Exploration of Wikipedia's categories for software languages with WikiTax

Tool demonstration

Ralf Lämmel and Dominik Mosen and Andrei Varanovich

University of Koblenz-Landau, Software Languages Team

Abstract. Wikipedia provides useful input for efforts on mining taxonomies or ontologies in specific domains. In particular, Wikipedia's category construct expresses classification. In this paper, we describe a workflow and corresponding tool support for exploring Wikipedia's category graph with the objective of better understanding a possible classification of software languages. The WikiTax tool supports exploration of Wikipedia's category graph in an interactive manner such that it is rendered in a tree-like manner, irrelevant edges and nodes may be excluded, comments on the corresponding judgmental decisions may be added, and visualization based on graph-based metrics is provided.

1 Introduction

Ever since 2008, the calls for papers for the *Software Language Engineering* (SLE) conference[1] have contained slightly different, more implicit or more explicit definitions of the term 'software language'. Other community material contains yet other definition attempts; see, for example, the IEEE TSE special section on SLE in 2009 [8]. At SLEBOK 2012 (i.e., an SLE 2012 satellite event dedicated to the the SL(E) body of knowledge), the attendees were also getting into the issue of what exactly a software language is and what classification may help in arriving at an accepted comprehensive definition.

Some classes of software languages are generally agreed upon. For instance, programming languages are definitely software languages; they are conceptually well understood and characterized in terms of programming language concepts; see, for example, textbooks on programming languages, programming paradigms, and programming language theory, e.g., [15,18]. Some classes of languages have been the target of scholarly work on language classification; see, for example, classifications of model transformation languages [5], business rule modeling languages [19], visual languages [3,12,4], architecture description languages [13], and programming languages [1,6].

In our work on the software chrestomathy '101' [9][2], we also aim at the classification of software languages, but we have failed to make a serious proposal

[1] `http://planet-sl.org/`

[2] `http://101companies.org/`

so far. We are simply not confident regarding classification style, expected level of detail, and treatment of multiple dimensions of classification. In fact, such a SL(E) classification challenge is by no means limited to software languages; it also applies to *software technologies* and *software concepts*. Perhaps, we may need to lower expectations and accept the use of simpler tagging schemes (as used on StackOverflow, for example) as opposed to hierarchically organized, consistent and comprehensive taxonomies.

Wikipedia contains substantial amounts of taxonomy-like (if not ontology-like) information—also for software languages, technologies, and concepts. For instance, there are hierarchically organized categories such as *Computer languages*, *Programming languages*, and *Programming language classification*, but yet other roots for exploration may be reasonable. We suggest that the SL(E) classification challenge shall be informed by the exploration of Wikipedia.

In this paper, we describe support for such exploration based on the WikiTax tool that was developed exactly for this use case. We also demonstrate exploration, without though any claim of having found a good candidate taxonomy for SL(E). Rather the expectation is that WikiTax and the associated paradigm add more structure to the ongoing community effort of addressing the SL(E) classification challenge. The source code of WikiTax, a comprehensive manual, and all data covered in this paper are available online.[3]

Road-map §2 describes the WikiTax tool. §3 explores some Wikipedia categories related to software languages. §4 concludes the paper.

2 Exploring Wikipedia with WikiTax

Wikipedia's category graph Wikipedia uses several means of organizing its information: plain links giving rise to an article graph, designated article lists, portals meant to introduce users to key topics, info-boxes for semantic ('typed') data, and categories giving rise to a category graph for the classification of articles. When it comes to taxonomy mining, the category graph is particularly relevant; the graph is accessible, for example, through the MediaWiki API[4], which is the access path chosen by WikiTax.

Graph extraction and reduction with WikiTax Initially, WikiTax is pointed to a root category (level 0) for extraction. Iteratively, subcategories and pages (in fact, page titles) can be extracted level by level or exhaustively. Exhaustive extraction may take minutes to hours depending on the root category. The Wikipedia category graph contains many surprising edges, which would easily imply inclusion of large, arguably irrelevant subgraphs. Thus, extraction needs to be controlled.

WikiTax supports reduction of the graph—both during (level-by-level) extraction and post extraction. Reduction is based on the selection of edges for

[3] `https://github.com/dmosen/wiki-analysis`

[4] `http://www.mediawiki.org/wiki/API:Main_page`

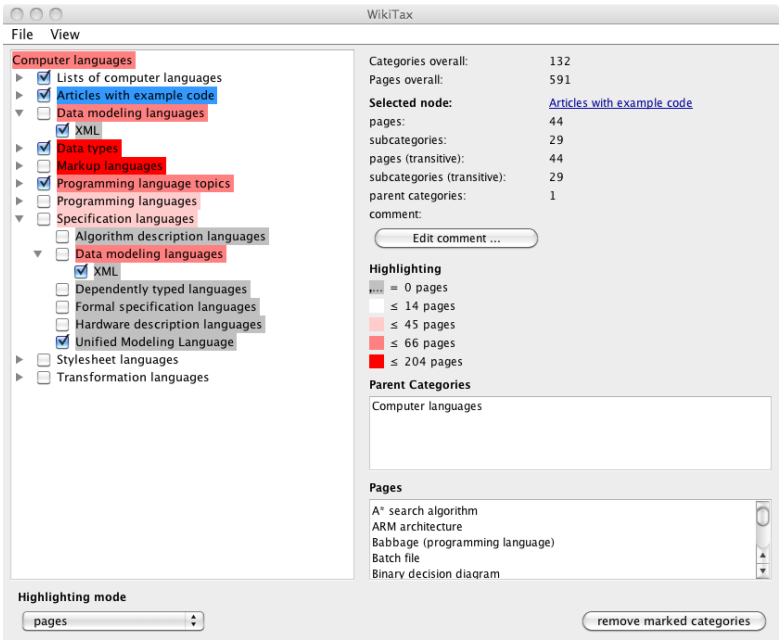

Fig. 1. Exploration of level 1 and 2 subcategories of *Computer languages*.

exclusion. If all edges to a given category are excluded, then the corresponding category node is also effectively excluded. (We note that a category may have multiple parent categories.) When reduction is performed during extraction, then the excluded edges (nodes) are not followed by subsequent extraction steps. When reduction is performed post extraction, then edges are only blacklisted so that all decisions can be easily revised later.

Figure 1 shows the WikiTax exploration view after the extraction of two levels (level 1 and 2) starting from the category *Computer languages*. Some edges are selected for exclusion. (Exclusion happens upon pushing the 'removal/blacklist' button.) In §3, we discuss reasons for exclusion systematically, but it suffices here to say that the selected categories are not proper language classifiers in a certain narrow sense. Highlighting is applied to the categories according to the metric of immediate member pages. We have selected the category *Articles with example code* for which some extra data is shown in the panel on the right. All categories and pages are clickable to navigate to Wikipedia.

WikiTax operates on an enhanced category graph; see the metamodel in Figure 2. Thus, each category associates with contained pages and subcategories. The subcategory associations are attributed to keep track of metadata as follows:

backwardArc Marker for cyclic edges in the category graph.
blacklisted Marker for categories blacklisted past extraction.
excluded Marker for categories excluded during reduction.
comment Label ('reason for exclusion') to be associated with the edge.

Categories are associated with measures as follows:

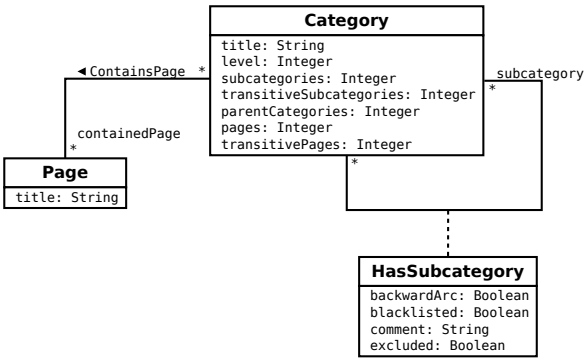

Fig. 2. Metamodel of the WikiTax category graph.

level The level 0, 1, 2, ... of the category in the graph with the root at level 0.
subcategories The number of immediate subcategories.
transitiveSubcategories The number of all subcategories.
pages The number of immediately contained pages.
transitivePages The number of all pages in this category.

Internally, WikiTax uses the Java-based JGraLab library[5] for the representation of (annotated) graphs with JSON as an export format.

3 Explorative study

In this study, we examine some Wikipedia categories with two objectives: a) to retrieve some candidate classifiers of an emerging taxonomy of software languages; b) to get some experience with Wikipedia's approach to classification and related issues of style and consistency.[6]

Designation of a root Wikipedia's classification hierarchies are complex and thus, it is not straightforward to determine a root for exploration unambiguously. However, we have established by an ad-hoc search that the category *Computer languages* may be a suitable root as its intended coverage may be similar to what the SL(E) community has in mind for the notion of software languages.

Category *Computer languages* only has a few immediate subcategories; see Figure 3. The figure shows the situation in the WikiTax dialog past selecting a few level 1 categories for exclusion. That is, *Lists of computer languages*, *Articles with example code*, *Data types*, and *Programming language topics* should be excluded because they are not directly concerned with the *classification* of languages.

We also observe that the category *Programming languages* is reachable in one step from *Computer languages* while there is another major classifier for programming languages, namely *Programming language classification*, which would be reachable through the excluded category *Programming language topics*.

[5] `https://github.com/jgralab`

[6] All Wikipedia access for this study was validated (again) during 7-18 June 2013 which is also when quotes were extracted from Wikipedia, as they appear in the text of this section.

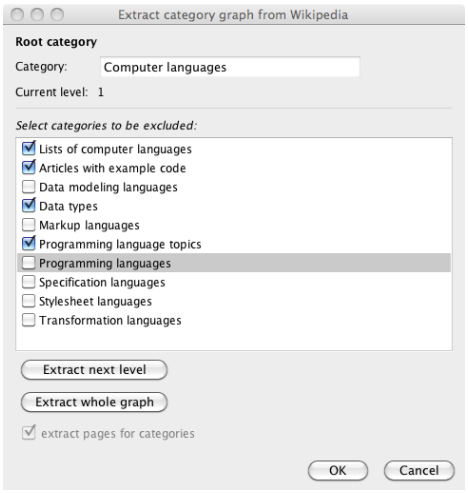

Fig. 3. Extraction and reduction of level 1 subcategories for *Computer languages*.

Level-by-level extraction We decided to extract another level to obtain a graph of manageable size. Again, we excluded several categories, if they did not meet our objective of language classification. As a result, we obtained the categories shown in Figure 4. This is a pretty manageable set of language classifiers. It happens that they all end on "... languages" except for two subcategories of *Markup languages* which end on "... formats". In contrast, most of the excluded categories (see below) do not end on "... languages" . We take this to provide a hint at the different classification styles of Wikipedia.

Classifier classification In order to obtain the reduced result of Figure 4, we had to exclude 29 categories. This may seem like a small number, but it is clear that we will need to exclude much more categories once we push extraction deeper into the category graph. Thus, we embarked on the classification of reasons for exclusion, thereby standardizing comments for exclusion, also suggesting a foundation for reproducing our results. We identified the following classifiers; see Figure 5 for the full list of excluded categories with the associated classifier:

Alternative classifier The category classifies software languages in a manner that is not related to software concepts. For instance, the category *Academic programming languages* describes itself as being concerned with languages that are "influential in computer science and programming language theory".

Deviating classifier The category does not actually classify software languages. It rather classifies something else. For instance, category *Articles with example code* describes itself as being concerned with "articles which include reference implementations of algorithms".

Singleton classifier The category is effectively concerned with a single software language for which it serves as a container of related entities such as technologies or standards. For instance, category *Cascading Style Sheets* contains pages on all kinds of topics related to the CSS language.

Category	Subcategories
Data modeling languages	–
Markup languages	*Declarative markup languages*, *GIS file formats*, *Knowledge representation languages*, *Lightweight markup languages*, *Mathematical markup languages*, *Musical markup languages*, *Page description markup languages*, *Playlist markup languages*, *User interface markup languages*, *Vector graphics markup languages*, *Web syndication formats*, *XML markup languages*
Programming languages	*.NET programming languages*, *Agent-based programming languages*, *Agent-oriented programming languages*, *Concatenative programming languages*, *Concurrent programming languages*, *Data-structured programming languages*, *Declarative programming languages*, *Dependently typed languages*, *Domain-specific programming languages*, *Dynamic programming languages*, *Extensible syntax programming languages*, *Formula manipulation languages*, *Function-level languages*, *Functional languages*, *High Integrity Programming Language*, *High-level programming languages*, *ICL programming languages*, *Intensional programming languages*, *Low-level programming languages*, *Multi-paradigm programming languages*, *Nondeterministic programming languages*, *Object-based programming languages*, *Pattern matching programming languages*, *Procedural programming languages*, *Process termination functions*, *Prototype-based programming languages*, *Reactive programming languages*, *Secure programming languages*, *Set theoretic programming languages*, *Statically typed programming languages*, *Synchronous programming languages*, *Term-rewriting programming languages*, *Text-oriented programming languages*, *Tree programming languages*, *Visual programming languages*, *XML-based programming languages*
Specification languages	*Algorithm description languages*, *Dependently typed languages*, *Formal specification languages*, *Hardware description languages*
Stylesheet languages	–
Transformation languages	*Macro programming languages*

Fig. 4. Reduced subcategory lists for subcategories of *Computer languages*.

List classifier The category collects lists or categories of lists (rather than plain categories) of software languages. For instance, category *Lists of computer languages* has *Lists of programming languages* as a subcategory, which in turn contains pages for some lists of languages, such as the *List of BASIC dialects*.

Maintenance classifier The category is used by the Wikipedia authors to capture some information related to the maintenance of content. For instance, category *Markup language stubs* describes itself as serving "for stub articles relating to markup languages".

An observation regarding Wikipedia style At this point, exploration already had led to a manageable view on the category graph for software languages. This view is, in fact, quite effective, which we illustrate with an inquiry that suggested itself during exploration.

Looking at Figure 3 and Figure 4, we may suspect an asymmetry between 'query' versus 'transformation'. That is, there is a category *Transformation languages* at level 1, but there is apparently no category for 'query languages', not even at level 2. Let us inspect the page for *SQL*, which is arguably a quite ob-

Category	Meta classifier
Academic programming languages	Alternative classifier
Articles with example code	Deviating classifier
Cascading Style Sheets	Singleton classifier
Data types	Deviating classifier
Discontinued programming languages	Alternative classifier
DocBook	Singleton classifier
Esoteric programming languages	Alternative classifier
Experimental programming languages	Alternative classifier
HTML	Singleton classifier
JSON	Singleton classifier
Lists of computer languages	List classifier
Lists of programming languages	List classifier
Markup language comparisons	Deviating classifier
Markup language stubs	Maintenance classifier
Non-English-based programming languages	Alternative classifier
Programming language families	Deviating classifier
Programming language standards	Deviating classifier
Programming language topics	Deviating classifier
Programming languages by creation date	Alternative classifier
Programming languages conferences	Deviating classifier
Software by programming language	Deviating classifier
SyncML	Singleton classifier
TeX	Singleton classifier
Text Encoding Initiative	Singleton classifier
Troff	Singleton classifier
Uncategorized programming languages	Maintenance classifier
Unified Modeling Language	Singleton classifier
Wikipedia categories named after programming languages	Deviating classifier
XML	Singleton classifier

Fig. 5. Exclusion summary for levels 1 and 2 of *Computer languages*; this list is produced by the WikiTax tool based on metadata (comments) entered by us interactively.

vious query language. It turns out that *SQL* is a member of various categories including a category *Query languages* which in turn is subcategory of various categories including the category *Domain-specific programming languages* which occurred in Figure 4. Let us compare this classification scheme with the one of *XSLT*, which is arguably a quite obvious transformation language: it is a member of the categories *Transformation languages*, *Declarative programming languages*, *Functional languages*, *Markup languages*, *XML-based programming languages*, and yet other categories that may count as 'alternative classifiers'. However, *XSLT* (unlike *SQL*) is not a member of the category *Domain-specific programming languages*.

We take this sort of observation to mean that the derivation of a highly consistent taxonomy for software languages would require some non-trivial effort in defining and enforcing principles. We were not able to observe this situation so clearly prior to using WikiTax.

Programming languages: all levels Figure 4 makes it obvious that the subcategory of *Computer languages* with by far the most subcategories is *Pro-*

Fig. 6. Metrics-based views on *Programming languages* graph.

gramming languages. Thus, we embarked on a more comprehensive exploration of category *Programming languages*:

Initially, we extracted 423 categories over 8 levels with 7515 pages. The automatic extraction took several minutes. We performed exclusion in two steps. First, we excluded direct subcategories of the category *Programming languages*—based on the list in Figure 5. After such initial pruning, 288 categories with 6671 pages remained. We completed reduction at all levels of the category graph. This process required about 2 hours of manual work—work that is mainly concerned with checking assumptions for exclusion by consulting corresponding category pages on Wikipedia. Ultimately, 79 categories over 4 levels with 1560 pages remained. Figure 6 visualizes the reduced taxonomy while applying two different metrics, as supported by WikiTax.

On the left, the metric for the *number of transitive member pages* is applied for visualization. No category is grayed out, which means that there is no category without members. Most of the categories are shown in a plain font, which means that they all carry members, but less than 25 % of the total members

in the category *Programming languages* (which has 1560 member pages). There is actually one heavyweight: category *Domain-specific programming languages* carries 976 members, which is more than 50 % of all members; this status is expressed by highlighting the category.

On the right, the metric for the number of transitive subcategories is applied for visualization. Most subcategories of *Programming languages* do not have any subcategories; thus, they are grayed out. 7 out of 36 level-1 categories carry subcategories. 6 out of these 7 categories carry only very few subcategories (less than 5). Category *Domain-specific programming languages* carries 18 subcategories, which is more than 25 % of all subcategories; this status is expressed by highlighting the category.

4 Conclusion

WikiTax supports the exploration of Wikipedia's category graph and its reduction to candidate taxonomies. Thus, WikiTax is a highly domain-specific exploration tool. In principle, such exploration could also be performed by means of search engines on Wikipedia (e.g., [14]) or plainly programmatically (by writing API-based queries against Wikipedia or DBpedia[7] or possibly Wikidata[8]), but this path, which we experimented with before designing WikiTax, would not enable convenient exploration and transparent judgements.

Any domain with large data to explore ('large' in terms of what the user needs to understand) may benefit from interactive exploration possibly with editing or annotation, see, e.g., tools for ontologies [2], graphs [10], semantic data [7], software bugs [11], API usage [17].

The key features of WikiTax are scalability in terms of access to Wikipedia's category graph, navigation thereupon, metrics-based visualization, link support to Wikipedia, and commented edge exclusion. The proposed paradigm of graph reduction is deliberately interactive and relies on (transparent) judgements by the user, as opposed to any means of automated ontology extraction / generation [21,20]. An important conceptual contribution is our proposal for classifying classifiers, thereby supporting the systematic (transparent) reduction of the category graph. This is again a more judgmental than automatic approach, when compared to related work on taxonomy or ontology mining, where categories are also classified and additional relationships are inferred, e.g., by analyzing the structure of compound category names [16].

To summarize, we have initiated a path towards derivation of an SL(E) taxonomy, thoroughly informed by Wikipedia. Collaborative work and presumably further improved tool support are needed to actually arrive at a comprehensive taxonomy. We imagine that we need powerful refactoring operations on the category graph to facilitate taxonomy extraction and enforcement of consistent style. Also, we need to generally better understand (perhaps based on analysis) the different classifier styles used by Wikipedia.

[7] `http://dbpedia.org`

[8] `https://www.wikidata.org/`

References

1. Babenko, L.P., Rogach, V.D., Yushchenko, E.L.: Comparison and classification of programming languages. Cybernetics 11, 271–278 (1975)
2. Baskaya, F., Kekäläinen, J., Järvelin, K.: A tool for ontology-editing and ontology-based information exploration. In: Proc. of ESAIR 2010. pp. 29–30. ACM (2010)
3. Bottoni, P., Grau, A.: A suite of metamodels as a basis for a classification of visual languages. In: Proc. of VL/HCC 2004. pp. 83–90. IEEE Computer Society (2004)
4. Burnett, M.M., Baker, M.J.: A classification system for visual programming languages. J. Vis. Lang. Comput. 5(3), 287–300 (1994)
5. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal 45(3), 621–646 (2006)
6. Doyle, J.R., Stretch, D.D.: The classification of programming languages by usage. International Journal of Man-Machine Studies 26(3), 343–360 (1987)
7. Dumas, B., Broché, T., Hoste, L., Signer, B.: ViDaX: an interactive semantic data visualisation and exploration tool. In: Proc. of AVI 2012. pp. 757–760. ACM (2012)
8. Favre, J.M., Gasevic, D., Lämmel, R., Winter, A.: Guest editors' introduction to the special section on software language engineering. IEEE Trans. Software Eng. 35(6), 737–741 (2009)
9. Favre, J.M., Lämmel, R., Varanovich, A.: Modeling the Linguistic Architecture of Software Products. In: Proc. of MODELS 2012. LNCS, vol. 7590, pp. 151–167. Springer (2012)
10. Haun, S., Nürnberger, A., Kötter, T., Thiel, K., Berthold, M.R.: CET: A tool for creative exploration of graphs. In: Proc. of ECML/PKDD (3) 2010. LNCS, vol. 6323, pp. 587–590. Springer (2010)
11. Hora, A., Anquetil, N., Ducasse, S., Bhatti, M.U., Couto, C., Valente, M.T., Martins, J.: Bug Maps: A tool for the visual exploration and analysis of bugs. In: Proc. of CSMR 2012. pp. 523–526. IEEE (2012)
12. Marriott, K., Meyer, B.: On the classification of visual languages by grammar hierarchies. J. Vis. Lang. Comput. 8(4), 375–402 (1997)
13. Medvidovic, N., Taylor, R.N.: A classification and comparison framework for software architecture description languages. IEEE Trans. Software Eng. 26(1), 70–93 (2000)
14. Milne, D.N., Witten, I.H.: Exploring Wikipedia with HMpara. In: Proc. of JCDL 2011. pp. 453–454. ACM (2011)
15. Mosses, P.D.: Action Semantics. Cambridge University Press (1992)
16. Nastase, V., Strube, M.: Decoding Wikipedia categories for knowledge acquisition. In: Proc. of AAAI 2008. pp. 1219–1224. AAAI Press (2008)
17. Roover, C.D., Lämmel, R., Pek, E.: Multi-dimensional exploration of API usage. In: Proc. of ICPC 2013. IEEE (2013), to appear. 10 pages.
18. Sebesta, R.W.: Concepts of Programming Languages. Addison-Wesley (2012), 10th edition
19. Skalna, I., Gawel, B.: Model driven architecture and classification of business rules modelling languages. In: Proc. of FedCSIS 2012. pp. 949–952 (2012)
20. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A large ontology from Wikipedia and WordNet. J. Web Sem. 6(3), 203–217 (2008)
21. Wu, F., Weld, D.S.: Automatically refining the Wikipedia infobox ontology. In: Proc. of WWW 2008. pp. 635–644. ACM (2008)