46.6	
September 2014	48.9	
October 2014	59.1	
November 2014	31.4	
December 2014	35.2	

F. Querying the Metadata

The metadata file can be downloaded from HERE, unzipped, and then loaded into mongoDB[9] using the following command: `mongoimport --db spreadsheets --collection fuse --file fuse.metadata.json` Standard mongoDB queries can then be done on the analysis, such as

```
db.fuse.aggregate(
    { "$match":
      {"Tika.Length" : { $gt: 1000000 } } },
    { "$group":
      {"_id": "Big Spreadsheets",
 "count": {"$sum": 1} } })
```

which counts how many spreadsheets are larger than 1 MB.

The metadata we collected for the indices was largely influenced by the summary statistics presented in [11]. For each spreadsheet, there are over 450 entries, so we will not list them all here. In general, the entries summarize the contents of the cells. To list a few examples, the number of times a given Excel function (such as `SUM` or `VLOOKUP`) is used, the total number of input or data cells, the number of numeric input cells, the number of formulas used more than 50 times, the most common formula used, etc.

how did we decide to filter? MSDN content-type

cleaning

IV. Description of Spreadsheet Corpus

since it's sdescription of corpus, should be descriptive? counts avg etc.

absfrequency, relativefrequency gov, 250211 org, 171845 com, 112573 edu, 63572 gov.au, 43490 pa.us, 7621 net, 7409 mn.us, 4641 ac.uk, 3423 ca.us, 3003

absfreq, relfreq census.gov, 143135 triathlon.org, 106486 amamanualofstyle.com, 45118 abs.gov.au, 42941 utah.gov, 22242 ohio.gov, 16739 usda.gov, 13016 worldbank.org, 11062 theahl.com, 10350 eia.gov, 8216

RQ: How many domains are represented?

4,381 domains. After dedup, 4,342.

RQ what types of headers do people use?

how canonical are urls?

not very

[9]https://www.mongodb.org/

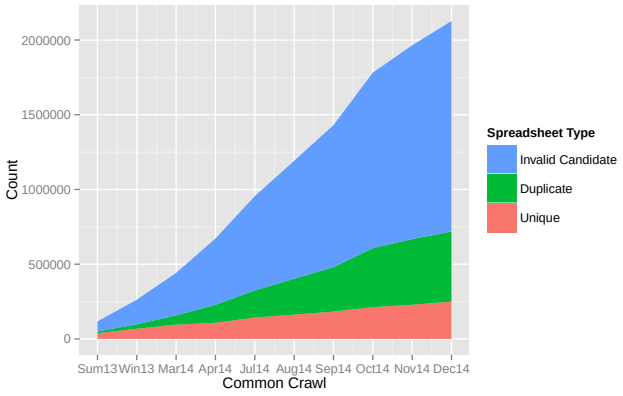

Fig. 2. Cumulative count of candidate spreadsheets with each additional crawl. One problem is that unless the diversity of the crawl increases, unique spreadsheets are reaching a local maximum.

results from this can be used to guide future crawls to increase the diversity of the spreadsheet corpus.

RQ: How much can you trust HTTP headers?

RQ: Train a text classifier to identify topicality.

RQ: Evolution of spreadsheets

Breakdown of analysis:

"OK", 220760 "BIFF5", 17643 "OTHER", 10782 "CORRUPT",129 "ENCRYPTED", 62

Total Input cells: 357210294

Total Formula cells: 10776903

Total Non-empty cells: 357210294 + 10776903 = 367987197 Average non-empty cells per workbook: 367987197/220760 = 1667

Number of workbooks with formulas: 14782

Number of formulas/workbook with formulas: 10776903/ 14782 = 729

Number of unique formulas: 894361

Number of unique formulas/workbook with formulas: 894361/ 14782 = 60

Number of different functions used: 219

Total Sheets: 346247

Maximum number of sheets: 147

Programming in spreadsheets:

154596 of our unique formulas contained `IF`, or one of its cousins like SUMIF, COUNTIF. 150461 of our unique formulas contained IF (and maybe other functions). 44584 of our unique formulas contained IF two or more times. 9816 unique formulas used IF 5 or more times, typically in a nested fashion 302 unique formulas used IF ten or more times, typically in a nested fashion. This may indicate a need for a more robust branching.

2496350 of our 10 million formula cells used an IF function, 94978 cells contained a SUMIF (these numbers may overlap as a cell may have both a SUMIF and an IF (we noted over 5000 unique formulas that had both an IF and a SUMIF)

The function breakdown between the three corpora is interesting. Some functions are used a similar amount (e.g. IF) and others are used differently (e.g. SUM, ISBLANK, VLOOKUP, ISTEXT). This underscores the need for large, diverse corpora. For example, all three corpora used IF about the same, whereas FUSE contains many more string-manipulation functions and Enron uses more financial functions. It may also be interesting to explore function "synonyms", which occur when there is more than one way to achieve the same result. For example, in FUSE and Enron, workbooks are more likely to use the + operator than the SUM function, but in EUSES, those tools are used at about the same rate.

TABLE III
SELECTED FUNCTIONS WITH COUNTS PER 1000 FORMULA CELLS

	FUSE	Enron	EUSES
IF	178.8	156.9	166.8
+ (operator)	166.4	217.5	167.6
SUM	87.7	80.3	153.6
ISBLANK	57.8	0.1	27.4
VLOOKUP	30.8	52.3	12.2
HLOOKUP	9.5	2.8	1.4
AVERAGE	32.0	9.2	7.9
AND	6.6	15.9	21.7
NOT	5.5	0.0	1.9

TABLE IV
SPREADSHEET CLASSIFICATION

Category	FUSE	EUSES
Database	4518	720
Finance	3058	780
Grade	2915	731
Homework	61	682
Inventory	2243	756
Model	2143	966

A summary of your spreadsheet corpus can be found in Table I.

V. RESULTS

This dataset suggestions many interesting questions.

A. Classification of spreadsheets

The results in this section are intended to demonstrate the essential properties of the corpus.

Top two formulas are the same across the three corpora: 1. Add up the three cells to my left 2. Add up the two cells to my left And most of the top 10 are Add up n cells to my left. Fuse #3 HYPERLINK("http://www.eia.doe.gov/totalenergy/data/monthly/dataunits.cfm","N Information about data precision and revisions.") Enron #3 NOW()

give warc record ids

VI. CHALLENGES AND LIMITATIONS

relevance is dependent on common crawl definition of relevant

VII. CONCLUSION

The conclusion goes here.

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 *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. IEEE, 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, V. Pipek, M. Rosson, B. de Ruyter, and V. Wulf, Eds. Springer Berlin Heidelberg, 2009, vol. 5435, pp. 15–28. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00427-8_2

[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. Washington, DC, USA: IEEE Computer Society, 2012, pp. 409–418. [Online]. Available: http://dx.doi.org.prox.lib.ncsu.edu/10.1109/ICSM.2012.6405300

[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. Washington, DC, USA: IEEE Computer Society, 2012, pp. 399–409. [Online]. Available: http://dx.doi.org.prox.lib.ncsu.edu/10.1109/ICSM.2012.6405299

[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. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2008.06.001

[10] R. Abraham and M. Erwig, "Inferring templates from spreadsheets," in *Proceeding of the 28th international conference on Software engineering - ICSE '06*. New York, New York, USA: ACM Press, May 2006, p. 182. [Online]. Available: http://dl.acm.org.prox.lib.ncsu.edu/citation.cfm?id=1134285.1134312

[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 M.-H. Emerson, "Enrons spreadsheets and related emails: A dataset and analysis," in *ICSE SEIP '15*, ser. ICSM '12, 2015.

[13] Z. Chen and M. Cafarella, "Automatic web spreadsheet data extraction," in *Proceedings of the 3rd International Workshop on Semantic Search Over the Web - SS@ '13*. New York, New York, USA: ACM Press, Aug. 2013, pp. 1–8. [Online]. Available: http://dl.acm.org.prox.lib.ncsu.edu/citation.cfm?id=2509908.2509909