Discovering Indicators for Classifying Wikipedia Articles in a Domain
A Case Study on Software Languages

Marcel Heinz[1] Ralf Lämmel[1] Mathieu Acher[2]
[1]SoftLang Team, CS Faculty, University of Koblenz-Landau, Germany
[2]Univ Rennes, Inria, CNRS, IRISA, France

Abstract

Wikipedia is a rich source of information across many knowledge domains. Yet, recovering articles relevant to a specific domain is a difficult problem since such articles may be rare and tend to cover multiple topics. Furthermore, Wikipedia's categories provide an ambiguous classification of articles as they relate to all topics and thus are of limited use. In this paper, we develop a new methodology to isolate Wikipedia's articles that describe a specific topic within the scope of relevant categories; the methodology uses supervised machine learning to retrieve a decision tree classifier based on articles' features (URL patterns, summary text, infoboxes, links from list articles). In a case study, we retrieve 3000+ articles that describe software (computer) languages. Available fragments of ground truths serve as an essential part of the training set to detect relevant articles. The results of the classification are thoroughly evaluated through a survey, in which 31 domain experts participated.

1 INTRODUCTION

Wikipedia is a very large-scale, continuous community effort to collect and organize (informal) knowledge in almost all domains. In general, the quality of the articles and the mere principles of collection and organization challenge automated procedures in the quest of extracting and structuring Wikipedia knowledge [1–4]. Recovering articles describing topics in a certain domain is a reoccurring problem [5–7]. This paper presents a supervised machine learning approach for recovering Wikipedia articles relevant to an ontological class.

We conduct a case study on *software languages* defined as a set of digital artifacts, for which syntax, type system, semantics, and pragmatics can be (in)formally defined, documented, and implemented [8]. For the purpose of this research, we assume that Wikipedia's notion of "computer language" – a notion that is used all across computer science and beyond – is essentially equivalent to the notion of "software language"; see also [9] for a discussion.

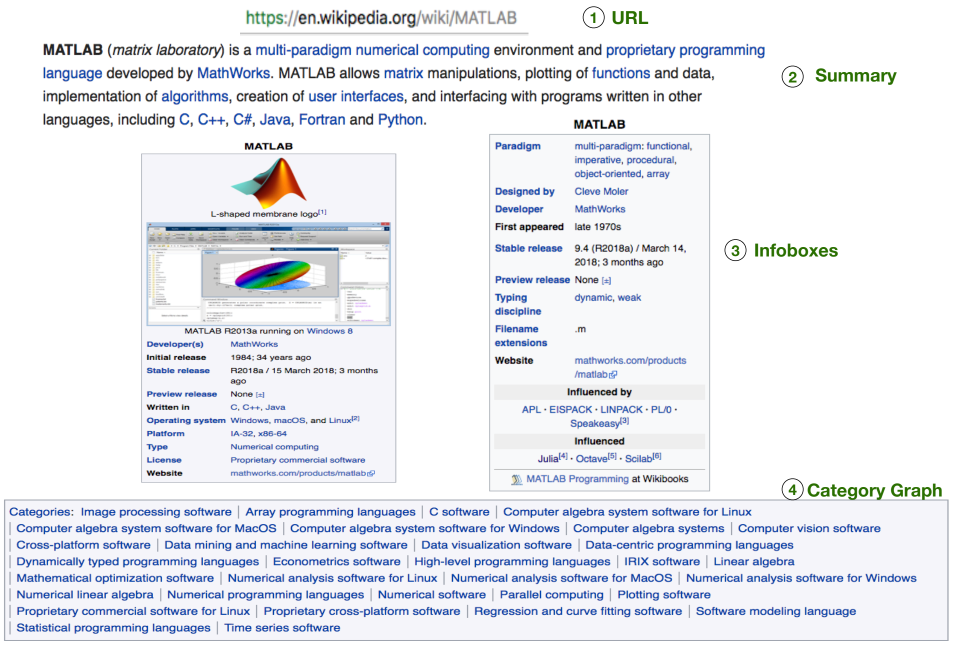
An important barrier is that, in a large number of cases, *a single Wikipedia article covers multiple subjects*. Looking at the very first sentences of the article about MATLAB (see Figure 1), several topics are mixed together (e.g., user interfaces, numerical computing, and software languages). The category graph is impacted as well (e.g., linear algebra and array-programming languages are both mentioned). There are also two infoboxes—one about MATLAB the *language*, the other about MATLAB the *software*.

Moreover, Wikipedia's category graph cannot be directly used for classification within the domain [10]; categories serve purposes other than classification, for example, collecting articles related to a common topic; see `https://en.wikipedia.org/wiki/Category:Java_(programming_language)`. Wikipedia's guidelines partly explain why multiple topics appear in an article: there should not be a new article, when the description benefits from explaining two or more subjects in the context of an existing article[1]. Such rules and editing practices, though reasonable, complicate *classification*, i.e., automated identification of articles (instances) and underlying categories (classifiers). For some domains, like animals [5], categories that represent scientific classification are consistently used and provide a decision ground for still identifying relevant articles. In the domain of software languages such crucial features do not exist for articles or are used inconsistently. Alas, the recovery of relevant articles becomes looking for needles in a hay stack.

Significance of the multiple-subject problem. We initially explored articles that link to (subcategories of) the category 'Computer languages' and search for the top ten frequent nouns ('NN' tag recovered with part-of-speech tagger). We noticed that nouns from the music and astronomy domain are dominant. Fig. 2 reports the number of articles for the top ten most frequent nouns in the category

DOI reference number: 10.18293/SEKE2019-126

[1]`https://en.wikipedia.org/wiki/Wikipedia:Notability#Whether_to_create_standalone_pages`


https://en.wikipedia.org/wiki/MATLAB

MATLAB (*matrix laboratory*) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

MATLAB

L-shaped membrane logo[1]

MATLAB R2013a running on Windows 8

Developer(s)	MathWorks
Initial release	1984; 34 years ago
Stable release	R2018a / 15 March 2018; 3 months ago
Preview release	None [±]
Written in	C, C++, Java
Operating system	Windows, macOS, and Linux[2]
Platform	IA-32, x86-64
Type	Numerical computing
License	Proprietary commercial software
Website	mathworks.com/products/matlab

MATLAB

Paradigm	multi-paradigm: functional, imperative, procedural, object-oriented, array
Designed by	Cleve Moler
Developer	MathWorks
First appeared	late 1970s
Stable release	9.4 (R2018a) / March 14, 2018; 3 months ago
Preview release	None [±]
Typing discipline	dynamic, weak
Filename extensions	.m
Website	mathworks.com/products/matlab

Influenced by

APL · EISPACK · LINPACK · PL/0 · Speakeasy[3]

Influenced

Julia[4] · Octave[5] · Scilab[6]

MATLAB Programming at Wikibooks

Categories: Image processing software | Array programming languages | C software | Computer algebra system software for Linux | Computer algebra system software for MacOS | Computer algebra system software for Windows | Computer algebra systems | Computer vision software | Cross-platform software | Data mining and machine learning software | Data visualization software | Data-centric programming languages | Dynamically typed programming languages | Econometrics software | High-level programming languages | IRIX software | Linear algebra | Mathematical optimization software | Numerical analysis software for Linux | Numerical analysis software for MacOS | Numerical analysis software for Windows | Numerical linear algebra | Numerical programming languages | Numerical software | Parallel computing | Plotting software | Proprietary commercial software for Linux | Proprietary cross-platform software | Regression and curve fitting software | Software modeling language | Statistical programming languages | Time series software

Figure 1. General structure of a Wikipedia article and some indicators.

tree of 'Computer languages' tree that we cut off at a depth of seven. Within the category tree, we observed subjects that belong to other domains, such as 'songs', 'stars', 'album' and 'music'. Except for the minority that describes domain specific languages in the respective domains, most of the article are obviously irrelevant to software languages. Fig. 2 also provides evidence that relatedness-based subcategorization connect to irrelevant subjects.

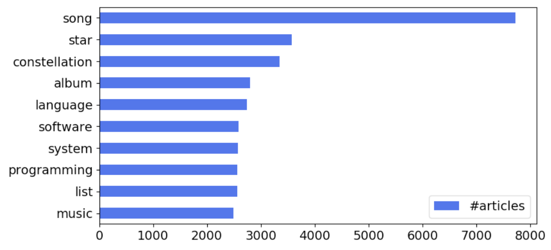


Figure 2. Top 10 most frequent nouns in articles below 'Formal languages' category.

Research question and contributions. Overall, our case study is an instance of the problem of extracting domain-specific knowledge from Wikipedia [7, 10, 11]: *How can we classify Wikipedia articles by their relevance to a given domain when relevant articles are rare and multiple main topics are covered by articles?* Our objective is essentially to detect members (here: Wikipedia articles) of a certain class (here: software languages).

We develop a seed-based learning methodology for identifying articles relevant to a domain-specific class while leveraging the available limited ground truth (based on Github and TIOBE) and identifying indicators by inspecting a learnt decision tree that include URL patterns, summary text, infoboxes, list articles and category graph. We then present a case study which, in itself, results in the most comprehensive corpus on software languages available today. The results are evaluated by domain experts through a survey.

2 SEED-BASED LEARNING

We now detail how we exploit fragments of ground truths and instrument a decision tree classifier. The datasets including plotted decision trees are available online[2]. Figure 3 provides an overview of our approach. i.) For training, we recover articles describing elements that appear in trustworthy external resources. ii.) Based on such seed, we define the scope in which we want to isolate relevant articles and label 4000 randomly sampled articles from this scope for additional training data. iii.) From the training data, we build a feature matrix with categorical values that state whether a structural feature, such as the programming language infobox template, is present. iv.) We configure a binary classifier that decides whether an article is relevant.

2.1 Seed Matching

As first step, we have identified and reused two trustworthy external data sets: both act as *seeds* for recognizing relevant articles. *GitHub* presents statistics on which languages

[2] https://github.com/softlang/wikionto/

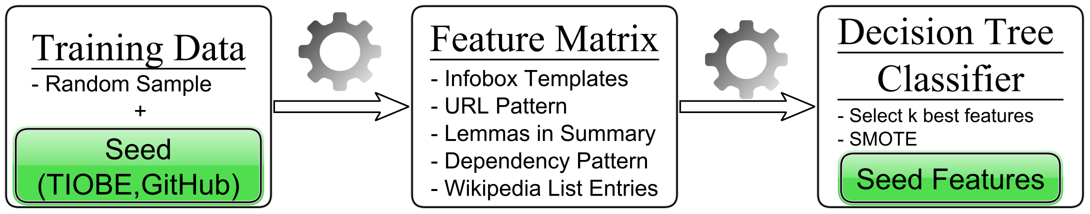


Figure 3. The ingredients of the approach. The seed in the training data influences decisions.

are used for any repository. The complete list of languages that are recognized can be extracted[3]. The *TIOBE* index presents statistics on how often software languages (mainly programming languages) are mentioned on the web. The list includes variations of names[4].

Identifying each language on Wikipedia from the lists is challenging. We encountered favorable and problematic cases. To our favor, most names can be matched with article titles by leaving out annotations (e.g. '(programming language)') or by using Wikipedia redirects. However, manual reviewing remains crucial to make sure that languages are matched correctly, e.g., 'Red' is matched with 'Red (programming language)' and not with the color. In some cases, we succeeded by varying the writing style for the name, e.g., by unfolding acronyms, such as EBNF. In less favorable cases, we had to rely on Wikipedia's search engine to detect mentions in the text of existing articles. Only if the existing article describes the seed language as its main topic in the summary, we take it as the recalled article. To ensure availability of the necessary features for classification, we exclude stub articles from further analysis.

The resulting merged seed contains 327 *recalled seed languages where* 110 *overlap*. 158 seed languages are only casually mentioned in articles and 99 seed languages remain unrecognized.

2.2 Category Scope Exploration

We narrow the scope of our analysis to articles that are reachable by links to (subcategories of) chosen upper categories. Technically, we have manually identified common upper categories so that the seed is linked to them. At first, we hypothesized that all relevant languages link to 'Computer languages', but we found 'Augmented BackusNaur form' as a seed member that links to 'Formal languages', where 'Computer languages' is a subcategory. Moreover, for seed articles such as 'CSV', we noticed the upper category 'Computer file formats' that is disconnected from 'Formal languages'. Such formats are categorized in a different manner, but they do conform to the definition of software language. Our experiments show that all seed articles are linked to (subcategories of) two disconnected roots in the category tree at a maximum depth of eight. Table 1 illustrates at what depth from the chosen root categories seed articles exist.

Table 1. Number of seed articles per depth.

	0	1	2	3	4	5	6-8
Formal languages	2	6	85	136	70	16	7
Computer file formats	8	22	11	7	0	0	3

2.3 Feature Extraction

Next, we describe which features we extract from the content types discussed in Section 1. To counter lack of decisive features and to reduce dimensionality, we exclude stub articles and only consider features that appear in at least ten articles. For now, we also exclude articles at a distance to the upper categories that is higher than 8. As a resulting dataset, the extraction returns an article-feature matrix with $104,186 \times 46,173$ entries with label '1' for present and '0' otherwise. Then, the set of 4000 randomly sampled articles and the seed form the training set. To avoid memory issues and enable loading the whole article-feature matrix, we use sparse matrices.

In the evaluation, we discuss that irrelevant articles in the training set are actually members of many other domain classes and have many different features. Hence, the resulting decision tree decides by features present in seed articles. In fact, GitHub and TIOBE seeds have a more general interest: **Seed articles provide representative features for recognizing relevant articles.** We show which features of seed articles provide a representative positive indication and hence may be found in a fit decision tree.

Infobox Templates. In the scope, there exist 864 distinct infobox template names. 263 seed articles use an infobox template. We found the templates on 'programming language' (215), 'file format' (32), 'technology standard' (3) and 'software license' (1). Especially, the templates 'programming language' and 'file format' provide a strong positive indication, but they do not exist for every relevant article.

URL Pattern. We extract all words separately in braces from every article's title as they provide a semantic anno-

[3]https://github.com/github/linguist/blob/master/lib/linguist/languages.yml
[4]https://www.tiobe.com/tiobe-index/programming-languages-definition/#instances

tation for disambiguation. Only around a third of the seed articles has such an annotation. The words in braces appearing more than once with their frequency are: 'language' (125), 'programming' (114), 'software' (6), 'stylesheet' (3) and 'markup' (2). 'programming' always appears together with 'language'.

Lemmas in Summary. From the summary, we extract all words, filter them by a stop word list and apply lemmatization on the remainder. We recovered 5449 lemmas from seed articles while $457,164$ distinct lemmas can be recovered from all articles. We enumerate the top five lemmas from the seed: 'language' (301), 'programming' (253), 'use' (216), 'develop' (120), 'design' (117). Lemmas are often used for topic models [12]. They are essential to distinguish the domain of software languages and co-occurring topics in articles from other unrelated domains such as natural languages.

Dependency Pattern. Following [10], we identified multiple kinds of relationships that can be extracted from the first sentence. Hypernyms provide a reliable feature as most articles on Wikipedia begin with a sentence containing 'is a'. Based on the dependency graph, we extract the hypernym [13], if the sentence does not start with 'A'. This allows us to ignore subset relationships as in 'A programming language is a formal language[...]'. To reach higher coverage, we infer instantiation from relationships such as part-of as in 'Perl 6 (also known as Raku[5]) is a member of the Perl family of programming languages', where 'languages' is the extracted hypernym. In the seed, we found 'language' (235), 'format' (16) and 'dialect' (10) as the most frequent hypernyms. The hypernym 'language' and 'dialect' are also recovered when analyzing articles on natural languages.

Wikipedia List Entries. As an internal resource, we consider Wikipedia lists of things [14]. Such lists collect names of entities of a certain kind and optionally provide additional information, e.g., 'List of dog breeds'. First, we recover the list of lists by searching for 'list of' in the title of articles in the scope. Binary Features state in which lists an article is linked. In the seed, we found 267 articles linked in such lists, such as the 'List of programming languages'.

3 EVALUATION

We investigate the results from evaluating a classifier based on labels by experts (RQ1). Since we chose to use a decision tree classifier, we can present technical insights on how decisions are made and explain the effects of using a representative seed (RQ2). We conclude with presenting how many software language articles and categories are estimated in the scope (RQ3).

3.1 Precision & Recall (RQ1)

We provide qualitative insights on **How well does the classifier perform beyond the seed?**

To gain an evaluation set that is as objective as possible, we conducted a survey in which 31 domain experts from our research groups and external collaborators participated. In each question in the survey, participants decided whether a presented article explicitly describes a software language as one primary topic. We received 990 articles labelled by at least two experts. In order to gain additional insights on problems with decision making, we emphasized the possibility of commenting on each question. For 43 articles, experts did not agree. The articles present border cases, for which experts are not sure. For example, articles do not directly refer to logic formulae as a software language, but such formulae are often digitally encoded, follow a syntax, well-formedness rules (thus, a type system) and have semantics and pragmatics. For the future of SLEBOK [8], such border cases can be used to further investigate on the borders of the term software language. A refinement of definitions would then again lead to better expert labels to, for now, problematic articles. Such refinement hopefully leads to a better categorization on Wikipedia itself.

As long as experts cannot reliably decide for an article whether a software language is described, a machine cannot as well. As a consequence, we exclude these articles. Since only ~4% is excluded, the threat to validity is reasonably low. We took the remaining expert labels as the evaluation set and explored several configurations with a k-best feature selection and synthetic oversampling with SMOTE [15].

With $k = 23$, the learned classifer performs with an *f1-score* of 0.7, *balanced accuracy* of 0.9, *recall* of 0.81 and *specificity* of 0.99.

3.2 Indicator Discovery (RQ2)

For imbalanced datasets, a classifier usually overfits towards the major class unless countermeasures are taken [15]. This problem is countered in two ways. i.) By adding the seed as an addition to the random sample to our training set, the rate of relevant articles in it is not representative which can be seen as a form of manual oversampling. ii.) Actually, we isolate articles relevant to a single class from articles relevant to $\underline{\text{many}}$ other classes (e.g., songs, stars, software). Feature $\overline{\text{selection}}$ based on *Gini index* or *entropy* pick features according to their score in discriminating classes, in our case relevant versus not relevant. Consequently, the features frequently appearing in seed articles are preferred, since the irrelevant articles have a huge amount of different features and thus single features recall a lesser percentage. At last, synthetic oversampling with SMOTE further improves results. This peculiarity mo-

Table 2. How many articles and categories exist in *total*, identified by the *seed*, and classified as *relevant*.

	Articles			**Categories**		
	Total	*Seed*	*Relevant*	*Total*	*Seed*	*Relevant*
Formal Languages	101641	301	2897	21822	353	1339
Computer File Formats	6116	46	745	235	18	79

tivates further inspection of the learned decision tree as an answer to **what features indicate articles on software languages**.

The templates 'programming language' and 'file format' provide a strong positive indication, but they do not exist for every relevant article. While the hypernym 'language' also provides strong indication, it cannot be used alone. Otherwise, natural language related articles are confused to be relevant. Lemmas, lists and URL pattern help in discriminating relevant articles from articles with overlapping features. Below, we list the most important indicators in the decision tree categorized by their source (see Section 2.3).

Infobox Templates: file format, programming language; **URL Pattern:** programming, language; **Lemmas:** syntax, code, programming, compiler, design, general-purpose, language, support, compile, object-oriented, use; **Dependency Pattern:** language; **Wikipedia List Entries:** List of programming languages, List of programming languages by type, List of file formats, List of C-family programming languages, List of object-oriented programming languages.

3.3 Article and Category Relevance (RQ3)

We give quantitative insights as an answer to the question: **How many articles and categories remain relevant for the domain of software languages?** Table 2 summarizes the degree of the reduction per root category based on the configuration from Section 3.1. **Based on the predicted reduction, we find 2797 more articles in the scope than there are already in the seed.** That is, we significantly augment the identification of software languages.

For the root category 'Computer file formats', the classifier predicts a lot of irrelevant articles. When inspecting infobox templates used in these articles, we observe a topic mix with, for example, software, websites and companies. Here, a threat remains that the articles do not directly address formats in several articles in a recognizable way. In comparison to 'Formal languages', a higher percentage of categories is estimated as relevant. Accordingly, we observe that the subcategory tree is more consistently maintained.

Inspecting single categories backs up subjective hypotheses based on manual inspection of the usefulness of specific categories. For example, out of 22546 articles that are (transitive) members 'Statistical data types', only 52 are classified as relevant. The number of articles indicated to be relevant in a category becomes a more objective estimation for its usefulness.

Table 2 provides a summary when inspecting all categories based on the assumption that the number of relevant articles hints at the relevance of categories. Out of 21822 categories below 'Formal languages', 353 categories contain seed members and 1339 categories remain relevant. **Only 6% of transitive subcategories under 'Formal languages' are estimated to be useful within the scope**.

4 RELATED WORK

Previous attempt: In [16], the category graph is pruned by manually excluding categories not serving for classification of software languages in a common sense, subject to a small suite of criteria. In contrast to our work, the approach relies on manual selection of categories instead of automatically classifying articles by combining several indicators.

Domain ontology: Wikipedia is a frequent target for knowledge discovery. *To the best of our knowledge, no approach tries to detect articles describing a class from an inconsistently maintained domain, where relevant articles are rare and require to combine multiple indicators.* For deriving domain ontologies, various distinct approaches can be found. The approaches range from supporting manual crafting [17, 18] to unsupervised crafting from text [12]. Wang et al [5] extract a domain ontology for animals from Wikipedia based on the category graph, article graph and section structure in articles. For the animals domain, most pages are maintained well and describe exactly one concept in a consistent order. Mirylenka et al [10] extract subset, membership, part-of, sub-topic and other relationships by analyzing the varying nature of subcategorization. Dong et al [6] learn subsumptions from articles describing domain concepts based on Hearst pattern. In a related approach [7], the concept set is learned by matching articles with Stackoverflow tags. Based on the aricle- and category graph, the concept set is expanded and further relationships are learned. Related works discover knowledge from different structural features, such as the title [10,19], text [13,20], an article's section structure [5], links to other articles [7,20], infoboxes [3], lists [14], etc.

External knowledge: The usefulness of WordNet varies depending on the coverage of terms in a domain. While WordNet is a known assistant in more common knowledge domains, low coverage is reported in more specific domains [2,5]. Our experience confirms it for software languages. Various knowledge graphs exist. *DBpedia Live* [21] mirrors Wikipedia while refining it with more ontological knowledge. YAGO [4] is a knowledge graph derived from Wikipedia, WordNet and Geonames that explicitly focuses only on structured knowledge aspect that is then again linked by Dbpedia as well. Wikidata [18] is widely manually crafted and tries to reach a clean knowledge graph from scratch. A type encompassing the term software language is not maintained by any of the mentioned resources.

5 CONCLUSION

In a domain where only a small set of positively labeled articles can be used as a ground truth, we did find strong indicators for classifying 3000+ articles as relevant for software languages. We showed that a learned decision tree classifier provides reasonably high recall and low false-positive-rate and allows one to inspect isolated articles inside Wikipedia. While learned random forests might provide higher accuracy, we are specifically interested in higher interpretability of decision trees. 31 domain experts thoroughly evaluated our classifier by labelling over 990 Wikipedia articles.

A natural follow up of this work is a collaborative engagement with experts to further extract and improve fine-grained classification knowledge into a high-quality domain ontology that will help students and professional developers to better understand software languages. We are in the process of repeating the experiments for the domain of software (in a broad sense). The general problem is the same: We are confident that our learning methodology can identify the body of domain knowledge within Wikipedia. There are also specific challenges ahead due to particularities of the domains (for example, 'Computing platforms' as a category is disconnected from 'Software').

Acknowledgements. This research was partially funded by the ANR-17-CE25-0010-01 VaryVary project.

References

[1] B. Stvilia, M. B. Twidale, L. C. Smith, and L. Gasser, "Information quality work organization in wikipedia," *JASIST*, vol. 59, no. 6, pp. 983–1001, 2008.

[2] Q. X. Do and D. Roth, "Exploiting the wikipedia structure in local and global classification of taxonomic relations," *Natural Language Engineering*, vol. 18, no. 2, pp. 235–262, 2012.

[3] F. Wu and D. S. Weld, "Automatically refining the wikipedia infobox ontology," in *Proc. WWW*, 2008, pp. 635–644.

[4] T. Rebele, F. M. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames," in *Proc. ISWC 2016*, 2016, pp. 177–185.

[5] H. Wang, X. Jiang, L. Chia, and A. Tan, "Wikipedia2Onto. Building Concept Ontology Automatically, Experimenting with Web Image Retrieval," *Informatica (Slovenia)*, vol. 34, no. 3, pp. 297–306, 2010.

[6] X. Dong, K. Chen, J. Zhu, and B. Shen, "Learning to discover subsumptions between software engineering concepts in wikipedia," in *Proc. SEKE 2016*, 2016, pp. 147–152.

[7] K. Chen, X. Dong, J. Zhu, and B. Shen, "Building a domain knowledge base from wikipedia: a semi-supervised approach," in *Proc. SEKE 2016*, 2016, pp. 191–196.

[8] B. Combemale, R. Lämmel, and E. V. Wyk, "SLEBOK: The Software Language Engineering Body of Knowledge (DS 17342)," *Dagstuhl Reports*, vol. 7, no. 8, pp. 45–54, 2018.

[9] J. Favre, D. Gasevic, R. Lämmel, and A. Winter, "Guest Editors' Introduction to the Special Section on Software Language Engineering," *IEEE Trans. Software Eng.*, vol. 35, no. 6, pp. 737–741, 2009.

[10] D. Mirylenka, A. Passerini, and L. Serafini, "Bootstrapping domain ontologies from wikipedia: A uniform approach," in *Proc. IJCAI 2015*, 2015, pp. 1464–1470.

[11] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, "The neon methodology framework: A scenario-based methodology for ontology development," *Applied Ontology*, vol. 10, no. 2, pp. 107–145, 2015.

[12] S. Sarencheh and A. Schiffauerova, "Automatic algorithm for extracting an ontology for a specific domain name," in *Proc. KEOD*, 2017, pp. 49–56.

[13] T. Flati, D. Vannella, T. Pasini, and R. Navigli, "Two Is Bigger (and Better) Than One: the Wikipedia Bitaxonomy Project," in *Proc. ACL*, 2014, pp. 945–955.

[14] P. Kuhn, S. Mischkewitz, N. Ring, and F. Windheuser, "Type inference on wikipedia list pages," in *46. Jahrestagung der Gesellschaft für Informatik*, 2016, pp. 2101–2111.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[16] R. Lämmel, D. Mosen, and A. Varanovich, "Method and Tool Support for Classifying Software Languages with Wikipedia," in *Proc. SLE 2013*, 2013, pp. 249–259.

[17] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *Proc. WWW*, 2006.

[18] D. Vrandecic, "Wikidata: a new platform for collaborative data collection," in *Proc. WWW*, 2012, pp. 1063–1064.

[19] R. Zarrad, N. Doggaz, and E. Zagrouba, "Title-based approach to relation discovery from wikipedia," in *Proc. KEOD*, 2013, pp. 70–80.

[20] V. Presutti, S. Consoli, A. G. Nuzzolese, D. R. Recupero, A. Gangemi, I. Bannour, and H. Zargayouna, "Uncovering the Semantics of Wikipedia Pagelinks," in *Proc. EKAW 2014*, 2014, pp. 413–428.

[21] M. Morsey, J. Lehmann, S. Auer, C. Stadler, and S. Hellmann, "Dbpedia and the live extraction of structured data from wikipedia," *Program*, vol. 46, no. 2, pp. 157–181, 2012.