Scaling OpenRefine

August 1, 2019

For nearly ten years, OpenRefine has served the needs of data science communities. As a leading open source power tool to work with messy data, it is taught in countless courses and workshops around the world. In biomedical research alone, OpenRefine is cited in hundreds of scientific articles, in fields such as genomics [1, 2, 3, 4], Alzheimer's disease [5, 6, 7], infectious diseases [8, 9, 10], oncology [11, 12, 13, 14] and clinical data management [15, 16, 17, 18]. To keep OpenRefine thriving in the next ten years, we need to grow its contributor community and undertake fundamental improvements to its architecture.

1 GOALS

The main objectives of this proposal are twofold:

1. **grow the community of OpenRefine contributors** by reaching out to seasoned users and helping them get involved more closely in the project.

2. **revamp the core architecture of the tool** to handle larger datasets and improve workflows.

Our goals will be reached if:

1. **We onboard at least 20 new contributors** over the course of the project, and get contributions from 20 existing contributors. Contributions are pull requests to any of the repositories in the GitHub organization. They include translations, documentation, website curation and code.

2. **OpenRefine can be used on datasets which do not fit in memory**. By this we mean that the deserialized data does not fit in the working memory allocated to the Java Virtual Machine. The dataset might still be smaller than the working memory when exported to a file.

2 GROWING THE CONTRIBUTOR COMMUNITY

One of the great assets of the OpenRefine project is its vibrant user community, which organizes training workshops, writes tutorials, provides high quality support and advertises the project.[1] This community is healthy, diverse socially and geographically.

Comparatively, the development team is of course small and but also less diverse. This is a very common issue in open source projects, but in the case of OpenRefine it can also be traced back to the project's history, as it was initially a product developed by Metaweb and then Google. The transition to an open source community required the formation of a group of committed volunteers to take over. With six stable releases since the transition to an open source community, the project has proved its viability in this form. However, we believe it would benefit from growing and diversifying its contributor community.

We propose to take the following steps to make it easier for seasoned users to join the project.

2.1 Developing an official user guide

OpenRefine's documentation is currently written in the GitHub wiki associated with the project,[2] which is imperfect in many ways: it is hard to discover, search and browse. There is no simple way to version it or translate it. As a consequence, users tend to write their own tutorials on external platforms rather than to improve the official documentation.[3] Some documentation efforts play the role of official guide in certain communities.[4] This results in a myriad of tutorials, all with a slightly different focus but with often a significant overlap in content, which gradually become outdated as the software evolves. As a user, this makes it hard to discover the best resources to learn about certain aspects of the tool.

We propose to migrate the official documentation to a proper platform such as ReadTheDocs, where it can be versioned and translated natively. By providing an appealing space to write documentation, we hope to encourage existing tutorial authors to contribute directly to it. This should help us grow our contributor base and make it more diverse.

2.2 Developing a contributor guide

The existing documentation to get started on OpenRefine development does not help contributors enough. Beyond the migration to a proper documentation platform as mentioned above, we need to give instructions to get started with the most common Java IDE platforms (Eclipse, IntelliJ, Netbeans) and provide detailed guidance about common tasks (writing new GREL functions, operations, importers). We can then link to these instructions from the issues, to encourage prospective contributors to step in.

1 This activity can be gauged by mentions of the tool on Twitter: `https://tinyurl.com/yyfj3d5y`
2 `https://github.com/OpenRefine/OpenRefine/wiki`
3 See for instance the Japanese translation of the reconciliation documentation.
4 Such as the Library Carpentry's OpenRefine lesson or Mathieu Saby's OpenRefine course.

3 REVAMPING THE CORE DATA MODEL

OpenRefine currently uses its own data storage backend. This choice is the root cause of important limitations in the tool, that we outline below. Changing our data model with these issues in mind is an important investment for the project to remain relevant in data-intensive fields, such as biology and medical research.

3.1 Working with large data sets

The requirement to load entire projects in memory to work on them makes it impractical to work on large datasets such as in genomics medicine. This requires users to adapt the hard memory limit given to the Java Virtual Machine depending on the size of their datasets and their available memory, when they can afford enough RAM.

3.2 Refining collaboratively

OpenRefine is a web-based tool that is designed to be run locally by the user. Although it can be hosted on a server, it is not designed for collaborative work. As operations are applied in sequence to the project, working simultaneously on disjoint parts of a dataset is rarely viable. The tool does not even have a notion of "user" which would let it track who performed each change.

3.3 Analyzing, sharing and reusing workflows

The ability to extract workflows as JSON objects and reapply them on other projects is a flagship feature of the tool. However, it has serious limitations. It is hard to understand what a workflow does by looking at its representation in JSON, or even at the project history in the tool itself. There is no simple way to reorganize a workflow, isolate reusable parts or undo selected operations buried in the history.

3.4 Running workflows in production

Once a workflow has been created, one could want to run it periodically as part of a wider pipeline. Although many of OpenRefine's operations can be easily parallelized [19], there is no simple way to run them on data streams discovered progressively. The scheduling of operations is also naive, as they are executed in sequence without any time sharing.

4 WORK PLAN

The first objective, to grow the contributor community, will be tackled by Owen Stephens. Owen is best placed for this work since he is a long standing contributor to the project on many aspects, writing documentation, contributing and reviewing code. He also runs many workshops in universities to teach OpenRefine to researchers and has therefore a deep understanding of the needs of the user community.

The second objective, to revamp the data model, will be handled by Antonin Delpeuch. As author of some of the research behind the proposed new architecture [20], Antonin is in a good position to implement it in OpenRefine. He has experience with carrying out large refactoring projects on OpenRefine's legacy code and cares deeply about how implementation decisions will be perceived by the end user.

4.1 Growing the community

Our strategy to attract new contributors and retain existing ones is as follows. The timescale is given on the assumption of Owen Stephens working part time on this goal (0.5 full time equivalent).

1. **Migrate the existing documentation to a new platform** - 1 month

   We obtain a versioned and translatable documentation on ReadTheDocs. Existing documentation pages redirect to the new platform to avoid creating dead links.

2. **Write tutorials for prospective contributors** - 2 months

   We document IDE set up, testing, making good pull requests. We show example workflows to implement new functions, importers or exporters.

3. **Improve the existing user documentation** - 6 months

   All operations, facets, expression languages, importers and exporters are documented.

4. **Organize events focused on contribution to the project** - 2 months

   On the occasion of OpenRefine's 10th anniversary (in 2020), we will organize local events in targeted communities[5] to show how to contribute to the translations, documentation and code. Use the feedback obtained at such events to inform the tasks above.

4.2 Revamping the data model

The new data model required to address all the issues mentioned in the previous section is ambitious. We anticipate that implementing it in full is likely to take significantly longer than a one year project. We decompose the work needed into packages with intermediate deliverables. We anticipate one year of full time development should cover packages 1 to 5. The other goals are mentioned to relate the work to the project's long term milestones. The timescale is given on the assumption of Antonin Delpeuch working full time on this goal for the entire duration of the project.

5 Given the growing popularity of OpenRefine in the Wikimedia community, the Wikimedia Hackathon is one possible venue for this.

1. **Preliminaries: expression languages** - 1 month

   Determining which column an expression depends on requires the ability to analyze expressions and extract the variables it reads.

   Performing this analysis requires changing the interface implemented by our expression languages (currently GREL, Jython and Clojure by default) to allow for such an inspection. This can be done easily for GREL but requires more work for other interpreters. Given that we already have plans to migrate from Jython to GraalPython, we propose to carry out this migration and implement this analysis using the Truffle library (which is used to define interpreters in the Graal framework).

2. **Column-based operations** - 2 months

   We introduce a new interface for row-wise, out of order operations reading from a selection of columns. Existing operations which satisfy these constraints can implement this interface. Similarly, facet configurations expose which columns they depend on. This new data model is based on the representation proposed in [20]. At this stage the data processing backend does not change yet: everything is still held in memory.

3. **New data processing backend** - 4 months

   We migrate workflow execution to a dataflow backend such as Apache Spark [21] through Apache Beam's API [22]. In this version, the tool can still execute workflows serialized in the current format, but cannot read project data from previous versions.

   We anticipate that the main benefits for end users at this stage are mostly felt in memory management (ability to work on datasets which do not fit in RAM) and marginal speed improvements. These benefits become important in data-intensive sciences, such as biomedical research.

4. **Workflow visualization** - 2 months

   We add diagrammatic representations of the list of operations that are implemented, letting users understand better the structure of their workflows. This is again based on [20].

5. **Operation reordering** - 2 months

   We introduce the possibility of reordering operations in the undo/redo history. This feature would only be available for row-wise operations working on different columns, implementing the new interface. Reordering could be triggered directly by dragging nodes on the graphical visualization.

6. **Concurrent operations** - if time allows

   We add the possibility of running independent operations concurrently, such as reconciling two different columns in parallel. The workflow representation should also make it possible to execute different operations on disjoint subsets of rows, but this is likely to be harder to expose to the user.

7. **Partial computations** - if time allows

   We return the control to the user before the full computation of long-running operations terminates. For instance, when reconciling a column, the user can start inspecting the first few reconciliation results immediately. They can perform other operations while reconciliation is progressing. These other operations are executed immediately even if they depend on the reconciliation results. This relies on the streaming capabilities of the underlying execution backend.

8. **Multi-user support** - if time allows

   We add a notion of a "user" to keep track of the author of each change and make necessary adjustments to the interface and server to let multiple users work on the same project seamlessly.

5 EXISTING SUPPORT

OpenRefine has been supported in 2018 by a 100,000 USD grant from the Google News Initiative.[6] These funds have been fully used for contract work by the core team in 2018. OpenRefine development is currently carried out on a voluntary basis.

The research which designed the new architecture for OpenRefine [20] was carried out by Antonin Delpeuch as part of his DPhil at the University of Oxford, which is supported by the Engineering and Physical Sciences Research Council[7]. in the United Kingdom.

6 https://newsinitiative.withgoogle.com/
7 https://epsrc.ukri.org/

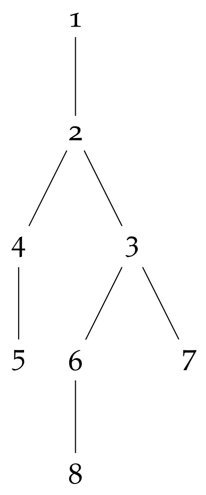

Figure 1: Dependencies between the work packages for the new architecture

REFERENCES

[1] Benedikt Kampgen, Horst Werner, Radwan Deeb, and Christof Bornhovd. Towards a Semantic Clinical Data Warehouse: A Case Study of Discovering Similar Genes. page 6, 2015.

[2] Gurnoor Singh, Arnold Kuzniar, Erik M. van Mulligen, Anand Gavai, Christian W. Bachem, Richard G.F. Visser, and Richard Finkers. QTLTableMiner++: Semantic mining of QTL tables in scientific articles. *BMC Bioinformatics*, 19(1):183, May 2018.

[3] Terje Klemetsen, Inge A. Raknes, Juan Fu, Alexander Agafonov, Sudhagar V. Balasundaram, Giacomo Tartari, Espen Robertsen, and Nils P. Willassen. The MAR databases: Development and implementation of databases specific for marine metagenomics. *Nucleic Acids Research*, 46(D1):D692–D699, January 2018.

[4] Jascha Silbermann, Catrin Wernicke, Heike Pospisil, and Marcus Frohme. RefPrimeCouch—a reference gene primer CouchApp. *Database*, 2013, January 2013.

[5] Xiaohui Yao, Jingwen Yan, Michael Ginda, Katy Börner, Andrew J. Saykin, Li Shen, and for the Alzheimer's Disease Neuroimaging Initiative. Mapping longitudinal scientific progress, collaboration and impact of the Alzheimer's disease neuroimaging initiative. *PLOS ONE*, 12(11):e0186095, November 2017.

[6] Vasco de Almeida Jorge Veríssimo. Data acquisition, curation and modeling for integration of Alzheimer's disease neuroimaging data from ADNI in the translational biomedicine platform tranSMART. 2015.

[7] I. P. A. Schilder. Dynamics of Search Spaces in the Drug Development Process of Alzheimer's Disease. http://dspace.library.uu.nl/handle/1874/355823, 2017.

[8] Jose Antonio Garrido-Cardenas, Concepción Mesa-Valle, and Francisco Manzano-Agugliaro. Genetic approach towards a vaccine against malaria. *European Journal of Clinical Microbiology & Infectious Diseases*, 37(10):1829–1839, October 2018.

[9] Christiane Hagel, Felix Weidemann, Stephan Gauch, Suzanne Edwards, and Peter Tinnemann. Analysing published global Ebola Virus Disease research using social network analysis. *PLOS Neglected Tropical Diseases*, 11(10):e0005747, October 2017.

[10] Maulik R. Kamdar and Michel Dumontier. An Ebola virus-centered knowledge base. *Database*, 2015, January 2015.

[11] Yaoyu E. Wang, Lev Kutnetsov, Antony Partensky, Jalil Farid, and John Quackenbush. WebMeV: A Cloud Platform for Analyzing and Visualizing Cancer Genomic Data. *Cancer Research*, 77(21):e11–e14, November 2017.

[12] Deborah L. McGuinness, Abdul R. Shaikh, Richard Moser, Bradford W. Hesse, Glen D. Morgan, Erik M. Augustson, Yvonne Hunt, Zaria Tatalovich, Gordon Willis, and Kelly Blake. A Semantically-enabled Community Health Portal for Cancer Prevention and Control. Technical report, RENSSELAER POLYTECHNIC INST TROY NY, June 2011.

[13] Jermaine Goveia, Andreas Pircher, Lena-Christin Conradi, Joanna Kalucka, Vincenzo Lagani, Mieke Dewerchin, Guy Eelen, Ralph J DeBerardinis, Ian D Wilson, and Peter Carmeliet. Meta-analysis of clinical metabolic profiling studies in cancer: Challenges and opportunities. *EMBO Molecular Medicine*, 8(10):1134–1142, October 2016.

[14] Karak&uuml;lah G&ouml;khan, Suner Asl&inodot;, Adlassnig Klaus-Peter, and Samwald Matthias. A Data-driven Living Review for Pharmacogenomic Decision Support in Cancer Treatment. *Studies in Health Technology and Informatics*, pages 688–692, 2012.

[15] Maria-Elena Hernandez, Sean M. Falconer, Margaret-Anne Storey, Simona Carini, and Ida Sim. Synchronized Tag Clouds for Exploring Semi-structured Clinical Trial Data. In *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, CASCON '08, pages 4:42–4:56, New York, NY, USA, 2008. ACM.

[16] Deepak K. Sharma, Harold R. Solbrig, Eric Prud'hommeaux, Kate Lee, Jyotishman Pathak, and Guoqian Jiang. D2Refine: A Platform for Clinical Research Study Data Element Harmonization and Standardization. *AMIA Summits on Translational Science Proceedings*, 2017:259–267, July 2017.

[17] Deepak K. Sharma, Harold R. Solbrig, Eric Prud'hommeaux, Jyotishman Pathak, and Guoqian Jiang. Standardized Representation of Clinical Study Data Dictionaries with CIMI Archetypes. *AMIA Annual Symposium Proceedings*, 2016:1119–1128, February 2017.

[18] A. Kiourtis, A. Mavrogiorgou, and D. Kyriazis. Aggregating Heterogeneous Health Data through an Ontological Common Health Language. In *2017 10th International Conference on Developments in eSystems Engineering (DeSE)*, pages 175–181, June 2017.

[19] Alberto Montresor, Andrii Bratus, and Giuliano Mega. *RefineOnSpark: A Simple and Scalable ETL Based on Apache Spark and OpenRefine*. PhD thesis, University of Trento, 2014.

[20] Antonin Delpeuch. A complete language for faceted dataflow programs. In *arXiv:1906.05937 [Cs, Math]*, To Appear in Proceedings of the Applied Category Theory 2019, Oxford, June 2019.

[21] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[22] Tyler Akidau, Eric Schmidt, Sam Whittle, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, and Frances Perry. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8(12):1792–1803, August 2015.