Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Wikifying Software Artifacts

Mathieu Nassif · Martin P. Robillard

Received: date / Accepted: date

Abstract
Context. The computational linguistics community has developed tools, called wikifiers, to identify links to Wikipedia articles from free-form text. Software engineering research can leverage wikifiers to add semantic information to software artifacts. However, no empirically-grounded basis exists to choose an effective wikifier and to configure it for the software domain, on which wikifiers were not specifically trained.
Objective. We conducted a study to guide the selection of a wikifier and its configuration for applications in the software domain, and to measure what performance can be expected of wikifiers.
Method. We applied six wikifiers, with multiple configurations, to a sample of 500 Stack Overflow posts. We manually annotated the 41 124 articles identified by the wikifiers as correct or not to compare their precision and recall.
Results. Each wikifier, in turn, achieved the highest precision, between 13% and 82%, for different thresholds of recall, from 60% to 5%. However, filtering the wikifiers' output with a whitelist can considerably improve the precision above 79% for recall up to 30%, and above 47% for recall up to 60%.
Conclusions. Results reported in each wikifier's original article cannot be generalized to software-specific documents. Given that no wikifier performs universally better than all others, we provide empirically grounded insights to select a wikifier for different scenarios, and suggest ways to further improve their performance for the software domain.

Keywords Wikipedia · Wikification · Comparative Evaluation · Stack Overflow · Knowledge-Driven Software Engineering

M. Nassif · M. P. Robillard
School of Computer Science
McGill University
Montréal, Canada
E-mail: {mnassif, martin}@cs.mcgill.ca

1 Introduction

Recent work in software engineering has used knowledge bases extensively to address various challenges associated with knowledge-intensive tasks. For example, Chen et al. (2018) and Ye et al. (2016c) learn word embeddings from Stack Overflow[A] and Wikipedia[B] (respectively) to improve recommender systems for libraries and software documents, respectively.[1] Patil (2017) aggregates bug reports based on an explicit semantic analysis approach using Wikipedia entries as concepts. Other research work proposes foundational techniques that can in turn help address software engineering problems. Examples include a technique to cluster source code elements (Schindler et al., 2015), and another to categorize technology terms (Nassif et al., 2020), both of which leverage Wikipedia and/or Stack Overflow as knowledge bases.

With prior efforts devoted to the creation of authoritative software engineering knowledge bases (ISO/IEC/IEEE, 2017; Bourque and Fairley, 2014) and to the design of techniques to automatically generate them (e.g., Wang et al., 2019; Zhao et al., 2017), software engineering research on knowledge management has a large space of resources to leverage. However, a common challenge in this area is to associate natural language resources, such as code comments, bug reports, blog posts, or tutorials sections, with relevant entries in a knowledge base.

Wikification, a prolific sub-area of computational linguistics that matured over the past decades, addresses this challenge: wikification tools (also known as *wikifiers*) can automatically associate concepts mentioned in a natural language document to a relevant entry in a knowledge base (typically Wikipedia). These associations are usually added to the document itself as hyperlinks to help readers understand unfamiliar terms. Thus, software engineering techniques can leverage mature wikification techniques to automatically link technical documents to supporting resources, either to help readers better understand the resource, or to improve other information retrieval techniques. However, wikification research targets well-written and general-domain documents (Hoffart et al., 2011; Cheng and Roth, 2013; Brank et al., 2017; Milne and Witten, 2008),[2] and it is unclear how well it can address the peculiarities of software-specific documents.

Software engineering terminology adds complexity to the wikification task, because many common terms have a specific technical sense. For example, a "lock" in the software domain can refer to a concurrent programming concept, or a file access restriction, but less likely to the physical security device. Similarly, "Python" more often refers to the programming language than a type of snake. The fast-growing list of technologies, many of them named using common terms, makes the problem of terminology even more challenging (Nassif et al., 2020).

[1] Uppercase letters in superscript (e.g., [A]) refer to URLs listed in Appendix A.

[2] There are a few exceptions of tools that target specific types of documents, such as Twitter messages (tweets) (Cassidy et al., 2012).

A second challenge originates from the peculiar format of software resources. They often include code fragments, either in distinct blocks or inserted directly in the text. Source code identifiers (e.g., names of variables and types) can also appear in the text without any special formatting, and often in different morphological forms (Chen et al., 2017), which makes them hard to distinguish from natural language words (Ye et al., 2018). This mix of code and natural language leads to an uncommon syntax and many out-of-vocabulary tokens, two challenging aspects of natural language processing. Thus, the impact of the peculiar format of software resources on wikification is hard to reliably estimate.

Recent work proposed software-specific techniques related to wikification (Ye et al., 2016a), but no end-to-end wikification technique exists yet for software resources, or even evaluations of general techniques on a software-related dataset. Thus, to facilitate the use of wikification technologies in software engineering, **the main contribution of this article is an in-depth independent comparative study of the effectiveness of six wikifiers on Stack Overflow posts**. The objective of this study is to determine precisely how well different wikifiers can identify relevant Wikipedia articles in software resources. This study gathers substantial empirical evidence on which to base the selection of a particular wikifier, as well as values for its configuration parameters.

Perhaps surprisingly, we found that different wikifiers are the most precise for different recall values, which demonstrates the need to carefully choose the right wikifier and configuration. Overall, however, the JSI wikifier (Brank et al., 2017) is the most flexible to accommodate various trade-offs between precision and recall. Additionally, we contribute:

- an assessment of the performance of wikifiers on Stack Overflow posts as an example of software resources, which, as the results show, differs greatly from the performance reported in the original articles;
- empirically grounded insights to improve the performance of wikifiers for the software domain, either with the use of white- and blacklists to filter the output, or based on an investigation of the synergy between the output of different wikifiers;
- a validated list of 1098 Wikipedia titles associated to computing, that can be used as a seed to scope the coverage of computing concepts of Wikipedia, especially for knowledge graph research in software engineering.

All data used in this study, as well as the results of the manual annotation phase and the list of 1098 validated Wikipedia computing titles, is available in an on-line appendix at `https://doi.org/10.5281/zenodo.3727035`. This on-line appendix supports the independent verification and full replication of the findings of this study.

Software system

From Wikipedia, the free encyclopedia

Not to be confused with System software.

A **software system** is a system of intercommunicating components based on software forming part of a computer system (a combination of hardware and software). It "consists of a number of separate programs, configuration files, which are used to set up these programs, system documentation, which describes the structure of the system, and user documentation, which explains how to use the system".[1]

The term "software system" should be distinguished from the terms "computer program" and "software". The term computer program generally refers to a set of instructions (source, or object code) that perform a specific task. However, a software system generally refers to a more encompassing concept with many more components such as specification, test results, end-user documentation, maintenance records, etc.[2]

Fig. 1: Excerpt from the Software system[D] Wikipedia article. Links to other articles appear in blue boxes.

2 Background on Wikification

The term *wikification* refers to the process of adding links to relevant Wikipedia articles from a natural language document, so that the reader of the document can easily find description of pertinent related concepts. The result of this process is similar to existing Wikipedia articles (hence the name) which contain internal links to other articles. The wikification process was originally a manual process: authors of Wikipedia articles would explicitly add links to other articles. As the popularity of Wikipedia grew, and authors of non-Wikipedia documents started to link to Wikipedia (e.g., from Stack Overflow[A] and Reddit[C] posts, Vincent et al., 2018), researchers designed tools, called *wikifiers*, to automatically perform wikification on custom text inputs. Organizations use wikifiers, for example, on news articles to refer readers of the document to contextual or prerequisite knowledge helpful to understand a news item.

For example, Figure 1 shows the first two paragraphs of the Wikipedia article Software system,[D] which contain fourteen links to eleven unique articles.[3] The objective of a wikifier is to automatically discover the same links as those identified by humans when given the raw text of the article as input.

In the context of wikification, a *mention* represents the fragment of text in the document that is associated with the Wikipedia article. In Figure 1, the first mentions in the first paragraph are *system*, *components*, and *software*. Mentions can consist of multiple words, such as *computer system*. In some cases, the Wikipedia article associated with a mention has a title that

[3] We mark titles of Wikipedia articles with a different Font.

differs from the mention, as is the case for the mention *components*, associated with Software component (itself redirecting to Component-based software engineering).

Typically, to evaluate wikification results, it is not sufficient to identify a set of related Wikipedia articles. Wikifiers must also associate these articles to the correct mention. For example, in Figure 1, a wikifier must associate the article System with the mention *system*, and not other terms such as *intercommunicating*. However, this study targets a variant of the wikification task that focuses only on the identification of a set of related articles, and disregards the mentions themselves. Thus, the expected outcome of wikifiers, for this study, is a set of articles (e.g., System software, System, and so on), rather than a list of mention–article pairs (e.g., <*system*, System>, <*components*, Software component>, and so on). Meij et al. (2012) originally described this variant of wikification, and Cornolti et al. (2013) named it *C2W*, for *Concepts to Wikipedia*.

This study focuses on C2W because it more naturally maps to potential applications in software engineering. Once the wikifier identifies relevant articles, the exact mentions are not useful to understand the concepts related to the software-related document. Therefore, these mentions should not affect the evaluation results. Nevertheless, when discussing the results of wikifiers, we occasionally refer to the mention associated with an article, when the context requires it (e.g., to interpret a result).

3 Study Preparation

The preparation for this study required the selection and preprocessing of a sample of Stack Overflow posts, as well as the selection of the wikifiers to compare, and which of their parameters to experiment with.

3.1 Types of Software Resources

Software-related resources can take many forms, such as source files, code comments, various forms of developer communications (e.g., emails, bug reports, forum posts), and technical or end-user documentation. These forms differ widely in many aspects, such as the level of explicit structure in the document, the formality and quality of the language, and the ratio of natural language to code.

At one end of the spectrum, well-written and highly edited software resources aimed at a general audience (e.g., end-user manuals) are similar to the general-domain documents, such as news articles, used to develop and train wikification techniques. Thus, we can expect the performance of wikifiers on these documents to be similar to the performance reported for general-domain documents.

At the other end of the spectrum, software resources composed entirely of source code are clearly outside the intended scope of wikifiers, which take

natural language as input. Hence, although the wikifiers may identify a few relevant concepts from identifiers in code artifacts, source code does not constitute an appropriate input for evaluation. The results would reflect the effectiveness of the preprocessing steps (e.g., identifier tokenization, aggregation into sentences) rather than the performance of wikifiers.

As a middle ground, the evaluation set consists of Stack Overflow posts, which are mainly written in natural language, but also contain code fragments. Some of the posts are long and well-structured documents, edited many times by the community to improve the quality of the language and add thorough descriptions of related concepts. These posts are similar to smaller versions of technical documentation and end-user manuals. At the other end of the spectrum, some posts are closer to fragments of informal discussions, with short replies and grammatically incorrect sentences. Posts also vary in their natural language-to-code ratio, from some that contain only text to others with only code, including posts with code formatted as natural language. In terms of information content, Stack Overflow covers virtually all computing domains. Finally, software engineering research often leverage Stack Overflow as a source of knowledge (Treude and Robillard, 2016; Barua et al., 2014; Ponzanelli et al., 2013), which makes the results of this study directly applicable to techniques that use Stack Overflow posts as input data.

3.2 Sample Selection

Our evaluation set contains 500 Stack Overflow posts from the September 2019 Stack Exchange archive.[E] This set is a uniform random sample from all 44 016 828 posts (questions and answers alike) with a nonnegative score,[4] considering only the most recent version of each post. We chose to discard posts with negative scores (3.9% of the population) to avoid the bias from documents explicitly flagged as problematic.

Typically, in quantitative research, the sample size ensures that statistics computed on the sample generalize to the population within a known error margin, at a predefined confidence level. This was not possible in our study, because the statistics used to compare wikifiers (precision and recall) measure proportions of Wikipedia articles, generated by wikifiers, rather than posts.[5] Because the articles are neither independent nor randomly generated, they do not meet the necessary assumptions for statistical generalization. This situation is not uncommon when evaluating wikification approaches. Prior work often reuses standard annotated corpora, which are not random at all, but instead allow for direct comparison of different approaches (e.g., Moro et al.,

[4] The score of a Stack Overflow post is visible next to the post on Stack Overflow, and represents the number of "upvotes" minus the number of "downvotes" attributed by Stack Overflow users based on the usefulness and quality of the post.

[5] One alternative is to aggregate each statistic per post first, then average them over all posts. However, this alternative gives more weight to small posts with few related articles, which would be detrimental to the interpretation and generalizability of the results.

Table 1: Properties of the 500 Selected Stack Overflow Posts (min = minimum value; Qn = n-th quartile; max = maximum value; avg = average; Edit = number of full days between the creation of the post and its last editing).

Property	**Value**					
Type of post	186 questions, 105 accepted answers, 209 non accepted answers					
Score	min: 0	Q1: 0	Q2: 1	Q3: 2	max: 96	avg: 2.5
# Words	min: 0	Q1: 29	Q2: 54	Q3: 102	max: 459	avg: 72.9
Creation year	08-09: 21	10-11: 74	12-13: 95	14-15: 118	16-17: 112	18-19: 80
Edit (days)	min: 0	Q1: 0	Q2: 0	Q3: 0	max: 3282	avg: 108

2014; Brank et al., 2017). We discuss the implications of the sampling procedure and the results of a sensitivity analysis with the threats to validity (Section 5.5).

Nevertheless, we designed our sample to be *representative* of the population, in the sense that insights gained from this sample should apply to the whole population. With this regard, despite its small size relative to the population, the sample is sufficiently large to contain a non trivial number of posts related to the popular topics discussed on Stack Overflow, as well as posts that exhibit common characteristics, such as long and short posts, posts with and without code fragments, and posts that are more or less well written and formatted. Hence, the sample size of 500 posts gives us reasonable confidence that our findings do not merely reflect spurious relations.

Table 1 shows summary statistics for the 500 posts in the sample: the first row indicates the number of questions and answers, the second and third rows, the distribution of score and number of words (excluding code blocks) per post, and the fourth and fifth rows, the year in which the post was created, and the last time it was edited, in days, relative to its creation date. Each type of post amounts to a non trivial proportion of the dataset. Almost half (42.8%) of the posts have a score of 0. Posts contain an average of 72.9 words, with six of them having no word (i.e., they contain only a code fragment). The sample contains posts reasonably distributed among the years, although the early years of the forum unsurprisingly contain fewer posts. Finally, a majority of posts (86.6%) were not further edited past one day after their creation, with one notable outlier edited almost nine years after its creation. Overall, these statistics show that the sample consists of posts that vary in length (from a few words to multiple paragraphs) and time (they do not only reflect a specific period). Most, but not all, posts are not heavily edited or highly scored, as would be, for example, popular blog posts.

3.3 Preprocessing of Posts

The Stack Overflow archive stores the post bodies as HTML-formatted documents. Although some wikifiers can take as input HTML documents, not all of them can. To perform an equivalent comparison, we used the jsoup[F] library,

version 1.11.2, to convert the HTML posts to plain text, and provided this text as input to all wikifiers.

Because wikifiers take natural language as input, not code, we removed code blocks (identified by HTML `pre` tags) from the post bodies, but kept inline code fragments (identified by HTML `code` tags without `pre` tags). The rationale behind this decision was to avoid breaking up sentences, and because users do not consistently format inline code fragments as such: some users use the "inline code" format option for terms other than code, such as names of technologies, whereas others do not use this formatting option at all, even for legitimate code fragments.

We excluded all other information, such as comments or edit messages, from the input. In particular, we did not include question titles, to avoid inconsistencies between questions and answers, and because most question bodies are self-contained.

3.4 Selection Procedure for Wikifiers

For this study, wikifiers must be able to generate a list of Wikipedia articles about concepts mentioned in an input Stack Overflow post. Prior work has proposed a large number of wikification techniques, but only a small subset of these techniques come with an implementation that can wikify custom texts. We used the following criteria to select wikifiers:

1. The wikifier must be usable immediately, without an intervention from the user (other than installing the necessary software and possibly writing short driver code). In particular, the wikifier must not require the user to provide their own training set.
2. The wikifier must identify Wikipedia articles from a input consisting of plain text. If the wikifier links concepts to another knowledge base, it must be possible to associate (most) entries of this knowledge base with Wikipedia articles in a straightforward manner.
3. The wikifier must be freely available (for non-commercial use), either as a web service, packaged executable, or source code. For web services, requests can be limited to a reasonable rate.
4. The wikifier must provide an interface that allows it to be programmatically integrated as a component of a larger software system. In particular, the wikifier must not be a tool for demonstration only or a replication package for a specific experiment. The interface must be sufficiently well documented to be usable with reasonable effort.
5. The wikifier must not be deprecated by a more recent wikifier developed by the same group.

3.5 Selection Procedure for Configuration Parameters

Each wikifier has a number of configuration parameters, used to tune their performance. Not all parameters, however, should be tuned by the end users:

for example, some hyperparameters affect the training phase of the wikifier, and other parameters have their values optimized during the training phase. For each wikifier, we carefully examined each of their parameters to understand whether the parameter would likely be tuned by an end user, or be fixed to an appropriate default value.

All wikifiers have at least one parameter that roughly estimates the confidence that an article is mentioned in the post, and is expressed as a number in the unit interval. Although this parameter has different names in different wikifier implementations, for consistency, we refer to it as the *confidence threshold*, except when referring to a specific wikifier. The confidence threshold is the main parameter that end users employ to tune the wikifier performance, and our comparative analysis focuses primarily on variations of this parameter.

Some wikifiers also have a *secondary numeric parameter* that allows to filter results based on a numerical property different from the confidence threshold. Other wikifiers have a *binary option* to select which of two components to use to perform one of the wikification subtask, or to enable or disable an optional algorithm. We studied the impact of these parameters on the wikifiers performance as well.

Finally, some of the configuration parameters that must be tuned by the end user relate to concrete and objective properties of the wikification task, such as the input language (e.g., English). Such parameters also include options to disable key components of the approach, to perform ablation studies on the wikifier. For those parameters, we selected the most appropriate value based on the sample of posts: we did not disable any key component, and we selected values that fit the properties of Stack Overflow posts (e.g., they are written in English).

We relied on the wikifier's documentation and preliminary experimentation to distinguish between parameters that should be tuned, parameters that should receive a fixed, appropriate value, and training hyperparameters or parameters that should keep their trained value.

3.6 Selected Wikifiers

We identified six tools that respect the criteria presented in Section 3.4. Each tool can be used on a computer with 16 Gb of memory and 400 Gb of storage. Table 2 shows, for each wikifier, the knowledge base it resolves mentions to and the names of the confidence threshold and additional tuning parameters studied. The following paragraphs describe each wikifier and their parameters in more details. The versions of both the wikifiers and their corresponding knowledge base is given when it is available. We also relate the performance measures reported in the original articles that present each wikifier.

AmbiverseNLU[G] (Ambiverse) is an open source natural language understanding Java suite that resolves entities to the YAGO knowledge base (Rebele et al., 2016). It can be added as a Maven dependency to Java projects, or run from a Docker container. Its entity recognition component, KnowNER (Seyler

Table 2: Wikifiers Compared in this Study

Wikifier	Knowledge Base	Confidence	Additional Parameter
Ambiverse	YAGO	confidence	NER: knowner/stanford
Babelfy	BabelNet	score	MCS: on/off
DBpedia	DBpedia	confidence	support
Illinois	Wikipedia	score	–
JSI	Wikipedia	1-pageRankSqThreshold	pageRank
WAT	Wikipedia/Wikidata	rho	tokenizer: opennlp/lucene

et al., 2018), uses several knowledge base derived features in a linear chain conditional random field (CRF) model to improve the state of the art. Its disambiguation component, AIDA (Hoffart et al., 2011), uses a linear combination of a prior probability based on popularity, a similarity score, and a coherence metric. In this study, we used the Maven artifact, version 1.1.0, with the database dump version `aida_20180120_cs_de_en_es_ru_zh_v18`. It is the most resource-consuming wikifier, requiring 16 Gb of memory to parse English texts and a little under 400 Gb to import the YAGO database. The performance of AIDA and KnowNER were evaluated separately. Hoffart et al. (2011) reported a precision of 82% at the 100% recall level for AIDA, and Seyler et al. (2018) reported F1 scores between 88% and 91% for KnowNER.

The confidence threshold of Ambiverse is named *confidence*. Ambiverse also has a secondary numeric parameter, *salience*, which indicates the relevance of the named entity to the document. However, because the salience is always 0 on non-named entities (i.e., concepts such as Computer and Debugging), which constitute most of the mentions, we did not consider it. Ambiverse also has a binary option: it allows users to use an alternative NER component (*stanford*), instead of KnowNER (*knowner*).

Babelfy[H] (Moro et al., 2014) is an online wikifier with a REST API that resolves entities to BabelNet (Navigli and Ponzetto, 2012), which also offers a REST API. To wikify a document, Babelfy first identifies a large set of candidate entities for each (possibly overlapping) mention using string matching heuristics, then leverages a densest subgraph algorithm to filter among the entities, so that the entities retained form a coherent set. Both Babelfy and BabelNet APIs share a daily quota of 1000 daily requests by default (after registration). This project also has a commercial counterpart, Babelscape. For this study, we used the non-commercial service. The version of the Babelfy REST API is not available, but the website states that it uses version 3.0 of BabelNet. Moro et al. (2014) reported an accuracy[6] between 72% and 82% for Babelfy on the entity linking task, and an F1 score of 87% for the disambiguation task only.

The confidence threshold of Babelfy is simply named *score*. Babelfy also offers the binary option to activate a fallback strategy, termed "most common sense" (MCS). When this option is enabled, if the main disambiguation

[6] Moro et al. (2014) do not provide an explicit definition for accuracy.

algorithm fails to identify a Wikipedia article for a mention, it uses its most common sense (i.e., the article with the highest prior probability). All articles linked with the MCS strategy have a confidence value of 0.

DBpedia Spotlight[I] (DBpedia) (Mendes et al., 2011; Daiber et al., 2013) is an online wikifier that resolves entities to DBpedia (Lehmann et al., 2015). DBpedia Spotlight also identifies candidates using simple string heuristics, then disambiguates between the candidates with a vector space model to represent knowledge base entities. The wikifier uses a custom measure derived from tf-idf, called tf-icf (for "inverse candidate frequency") and the cosine similarity metric to weigh and compare vectors. DBpedia Spotlight's REST API does not impose any rate limit, but during our study, the server was unstable and would often return HTTP 502 or 503 errors for valid requests. The website does not mention the version of the wikifier, nor the backing knowledge base. Mendes et al. (2011) reported a precision-recall plot of DBpedia, where precision varies between 40% and 82% for recall values between 12% and 61%, with a maximal F1 score of 56%.

Similar to Ambiverse, DBpedia's confidence threshold is named *confidence*. However, contrary to all other wikifiers, DBpedia's confidence score must be set a priori. We used values between 0 and 1, with 0.1 increment (i.e., 0, 0.1, 0.2, ..., 0.9, 1.0) to sample the parameter space. DBpedia returns a secondary numeric parameter, *support*, which is a positive integer that is a property of the entity (i.e., independent of the associated mention or input text).

Illinois Wikifier[J] (Illinois) (Cheng and Roth, 2013; Ratinov et al., 2011) is an open source Java wikifier that resolves entities to Wikipedia. To disambiguate mentions to articles, Illinois includes several local and global features into a linear combination. Local features, which are defined between a mention and a Wikipedia article, use the cosine similarity on tf-idf vectors of the mention and the article. Global features use the pointwise mutual information (PMI) and normalized Google distance (NGD) metrics on the Wikipedia link graph between all pairs of entities. Finally, Illinois improves the results by using semantic relations, extracted from the input text, between entities. Due to many dependencies to outdated maven artifacts, we experienced some difficulty in importing the wikifier into our project. To solve our issues, we downloaded a compiled distribution, version 3.0, that includes all dependencies. Cheng and Roth (2013) reported F1 scores between 81% and 93% on four datasets for Illinois.

Similar to Babelfy, Illinois' confidence threshold is named *score*. Illinois also offers several predefined configurations, of which *STAND_ALONE_NO_INFERENCE* was the most appropriate for custom text (the "with inference" configuration requires the commercial Gurobi Optimizer software).

JSI Wikifier[K] (JSI) (Brank et al., 2017) is an online wikifier with a REST API that resolves entities to Wikipedia. At its core, JSI uses an augmented mention–entity graph of Wikipedia: first, JSI constructs the bipartite graph with anchor text (mentions) on one side, and articles (entities) on the other, and edges linking each anchor text to the article they point to. Then, it augments this graph with edges between articles that are similar, according to a

similarity metric based on internal links. JSI computes pagerank values (Page et al., 1999) on the nodes of this graph, and returns the set of articles with the highest pagerank values. JSI's REST API requires registration, but does not impose any rate limit. It does not provide version information on either the software or backing knowledge base. Brank et al. (2017) reported an F1 score of 59% for JSI.

Instead of using an absolute pagerank value as its confidence threshold, JSI computes the threshold as a proportion of the sum of the squares of all pagerank values. Thus, the user-defined proportion constitutes the confidence threshold (named *pageRankSqThreshold*). Because a lower proportion leads to a lower recall, for this study, the confidence threshold is actually *1 - pageRankSqThreshold*. JSI also returns *pageRank* values with the results, so we considered them as a secondary numeric parameter.

WAT^{L} (Piccinno and Ferragina, 2014) is an online wikifier with a REST API that resolves entities to Wikipedia and Wikidata. WAT is the successor of another wikifier, TagME (Ferragina and Scaiella, 2010). To identify mentions and candidates, WAT trained a binary classifier with features from Wikipedia articles, such as redirect titles and anchor text of internal links. To disambiguate between candidates, WAT leverages two algorithms. The "voting" algorithm developed for TagME computes the sum of a relatedness measure between pairs of extracted entities, weighted by a priori probabilities. The other algorithm is based on a mention–entity graph similar to that of JSI, but computed only over mentions and entities extracted in the first step. Similar to JSI, WAT imposes no rate limit, but requires registration. It also does not provide any version information. Piccinno and Ferragina (2014) reported precision and recall values respectively between 46% and 49%, and between 51% and 59%, for different configurations of WAT.

WAT's confidence threshold is named ρ (*rho*). The only other parameter (apart from the input language) is the *tokenizer* to use: *opennlp* or *lucene*.

A notable tool missing from this list is Milne and Witten's **Wikipedia Miner** (Milne and Witten, 2008, 2013). This is one of the first available wikifiers, and it has served as a baseline for the evaluation of many other wikifiers. However, the web service is no longer operational, and we could not find the source code with pre-trained models.

4 Data Annotation

The evaluation of wikifiers consisted of executing all of them on the posts from the dataset, collecting all generated pairs of articles and posts, and manually annotating each pair as correct or incorrect. This procedure generated a *reference set* to assess the precision and recall of each wikifier.

The usual method to evaluate a wikifier is to compare its output with a gold standard (e.g., Moro et al., 2014; Brank et al., 2017), which allows to automatically discriminate false positives (FP) from true positives (TP) and list all false negatives (FN). Figure 2 shows a sentence with a corresponding

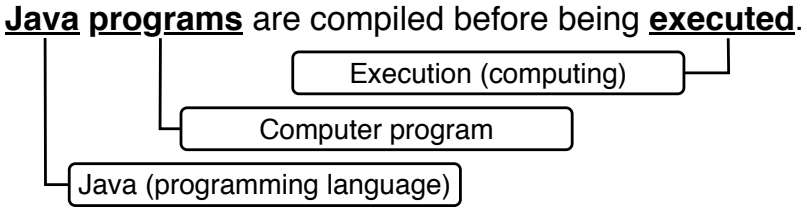


Fig. 2: Sample sentence with a gold standard wikification: Titles in round boxes indicate the expected Wikipedia article that wikifiers should identify.[7] Exact mentions (in bold and underlined) are shown for better understandability, but are not evaluated.

gold standard.[7] Three articles are expected to be found by the wikifiers. If a wikifier outputs the two articles Java (programming language) (correct) and Compilation (album) (incorrect), there would be one true positive, one false positive, and two false negatives. From these counts, it is possible to compute the precision and recall.[8] The wikifier in the previous example would have a precision of 0.5 (i.e., 1/2) and a recall of 0.3 (i.e., 1/3). F1 scores[9] are also commonly reported for wikifier evaluations, as a single objective function that balances precision and recall. However, in the context of our study, a single arbitrary objective function would only hide the more precise information captured by precision and recall, so we chose to present the latter metrics separately.

With such a method of evaluation, the most effort-intensive and critical task is to generate the gold standard. Creating a high quality evaluation dataset is hard, especially when such concepts are not named entities, as humans do not always agree on the concepts that should be linked. Because of this difficulty, wikifiers are commonly evaluated against standard golden datasets, such as AIDA-CoNLL, which consists of 1393 news articles manually annotated by Hoffart et al. (2011).[10] For example, Brank et al. (2017) used this dataset as a gold standard, and Moro et al. (2014) used six different gold standards, including AIDA-CoNLL.

None of these available gold standards are specific to software engineering, and in particular, none of them uses Stack Overflow posts. To mitigate the cost of creating a new, high quality evaluation dataset, we used a slightly different approach. We generated a *reference set*, rather than a gold standard, to compare the relative, rather than absolute, performance of wikifiers. We exe-

[7] At the time of writing, Wikipedia did not have an article specifically on source code compilation, but only an article about Compiler. For this study, we considered the distinction between the two concepts (a process and tools to perform it) significant enough to reject the article Compiler for the mention *compiled*.

[8] precision $= \frac{TP}{TP+FP}$; recall $= \frac{TP}{TP+FN}$

[9] F1 score is the unweighted harmonic mean of precision and recall, or $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision}+\text{recall}}$

[10] The AIDA-CoNLL dataset only links proper nouns (i.e., named entities), but other datasets exist for both named entities and concepts.

cuted all wikifiers on each post, with parameters that maximize recall. We then manually annotated each association between an article and a post as correct or incorrect (without knowing which wikifier produced which associations).[11] The reference set generated with this procedure is sufficient to discriminate false positives from true positives, and have a consistent list of false negatives across all wikifiers.

The precision values computed with this procedure are equivalent to those that would be derived from a gold standard. However, recall values are only *relative* to all true positives found by any of the six wikifiers, instead of all theoretically possible true positives. Thus, the ratios of recall values between wikifiers is consistent with the ratios that would be derived from a gold standard, but their absolute magnitudes are not.

Using this procedure, it is possible to accept multiple articles for the same mention, which would not be possible with a gold standard. For example, for the sentence *"Paypal is a payment system."*, standard datasets would associate the mention *payment system* with either Payment system or E-commerce payment system. But, using our reference set, both of them are correct associations.

4.1 Annotation Task

The annotation task consisted of answering, for each article–post pair, the single question

Is the concept represented by the Wikipedia article related to computing and explicitly mentioned in the Stack Overflow post?

The above question is the result of an iterative process over a pilot set to mitigate the subjectivity of the task as much as possible, so that the results would be objective and lead to an unambiguous interpretation. It requires two conditions for an article–post pair to be *correct* (i.e., for a positive answer to the question): the article must be *explicitly mentioned* in the post, and it must be *related to computing*. The following paragraphs discuss in details why we needed to include these conditions.

Condition 1 (Explicit Mention): Some concepts do not have an associated Wikipedia article. For example, there is no article dedicated to the HTTP `GET` method (it is described in a section of Hypertext Transfer Protocol). When concepts with no Wikipedia article appear in a document, wikifiers sometimes link the mention to a related article (e.g., linking *GET* to Hypertext Transfer Protocol). However, for the annotation task, we require that related articles (including articles about hypernyms and holonyms) be marked as "incorrect".

[11] The term "annotate", in the wikification community, often refers to the wikification process itself, or a variant. In this article, we use "annotate" (and its derivatives) only when referring to the manual annotation of human experts to assess the correctness of an article–post pair.

Only articles about concepts that are actually mentioned in the post can be "correct". This condition is necessary to avoid an ambiguous or subjective definition of relatedness threshold, and favors wikifiers that link specific mentions (e.g., *Java*) to specific articles (e.g., Java (programming language)) instead of general ones (e.g., Programming language).

Condition 2 (Related to Computing): The pilot annotation task revealed that wikifiers often correctly identify mentions that are nonetheless irrelevant to the context, such as phatic expressions (e.g., *Thanks*), figures of speech (e.g., idioms and metaphors), common functional phrases (e.g., *There exists*), and terms related to the Q&A nature of Stack Overflow (e.g., *question* and *answer*). Typical wikification guidelines, such as those from the Wikipedia Manual of Style on linking,[N] suggest to only link relevant mentions, and avoid "everyday words understood by most readers in context" (Wikipedia, 2019). However, relevance is often subjective. To avoid this subjectivity, an article is considered relevant if and only if it is related to computing.[12] Consequently, we reject genuinely relevant but non-computing articles, such as Azimuth in a post about astronomy-related libraries,[O] but such articles are rare in a computing-related forum.[13]

To help annotators, we designed an extensive annotation guide that discusses corner cases (e.g., different parts of speech, synonyms, and antonyms) and what exactly is *related to computing*. Elements of this guide are motivated in part by the structure and conventions of Wikipedia, and in part by the lessons learned during the pilot annotation task. The guide also contains a curated list of five Stack Overflow posts from the pilot set, with all associated concepts annotated jointly by the authors, to serve as an example for annotators. This annotation guide is available in our on-line appendix.[M]

4.2 Annotators

Despite the extensive annotation guide, the annotation task still required a considerable effort from the annotators. The difficulty of the annotation task

[12] We express our criterion in terms of *computing*-related articles, rather than only those specific to *software engineering*, because the precise boundary of software engineering is less well defined than that of computing, especially among Wikipedia articles.

[13] There are concepts that can conceivably be related to computing in some contexts, but part of a general body of knowledge in others. Blog is such a concept: We consider that, when referring to a specific post, the concept is not a technical term, but when discussing the creation of a blogging platform, this term becomes related to computing.

Table 3: Agreement between annotators, using Cohen's κ statistic (Cohen, 1960). The number of common article–post pairs are shown in parentheses.

	Author A	**Author B**	**External 1**	**External 2**
Author A	–	0.83 (940)	0.72 (1007)	0.74 (1007)
Author B	0.83 (940)	–	0.60 (569)	0.64 (569)
External 1	0.72 (1007)	0.60 (569)	–	0.67 (1576)
External 2	0.74 (1007)	0.64 (569)	0.67 (1576)	–

arises from quality issues from the Stack Overflow posts,[14] as well as potentially misleading Wikipedia titles.[15]

These difficulties related to the annotation task result in a steep learning curve for the annotators, and an increased threat of quality degradation that would be hard to detect if a careless external annotator wishes to finish the task too quickly. Consequently, we could not efficiently distribute the annotation task to many external annotators, and the two authors annotated all pairs. After the initial learning curve, the authors were able to annotate roughly 1000 pairs per hour.

Investigator bias, common in situations where the investigators perform the annotation, is minimal in this study, because it is an independent evaluation of already existing tools. Nevertheless, we took great care to mitigate even the threat of this bias as much as possible. Annotators were not able to discern which concept was produced by which wikifier, preventing unintentional biases. Furthermore, the set of pairs to annotate was split into two mostly disjoint sets, for efficiency reasons, but both annotators annotated a common, unmarked set, to estimate their agreement on the complete set. Finally, we hired two additional annotators, external to our research group, to re-annotate a subset of pairs, to better understand the difficulty and subjectivity of the task, and further control any biases that could arise. Table 3 shows the agreement between each annotator, using Cohen's κ statistic (Cohen, 1960). The agreement between external coders and the authors are between 0.60 and 0.74, which Landis and Koch (1977) consider "substantial", and the agreement between both authors, who annotated the complete dataset, is "almost perfect" (0.83).

[14] In addition to making grammatical errors, post authors often use natural language shortcuts such as abbreviations, acronyms, omissions, and ambiguous terms, which require additional effort from annotators to resolve. The misuse of formatting options, such as formatting code blocks as inline code, also makes posts harder to understand. For example, the scope keyword "until successful" can easily be misread as a preposition and an adjective if it is not formatted as inline code.[Q]

[15] Annotators cannot assume the topic of a Wikipedia article only by its title. For example, the article Java does not describe the programming language, but the Indonesian island. Also, although most titles that consist only of three capital letters lead to disambiguation pages, the article URL does not. Redirect titles further add to the possible disconnect between titles and article content. Thus, annotators must make the effort to scan the article to verify what it actually describes.

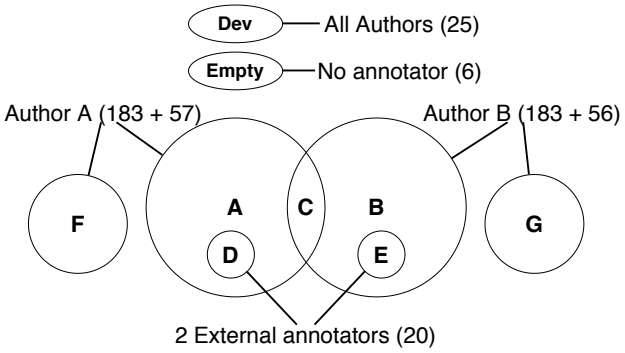


Fig. 3: Annotation sets, with their respective annotator(s). The letters identify each partition, and the total number of posts for each annotation set is shown in parentheses.

Following the computation of the agreement scores, the authors jointly resolved all conflicts in the overlapping sets by discussing each conflicting pair and deciding on the correct annotation.

4.3 Annotation Sets

The wikification of all 500 posts produced 41 124 article–post pairs to annotate. We performed the annotation in two phases. In the initial phase, we annotated the pairs from 385 posts. To gain additional confidence in the reliability of our results, we then conducted a second annotation phase, in which we annotated the pairs from the remaining 115 posts. These two phases allowed us to assess the generalizability of our conclusions by observing the variation in the performance of wikifiers when augmenting the sample by 29.9%. Indeed, we observed very little variation (less than 2 percentage points of precision and recall).

Pairs formed four (non disjoint) sets: one *development* set, two *main* sets, and one *external validation* set.

To improve the efficiency of the annotation task, all pairs from the same post are put in the same set, so that each annotator has fewer posts to read. Therefore, in the following, we describe annotation sets by referring to the Stack Overflow posts they contain, rather than the article–post pairs. Figure 3 summarizes these sets.

Six of the 500 posts (four from the first phase and two from the second phase) happened to contain only a code block, which is removed at the preprocessing stage. Thus, these six posts produced no article for any wikifier, and were naturally not considered when creating the annotation sets.

Of the remaining 494 posts, 25 constituted the development (dev) set, used for the iterative pilot. Both authors annotated this set to refine the annotation task and guide. In contrast to many other evaluations, where development sets

are discarded from the final results, the results include the final annotations of this set, as they constitute valid information that does not threaten the validity of the results. In fact, these annotations are possibly of higher quality due to the many iterations to reach a consensus.

The empty and development sets are disjoint from the other sets. We split the remaining 471 posts into seven partitions: two large partitions of 163 posts each (A and B), three small partitions of 10 posts each (C, D, and E), and two additional partitions of 57 and 56 posts for the second phase (F and G). Author A annotated the first main set, consisting of partitions A, C, and D in the first phase, and partition F in the second phase. Author B annotated the second main set, consisting of partitions B, C, and E in the first phase, and partition G in the second phase. Each external annotator annotated the same validation set, consisting of partitions D and E. We designed this strategy so that the overlap between any two annotators consists of all pairs from at least ten distinct posts, to compute the inter-rater agreement scores shown in Table 3.

5 Results

Of all 500 posts, 41 (8.2%) did not link to any correct article, and the two posts^{RS} with the most links have 37 and 28 linked articles (average: 6.1, median: 5). We analyze the results according to three perspectives, each related to one of the following research questions:

1. How do wikifiers compare to each other, in terms of performance, to wikify Stack Overflow posts? (Section 5.1)
2. How do the additional parameters affect the performance of each wikifier? (Section 5.2)
3. To what degree do different wikifiers identify similar sets of articles? (Section 5.3)

A considerable artifact of this study is the outcome of the annotation task, a manually verified list of 1098 computing-related Wikipedia titles, and 10 854 negative examples.[16] Section 5.4 explores possible ways to use this list to further improve the performance of wikifiers.

5.1 Wikifiers Performance

This section describes how wikifiers compare to each other, with the explicit objective of helping software engineering researchers select a wikifier that best suits their need. For this comparison, only the confidence threshold of each wikifier varies. Additional parameters are set to optimal values, described in Section 5.2. To compare wikifiers, we use the precision and relative recall

[16] The number of distinct articles is slightly less than 1098 and 10 854, respectively, because some titles are actually redirect pages to other articles.

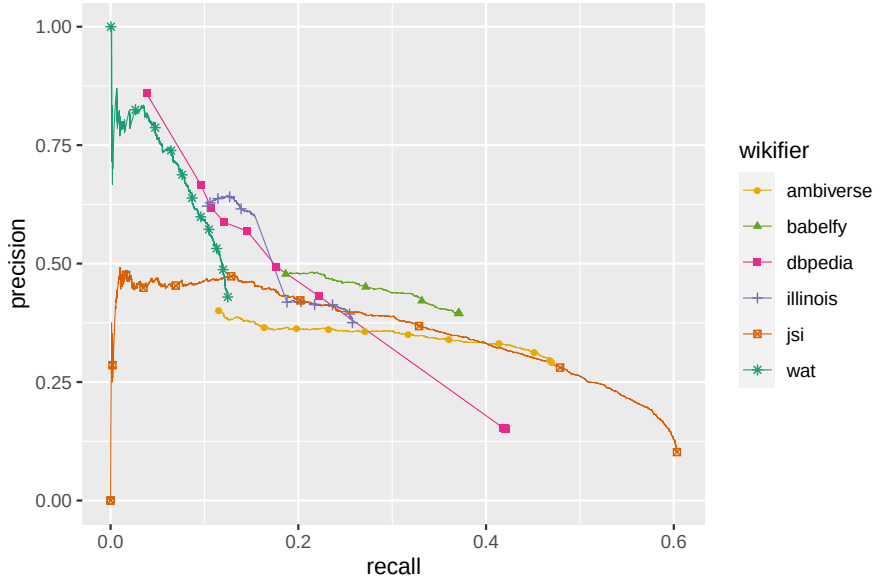


Fig. 4: Precision-Recall curves of all six wikifiers. Markers indicate each 0.1 increment of the confidence threshold (some markers overlap). For DBpedia, a linear interpolation estimates precision and recall values for confidence values that are not exact multiple of 0.1.

metrics (see Section 4), which have an intuitive interpretation and are not affected by true negatives, irrelevant in the context of wikification.

Figure 4 compares the performance of all six wikifiers, in terms of precision and recall, for all confidence thresholds between 0 and 1. Each marker indicates an increment of 0.1 of the confidence. In the case of Babelfy, no confidence score was below 0.7, which explains why only four markers appear on the plot. The performance of all six wikifiers shows the difficulty of wikifying software resources, with precision levels hardly reaching 70% even for low relative recall values (10%), and rapidly decreasing under 50% for higher recall values. Interestingly, no single wikifier completely outperforms another: each one achieves the highest precision for a portion of the recall range. Therefore, before choosing a wikifier, it is important to decide on the acceptable precision or recall values for the specific application, as the optimal wikifier (and confidence threshold) depends on this decision.

Another interesting observation is the difference in the precision and recall ranges. Variations in the confidence threshold of some wikifiers, such as DBpedia and JSI, will greatly affect the recall, with the expected degradation in precision as recall increases, whereas other wikifiers, such as Babelfy and Illinois, have a much lower degradation in precision, but cannot achieve the

Table 4: Comparison of the precision (Pr.) of each wikifier for selected recall (Rec.) values, along with the respective confidence threshold (CT). Dashes [-] indicate that the wikifier did not achieve such recall. DBpedia values (marked by an asterisk [*]) are linear interpolations. For each recall, the best precision is shown in bold. All precision and recall values are percentages.

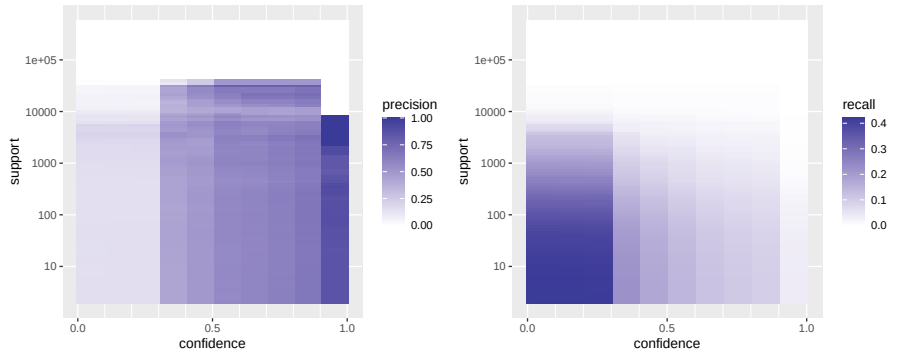
	Ambiverse		**Babelfy**		**DBpedia***		**Illinois**		**JSI**		**WAT**	
Rec.	CT	Pr.	CT	Pr.	CT	Pr.	CT	Pr.	CT	Pr.	CT	Pr.
5	-	-	-	-	0.98	**82**	-	-	0.56	47	0.78	76
10	-	-	-	-	0.87	**65**	-	-	0.44	46	0.36	59
15	0.94	38	-	-	0.59	56	0.52	**61**	0.37	46	-	-
20	0.79	36	0.99	**48**	0.45	46	0.41	42	0.30	42	-	-
25	0.66	36	0.93	**46**	0.39	39	0.13	41	0.26	40	-	-
30	0.54	36	0.85	**44**	0.36	32	-	-	0.22	39	-	-
35	0.42	34	0.75	**41**	0.33	25	-	-	0.18	36	-	-
40	0.32	33	-	-	0.31	18	-	-	0.15	**33**	-	-
45	0.20	**31**	-	-	-	-	-	-	0.12	30	-	-
50	-	-	-	-	-	-	-	-	0.08	**26**	-	-
55	-	-	-	-	-	-	-	-	0.05	**22**	-	-
60	-	-	-	-	-	-	-	-	0.00	**13**	-	-

same recall. The former wikifiers can be used in more varied contexts, but the latter wikifiers perform more consistently.

Finally, some wikifiers, including Illinois and JSI, exhibit a decline in precision for the highest confidence values. This counter-intuitive drop in precision is possibly due to these wikifiers typically giving higher confidence values only to concepts from popular domains, due to their high prior probability. For example, two of the articles associated with the highest confidence values by JSI are Worldwide Exchange (about a television business news program) and Midfielder (a position of football/soccer). Such concepts are more likely to appear in news articles targeted to a general audience, which are used to train wikifiers, than computing concepts, so they are more likely to receive favorable biases during the training phase of wikifiers.

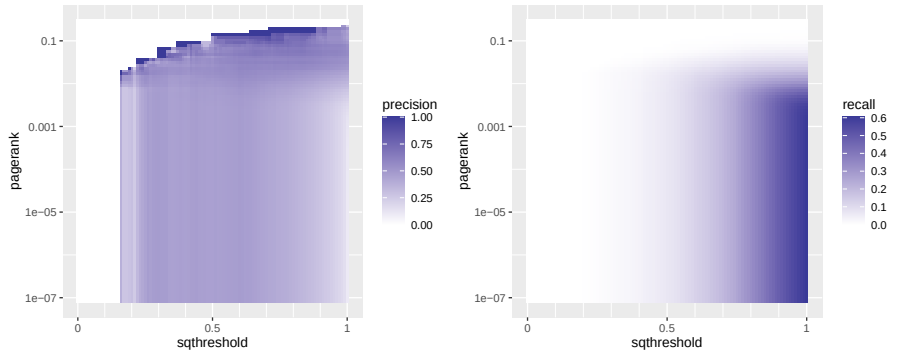
Table 4 shows the value of the confidence threshold needed to achieve different recall values (from 5% to 60%, in steps of 5%), with the associated precision. The highest precision in each row is indicated in bold. For example, a user who wants to wikify Stack Overflow posts with 25% recall should use Babelfy with a confidence of 0.93, and expect a precision around 46%. Conversely, if the user requires a precision of at least 60%, they should choose Illinois, with a confidence around 0.52, and expect a recall around 15%.

Result: All wikifiers have a precision under 50% for relative recall values over 20%. These values greatly differ from the performance reported in the original articles that describe each wikifier. For different recall values, all wikifiers in turn achieves the highest precision, which means that no wikifier is irrelevant.



(a) DBpedia, Precision

(b) DBpedia, Recall



(c) JSI, Precision

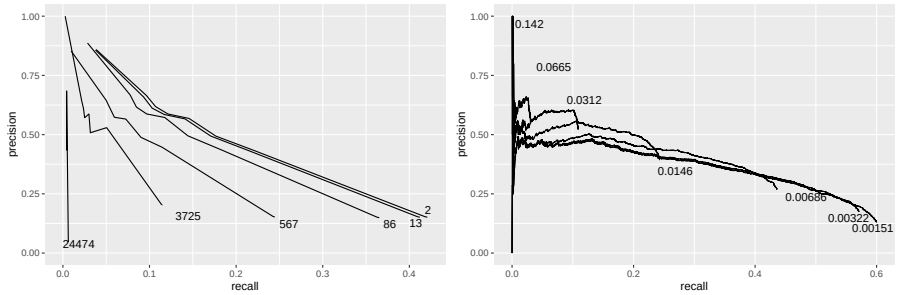
(d) JSI, Recall

Fig. 5: Precision and Recall of DBpedia and JSI for different combinations of their main and secondary numeric parameters. When the wikifier does not return any Wikipedia article for all Stack Overflow posts, we define the precision (typically undefined) to be 0.

5.2 Effect of Additional Parameters

The previous section compares all wikifiers, varying only the confidence threshold. Additional parameters allow to further fine-tune the wikifiers, but, as the result will show, at least in the context of Stack Overflow posts, there is generally little incentive to modify the default parameter values.

Secondary Numeric Parameters (DBpedia/support and JSI/pageRank): Both DBpedia and JSI have a secondary numeric parameter, respectively *support* and *pageRank*, that can further influence the balance between precision and recall. To visualize the effect of these parameters, Figure 5 shows their effect, combined with the confidence threshold, on precision and recall. The axes range from the lowest to the highest value that impacts the results.



(a) DBpedia (secondary: *support*) (b) JSI (secondary: *pageRank*)

Fig. 6: Precision-Recall curves of DBpedia and JSI for various values (equidistant on a logarithmic scale) of their secondary numeric parameter. The numeric labels on the graph indicate the parameter value associated with each curve.

In the case of DBpedia (Figures 5a and 5b), we observe that applying a minimum support threshold has little impact on the precision, but gradually reduces the recall. Around a minimum support value of 10 000, the precision slightly increases, but at the cost of near-zero recall. Similarly, for the highest confidence level, increasing the support threshold slightly increases the precision, which reaches almost 100%, but again leads to a very low recall. A possible explanation for the ineffectiveness of support to increase precision is that the distribution of the support values of computing-specific articles, which are the focus of this study, is too similar to the distribution of more general domain articles to be an effective discriminating factor.

The case of JSI differs. Below 0.001, the minimum pageRank threshold barely affects the precision and recall. However, between 0.1 and 0.001, increasing the pageRank threshold leads to the expected increase in precision, with a corresponding decrease in recall. This suggests that some combination of minimal pageRank and pageRankSqThreshold could allow JSI to achieve precision levels similar to the other tools, without entirely sacrificing recall.

Figure 6 gives a different view of these phenomena by showing the effect of the secondary parameters on the precision-recall curves from Figure 4. To keep the graphs readable, only the most interesting portion of the secondary parameter range is shown, rather than the same range as in Figure 5.

DBpedia's curves strengthen the interpretation from the heatmaps. Increasing the minimum support brings the curve more to the left (i.e., decreases the recall), without considerably lifting it up (i.e., increasing precision). In the case of JSI, increasing the pageRank threshold moves the curve closer to the left and up. This makes JSI a very flexible wikifier, that can adapt to various needs. However, comparing Figure 6b with Figure 4, the gain in precision is generally not worth the loss in recall, as other tools can achieve higher recall for similar levels of precision.

Result: A higher threshold for DBpedia's *support* parameter does not increase precision in most cases. We suggest keeping this value to 0.

Result: Using a minimum *pageRank* threshold for JSI, in addition to *pageRankSqThreshold*, makes JSI a very flexible wikifier. However, when optimized for precision (non zero *pageRank* threshold), JSI is still less precise than other wikifiers. We suggest either using JSI with a *pageRank* minimum of 0, or other wikifiers.

Babelfy (MCS on or off): Because Babelfy's MCS option is a fallback strategy that links unmatched mentions to their most common sense, they do not affect results previously matched by the main algorithm. Furthermore, because MCS matches do not have an associated confidence score, it is impossible to filter among them: a user must either take all or them, or none.

With MCS disabled, and with the lowest value for the confidence threshold, Babelfy identified 2842 articles for the 500 posts, with a precision of 40% and a recall of 37%. Enabling MCS adds an additional 2599 articles,[17] nearly doubling the number of articles. However, the precision of the MCS articles is only 5.9%. This means that the large decrease in precision (40% to 23%) is balanced by only a small increase in recall (37% to 42%). Furthermore, with MCS enabled, increasing the confidence threshold will actually decrease precision, which will converge towards the precision of only the MCS matches, 5.9% (with a confidence threshold of 1, almost all articles are identified by the MCS strategy). Therefore, in most cases, the slight increase from the MCS strategy is not worth the loss in recall.

Result: Babelfy's MCS option has very low precision. Because there is no way to distinguish between the MCS matches, we suggest not to use this option.

Ambiverse (Stanford or KnowNER recognition): Ambiverse includes two components to perform entity recognition: a Stanford NER tool and the KnowNER tool by the Ambiverse authors. We observed that KnowNER leads to a better performance, but the difference is minimal: for any given recall value, the precision with KnowNER is between 0.3 and 2.6 percentage points above that of Stanford NER. The superiority of KnowNER is consistent over the range of recall. Both components achieve a maximum recall just under 47%.

Result: Ambiverse's KnowNER component performs slightly, but consistently, better than the Stanford NER alternative.

WAT (OpenNLP or Lucene tokenizer): WAT offers a choice between two components for the tokenization step: OpenNLP, which is the default option, and Lucene, which is described as better suited for ill-formed text. Figure 7 shows

[17] An additional 15 previously identified articles become MCS matches, due to unpredictable factors of the wikification algorithms.

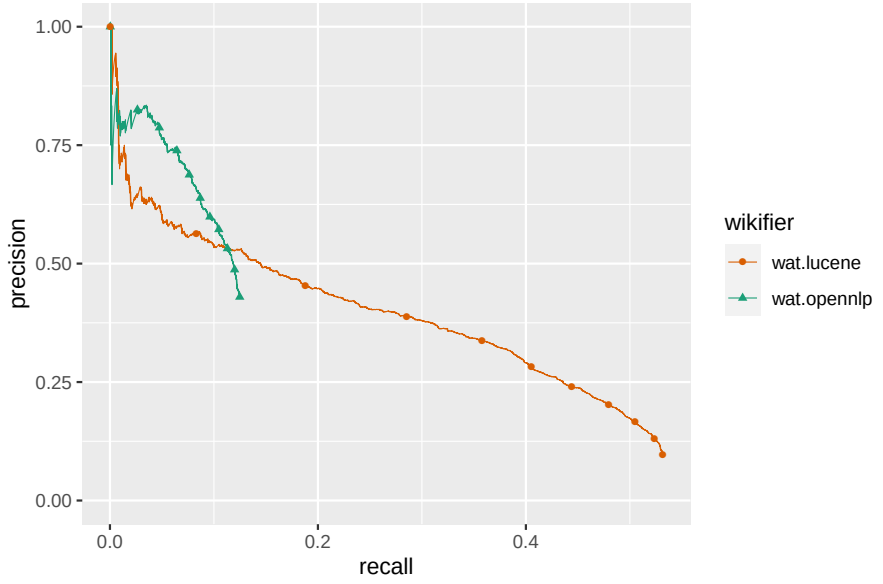


Fig. 7: Precision-Recall curves for WAT with two different tokenizers: Lucene and OpenNLP

the precision-recall curves of both tokenizers. Contrary to the choice of NER component for Ambiverse, the choice of tokenizer has a noticeable impact on the peformance of WAT. Lucene allows WAT to achieve much greater recall, but quickly drops in precision, whereas OpenNLP generally retains a higher precision, but at a much lower recall. Therefore, depending on which of the precision or recall is preferred, both tokenizers can be relevant. However, when compared to other wikifiers, Lucene's gain in recall is not worth its loss in precision. Therefore, in most cases, either OpenNLP is more appropriate, or another wikifier is.

Result: WAT's OpenNLP tokenizer achieves higher precision, whereas the Lucene tokenizer leads to higher recall. However, the gain in recall is less impressive when compared to other wikifiers. We suggest using WAT with OpenNLP, or other wikifiers.

5.3 Correlation Between Wikifiers

Although all six wikifiers claim to solve the same general task, wikification, it is not clear whether all approaches converge towards the same results, or the results of one can cover the blind spots of another. Understanding such correlation between the results of wikifiers can help to improve each individual

Table 5: Actual and expected overlap between the correct results of each wikifier. The expected overlap (in parentheses) assumes a uniform random sampling.

Ambiverse	**Babelfy**	**DBpedia**	**Illinois**	**JSI**	**WAT**	
1423	703 (533)	586 (605)	353 (370)	965 (867)	206 (179)	**Ambiverse**
–	1123	596 (477)	432 (292)	824 (685)	252 (177)	**Babelfy**
–	–	1274	475 (332)	898 (777)	261 (161)	**DBpedia**
–	–	–	780	529 (475)	229 (98)	**Illinois**
–	–	–	–	1827	324 (230)	**JSI**
–	–	–	–	–	378	**WAT**

Table 6: Correlation between the wikifiers results. The correlation is expressed as Kendall's τ_b statistics (Kendall, 1938) computed over the overlap of results of each pair of wikifers (see Table 5). The confidence interval at the 0.95 level is shown in parentheses.

Babelfy	**DBpedia**	**Illinois**	**JSI**	**WAT**	
0.43 (±0.05)	0.37 (±0.05)	0.44 (±0.06)	0.19 (±0.04)	-0.21 (±0.10)	**Ambiverse**
–	0.33 (±0.06)	0.36 (±0.07)	0.22 (±0.05)	-0.07 (±0.09)	**Babelfy**
–	–	0.04 (±0.07)	0.36 (±0.04)	0.01 (±0.09)	**DBpedia**
–	–	–	0.09 (±0.06)	0.02 (±0.09)	**Illinois**
–	–	–	–	-0.06 (±0.07)	**JSI**

approach, as well as develop ensemble methods to mitigate the blind spots of different wikifiers.

Table 5 presents, for each pair of wikifiers, the number of articles correctly identified by both wikifiers (with a confidence threshold of 0), as well as the *expected* number of articles, in parentheses. The expected value assumes that all wikifiers select independently and uniformly randomly articles from the set of correct matches, keeping the recall constant. For example, Ambiverse correctly identified 1423 of the 2997 articles (recall = 47.48%), and Babelfy correctly identified 1123 articles (recall = 37.47%). Therefore, assuming independence, the number of articles identified by both Ambiverse and Babelfy should be $47.48\% \times 37.47\% \times 2997 = 533$.

Typically, one could expect the actual overlap between wikifier results to be greater than the expected overlap under the assumption of independence, because they attempt to solve the same task. However, Ambiverse with both DBpedia and Illinois actually has a smaller overlap than expected. This could suggest that Ambiverse's results complement those of DBpedia and Illinois, and that combining the ideas from Ambiverse and Illinois or DBpedia into a mixed approach would result in a higher improvement than combining other approaches. For other wikifiers, with an overlap larger than the expected value, a simple ensemble approach, such as majority voting, can improve individual performances.

To further understand the correlation between wikifiers, Table 6 shows Kendall's τ_b statistic between each pair of wikifiers, computed over the overlap between the results of the two wikifiers.[18] Kendall's τ coefficient is a non parametric rank correlation statistic that estimates the probability that the two wikifiers will agree on which of two random articles have a higher confidence score (Kendall, 1938). The probability value is rescaled from $[0, 1]$ to the $[-1, 1]$ range to produce a correlation score. Therefore, $\tau = 0$ indicates an agreement probability of 50% (no correlation), $\tau = -1$ indicates a 0% probability of agreement, or 100% of disagreement (perfect negative correlation), and $\tau = 0.43$, as for Ambiverse and Babelfy, indicates a probability of 72% of agreement.[19]

Table 6 provides a more detailed insight into the complementarity of different wikifiers. WAT shows a peculiar behavior, with correlation scores hovering around zero for most wikifiers, except for Ambiverse, where the correlation is negative. This surprising result indicates that among articles correctly identified by both Ambiverse and WAT, those with a high confidence Ambiverse score tend to have a low WAT score, and vice versa.

Apart from WAT, almost all pairs of wikifiers have a significant positive correlation, ranging from 0.19 to 0.44. The only exception is Illinois, with DBpedia and JSI, which has a near-zero correlation. In particular, Ambiverse with DBpedia and Illinois show a strong positive correlation, despite having a smaller than expected overlap. These results are encouraging, as they suggest that different wikifiers (with the exception of WAT) actually have a similar objective. Therefore, choosing one over another is less likely to introduce unwanted biases.

Result: The confidence score used by all wikifiers is positively correlated, with the notable exception of WAT. Confidence scores between Ambiverse and WAT are even negatively correlated, which is a surprising result.

5.4 Validated List of Computing Concepts

The list of validated computing-related Wikipedia titles, obtained as a result of the annotation task, is an important contribution of this study. Because we generated the list using six state-of-the-art wikifiers configured to optimize recall, with a representative sample of a large programming forum, it represents the current range of Wikipedia articles on computing and can serve as a basis to precisely identify the full extent of computing articles.

[18] Instead of restricting the correlation to the overlap, another possible approach would have been to use all articles, and assign a confidence of 0 to articles not found by wikifiers. This approach, however, is sensible to noise due to differences in the knowledge bases (and their version) used to train the wikifiers. Using this approach, we observed correlation scores all near zero. Therefore, we present the more useful results using only the overlaps.

[19] The τ_b variant of the statistics is explicitly tuned to account for ties, which is especially important for the discretized confidence scores of DBpedia's results.

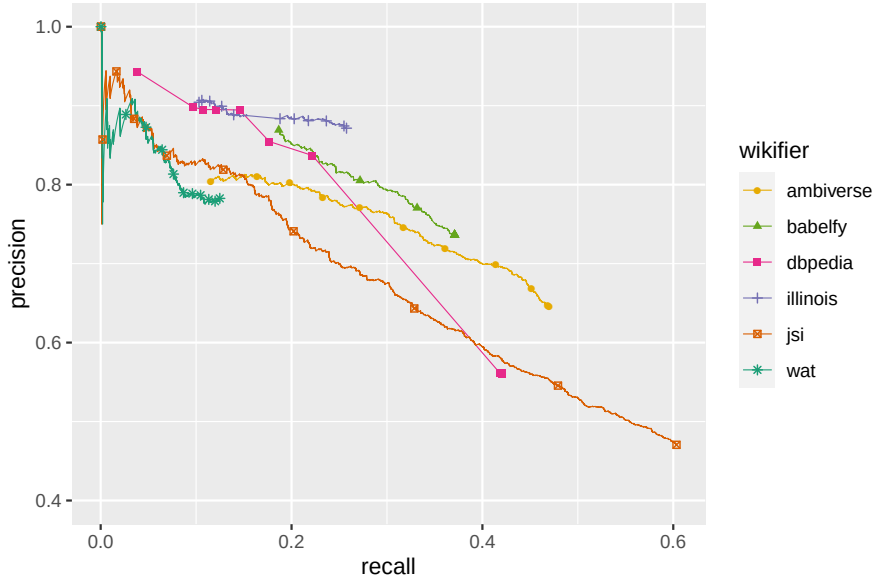


Fig. 8: Precision-Recall curves of all six wikifiers, using the whitelist strategy to improve precision. Note that the y axis differ from Figure 4, to improve readability.

We studied two approaches to use this list as a filter to improve the output of wikifiers. The first approach uses the validated computing titles as a whitelist, and rejects other articles. The second approach instead uses negative examples (i.e., articles marked as "incorrect" by the annotators) as a blacklist of articles to reject. To allow others to use the same strategies to improve the performance of wikifiers, we distribute the complete annotated dataset in our on-line appendix.[M]

Computing Titles as a Whitelist: The first strategy is to remove from the output of wikifiers all articles not on a whitelist, i.e., the validated list of computing-related articles. In the ideal case, where the whitelist contains all computing articles of Wikipedia, this strategy will increase precision with no impact on recall, because it filters out only non-computing, thus incorrect, articles.

To evaluate this strategy, we used the list of 1098 articles to filter the output of all six wikifiers. Figure 8 shows the result of this strategy on the precision-recall curves. The whitelist considerably augments the precision. For example, for a recall threshold of 20%, the precision of all wikifiers goes from less than 50% to over 74% (except for WAT, which does not achieve such recall). Nevertheless, there is still a non trivial proportion of computing-related false positives, especially for low confidence values (e.g., to achieve 60% recall,

JSI's precision drop to 47%). For example, although Source code is often correctly linked to the mentions *source* and *code*, it can be linked to misleading terms, for example when *source* indicates the provenance of information in a post.[T]

A limitation of this approach is the need for an extensive whitelist. Although our list of 1098 articles is a good start, it may not contain articles about rarer computing concepts or technologies. For example, we took a random sample of 30 articles marked as "incorrect" only once, and found four computing articles among them: Message authentication code (matched to HTML codes), 32-bit (matched to the number 32), Arithmetic overflow (matched to HTML content overflow), and Graphics Environment Manager (matched to Ruby's `gem` command). To avoid the possibility of rejecting such articles, users can combine two configurations (or two wikifiers): one optimized for recall that uses the whitelist, and the other optimized for precision which does not use the whitelist.

Result: Using the list of articles as a whitelist showed a precision improvement, reaching values between 74% and 89%. However, computing-related false positives are still present in non-negligible ratios for configurations that optimize recall.

Negative Examples as a Blacklist: To circumvent the limitations of the whitelist strategy, another strategy is to only reject articles from an explicit blacklist. An extensive list of all non-computing articles would be equivalent to an extensive whitelist. However, the blacklist can focus on articles that are especially problematic for a wikifier, i.e., articles that are most often false positives. The objective of this strategy is to maximize the increase of precision by excluding common irrelevant terms (e.g., Canning, which is often matched to the modal auxiliary verb *can*) and artifacts of the training phase (e.g., Burmese language[20] and On Your Toes[21]).

In addition to the list of 1098 computing articles, the annotation task generated a set of 10 854 distinct articles marked as "incorrect" at least once. Articles that were rejected the most often (e.g., 781 articles were rejected at least ten times) can serve as a basis to create the blacklist. However, even some of these most often rejected articles are related to computing. For example, although This (computer programming) is a computing concept, WAT with the Lucene tokenizer often associates it with the demonstrative pronoun *this*, creating many false positives. Including these articles in the blacklist can improve precision at the cost of known blind spots, but this tradeoff must be tailored to each wikifier (e.g., associating This (computer programming) is not a common issue for other wikifiers than WAT).

[20] The ISO two-letter code for the Burmese language is "my", an homograph of the possessive determiner, which appears in many posts.

[21] This musical is often linked to the word *your*, an obvious incorrect artifact from the training phase.

Result: A blacklist does not prevent rarer articles from being detected. It optimizes the gain in precision by targeting the weaknesses of wikifiers. However, some computing articles are often confounded with other terms, and thus often marked as "incorrect". With a careful configuration, the blacklist strategy can optimize the gain in precision at the cost of known blind spots for such articles.

5.5 Threats to Validity

Internal Validity: The dynamic nature of Wikipedia affects the internal validity of the results. Because Wikipedia is in constant evolution, identifying an article by title is ambiguous, as it can refer to multiple versions. This evolution impacts wikifiers trained with older versions of Wikipedia, or with other knowledge bases which themselves rely on old Wikipedia versions. In addition to being oblivious to more recent articles, these wikifiers may refer to articles that were significantly changed. As an example, the article Database management system was merged in 2013 with the article Database.[U] Therefore, a wikifier trained on a Wikipedia version older than 2013 will consider Database and Database management system as two different articles, whereas more recent wikifiers will only consider them as one. This divergence, however, is inevitable, as researchers who may wish to use a wikifier without having to re-train it will be similarly affected.

By the same argument, the evolution of Stack Overflow posts between the wikification and the completion of the annotation task causes another threat. To mitigate it, all data used in this study was only a few months old, greatly reducing the threat of divergence.

External Validity: Threats to external validity depend on the level of generalization considered. The first level of generalization is from the sample of 500 posts to all Stack Overflow posts. The simple random sample of posts is sufficient to support a statistical generalization of proportions *computed over posts* to the whole population within an error of 0.05 with 95% confidence. However, most of the results of this study are proportions *computed over article–post pairs*. This situation benefits us, as proportions of pairs represent a much larger number of annotations (41 124 article–post pairs in total), but statistical generalization from these proportions is more complex, because the sample of pairs is not random. Despite the lack of statistical generalization, the unbiased sampling of posts and the absence of evident imbalance in the wikifications (e.g., there was not a small subset of posts that received all correct wikifications) give sufficient confidence in the representativeness of the findings. Furthermore, the objective of this study is not to identify the best performing wikifier, but to understand their relative performance to help potential users make informed decision on which wikifier to use. In such context, small performance differences, statistically significant or not, should not influence the choice of a wikifier the way larger differences would.

We performed a sensitivity analysis to further assess the threat of spurious results. We generated 300 trimmed samples by removing 5%, 10%, and 20% of the 500 posts, 100 times for each proportion. For each trimmed sample, we computed the precision and recall of all wikifiers, for 21 confidence thresholds ranging from 0 to 1, in increments of 0.05. We then compared those precision and recall values to the precision and recall computed on the full sample, for the same confidence thresholds. For all six wikifiers, the maximum variation in recall, for all three proportions, was 1.8%. In terms of precision, the maximum variation was of 4.2% for Ambiverse, Babelfy, DBpedia, and Illinois for all three proportions. We observed larger variations in precision for JSI and WAT when the confidence threshold was near 1, which can be expected, as very few results are returned (and thus, the effect of one additional true or false positive is much larger). However, for lower thresholds (0.70 or less for JSI and 0.95 or less for WAT), the variation is similar to that of the other tools (below 7.5%). Considering that these values represent the *maximum variation* over 300 independent perturbations of up to 20% of the sample, the small observed variations suggest that the results can reliably be generalized to the population of Stack Overflow posts within a small margin of error.

The study design supports an analytic generalization from the sampled population, i.e., Stack Overflow posts, to the concept of a "software resource". In contrast to other types of documents, a software resource contains a mix of natural language and code, either in blocks or embedded in the main text, software-specific jargon, and references to a fast growing list of technologies. The considerable variance of posts in a number of dimensions, such as the formality and quality of the language, the length of a post, the sub-area of computing, and the intent of the post (e.g., question, description, explanation, step-by-step guide), allows this analytic generalization. Nevertheless, because this study is the first to focus on the wikification of software resources, further replication studies with resources from other sources are required to strenghten this generalization. In particular, Stack Overflow posts exhibit a number of idiosyncrasies. For example, posts edited to add content often include the new content under an "Edit" header, and questions sometimes end with a "Thank you", which wikifiers would link to the Wikipedia articles Editing and Gratitude, respectively. Although such phrases are unlikely to appear in other types of software resources, it is reasonable to assume that any source has its own idiosyncrasies, and wikifiers should be able to ignore them.

Construct Validity: Construct validity concerns the relation between the metrics used (precision and recall) and the constructs under study (wikifier performance). The two metrics are commonly used in software engineering research, and both have an intuitive interpretation that allows the reader to understand what exactly is measured (as opposed to, e.g., F1 score). The more important threat to construct validity comes from the formulation of the annotation task. The details of the task, discussed in Section 4.1, may diverge from others' interpretation of wikification, especially regarding the rejection of relevant, but

non-computing articles. This decision was however necessary to mitigate the subjectivity of the task and the influence of irrelevant results.

Conclusion Validity: The findings of this study suggest different strategies to use wikifiers. These suggestions are backed by empirical evidence, but there are many other factors that can influence the choice of a wikification procedure, such as the time budget, the number of documents to wikify, and the amount of human effort that can be devoted to the task (e.g., to experiment between different configurations or manually validate some of the output). To mitigate this threat, we discussed findings from all wikifiers, not just the best performing ones.

6 Related Work

The wikification task finds its roots in two related sub-areas of computational linguistics: *named entity recognition* (NER) and *named entity disambiguation* (NED). The recognition task aims at identifying, from a natural language text, mentions of named entities. The disambiguation task aims at identifying, for a given mention in a natural language document, the correct sense of this term, usually by associating the term with an entry of a knowledge base. Named entities are real life entities with a proper noun, such as people, organizations, and geographical locations.

The two tasks, NER and NED, are sometimes jointly referred to as *named entity recognition and disambiguation* (NERD) or *entity linking* (EL). When considering any term, not just named entities, the disambiguation task is also called *word sense disambiguation* (WSD). The equivalent of NERD for any term is called *concept linking* (CL), which thus subsumes the NERD problem. For any of the disambiguation tasks, different knowledge bases can be used, such as WordNet (Fellbaum, 1998), DBpedia (Lehmann et al., 2015), and YAGO (Rebele et al., 2016). Hence, *wikification* is simply concept linking with Wikipedia as the knowledge base.

A large number of different techniques exist to solve any of these tasks. The popularity of concept linking is in part due to several challenges and shared tasks developed by the community, such as those of the Message Understanding Conferences (see, for example, Sundheim, 1995), Senseval/SemEval (e.g., Mihalcea et al., 2004; Navigli et al., 2013), and CoNLL (e.g., Tjong Kim Sang and De Meulder, 2003). Lists of concept linking techniques can be found in the proceedings of any of the shared tasks. Shen et al. (2015) present a synthesized summary of the state of concept and entity linking (as of 2015), in terms of techniques, evaluation, and applications. Although not specifically focusing on concept linking, Xun et al. (2017) discuss many of the techniques underlying concept linking. More recently, Szymański and Naruszewicz (2019) surveyed wikification techniques specifically. However, to the best of our knowledge, no prior work surveys or independently evaluates the problem of concept linking specifically in the context of software engineering.

The large amount of concept or entity linking techniques, each described with inconsistent terms, led to a community effort to generate a new, standard terminology to clearly discuss each variant of the concept linking task (Usbeck et al., 2015). According to Usbeck et al.'s terminology, the exact task evaluated in this study is *Concepts to Knowledge Base* (C2KB), or more precisely *Concepts to Wikipedia* (C2W). However, all wikifiers from this study actually solve the *Annotation to Knowledge Base* (A2KB) task, which subsumes C2KB. The outcome of this community effort also includes a framework, GERBIL, to evaluate concept linking techniques with consistent methodology and metrics. In the present study, however, we could not use GERBIL to compare wikifiers, because it requires a gold standard as input to the evaluation procedure.

In the software engineering community, prior work has addressed some of the specific issues related to natural language processing for software resources. To leverage the information contained in identifiers, Carvalho et al. (2015) developed an approach to expand and tokenize source code identifiers that goes beyond the trivial *camelCase* and *snake_case*-based splitting. Chen et al. (2017) present an approach to aggregate the many morphological forms software entities can take (e.g., the authors give the example of the *Microsoft Visual C++* IDE, which is commonly referred to by *vc++*, *msvc*, *ms vc++*, and other forms). Closer to an eventual software-specific concept linking tool, Ye et al. (2016a) developed S-NER, a named entity recognition technique specifically tailored for software entities. We are not aware, however, of an end-to-end approach for concept linking of software-related concepts.

Finally, the wikification task, which links mentions of concepts to Wikipedia articles, can be construed as a *traceability problem*. Traceability has been extensively studied in software engineering, to explore how different types of software artifacts can be linked (Cleland-Huang et al., 2014). Our study explores the use of state-of-the-art wikifiers for this purpose, so that future work can build on a mature foundation to link software artifacts to knowledge base entries, or leverage ideas from the wikification domain to improve other traceability recovery approaches, such as resolving mentions of API elements in natural language documents (Rigby and Robillard, 2013; Ye et al., 2016b, 2018; Ma et al., 2019). These works, however, focus on named entities (e.g., API elements or assets of a software project), which are typically better defined than abstract concepts. Because wikifiers can identify both kinds of entities, their use can complement other software-specific information retrieval techniques.

7 Conclusion

Software engineering techniques leverage wikification to automatically embed documents into the rich semantic space defined by Wikipedia articles. However, although research on wikification techniques continues to grow, there is little evidence on the ability of state-of-the-art wikifiers to overcome issues specific to software resources, such as the quickly expanding software jargon, which includes many terms that have another generic sense (e.g., the *Python* language,

the *Spring* framework), and the presence of code integrated directly into documents. Furthermore, the number of available wikifiers, each with different configuration parameters, makes it harder to choose an optimal wikification procedure.

To address these issues and encourage software engineering researchers to use available wikifiers, we conducted an empirical comparison of six wikifiers that can wikify custom text inputs. The comparison uses a random sample of 500 Stack Overflow posts as input for the wikifiers, and we manually annotated the 41 124 Wikipedia articles identified for each post as correct or incorrect. The results of this comparison confirm that it is challenging to select an optimal wikifier: no wikifier outperforms another wikifier completely, and none of them perform as well as reported in their associated research articles. However, it is possible to achieve decent wikification precision and recall on software documents. In addition to the empirical evidence to support the choice of wikifier and configuration, we also contribute suggestions to improve the performance of wikifiers for software resources, including by leveraging the manually validated list of 1098 Wikipedia titles related to computing obtained as a result of the annotation task.

Acknowledgements We are grateful to the external annotators for helping with the manual annotation of the wikifiers output. This work is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

A List of Links to External Resources

A. https://stackoverflow.com/questions
B. https://en.wikipedia.org/wiki/Main_Page
C. https://www.reddit.com/
D. https://en.wikipedia.org/wiki/Software_system
E. https://archive.org/details/stackexchange
F. https://jsoup.org/
G. https://github.com/ambiverse-nlu/ambiverse-nlu
H. http://babelfy.org/
I. https://www.dbpedia-spotlight.org/
J. https://cogcomp.seas.upenn.edu/page/software_view/Wikifier
K. http://wikifier.org/
L. https://services.d4science.org/web/tagme/wat-api
M. https://doi.org/10.5281/zenodo.3727035
N. https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking
O. https://stackoverflow.com/questions/2348415
P. https://stackoverflow.com/questions/55228245
Q. https://stackoverflow.com/questions/41998618
R. https://stackoverflow.com/questions/55893389
S. https://stackoverflow.com/questions/21646135
T. https://stackoverflow.com/questions/16516936
U. https://en.wikipedia.org/w/index.php?title=Database_management_system&diff=544579037&oldid=544577010

References

Barua A, Thomas SW, Hassan AE (2014) What are developers talking about? an analysis of topics and trends in stack overflow. Empirical Software Engineering 19(3):619–654

Bourque P, Fairley RE (2014) Guide to the Software Engineering Body of Knowledge, 3rd edn. IEEE Computer Society Press, URL `www.swevok.org`

Brank J, Leban G, Grobelnik M (2017) Annotating documents with relevant wikipedia concepts. In: Proceedings of the Slovenian Conference on Data Mining and Data Warehouses, p 4 p.

Carvalho NR, Almeida JJ, Henriques PR, Varanda MJ (2015) From source code identifiers to natural language terms. Journal of Systems and Software 100:117–128

Cassidy T, Ji H, Ratinov LA, Zubiaga A, Huang H (2012) Analysis and enhancement of wikification for microblogs with context expansion. In: Proceedings of the 24th International Conference on Computational Linguistics, pp 441–456

Chen C, Xing Z, Wang X (2017) Unsupervised software-specific morphological forms inference from informal discussions. In: Proceedings of the 39th International Conference on Software Engineering, p 450–461

Chen C, Xing Z, Liu Y (2018) What's Spain's Paris? Mining analogical libraries from Q&A discussions. Empirical Software Engineering 24(3):1155–1194

Cheng X, Roth D (2013) Relational inference for wikification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp 1787–1796

Cleland-Huang J, Gotel OCZ, Huffman Hayes J, Mäder P, Zisman A (2014) Software traceability: Trends and future directions. In: Proceedings of the on Future of Software Engineering, p 55–69

Cohen J (1960) A coefficient of agreement for nominal scales. Educational and psychological measurement 20(1):37–46

Cornolti M, Ferragina P, Ciaramita M (2013) A framework for benchmarking entity-annotation systems. In: Proceedings of the 22nd International Conference on World Wide Web, pp 249–260

Daiber J, Jakob M, Hokamp C, Mendes PN (2013) Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems, pp 121–124

Fellbaum C (1998) WordNet: An electronic lexical database. MIT press

Ferragina P, Scaiella U (2010) TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp 1625–1628

Hoffart J, Yosef MA, Bordino I, Fürstenau H, Pinkal M, Spaniol M, Taneva B, Thater S, Weikum G (2011) Robust disambiguation of named entities in text. In: Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing, pp 782–792

ISO/IEC/IEEE (2017) International standard – Systems and software engineering – Vocabulary. Standard 24765:2017, ISO/IEC/IEEE

Kendall MG (1938) A new measure of rank correlation. Biometrika 30(1/2):81–93

Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174

Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, van Kleef P, Auer S, Bizer C (2015) Dbpedia – a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6(2):167–195

Ma S, Xing Z, Chen C, Chen C, Qu L, Li G (2019) Easy-to-deploy api extraction by multi-level feature embedding and transfer learning. IEEE Transactions on Software Engineering p 15 pages, to appear

Meij E, Weerkamp W, de Rijke M (2012) Adding semantics to microblog posts. In: Proceedings of the 5th ACM International Conference on Web Search and Data Mining, pp 563–572

Mendes PN, Jakob M, Garcia-Silva A, Bizer C (2011) DBpedia Spotlight: Shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems, pp 1–8

Mihalcea R, Chklovski T, Kilgarriff A (2004) The senseval-3 English lexical sample task. In: Proceedings of the third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, pp 25–28

Milne D, Witten IH (2008) Learning to link with wikipedia. In: Proceedings of the 17th ACM conference on Information and Knowledge Management, pp 509–518

Milne D, Witten IH (2013) An open-source toolkit for mining wikipedia. Artificial Intelligence 194:222–239

Moro A, Raganato A, Navigli R (2014) Entity linking meets word sense disambiguation: a unified approach. Transactions of the Association for Computational Linguistics 2:231–244

Nassif M, Treude C, Robillard MP (2020) Automatically categorizing software technologies. IEEE Transactions on Software Engineering 46(1):20–32

Navigli R, Ponzetto SP (2012) BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence 193:217–250

Navigli R, Jurgens D, Vannella D (2013) SemEval-2013 task 12: Multilingual word sense disambiguation. In: Second Joint Conference on Lexical and Computational Semantics, Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation, pp 222–231

Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab

Patil S (2017) Concept-based classification of software defect reports. In: Proceedings of the 14th International Conference on Mining Software Repositories, p 182–186

Piccinno F, Ferragina P (2014) From TagME to WAT: a new entity annotator. In: Proceedings of the first International Workshop on Entity Recognition & Disambiguation, pp 55–62

Ponzanelli L, Bacchelli A, Lanza M (2013) Seahawk: Stack overflow in the ide. In: Proceedings of the 35th International Conference on Software Engineering, pp 1295–1298

Ratinov L, Roth D, Downey D, Anderson M (2011) Local and global algorithms for disambiguation to wikipedia. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1, pp 1375–1384

Rebele T, Suchanek F, Hoffart J, Biega J, Kuzey E, Weikum G (2016) YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In: Proceedings of the International Semantic Web Conference, pp 177–185

Rigby PC, Robillard MP (2013) Discovering essential code elements in informal documentation. In: Proceedings of the 35th IEEE/ACM International Conference on Software Engineering, pp 832–841

Schindler M, Fox O, Rausch A (2015) Clustering source code elements by semantic similarity using wikipedia. In: Proceedings of the Fourth International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, p 13–18

Seyler D, Dembelova T, Del Corro L, Hoffart J, Weikum G (2018) A study of the importance of external knowledge in the named entity recognition task. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp 241–246

Shen W, Wang J, Han J (2015) Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE Transactions on Knowledge and Data Engineering 27(2):443–460

Sundheim BM (1995) Overview of results of the MUC-6 evaluation. In: Proceedings of the 6th Conference on Message Understanding, p 13–31

Szymański J, Naruszewicz M (2019) Review on wikification methods. AI Communications 32(3):235–251

Tjong Kim Sang EF, De Meulder F (2003) Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL, pp 142–147

Treude C, Robillard MP (2016) Augmenting API documentation with insights from stack overflow. In: Proceedings of the IEEE/ACM 38th International Conference on Software Engineering, pp 392–403

Usbeck R, Röder M, Ngonga Ngomo AC, Baron C, Both A, Brümmer M, Ceccarelli D, Cornolti M, Cherix D, Eickmann B, Ferragina P, Lemke C, Moro A, Navigli R, Piccinno

F, Rizzo G, Sack H, Speck R, Troncy R, Waitelonis J, Wesemann L (2015) GERBIL: General entity annotator benchmarking framework. In: Proceedings of the 24th International Conference on World Wide Web, pp 1133–1143

Vincent N, Johnson I, Hecht B (2018) Examining Wikipedia with a broader lens: Quantifying the value of Wikipedia's relationship with other large-scale online communities. In: Proceedings of the CHI Conference on Human Factors in Computing Systems, pp 1–13

Wang C, Peng X, Liu M, Xing Z, Bai X, Xie B, Wang T (2019) A learning-based approach for automatic construction of domain glossary from source code and documentation. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, p 97–108

Wikipedia (2019) Wikipedia:Manual of Style/Linking. URL `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking`, accessed 2020-01-06

Xun G, Jia X, Gopalakrishnan V, Zhang A (2017) A survey on context learning. IEEE Transactions on Knowledge and Data Engineering 29(1):38–56

Ye D, Xing Z, Foo CY, Ang ZQ, Li J, Kapre N (2016a) Software-specific named entity recognition in software engineering social content. In: Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, pp 90–101

Ye D, Xing Z, Foo CY, Li J, Kapre N (2016b) Learning to extract api mentions from informal natural language discussions. In: IEEE International Conference on Software Maintenance and Evolution, pp 389–399

Ye D, Bao L, Xing Z, Lin SW (2018) APIReal: an api recognition and linking approach for online developer forums. Empirical Software Engineering 23(6):3129–3160

Ye X, Shen H, Ma X, Bunescu R, Liu C (2016c) From word embeddings to document similarities for improved information retrieval in software engineering. In: Proceedings of the 38th International Conference on Software Engineering, p 404–415

Zhao X, Xing Z, Kabir MA, Sawada N, Li J, Lin SW (2017) HDSKG: Harvesting domain specific knowledge graph from content of webpages. In: Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, pp 56–67