Co-AI: A Colab-Based Tool for Abstraction Identification

Zedong Peng
Department of Electrical Engineering and Computer Science
University of Cincinnati
Cincinnati, OH, USA
pengzd@mail.uc.edu

Nan Niu
Department of Electrical Engineering and Computer Science
University of Cincinnati
Cincinnati, OH, USA
nan.niu@uc.edu

***Abstract*— Abstraction identification is aimed at discovering significant domain terms. Prior work, notably AbstFinder and RAI (relevance-driven abstraction identification), has introduced the core ideas, but offered only limited tool support. This paper presents our abstraction identification tool, Co-AI, built on the Google Colab environment allowing the users to run the tool within their web browsers, promoting tool adoption and extension. Co-AI integrates the Wikipedia pages as the domain corpus, and identifies the candidate abstractions with a set of natural language processing (NLP) patterns. Co-AI is available at: https://colab.research.google.com/drive/1ur5KILoi_n-3KY0_vJcMBQDtiSYgcYeP?usp=sharing and we welcome the community's feedback of our tool.**

***Index Terms*—abstraction identification, natural language processing, Google Colab.**

I. Introduction

In requirements engineering (RE), an *abstraction* is referred to as a general idea or concept that has a particular significance in a given domain [1]. For example, "strip", "radar", and "sector" are all recognized as abstractions in the air traffic control problem domain [2]. Abstraction identification can serve a couple of purposes in RE [1]: (i) the act of recognizing and organizing the abstractions helps the requirements engineer understand the problem domain, the entities it encapsulates, and their relationships, and (ii) the set of abstractions provides a lexicon of terms or project dictionary that helps stakeholders and requirements engineers communicate effectively.

In order to reduce the requirements engineer's manual effort in searching for the significant terms, researchers have developed methods to automatically identify the abstractions from the natural language (NL) documents. The seminal work of AbstFinder treats text as byte streams and searches for patterns of co-occurring byte sequences within pairs of sentences using a series of circular shifts [3]. Fig. 1 illustrates AbstFinder's idea that important abstractions would recur frequently as repeated words within the target document.

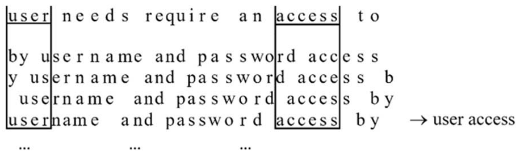

user needs require an access to
by username and password access
y username and password access b
sername and password access by
username and password access by
→ user access

Fig. 1. Illustration of AbstFinder.

Compared to the relatively straightforward implementation of AbstFinder, the relevance-driven abstraction identification (RAI) proposed by Gacitúa *et al.* [1] is built on the idea of corpus-based frequency profiling. Specifically, the log-likelihood (LL) value for a word w is computed as:

$$LL_w = 2\left(w_d \cdot \ln\frac{w_d}{E_d} + w_c \cdot \ln\frac{w_c}{E_c}\right) \tag{1}$$

where w_d is the number of times w appears in the domain documents, and w_c is the number of times w appears in the normative corpus. While w_d and w_c are observed values of w, E_d and E_c in equation (1) are expected values: $E_d = n_d \bullet (w_d + w_c) / (n_d + n_c)$ and $E_c = n_c \bullet (w_d + w_c) / (n_d + n_c)$, where n_d is the total number of words in the domain documents, and n_c is the total number of words in the normative corpus. The normative corpus used in [2] is the British National Corpus (BNC) which contains around 100 million words of running English text. LL_w thus measures w's deviation from its frequency in the normative corpus: the greater LL_w is, the more significant w is in the domain corpus with respect to the normative corpus.

Although equation (1) underpins RAI, using the approach without tool support is difficult [4], e.g., incorporating any normative corpus of BNC's scale presents non-trivial challenges for the end user. At any rate, having an abstraction identification tool that is free, available, and extensible will offer direct value to the requirements engineers.

To offer such support, we have leveraged Google Colab[1] to build an automated tool. We call our tool Co-AI, standing for **Co**lab-Based **A**bstraction **I**dentification. Google Colab, also known as Google Colaboratory, is a cloud service based on Jupyter Notebooks[2] for developing and disseminating Python programs. It provides significant functionality for software development and free-of-charge access. Some researchers regard it as a high-quality platform [5], and abundant benefits can be observed:

- **Easy setup**. It takes a couple of minutes to set up a new account, and some most commonly used libraries have been installed, such as panda, NumPy, etc.
- **Programming within a web browser**. The platform's code execution can be treated as a real-time web application and can be run cross-platform on Windows, Linux, etc. Similar to web applications, it shares some standard features and advantages, such as zooming, adaptive window, and page inspect.
- **Flexible access and client secure login.** Colab integrates seamlessly with other services like Google Drive, enabling authorized view and co-edit.
- **Easy to collaborate.** Everyone is accessing the same version of the website (Google Colab) with the same URL. The user will always be accessing the most up-to-date version of the software.

The above list makes Google Colab a suitable environment for building research prototypes. We therefore designed and

[1]Google Colaboratory: https://research.google.com/colaboratory/
[2]Jupyter Notebooks: https://jupyter.org/

developed Co-AI on top of this environment to increase transparency and reduce the user's cost of carrying out abstraction identification.

II. Co-AI

Having introduced Google Colab's benefits, we now discuss some key features of Co-AI as follows.

(1) *Integration with Wikipedia pages, which serve as Co-AI's underlying domain corpus.* The data source that we use to identify the abstractions is Wikipedia[3]. Wikipedia articles focus on explaining the natural human concepts and shared knowledge about specific and related topics. Compare to using a textbook as the corpus [1], the range of expertise on Wikipedia articles are vertically integrated from broader to narrower views of the world [6].

(2) *Allowing the user to specify "seed page" and the size of corpus.* In Co-AI, we use Python web wrapper named Wikipedia-API[4] to extract a set of Wikipedia pages. As shown in Fig. 2 (area A), the user is asked to enter a validate seed page name and the number of pages to be collected. Co-AI will then automatically scrap the current seed page and related pages. The example on Fig. 2 (area A) illustrates that 6 pages will be scrap (1 seed page + 5 related pages) . The priorities considered for related pages will be: (a) seed page, (b) the content pages belonging to the seed page's topic, (c) the content pages belonging to the sub-categories of the seed page's topic, and (d) the content pages of the hyperlinks.

(3) *Built-in and extensible NLP patterns.* We define five patterns by exploiting the syntactic and grammatical roles that words play in a sentence. As shown in Fig. 2 (area B), each pattern can be selected alone or together to better identify abstractions. For better RE support, we operationalize to find testable abstraction candidates [6] by formatting an abstraction as a <key, value> pair. We expect the "key" to help locate "what to test", and the "value" to guide "how to test it" by feeding in concrete data. The example results are shown in Fig. 2 (area C), and pattern usages are also displayed.

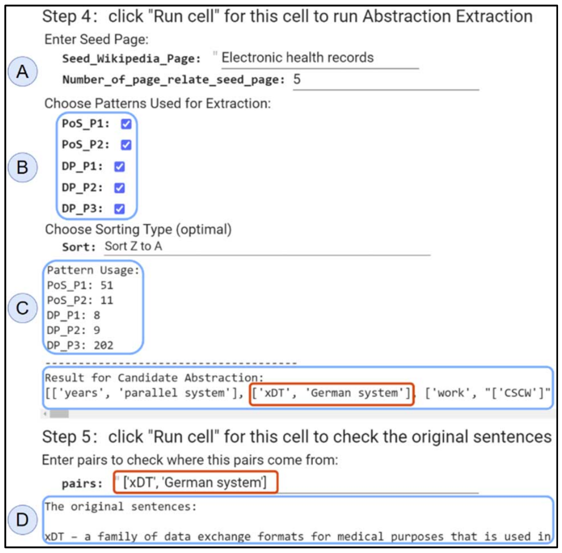
*(*4) *Sorting and explaining the candidate abstractions*. For result analysis, the <key, value> pairs can be sorted alphabetically. In Fig. 2 (area D), the user can enter one output pair from step 4, and an explanation will be displayed to show an original Wikipedia sentence where the pair appears. This will provide more context for the abstraction [7].

To quantitatively and practically evaluate different abstraction identification techniques, we compared AbstFinder, RAI, and Co-AI, and measured precision of the topmost 20, 100, and 200 results. For the "Electronic health records" domain, results show the average precision is 16.78% (AbstFinder), 18% (RAI), and 25.44% (Co-AI) [6].

III. SUMMARY

Abstraction identification can be thought of as where the expertise of domain expert and requirements engineer meet. Previous approaches, such as AbstFinder and RAI, offer limited tool support. In this paper, we have presented an automated tool that is freely available, easy to access, and extensible. Our Co-AI tool is built on the Google Colab environment, and leverages the NLP patterns to identify

[3]Wikipedia: https://en.wikipedia.org
[4]Wikipedia-API: https://github.com/martin-majlis/Wikipedia-API


Step 4: click "Run cell" for this cell to run Abstraction Extraction

Enter Seed Page:

Seed_Wikipedia_Page: Electronic health records

Number_of_page_relate_seed_page: 5

Choose Patterns Used for Extraction:

PoS_P1: ✓
PoS_P2: ✓
DP_P1: ✓
DP_P2: ✓
DP_P3: ✓

Choose Sorting Type (optimal)

Sort: Sort Z to A

```
Pattern Usage:
PoS_P1: 51
PoS_P2: 11
DP_P1: 8
DP_P2: 9
DP_P3: 202
--------------------------------------
Result for Candidate Abstraction:
[['years', 'parallel system'], ['xDT', 'German system'], ['work', "['CSCW']"
```

Step 5: click "Run cell" for this cell to check the original sentences

Enter pairs to check where this pairs come from:

pairs: ['xDT', 'German system']

```
The original sentences:

xDT - a family of data exchange formats for medical purposes that is used in
```

Fig. 2. Some Features of Co-AI.

candidate abstractions in the form of <key, value> pairs. Co-AI can be run inside a user's web browser, making use of Google Colab's cloud-based computing engines to perform NLP and RE tasks. While we have a backlog of features [8] to explore (e.g., clustering the candidate abstractions [9], visual exploration of the abstractions [10], and formulating environment assertions [11]), we welcome the community's feedback on Co-AI.

References

[1] R. Gacitúa, P. Sawyer, and V. Gervasi. "Relevance-Based Abstraction Identification: Technique and Evaluation", *Requir. Eng.*, 16(3): 251–265, 2011.

[2] P. Sawyer, P. Rayson, and K. Cosh. "Shallow Knowledge as an Aid to Deep Understanding in Early Phase Requirements Engineering", *IEEE Trans. on Software Eng.*, 31(11): 969–981, 2005.

[3] L. Goldin and D. M. Berry. "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation", *Autom. Softw. Eng.*, 4(4): 375–412, 1997.

[4] A. Dwarakanath, R. R. Ramnani, and S. Sengupta. "Automatic Extraction of Glossary Terms from Natural Language Requirements", In *RE*, 2013, pp. 314–319.

[5] T. Carneiro, R. da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. de Albuquerque, and P. P. R. Filho. "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications", *IEEE Access*, 6: 61677-61685, 2018.

[6] Z. Peng, P. Rathod, N. Niu, T. Bhowmik, H. Liu, L. Shi, and Z. Jin. "Environment-Driven Abstraction Identification For Requirements-Based Testing", In *RE*, 2021 (to appear).

[7] N. Maltbie, N. Niu, M. Van Doren, and R. Johnson. "XAI Tools in the Public Sector: A Case Study on Predicting Combined Sewer Overflows", In *ESEC/FSE*, 2021 (to appear).

[8] N. Niu, S. Brinkkemper, X. Franch, J. Partanen, and J. Savolainen. "Requirements Engineering and Continuous Deployment", *IEEE Softw.*, 35(2): 86–90, 2018.

[9] A. Mahmoud and N. Niu. "TraCter: A Tool for Candidate Traceability Link Clustering", In *RE*, 2011, pp. 335–336.

[10] S. Reddivari, Z. Chen, and N. Niu. "ReCVisu: A Tool for Clustering-Based Visual Exploration of Requirements", In *RE*, 2012, pp. 327–328.

[11] T. Bhowmik, S. R. Chekuri, A. Q. Do, W. Wang, and N. Niu. "The Role of Environment Assertions in Requirements-Based Testing", In *RE*, 2019, pp. 75–85.