Extraction Guideline

This document describes the attributes to extract from primary studies relevant to the NLP4RE tools book chapter.

Overview

The following attributes of NLP4RE tools are relevant to the extraction:

Attribute	Type	Values
Name	str	
Description	str	
Source	str	URL
Release	enum	[source code, executable]
License	enum	[cc, cc-by-4.0, cc-by-sa-4.0, gpl, lgpl, mit, …]
Availability	enum	[archived, open source, reachable, upon request, broken, unavailable, private, proprietary]
Requirements Engineering Phase	enum	[elicitation, analysis, modeling, validation & verification, management, other, multiple]
Task Type	enum	[detection, extraction, classification, transformation, tracing & relating, search & retrieval, generation]

str = String/text, enum = enumerable/discrete

Extraction Rules

The following rules guide the extraction of the tools from eligible articles. Every attribute is specified by a type (string or enum), description, purpose, extraction rule, and examples.

Name

String

Description: The name of a tool is a short, multi-word identifier, sometimes accompanied by an additional acronym. In several but not all cases, the name is explicitly provided by the authors of the paper containing the tool.

Purpose: The name acts as the identifier of the tool and should distinguish it from all other tools.

Extraction rule:

- If the authors explicitly state a name (and optionally an acronym), record it verbatim.
- If the authors do not state a name or acronym, select a name by nominalizing the main purpose of the tool.

Examples:

- In [2], the authors present a tool called the *Completeness Assistant for Requirements (CAR)*. This name and acronym are extracted.
- In [3], the authors present a tool with the words, “We developed an automated system to detect nocuous anaphoric ambiguities from full-text documents.” The name *Nocuous Anaphoric Ambiguities Detector* can be derived from this description.

Description

String

Description: The description of a tool is a short, verbose explanation of its main purpose and technology.

Purpose: This attribute acts as a short summary of the tool and should contain the following two pieces of information:

1. The main objective of the tool (i.e., a black-box view of the tool's functionality), to inform potential users of the tool whether it fits their use case.
2. The key technologies used to inform potential contributors about opportunities to improve the underlying technology.

Extraction rule: Fill the following template with all information provided by the authors: "A tool that takes <input> and <task> to produce <output> using <technology>." In case the authors provide a semantically comparable version of this information, use the verbatim description.

Examples:

- In [2], the authors describe their tool as follows: "We have implemented a prototype tool named Completeness Assistant for Requirements (CAR), which addresses these problems by automatically suggesting possible relevant terms and possible relevant relations among terms to be used in the requirements."
- In [3], the authors describe their tool as follows: "We developed an automated system to detect nocuous anaphoric ambiguities from full-text documents. [...] The initial input is a complete requirements document. The output is the set of selected sentences that contain potentially nocuous anaphoric ambiguities."

Source

String

Description: The source of a tool is the URL under which it can be found.

Purpose: This attribute acts as the pointer to access the tool in case it has been made available.

Extraction rule:

- If the authors explicitly provide a link to their tool, record that link.
- If the authors do not provide a link to their tool, manually search for it. In case it is findable, record that link.

Examples:

- In [8], the authors present an unnamed tool for identifying ambiguous terms and refer to https://doi.org/10.5281/zenodo.1476902 for the tool's location.
- In [4], the authors present their tool AQUSA. It is not directly referenced in the paper, but by searching for the tool name, the GitHub repository can be found (https://github.com/gglucass/AQUSA).

Release

Enum

Description: This attribute classifies the artifact (source code or executable) that has been released.

Purpose: The release determines how the tool can be evolved. While an executable can be easily reused, only the distribution of source code allows for the evolution of the tool.

Extraction rules:

Categories	Criteria
Source Code	The source code of the tool is available
Executable	A pre-compiled executable of the tool is available

A tool can have both source code and an executable connected to it.

Examples:

- In [4], the authors present their tool AQUSA, of which the source code is available on GitHub (https://github.com/gglucass/AQUSA).
- In [7], the authors present an extension to the tool ConQAT(https://www.cqse.eu/en/news/blog/conQAT-end-of-life/). While the source code is unavailable, binaries (an *executable*) are.

License

Enum

Description: This attribute classifies the license under which the tool can be reused.

Purpose: The license of a tool clarifies how it can be reused. This information is extracted to be able to filter the resulting list of tools for those that are legally available for reuse and evolution.

Extraction rules:

License	Extraction Criteria
None	The tool is available, but no license has been provided
Unknown	The tool is unavailable and the licensing is unclear
MIT	The tool contains an explicit MIT license
CC2.0	The tool contains a Creative Commons Attribution 2.0 Generic license
…	

Further licenses (to be added on demand):
https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository#searching-github-by-license-type

Examples:

- In [4], the authors present their tool AQUSA (https://github.com/gglucass/AQUSA), which can be reused under the **MIT license**.
- In [8], the authors present an unnamed tool that is accessible at https://doi.org/10.5281/zenodo.1476902 and licensed under the **Creative Commons Attribution 2.0 Generic license (CC2.0)**.
- In [9], the authors present a tool for automated CEG creation, which is located at https://github.com/fischJan/Automated_CEG_Creation. However, neither the repository itself nor the Readme file contains a license. Hence, the license is **None**.
- In [2], the authors present a tool but do not provide any link to it. The license remains **Unknown**.

Availability

Enum

Description: This attribute classifies a tool's degree to be accessed.

Purpose: The availability can be used as a filter out extracted tools that were only described in the paper but not disclosed. Available tools are relevant for reuse and evolution, while unavailable tools are relevant for artifact recovery attempts.

Extraction rules:

Characteristic	Criteria
Archived	The tool is hosted in a service satisfying all of the following criteria: • **Immutable URL**: cannot be altered by anyone • **Permanent**: the hosting organization has a mission to maintain artifacts for the foreseeable future • **Accessible**: There is a DOI pointing to the real approach URL
Open Source	The source code is available and has a proper license which grants access and re-use of data, material, and source code
Reachable	The tool is reachable now but is missing some aspect above to be considered Open Access.
Upon Request	Authors say the tool is available upon request.
Broken	A link is given in the paper, but does not resolve.
Unavailable	A tool is discussed in the paper, but no link is provided.
Private	The authors say that a tool exists, but it is private for some reasons (such as industry collaboration with private data, etc.).
Proprietary	The tool is available but proprietary

Examples:

- In [10], the author presents their tool TT-RecS located at https://zenodo.org/record/3827169. The tool is licensed under an open-source license (Apache 2.0), has an immutable DOI, and the hosting platform (zenodo) is dedicated to permanence. Consequently, the tool's availability counts as **archived**.
- In [4], the authors present their tool AQUSA (https://github.com/gglucass/AQUSA). The tool is **open source** because it contains an OS license (MIT), but hosting it via GitHub does not count as archived.
- In [6], the authors present the Desiree Framework and point toward https://goo.gl/oeJ9Fi to access the tool. This Google Drive folder requires to request access to the tool. Hence, the tool is available **upon request**.
- In [5], the authors present the QuARS tool and point toward http://fmt.isti.cnr.it/quars for the source, but this is a **broken link**.
- In [2], the authors present a tool but do not provide any link to it. As they do not mention any other reason for the unavailability of the tool, it is coded as **no link**.

Requirements Engineering Activity

Enum

Description: This element classifies in which RE phase the tool is intended to be used.

Purpose: This information allows identifying relevant RE tools for specific activities of the RE process.

Extraction rules: Select the RE phase that fits the tool best.

Phase	Description
Elicitation	This phase comprises activities that enable the understanding of the goals, objectives, and motives for building a proposed software system.
Analysis	This phase involves evaluating the quality of recorded requirements (i.e., identifying anomalies in requirements, such as ambiguity, inconsistency, and incompleteness) as well as general analysis including attribution.
Modeling	This phase involves building conceptual models (including architecture and code) of requirements that are amenable to interpretation.
Validation & Verification (V&V)	Requirements validation ensures that models and documentation accurately express the stakeholders' needs. Validation usually requires stakeholders to be directly involved in reviewing the requirements artifacts. Verification entails proving that the software specification meets these requirements. Such proofs often take the form of checking that a specification model satisfies some constraint (model checking).
Management	This is an umbrella activity that comprises a number of tasks related to the management of requirements, including the evolution of requirements over time and across product families and the task of identifying and documenting traceability links among requirements artifacts and between requirements and downstream artifacts.
Other	This is an open-end category that allows us to record other NLP4RE-related software development activities.
Multiple	The tool is not designed to aid a specific RE activity but rather multiple, e.g., as a preprocessor or precursor to different activities.

The categories were initially adopted from Zhao et al. [1] and further evolved based on the observed data.

Examples:

- In [11], the authors present a tool that classifies user comments as either problem reports, feature requests, or irrelevant. The latter two contain relevant information for new requirements. Hence, the tool supports the **elicitation** activity.
- In [4], the authors present AQUSA, which assesses the quality of NL requirements and, therefore, supports the **analysis** activity.
- In [12], the authors present T-Star, a tool that generates iStar models from NL requirements, which is a **modeling** activity.

- In [9], the authors present a tool that automatically generates cause-effect graphs from NL requirements. While this in itself is a modeling task, the purpose of these graphs is to automatically generate test cases. Hence, the activity that the tool supports is the **validation & verification** activity.
- In [10], the author presents TT-RecS, a tool to recover trace links, which supports the **management** activity.
- In [13], the authors present a classification tool based on zero-shot learning. The tool does not have a dedicated purpose in requirements engineering but is rather a precursor to several different NLP4RE tools. Consequently, it supports **multiple** activities.

Task Type

Enum

Description: This attribute classifies the type of NLP4RE task that the tool is supposed to solve. The types of tasks are common NLP tasks.

Purpose: The task type differentiates tools in their responsibility from each other.

Extraction rules: Select the NLP4RE task type that fits the tool best.

Task	Description
Detection	Detect linguistic issues in requirements documents
Extraction	Identify key domain abstractions and concepts
Classification	Classify requirements into different categories
Transformation	Derivation of a semantically similar specification using a different syntax
Tracing & Relating	Establish traceability links or relationships between requirements, or between requirements and other software artifacts
Search & Retrieval	Search and retrieve requirements-relevant information from existing corpora
Generation	New, requirements-relevant text is formed

The categories were initially adopted from Zhao et al. [1] and further evolved based on the observed data.

Examples:

- In [4], the authors present the AQUSA tool, which **detects** a specific set of linguistic issues in user stories.
- In [14], the authors present a tool to **extract** privacy requirements from user stories.
- In [11], the authors present a transfer-learning-based **classification** algorithm to filter requirements-relevant data from user comments.
- In [12], the authors present a tool that generates an iStar model from NL requirements. This is a **transformation** task, as the source semantics are preserved.
- In [10], the author presents a tool to recover trace links, which is a **tracing & relating** task.
- In [15], the authors present a tool that finds GUI prototypes from a large repository using NL queries. This is a **search & retrieval** activity.
- In [16], the authors present ScenarioAmigo, which recovers missed steps from use cases. This **generates** new requirements.

References

[1] Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., & Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: A systematic mapping study. *ACM Computing Surveys (CSUR)*, *54*(3), 1-41.

[2] Ferrari, A., Spagnolo, G. O., & Gnesi, S. (2014, April). Measuring and improving the completeness of natural language requirements. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 23-38). Springer, Cham.

[3] Yang, H., De Roeck, A., Gervasi, V., Willis, A., & Nuseibeh, B. (2011). Analysing anaphoric ambiguity in natural language requirements. Requirements engineering, 16(3), 163-189.

[4] Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2017, February). Improving user story practice with the Grimm Method: A multiple case study in the software industry. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 235-252). Springer, Cham.

[5] Lami, G., Gnesi, S., Fabbrini, F., Fusani, M., & Trentanni, G. (2004). An automatic tool for the analysis of natural language requirements. Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre.

[6] Li, F. L., Horkoff, J., Liu, L., Borgida, A., Guizzardi, G., & Mylopoulos, J. (2016, June). Engineering requirements with desiree: An empirical evaluation. In International Conference on Advanced Information Systems Engineering (pp. 221-238). Springer, Cham.

[7] Juergens, E., Deissenboeck, F., Feilkas, M., Hummel, B., Schaetz, B., Wagner, S., ... & Streit, J. (2010, May). Can clone detection support quality assessments of requirements specifications?. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2 (pp. 79-88).

[8] Ferrari, A., & Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. Automated Software Engineering, 26(3), 559-598.

[9] Fischbach, J., Junker, M., Vogelsang, A., & Freudenstein, D. (2019, September). Automated generation of test models from semi-structured requirements. In 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW) (pp. 263-269). IEEE.

[10] Unterkalmsteiner, M. (2020, September). TT-RecS: The Taxonomic Trace Recommender System. In 2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE) (pp. 18-21). IEEE.

[11] Henao, P. R., Fischbach, J., Spies, D., Frattini, J., & Vogelsang, A. (2021, September). Transfer learning for mining feature requests and bug reports from tweets and app store reviews. In 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW) (pp. 80-86). IEEE.

[12] Chen, Y., Wang, Y., Hou, Y., & Wang, Y. (2019, September). T-star: a text-based istar modeling tool. In 2019 IEEE 27th international requirements engineering conference (RE) (pp. 490-491). IEEE.

[13] Alhoshan, W., Ferrari, A., & Zhao, L. (2023). Zero-shot learning for requirements classification: An exploratory study. Information and Software Technology, 159, 107202.

[14] Casillo, F., Deufemia, V., & Gravino, C. (2022). Detecting privacy requirements from User Stories with NLP transfer learning models. Information and Software Technology, 146, 106853.

[15] Kolthoff, K., Bartelt, C., & Ponzetto, S. P. (2020, December). GUI2WiRe: rapid wireframing with a mined and large-scale GUI repository using natural language requirements. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (pp. 1297-1301).

[16] Ko, D., Kim, S., & Park, S. (2019). Automatic recommendation to omitted steps in use case specification. Requirements Engineering, 24, 431-458.