

An Exploratory Study of Documentation Strategies for Product Features in Popular GitHub Projects



Tim Puhlfürß
@TimPuhl



Lloyd Montgomery
@LloydThinks



Walid Maalej
@maalejw

Research Objective

35 contributors



README.md

Overview

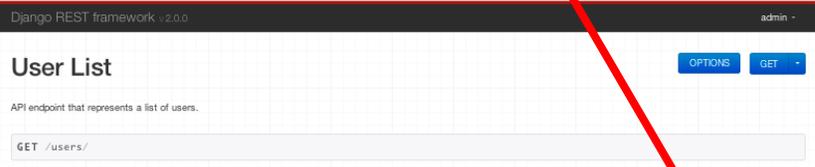
Django REST framework is a powerful and flexible toolkit for building Web APIs.

Some reasons you might want to use REST framework:

- The [Web browsable API](#) is a huge usability win for your developers.
- [Authentication policies](#) including optional packages for [OAuth1a](#) and [OAuth2](#).
- [Serialization](#) that supports both [ORM](#) and [non-ORM](#) data sources.
- Customizable all the way down - just use [regular function-based views](#) if you don't need the [more powerful features](#).
- [Extensive documentation](#), and [great community support](#).

There is a live example API for testing purposes, [available here](#).

Below: Screenshot from the browsable API



(RQ2a)
Connection?

(RQ2b)
Connection?

master | django-rest-framework / rest_framework / authentication.py

puittenbroek #7157: Fix RemoteUserAuthentication calling django authenticate with ...

35 contributors

232 lines (178 sloc) | 7.56 KB

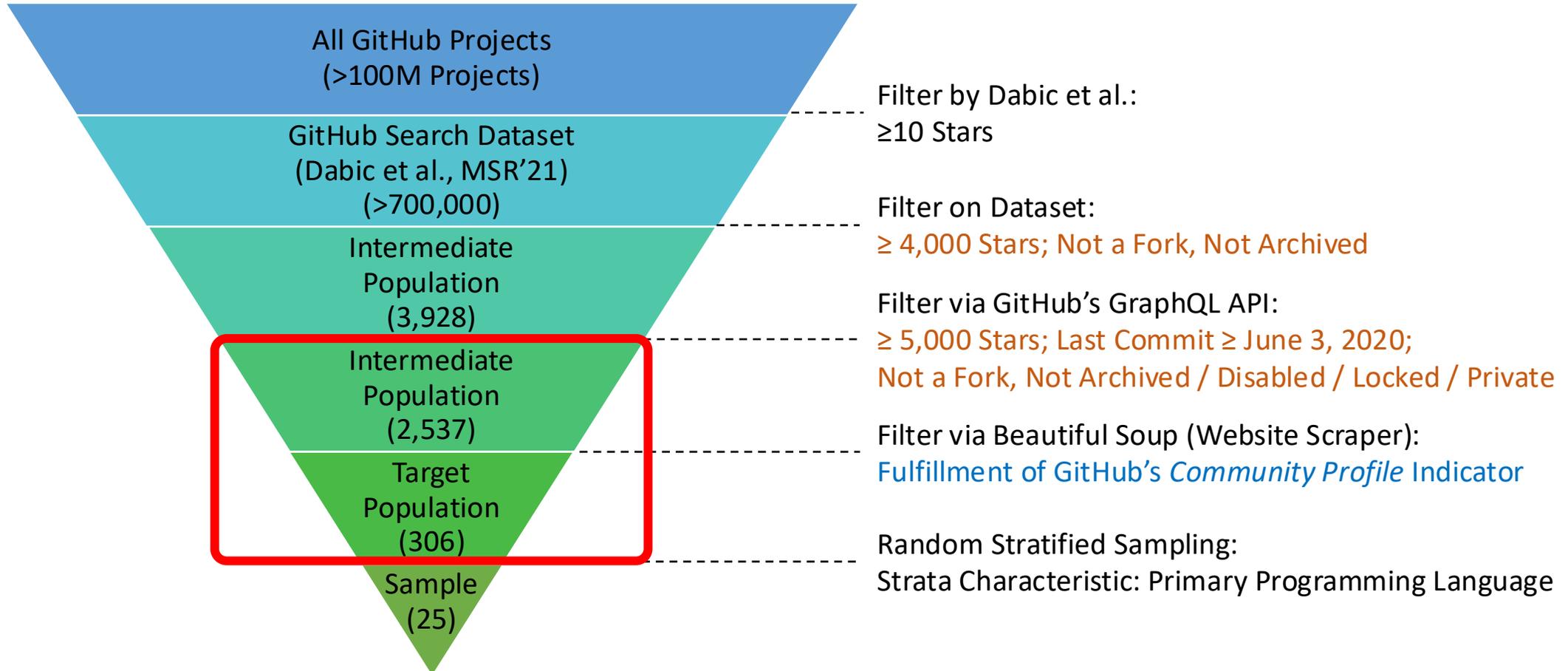
```
1 """
2 Provides various authentication policies.
3 """
4 import base64
5 import binascii
6
7 from django.contrib.auth import authenticate, get_user_model
8 from django.middleware.csrf import CsrfViewMiddleware
9 from django.utils.translation import gettext_lazy as _
10
11 from rest_framework import HTTP_HEADER_ENCODING, exceptions
12
13
14 def get_authorization_header(request):
15     """
16     Return request's 'Authorization:' header, as a bytestring.
17
18     Hide some test client ickyness where the header can be unicode.
19     """
```

↑
Textual artefacts to document product features? (RQ1a)
Descriptive elements? (RQ1b)

Research Method: In-depth Content Analysis of 25 GitHub Projects

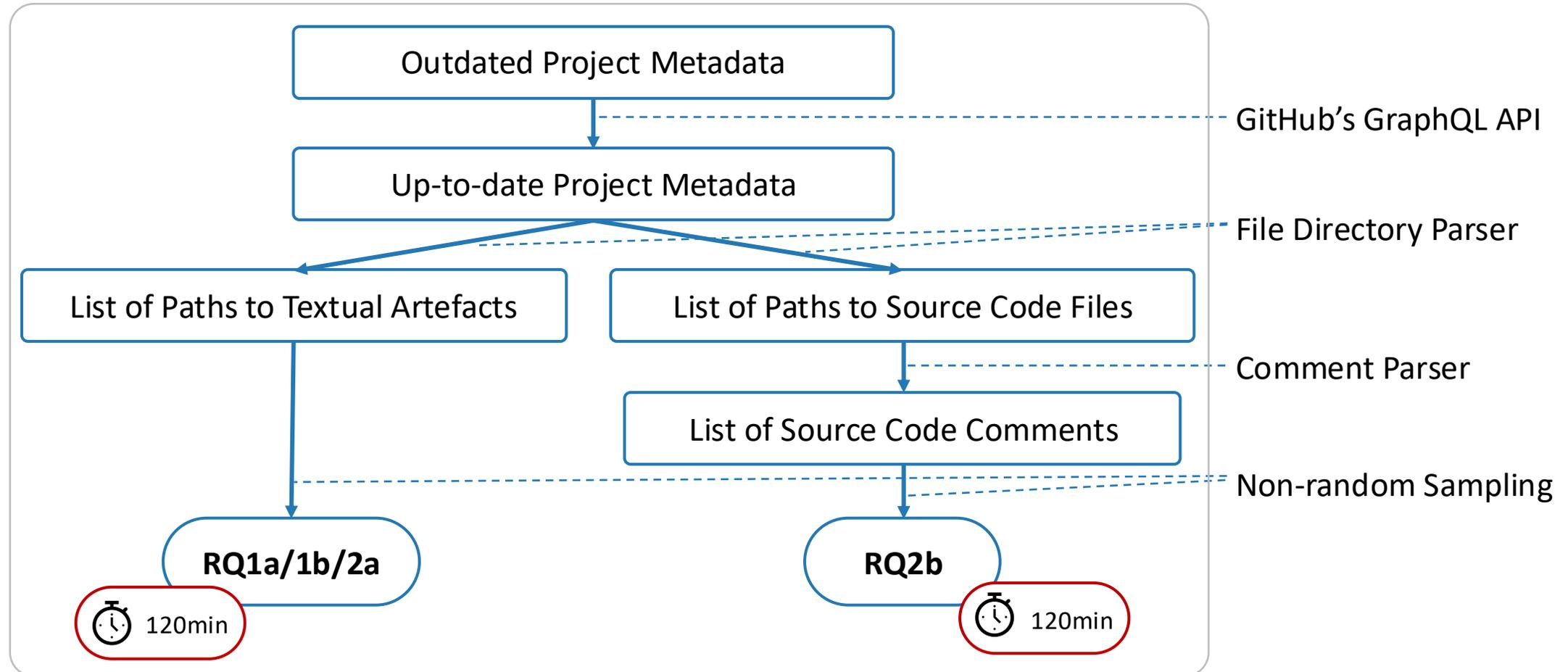
Methodology – Population and Sampling

Scope: **Very Popular Projects** with **Documentation of High Quality**



Methodology – Analysis Approach

Per Project

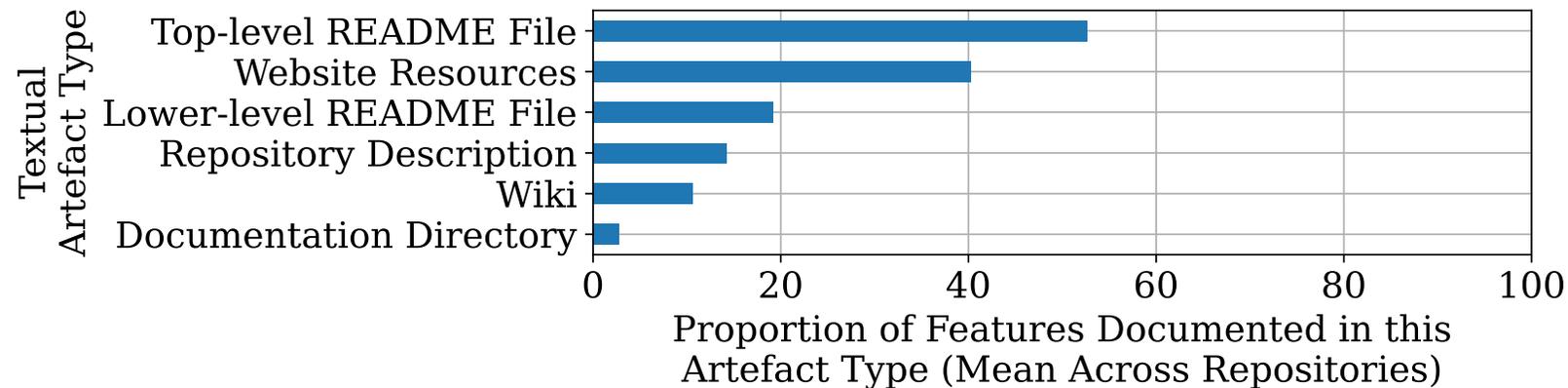


RQ1a: Textual Artefacts Used to Document Product Features

„Features [...]

- Full Navigation Control: Swiper comes with all required built-in navigation elements, such as Pagination, Navigation arrows and Scrollbar“

Top-level README File of *swiper*



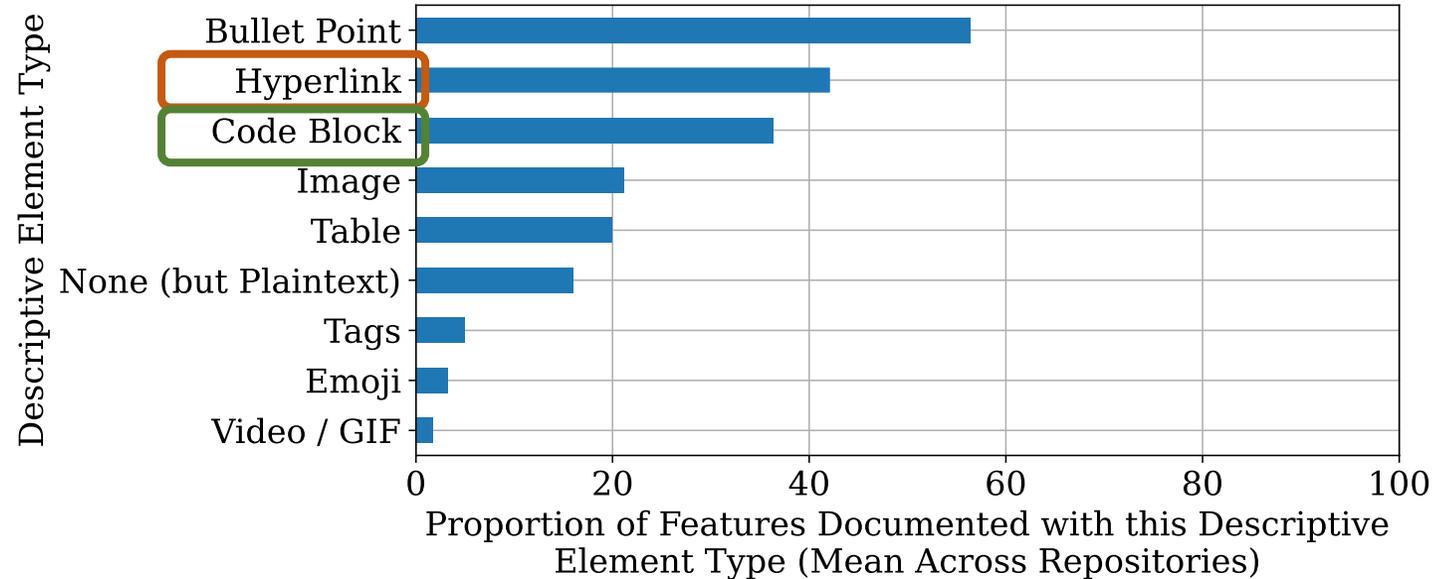
RQ1a: Textual Artefacts Used to Document Product Features

Usage of GitHub-supported documentation means: relatively rare
(in comparison to the usage of website resources)

Recommendation: Use beneficial combinations of textual artefacts.

E.g., top- and lower-level README files

RQ1b: Descriptive Elements Used to Document Product Features



Website Resources of *django-rest-framework*

Schema

A machine-readable [schema] describes what resources are available via the API, what the operations they support.

Heroku, [JSON Schema for the Heroku Platform API](#)

API schemas are a useful tool that allow for a range of use cases, including generating reference libraries that can interact with your API.

Django REST Framework provides support for automatic generation of [OpenAPI](#) schemas.

Generating a dynamic schema with `SchemaView`

If you require a dynamic schema, because foreign key choices depend on other data, you can use `SchemaView` to generate and serve your schema on demand.

To route a `SchemaView`, use the `get_schema_view()` helper.

In `urls.py`:

```
from rest_framework.schemas import get_schema_view

urlpatterns = [
    # ...
    # Use the `get_schema_view()` helper to add a `SchemaView` to
    # * `title` and `description` parameters are passed to `SchemaView`.
    # * Provide view name for use with `reverse()`.
    path('openapi', get_schema_view(
        title="Your Project",
        description="API for all things ...",
        version="1.0.0"
    ), name='openapi-schema'),
    # ...
]
```

RQ1b: Descriptive Elements Used to Document Product Features

Features often “entangled” in unstructured text

Recommendation: Provide a structured presentation of product features / functional requirements.

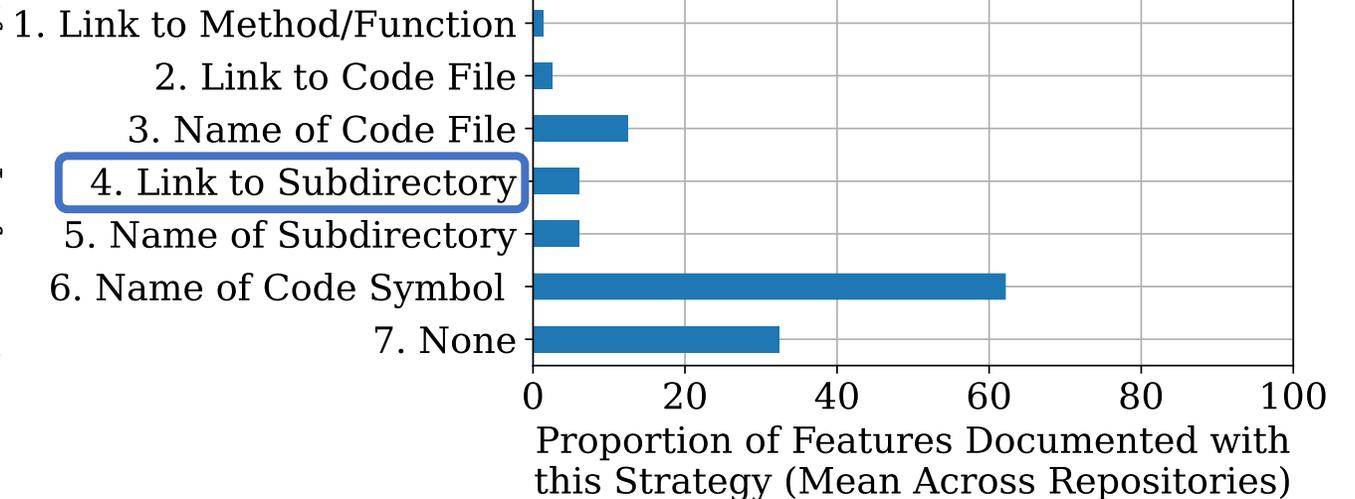
E.g., use tables, bullet points, and emojis to improve structure

RQ2a: Strategies to Link from Textual Artefacts to Source Code

„The code is set up into several main directories: [...]
- **zoo**: contains code to directly download and use pretrained models from our model zoo“

Top-level README file of *ParlAI*

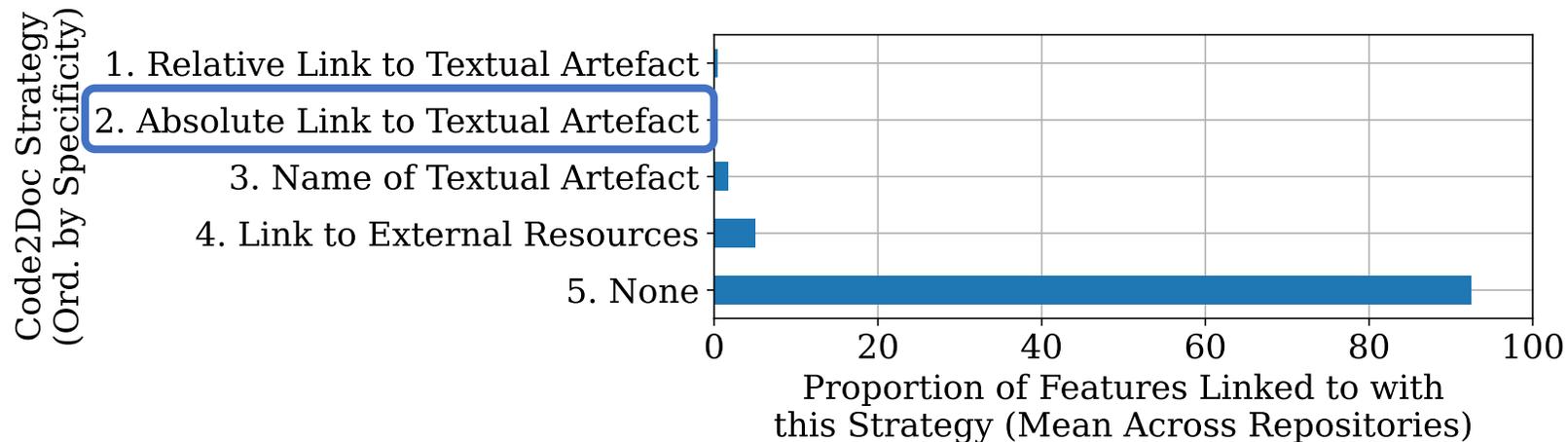
Doc2Code Strategy
(Ord. by Specificity)



RQ2b: Strategies to Link from Source Code to Textual Artefacts

```
„ * Register logs source configuration, used by LogStream components  
* @see https://github.com/elastic/kibana/blob/master/x-pack/plugins/infra/public/components/log\_stream/log\_stream.stories.mdx#with-a-source-configuration“
```

Comment in a code file of *kibana*



RQ2a/b: Strategies to Link between Textual Artefacts and Source Code

Sparse usage of tracing strategies to link textual artefacts and code

Recommendation:

Improve tracing information available in textual artefacts and source code files

- (1) Hyperlinks → direct navigation between files
- (2) Use documentation-focused Tools

An Exploratory Study of Documentation Strategies for Product Features in Popular GitHub Projects

Tim Puhlfürß

@TimPuhl

tim.puhlfuerss@uni-hamburg.de

Lloyd Montgomery

@LloydThinks

lloyd.montgomery@uni-hamburg.de

Walid Maalej

@maalejw

walid.maalej@uni-hamburg.de



Artefacts:

<https://doi.org/10.5281>

[/zenodo.7107674](https://zenodo.org/record/7107674)



Preprint:
[https://doi.org/10.48550/a](https://doi.org/10.48550/arXiv.2208.01317)
[rXiv.2208.01317](https://doi.org/10.48550/arXiv.2208.01317)

