

Searching for Software in NSF Awards

Anonymous Submission

ABSTRACT

Software is an important tool for scholarly work, but software produced for research is in many cases not easily identifiable or discoverable. A potential first step in linking research and software is software identification. In this paper we present two datasets to study the identification and production of research software. The first dataset contains almost 1000 human labeled annotations of software production from National Science Foundation (NSF) awarded research projects. We use this dataset to train models that predict software production. Our second dataset is created by applying the trained predictive models across the abstracts and project outcomes reports for all NSF funded projects between the years of 2010 and 2023. The result is an inferred dataset of software production for over 150,000 NSF awards. We release the NSF-Soft-Search dataset to aid in identifying and understanding research software production: <https://github.com/redacted-for-submission>

CCS CONCEPTS

• **Information systems** → **Digital libraries and archives**; *Clustering and classification*; • **Software and its engineering** → *Software creation and management*; Software libraries and repositories.

KEYWORDS

datasets, text classification, research software

ACM Reference Format:

Anonymous Submission. 2023. Searching for Software in NSF Awards. In *Proceedings of Joint Conference on Digital Libraries (JCDL)*. ACM, New York, NY, USA, 4 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

Software production, use, and reuse is an increasingly crucial part of scholarly work [1, 10]. While historically underutilized, citing and referencing software used during the course of research is becoming common with new standards for software citation [2, 6] and work in extracting software references in existing literature [5]. However, records of software production are not readily identifiable or available at scale in the way that peer-reviewed publications or other scholarly outputs are [8]. To make progress on this problem, we introduce two related datasets for studying and inferring software produced as a part of research, which we refer to as the NSF-Soft-Search dataset.

The NSF-Soft-Search dataset is aimed at identifying research projects which are likely to have produced software while funded

by a federal grant. We start by identifying GitHub repositories that acknowledge funding from at least one National Science Foundation (NSF) award. We then annotate each GitHub repository found with a binary decision for its contents: software or not-software (e.g. not all github repositories contain software, they might include research notes, course materials, etc.). We then link each annotated GitHub repository to the specific NSF award ID(s) referenced in its README.md file. Finally, we compile the NSF-Soft-Search Training dataset using the annotations for each GitHub repository, and the text from the linked NSF award abstract and the project outcomes report.

Using the NSF-Soft-Search Training dataset, we train a variety of models to predict software production using either the NSF award abstract or project outcomes report text as input. We use the best performing models to then infer software production against all awards funded by the National Science Foundation from 2010 to 2023 (additional details are offered in Section 2). The predictions and metadata for each NSF award between the 2010 and 2023 period are compiled to form the NSF-Soft-Search Inferred dataset.

In total, our new NSF-Soft-Search dataset includes the following contributions:

1. NSF-Soft-Search Training: A ground truth dataset compiled using linked NSF awards and GitHub repositories which have been annotated for software production.
2. Multiple classifiers which infer software production from either the text of an NSF award’s abstract or project outcomes report.
3. NSF-Soft-Search Inferred: A dataset of more than 150,000 NSF funded awards from between 2010 and 2023. Each award has two predictions for software production: one from prediction using the abstract text and the other from prediction using the project outcomes report text.

The rest of the paper proceeds as follows. In Section 2 we detail the data collection and annotation process used for creating the NSF-Soft-Search Training dataset. In Section 3 we briefly describe the model training process and report results. In Section 4 we provide summary statistics for the NSF-Soft-Search Inferred dataset and observe trends in software production over time. We conclude with discussion regarding the limitations of our approach and opportunities for future work.

2 DATA COLLECTION AND ANNOTATION

2.1 Finding Software Produced by NSF Awards

The first step in our data collection process was to find software outputs from National Science Foundation (NSF) funded research. This step has two potential approaches. The first approach is a manual search for references and promises of software production within NSF award abstracts, project outcome reports, and papers supported by each award. This first approach is labor intensive and may be prone to labeling errors because while there may be a promise of software production in these documents, it may not be possible to verify such software was ultimately produced. The

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

JCDL, June 26–30, 2023, Santa Fe, NM

© 2023 Association for Computing Machinery.

<https://doi.org/XXXXXXX.XXXXXXX>

other approach is to predict software production using a trained model. We pursue this approach with the caveat that there are also potential label errors.

To gather examples of verifiable software production, we created a Python script which used the GitHub API to search for repositories which included reference to financial support from an NSF award in the repositories README.md file. Specifically our script queried for README.md files which contained any of the following text snippets: ‘National Science Foundation’, ‘NSF Award’, ‘NSF Grant’, ‘Supported by NSF’, or ‘Supported by the NSF’. GitHub was selected as the basis for our search because of its widespread adoption and mention in scholarly publication [3]. This search found 1520 unique repositories which contained a reference to the NSF in the repository’s README.md file.

2.2 Software Production Annotation

The next step in our data collection process was to annotate each of the GitHub repositories found as either “software” or “not software.” In our initial review of the repositories we had collected, we found that the content of repositories ranged from documentation, experimental notes, course materials, collections of one-off scripts written during a research project, to more typical software libraries with installation instructions, testing, and community support and use.

Using existing definitions of what constitutes research software to form the basis of our annotation criteria [7, 9], we conducted multiple rounds of trial coding on samples of the data. Fleiss’ kappa was used to determine if there was agreement between our research team on whether ten GitHub repositories contained ‘software’ or not. On each round of trial coding ten GitHub repositories were randomly selected from our dataset for each member of our research team to annotate independently. When assessing a repository, members of the research team were allowed to use any information in the repository to determine their annotation (i.e. the content of the README.md file, the repository activity, documentation availability, etc.)

Our final round of trial coding showed that there was near perfect agreement between the research team ($K=0.892$) [11].

Our final annotation criteria was generally inclusive of labeling repositories as software, rather there were specific exclusion criteria that resulted in a repository being labeled as “not software”. Specifically repositories were labeled as “not software” when a repository primarily consisted of:

1. project documentation or research notes
2. teaching materials for a workshop or course
3. the source code for a project or research lab website
4. collections of scripts specific to the analysis of a single experiment without regard to further generalizability
5. utility functions for accessing data without providing any additional processing capacity

We then annotated all GitHub repositories in our dataset as either “software” or “not software” according to our agreed upon annotation criteria.

Table 1: Predictive Model Results (Trained with Abstract Text)

| | model | accuracy | precision | recall | f1 |
|---|----------------|----------|-----------|--------|-------|
| 0 | tfidf-logit | 0.674 | 0.674 | 0.674 | 0.673 |
| 1 | transformer | 0.636 | 0.608 | 0.697 | 0.649 |
| 2 | semantic-logit | 0.630 | 0.630 | 0.630 | 0.630 |
| 3 | regex | 0.516 | 0.515 | 0.516 | 0.514 |

2.3 Linking GitHub Repositories to NSF Awards

Our final step in the data collection process was to link the annotated GitHub repositories back to specific NSF awards. To do so, we created a script which would load the webpage for each GitHub repository, scrape the content of the repository’s README and find the specific NSF award ID number(s) referenced. While annotating the dataset, and with this script, our dataset size was reduced as we found some repositories were returned in the initial search because of the “NSF” acronym being used by other, non-United-States governmental agencies which also fund research.

When processing each repository, our Python script would load the README content, search for NSF Award ID patterns with regular expressions, and then verify that each NSF award ID found was valid by requesting metadata for the award from the NSF award API.

We then retrieved the text for each award’s abstract and project outcomes report. This was the final step of our data collection process and allowed us to create a dataset of 446 unique NSF awards labeled as ‘produced software’ and 471 unique NSF awards labeled as ‘did not produce software’.

3 PREDICTIVE MODELS

Using the compiled NSF-Soft-Search Training dataset, we trained three different models using the text from either the award abstract or project outcomes report. The models trained include a logistic regression model trained with TF-IDF word embeddings (tfidf-logit), a logistic regression model trained with semantic embeddings (semantic-logit), and a fine-tuned transformer (transformer). The semantic embeddings and the base model from which we fine-tuned our own transformer model was the ‘distilbert-base-uncased-finetuned-sst-2-english’ model [4]. Each model was trained with 80% of the NSF-Soft-Search Training dataset. We then test each of the models and use F1 to rank each model’s performance.

Table 1 reports the results from training using the abstract text as input. The best performing model was the tfidf-logit which achieved an F1 of 0.673.

Table 2 reports the results from training using the project outcomes reports as input. The best performing model was the tfidf-logit which achieved an F1 of 0.745.

While the models trained with the project outcomes reports were trained with less data, the best model of the group achieved a higher F1 than any of the models trained with the abstracts. While we have not investigated further, we believe this to be because the

Table 2: Predictive Model Results (Trained with Project Outcomes Report Text)

| | model | accuracy | precision | recall | f1 |
|---|----------------|----------|-----------|--------|-------|
| 0 | tfidf-logit | 0.745 | 0.745 | 0.745 | 0.745 |
| 1 | transformer | 0.673 | 0.638 | 0.771 | 0.698 |
| 2 | semantic-logit | 0.633 | 0.633 | 0.633 | 0.632 |
| 3 | regex | 0.510 | 0.507 | 0.510 | 0.482 |

Table 3: Composition of the NSF Soft Search Dataset

| | Program | # Awards | # Software | % Software |
|---|---------|----------|------------|------------|
| 0 | MPS | 32885 | 19178 | 0.583184 |
| 1 | CISE | 24633 | 13274 | 0.538871 |
| 2 | ENG | 22900 | 11242 | 0.490917 |
| 3 | GEO | 17822 | 5142 | 0.288520 |
| 4 | BIO | 16990 | 6013 | 0.353914 |
| 5 | EHR | 13703 | 575 | 0.041962 |
| 6 | SBE | 13318 | 1966 | 0.147620 |
| 7 | TIP | 8597 | 4501 | 0.523555 |
| 8 | OISE | 2329 | 636 | 0.273079 |
| 9 | OIA | 498 | 123 | 0.246988 |

project outcomes reports contain more direct citation of produced software rather than an abstract’s promise of software production.

The data used for training, and functions to reproduce these models, are made available via our Python package: redacted.

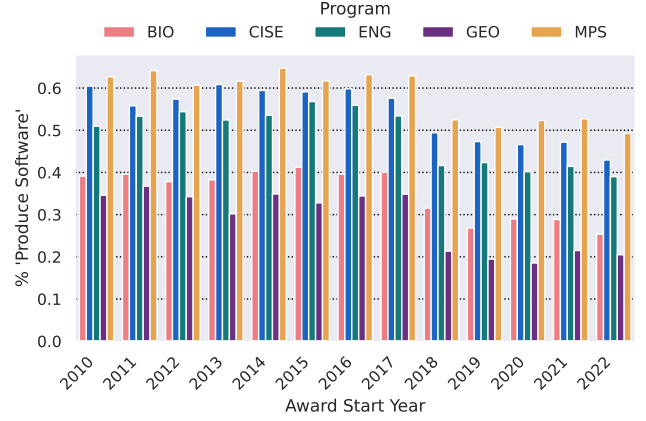
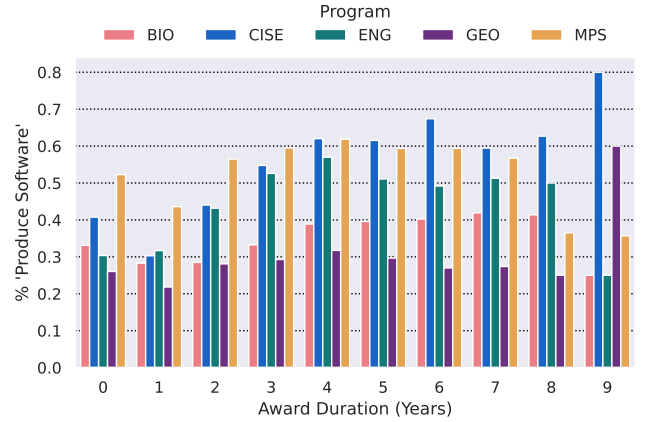
4 THE NSF-SOFT-SEARCH DATASET

Using the predictive models, we compile the NSF-Soft-Search Inferred dataset which contains the metadata, abstract text, and project outcomes report text, for all NSF awarded projects during the 2010-2023 period. The NSF-Soft-Search Inferred dataset additionally contains our predictions for software production using both texts respectively.

4.1 Trends and Observations

Using the NSF-Soft-Search Inferred dataset we can observe trends in software production over time. Figure 1 plots the percent of awards which we predict to have produced software (using the award’s abstract) over time. While there are minor year-to-year deviations in predicted software production, we observe the “Math and Physical Sciences” (MPS) funding program as funding the most awards which we predict to produce software, with “Computer and Information Science and Engineering” (CISE), and “Engineering” (ENG) close behind.

We can additionally observe trends in software production as award duration increases. Figure 2 plots the percent of awards which we predict to have produced software (using the award’s abstract) grouped by the award duration in years. We note that as award duration increases, the percentage of awards which are predicted to have produced software also tends to increase.

**Figure 1: Software Production Over Time (Using Predictions from Abstracts)****Figure 2: Software Production Grouped By Award Duration (Using Predictions from Abstracts)**

5 CONCLUSION

We introduce NSF-Soft-Search, a pair of novel datasets for studying software production from NSF funded projects. The NSF-Soft-Search Training dataset is a human-labeled dataset with almost 1000 examples used to train models which predict software production from either the NSF award abstract text or the project outcomes report text. We used these models to generate the NSF-Soft-Search Inferred dataset. The NSF-Soft-Search Inferred dataset includes project metadata, the awards abstract and project outcomes report, and predictions of software production for each NSF funded project between 2010 and 2023. We hope that NSF-Soft-Search helps further new studies and findings in understanding the role software development plays in scholarly publication.

All datasets and predictive models produced by this work are available from our GitHub repository: redacted.

5.1 Limitations

As discussed in Section 2, the NSF-Soft-Search Training dataset was entirely composed of NSF awards which ultimately released or hosted software (and other research products) on GitHub. Due to our data collection strategy, it is possible that each of the predictive models learned not to predict if an NSF award would produce software, but rather, if an NSF award would produce software hosted on GitHub.

5.2 Future Work

As discussed in Section 2.1, our initial method for attempting to find research software produced from NSF supported awards was to search for references and promises of software production in the abstract, project outcomes report, and attached papers of each award. While attempting this approach to create the dataset, we found that many awards and papers that reference computational methods do not provide a reference web link to their code repositories or websites. In some cases, we found repositories related to an award or paper via Google and GitHub search ourselves. While we support including references to code repositories in award abstracts, outcomes reports, and papers, future research should be conducted on how to enable automatic reconnection of papers and their software outputs.

6 ACKNOWLEDGEMENTS

Redacted for anonymous submission.

REFERENCES

- [1] Prajwal Bhattarai, Mohammed Ghassemi, and Tuka Alhanai. 2022. Open-Source Code Repository Attributes Predict Impact of Computer Science Research. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries* (Cologne, Germany) (JCDL '22). Association for Computing Machinery, New York, NY, USA, Article 16, 7 pages. <https://doi.org/10.1145/3529372.3530927>
- [2] Cai Fan Du, Johanna Cohoon, Patrice Lopez, and James Howison. 2022. Understanding progress in software citation: a study of software citation in the CORD-19 corpus. *PeerJ Computer Science* 8 (2022).
- [3] Emily Escamilla, Martin Klein, Talya Cooper, Vicky Rampin, Michele C. Weigle, and Michael L. Nelson. 2022. The Rise of GitHub in Scholarly Publications. <https://doi.org/10.48550/ARXIV.2208.04895>
- [4] HF Canonical Model Maintainers. 2022. distilbert-base-uncased-finetuned-sst-2-english (Revision bfdd146). <https://doi.org/10.57967/hf/0181>
- [5] Ana-Maria Istrate, Donghui Li, Dario Taraborelli, Michaela Torkar, Boris Veytsman, and Ivana Williams. 2022. A large dataset of software mentions in the biomedical literature. <https://doi.org/10.48550/ARXIV.2209.00693>
- [6] Daniel S. Katz, Neil P. Chue Hong, Tim Clark, August Muench, Shelley Stall, Daina R. Bouquin, Matthew Cannon, Scott C. Edmunds, Telli Faez, Patricia Feeney, Martin Fenner, Michael Friedman, Gerry Grenier, Melissa Harrison, Joerg Heber, Adam Leary, Catriona J. MacCallum, Hollydawn Murray, Erika Pas-trana, Kath Perry, Douglas C. Schuster, Martina Stockhause, and Jake S. Yeston. 2021. Recognizing the value of software: a software citation guide. *F1000Research* 9 (2021).
- [7] Carlos Martinez-Ortiz, Paula Martinez Lavanchy, Laurents Sesink, Brett G. Olivier, James Meakin, Maaïke de Jong, and Maria Cruz. 2022. Practical guide to Software Management Plans. <https://doi.org/10.5281/zenodo.7185371>
- [8] David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2022. The role of software in science: a knowledge graph-based analysis of software mentions in PubMed Central. *PeerJ Computer Science* 8 (2022).
- [9] Vanessa Sochat, Nicholas May, Ian Cosden, Carlos Martinez-Ortiz, and Sadie Bartholomew. 2022. The Research Software Encyclopedia: a community framework to define research software. *Journal of Open Research Software* (2022).
- [10] Ana Trisovic, Matthew K. Lau, Thomas Pasquier, and Mercè Crosas. 2021. A large-scale study on research code quality and execution. *Scientific Data* 9 (2021).
- [11] Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med* 37, 5 (2005), 360–363.