Harvard Data Science Review • Special Issue 4: Democratizing Data

Discovering Data Sets Through Machine Learning: An Ensemble Approach to Uncovering the Prevalence of Government-Funded Data Sets

Ryan Hausen[1] **Hosein Azarbonyad**[2]

[1]Department of Physics and Astronomy, Johns Hopkins University, Baltimore, Maryland, United States of America,
[2]Elsevier BV, Amsterdam, Netherlands

Published on: Apr 02, 2024

DOI: https://doi.org/10.1162/99608f92.18df5545

Abstract

The prevalence of government-funded data set usage has yet to be comprehensively tracked and understood. The lack of a standardized citation methodology has thus far prevented the government from understanding data set usage in a transparent, accessible way. In this work, we seek to build on recent successes in natural language processing techniques and a recent Kaggle competition to develop an extensible framework for extracting government data set usage from scientific publications. Further, we apply the developed techniques to over 50,000 scientific articles from Elsevier's Scopus collection. Finally, we show that improvements to the submitted algorithms along with ensembling improves overall performance on an evaluation data set.

Keywords: data sets, machine learning, artificial intelligence, natural language processing

1. Introduction

Understanding how government-funded data is used in science is integral to deciding how to most effectively allocate government resources. The reporting of data set usage has not yet been standardized or widely adopted in the scientific literature. Therefore, uncovering data set usage requires manual review and best-guess estimation. Lane et al. (2022) ran a Kaggle Competition[1] (both referred to as COL22 hereafter) to develop automated approaches to discovering government data sets in texts. In this work, we seek to analyze and improve the submitted methods and describe the application of these methods at the scale of tens of thousands of documents. The code and model parameters are freely available at https://purl.archive.org/democratizing-data/code.

At a high level, the participants in the COL22 competition were tasked with providing a Jupyter Notebook[2] that when run iterates over a list of documents and outputs a comma-separated value (CSV) file containing a column for the document IDs and a column containing a pipe (|) delimited string containing the detected data sets for each of the documents. In this work, we refactor the submitted methods out of the notebook format into a code base with a consistent interface and improve on the core methods submitted. All submitted and newly developed methods are compared using precision, recall, and F1 metrics.

The remainder of the work is laid out as follows: Section 2 discusses related work, Section 3 describes the data, Section 4 details the models used and a framework for future development and evaluation of models, Section 5 reports the results, Section 6 discusses the applications of the models on tens of thousands of documents, and Section 7 reviews the work done and proposes future research directions.

2. Related Work

There has been a growing interest in methods to detect data set references in scientific texts. Previous work has identified informally cited references like data sets and software in the scientific literature. These works differ from ours in either scale or focus. Duck et al. (2013) developed a technique for detecting bioinformatics software and database resources in scientific texts. They use a data set of 60 full text articles. Twenty-five articles for training, 5 articles for development, and 25 articles for evaluation. Luan et al. (2018) examined 500 abstracts and builds a knowledge graph that identifies scientific entities. Zhao et al. (2019) developed methods for identifying the role and function of hyperlinked resources in scientific papers. Otto et al. (2023) developed a technique and data set for entity extraction of machine learning models and data sets. Ghavimi, Mayr, Lange, et al. (2016) and Ghavimi, Mayr, Vahdati, and Lange (2016) describe methods for detecting data sets in social science papers and evaluate the efficacy of their models using 15 hand-curated documents.

Several studies have experimented with the detection of data set mentions in natural language processing (NLP) as a named entity recognition (NER) task. Early work mostly focused on architectures that combined word representation vectors with recurrent neural networks (RNN), like long short-term memory networks (LSTM) and bidirectional LSTMs (BiLSTM) (Hou et al., 2021; Li et al., 2021). More recent studies have experimented with data set mention detection as a downstream task for pretrained language models such as BERT (Bidirectional Encoder Representations from Transformers), outperforming the previously mentioned models (Heddes et al., 2021). Some recent work has focused on improving the scale of training and test data along with model development. Hou et al. (2019) developed an automated approach for generating leader boards for data sets and tasks from NLP-focused papers. Prasad et al. (2019) develops convolutional neural network and recurrent neural network–based approaches to the Coleridge Initiative's Rich Context Competition data set. Heddes et al. (2021) developed an approach to detect data sets in artificial intelligence (AI) papers using a corpus of $\sim 6,000$ sentences. Younes and Scherp (2023) used the COL22 data to compare the efficacy of question answering large language models and named entity recognition models for extracting unknown data sets from extracted snippets in the COL22 documents. Yao et al. (2019) developed a method for extracting the methods and data sets from a collection of $\sim 6,000$ sentences extracted from 430 papers from the Pacific-Asia Conference on Knowledge Discovery and Data Mining and 266 papers from the Meeting of the Association for Computational Linguistics. Fan et al. (2023) developed a method to extract data sets from sentence-level text while trying to provide a rich characterization of how the data were used. Pan et al. (2023) developed and released the Dataset Mentions Detection Dataset. A large-scale data set consisting of documents from Papers With Code and S2ORC with data set annotations and entity linking. Bassinet et al. (2023) developed the French Open Science Monitor to detect software and data set usage in French publications.

3. Data

For model training and evaluation, we leveraged the COL22 Kaggle competition data consisting of $\sim 14,000$ documents with labels indicating which data sets that are within each document. The labels in the data set are derived from five government agencies (Lane et al., 2022): U.S. Department of Agriculture, National Oceanic and Atmospheric Agency (NOAA), National Center for Education Statistics (NCES), NCSES at the National Science Foundation (NSF), U.S. Geological Survey, and National Institutes of Health and found in open access documents using a search process discussed in Lane et al. (2022).

All data set labels were preprocessed according to the requirements of the competition. Specifically, all data set labels are converted to all lowercase characters and any non-alphanumeric characters are replaced with single-space characters. An example of the preprocessing is shown below,

`Aging Integrated Database (AGID) → aging integrated database agid`.

Though cased versions of the labels are available and some of the implemented models reproduce the mentions in their cased form, evaluations are performed on lowercase versions to fairly compare between models submitted to the competition and further developed methods.

The training labels from the COL22 competition consists of the raw string representation of the target data set associated with the text (i.e., "our world in data," "beginning postsecondary students," and "baccalaureate and beyond longitudinal study"). The training labels do not include information about their location in the text or any categorical information about them.

4. Methods

The methods developed in this work build upon the top three submissions to the COL22 competition by extracting the core methods from the submissions, analyzing and improving the core methods, and building an ensemble that benefits from the strengths of each method. The submissions were in the form of a Jupyter Notebook[3] that is executed in a Python environment. The algorithms for each submission are refactored into a single code base with a unified interface for each submitted and newly developed model. The algorithmic approaches are organized into four categories: *string matching*, *entity classification*, *token classification*, and *ensemble*.

4.1. Core Methods and Implicit Ensembling

Each of the submitted methods, as noted above, was submitted in the form of a Jupyter Notebook. Each submission was not restricted to submitting a simple function that consists of a model only but included additional heuristics wrapped around the model to improve the final competition score. With the exclusion of the simple string-matching method, the submitted models include additional heuristics. Leveraging additional

heuristics creates an implicit ensemble that includes the core model and the additional heuristics. Below, in Equation 4.1, is an example of what an ensemble might look like.

$$f(\text{text}) = \text{ensemble}(\text{text}) = \begin{cases} \text{text } \textbf{is} \text{ dataset} & \text{If model}(\text{text}) \geq 0.9, \\ \text{text } \textbf{is} \text{ dataset} & \text{If “dataset” is in text}, \\ \text{text } \textbf{is not} \text{ dataset} & \text{otherwise}, \end{cases} \quad (4.1)$$

The additional heuristics submitted alongside the core methodology, though helpful in improving the competition score, obfuscate the efficacy of the core method. Consider the ensemble described in Equation 4.1, f defines some function that works on a subset of the document text that may or may not represent a data set, suppose that two different deep learning models were trained and substituted in for the ‘model’ on the top line of the conditional. Model efficacy for any data set with the text ‘dataset’ within it cannot be tested within the ensemble because those sets of text will always be evaluated as ‘is dataset’ due to the second condition in Equation 4.1. To fairly compare methodologies, comparisons are made between core methodologies without additional heuristics.

4.2. Model Development Framework

To evaluate the core component of the submitted and proposed methods, we developed an extensible framework for developing and evaluating methods on the COL22 data. The submission criteria for the COL22 competition included that the format be a Jupyter Notebook and that the notebook finish running on the test data within 9 hours. Clearly analyzing the core components of the submitted algorithms was difficult in the submitted Jupyter Notebooks due to different types of implementations and coding practices. We refactor and reimplement the core methods such that they follow a unified interface. Further, we implement functions to evaluate methods on the COL22 data including ensembles of multiple methods. To promote openness and contributions from the community, we publicly release our code and model parameters, which are available at https://purl.archive.org/democratizing-data/code.

The methods examined in this work are organized into three categories: *String Matching* (Section 4.3), *Entity Classification* (Section 4.4), and *Token Classification* (Section 4.5). In Section 4.6 we discuss an ensemble approach leveraging models from each of the categories.

4.3. String Matching

The third-place submission to the COL22 competition was a simple string-matching algorithm (Mikhail, 2021) where a given list of data sets[4] is compared to the text of documents with some preprocessing (see Algorithm 1). Though this algorithm is fast, it is fragile in two important ways. First, the algorithm makes all data sets and text lowercase before matching. This causes the algorithm to generate false positives when an acronym shares a spelling with another word. An example of this is the Higher Education Research and Development Survey[5], abbreviated *HERD*, which can also, and more commonly, refer to a group of animals. Homographs like these

result in a large number of false positives. Second, the submitted algorithm splits the input text into sentences on every occurrence of a period. Each ‘sentence’ is then searched for all the data sets. However, periods do not always indicate the end of a sentence and breaking up the text this way can cause the algorithm to miss data sets that were improperly separated.

To improve this algorithm, we replace simple string matching with a regular expression-based matching algorithm (see Algorithm 2). Using a regular expression allows the algorithm to selectively enforce casing. By selectively enforcing casing we eliminate the false positives generated by acronyms that share a spelling with another more common word. Further, we run the regular expression on the entirety of the text, which will avoid missing data sets from improperly splitting the text into sentences. Finally, using regular expressions offers more flexibility beyond casing for variations on data set names. Some data sets may have numerical variations such as year or version that need special rules beyond casing that can be better handled by a regular expression.

4.4. Entity Classification

The second-place submission to the COL22 competition was an entity classification–based method (Lee, 2021). Entity classification is an approach comprised of two steps. The first step is to extract all entities from a target document (extraction). The second step is to classify an entity as being a data set or not (classification). Improving the performance of the algorithm can be done by improving the extraction and classification algorithms.

The submitted extraction algorithm is an implementation of the Schwartz and Hearst (2003) (SH03) algorithm for detecting abbreviation definitions. The SH03 algorithm seeks to find abbreviations in the form of: *Abbreviated Entity Definition (AED)*. Though generally effective, not all data sets are referenced using that form. Some data sets like the USDA Census of Agriculture do not have a common acronym. Other data sets like the OECD’s PISA (Programme for International Student Assessment) data set are so commonly used that the acronym may not be defined in the text at all. Additionally, the submitted implementation was not robust. For example, nested parentheses and abbreviations that include year references, i.e., *Abbreviated Entity Definition (AED;95)* are not correctly handled.

In our approach, we opt to replace SH03 with a regular expression for entity extraction shown below (split into multiples lines for visibility),

```
((((([A-Z][a-z]{2,}|[A-Z]{3,})('s)?)\ )
(([A-Z][a-z]{2,}|[A-Z]\3,}|and\ |for\ |in\ |of\ |the\ |on\ |to\ )('s)?(\ )?){2,}
(?<!and\ |for\ |in\ |of\ |the\ |on\ |to\ )(\([A-Z]{3,}\))?)
```

The above regular expression can be translated semantically as the following:

1. Match a sequence of characters that start with a capital character followed by at least two lowercase letters OR at least three capital letters. Followed by an optional apostrophe s [‘s]. Followed by a space.

2. Pattern (1) or one of the following words “and”, “for”, “in”, “of”, “the”, “to” followed by an optional apostrophe s [‘s] or a space. Pattern (2) repeated 2 or more times.
3. Match a sequence of at least 3 capital letters in between parenthesis, so long as it is not preceded by one of: “and,” “for,” “in,” “of,” “the,” “on,” or “to.”

In designing a regular expression, we sought to capture as many entities as possible. Our proposed regular expression matches strings that would be detected by SH03 and additional entities that would have otherwise been missed, most obviously those that do not end with an acronym wrapped in parenthesis. The regular expression captures many more non-data set entities than SH03 (i.e., “United States of America (USA)” and “Jenny and John”). However, the emphasis for the extraction algorithm is to improve recall as it should be paired with a classifier that is responsible for the precision of the output data sets.

The submitted classifier was fine-tuned for sentence classification from a RoBERTa-based (Liu et al., 2019) model. The training data for the classifier, also submitted by the participant,[6] consists of entities with their labels. In training our model, we also use a RoBERTa-based model but leverage an additional 29 known agency data sets for additional training samples that were provided from agency stakeholders.[7] Leveraging additional training samples produces more reliable deep learning models that can better capture the distribution of data set entities.

4.5. Token Classification

The first-place submission to the COL22 competition was a token classification–based method (Nguyen & Nguyen, 2021). Token classification is an approach where each token within a text is classified. Tokens can be words or subwords, pieces depending on the tokenizer. Applied in this setting, token classification indicates whether or not a token refers to a data set. The first-place submission to the COL22 competition used a masked token classification approach.

The first-place submission to the COL22 competition used an ensemble of two models utilizing a custom token classification approach based on masked language models (Nguyen and Nguyen [2021] and Figure 1). The model is trained to learn the context of sentences that typically describe data sets. The specific training details can be found in Nguyen and Nguyen (2021). At inference time, the model converts the tokens into an embedding space that is compared to the average embedding for 100 randomly sampled data set-like tokens and 100 randomly sampled non-data set like tokens. The strength of this approach is that the model is trained to recognize tokens that surround data sets as part of the classification process. Data set names are not completely obscured from the model during training and so the model does not completely rely on context. The drawback of this approach is that the model requires data set and non–data set embedding prototypes that can be compared to the model output emeddings for inference. The submitted algorithm randomly samples from a collection containing 71,627 embeddings for both data set and non–data set embeddings. The collection is sampled for each batch inference.

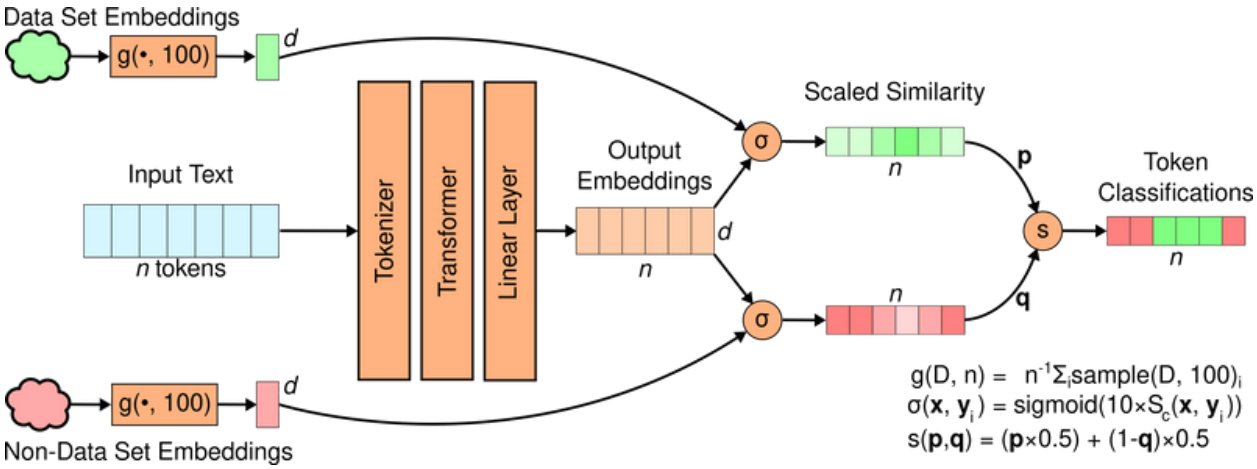

Figure 1. A diagrammatic representation of the inference process for the submitted masked language token classification algorithm described in Section 4.5. *Data Set Embeddings* and *Non-Data Set Embeddings* are collections of *d*-dimensional vectors that are precalculated from the training set and stored on disk. An average over $m = 100$ randomly sampled vectors from the precomputed collections are used per batch at inference time. *g* is a function that randomly samples *m* vectors from collection D and returns the average vector. *d* is the embedding dimensionality and S_C is the cosine similarity function. In the function $\sigma(\mathbf{x},\mathbf{y}_i)$, $\mathbf{x}$ refers to one of Data Set Embeddings or Non-Data Set Embeddings and $\mathbf{y}_i$ refers to a single embedded token, *i*, from Output Embeddings.

In our experiments, we fine-tune a SciBERT-based (Beltagy et al., 2019) transformer for Named Entity Recognition (NER) on the COL22 competition data. Using an NER model removes the dependency on generating data set and non–data set embeddings. The original COL22 data does not include any part-of-speech or data set tagging. Therefore, to train a NER model, we generate Part-of-Speech (POS) tags using spaCy (Montani et al., 2021) and convert the POS tags to data set label tags by searching the text for the given data set names and labeling the tags ("B-DAT" and "I-DAT") when they match a data set and the generic 'O' tag for all other tokens.

We additionally train another model of the same type as Nguyen and Nguyen (2021) using a RoBERTA-based transformer using a refactored version of the Nguyen and Nguyen (2021) submission. Our implementation serves as a prototype should other contributors wish to further develop and train models in that family of models.

4.6. Ensemble

In practice, we use an ensemble classifier (Polikar, 2012) composed of models from different categories. For the calculation of metrics, we ensemble multiple models by merging the evaluation statistics for each ensembled model on each of the test documents from the COL22 competition. We take the set union of any ensembled models. Any time two models have a shared entry where one has the data set evaluated as a false negative and the other has the entry evaluated as a true positive, that entry is considered a true positive. All

false positives after the set union is performed are kept in the ensemble's calculation. Ensemble methods have been demonstrated to generally perform better than any single classifier on its own. As shown in Table 1, the ensemble method has a better F_1 score than any single method on its own. We therefore use an ensemble in our large-scale analysis.

5. Results

The methods were evaluated using the private validation data set from the COL22 competition. The data is validated by comparing the set of labels from the human labelers, D_y, with the set of labels from the model, $D_{\hat{y}}$. The statistics used to evaluate model performance are precision, recall, and F_1 scores. To calculate the evaluation statistics requires a measure of string similarity to compare elements from $D_{\hat{y}}$ to D_y. The COL22 competition applied the preprocessing described in Section 3 to all the elements of both sets, then compared elements from $D_{\hat{y}}$ to D_y using the Jaccard index (Jaccard, 1912), also called the intersection-over-union,

$$J(d_y, d_{\hat{y}}) = \frac{d_y \cap d_{\hat{y}}}{d_y \cup d_{\hat{y}}} \qquad (5.1)$$

where $d_y \in D_y$ is a single data set mention from the human labelers and $d_{\hat{y}} \in D_{\hat{y}}$ is a predicted data set from a model. Both d_y and $d_{\hat{y}}$ are themselves sets of string tokens generated by splitting the string by whitespace (i.e. "aging integrated database" becomes {"aging," "integrated," "database"}). Elements from d_y and $d_{\hat{y}}$ are compared using strict string equality to evaluate the intersection and union seen in Equation 5.1. In COL22 d_y and $d_{\hat{y}}$ are considered to be referring to the same data set if $J(d_y, d_{\hat{y}}) \geq 0.5$.

Using the Jaccard index is computationally efficient and easy to understand, however, it can be rigid and flexible in ways that are not necessarily helpful for our purposes. With respect to its rigidity, simple strict string matching does not have any tolerance for spelling differences either by mistake from the human labelers, the model, or from some other external error (i.e., a mistake by an optical character recognition program that digitized the document). Concerning flexibility, the Jaccard index operates on sets, which by definition are unordered, excluding an important facet of language.

To more robustly compare d_y and $d_{\hat{y}}$, we adopt a normalized Levenshtein distance. The Levenshtein distance (Levenshtein, 1966; Section A.1), describes the number of edits that would transform d_y into $d_{\hat{y}}$. Edits are one of the following operations: *insertion*, *deletion*, and *substitution*. To compute the final matching score, the Levenshtein distance is calculated using `partial_ratio` from the `thefuzz` Python package. When two strings are different lengths the `partial_ratio` function finds the substring in the longer string with the shortest distance to the shorter string and normalizes that distance by the length of the shorter string, yielding a value between 0 and 100. We divide this value by 100, yielding a value between 0 and 1. First, any single character variants, however introduced, cost a single character distance rather than the entire word. Further, order is enforced in the normalized Levenshtein distance, which will reduce false positives for sequences that utilize common words. In our experiments, we consider two strings to be referring to the same data set if

`partial_ratio` $(d_y, d_{\hat{y}})/100 \geq 0.90$. In practice, we find the normalized Levenshtein distance to not share the weaknesses of the Jaccard index while producing robust comparisons. Using the normalized Levenshtein distance the performance of the submitted and retrained models on the COL22 Kaggle competition private validation data set is shown in Table 1. The private validation data set contains 8,063 documents annotated with labels indicating which data sets are within each document. The data sets follow the same format as the COL22 Kaggle competition training data set described in Section 3.

Table 1. Model evaluation results.

Model	Precision	Recall	F_1
String Matching			
(1) Submitted Simple String Match	0.90	**0.07**	**0.13**
(2) Regex Enhanced Match	**1.0**	0.06	0.11
Entity Classification			
(3) Submitted $\text{Model}_{0.90}$ + Submitted Entity Extractor	**0.70**	0.08	0.15
(4) Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor	0.45	**0.51**	**0.48**
Token Classification			
(5) NER $(\text{SciBERT})_{0.70}$	**0.86**	0.13	0.22
Ensemble			
(2) + (4)	0.45	0.52	0.48
(2) + (5)	**0.87**	0.14	0.25
(4) + (5)	0.45	0.52	0.48
(2) + (4) + (5)	0.45	**0.53**	**0.49**

Note. Three of the token classification approaches: *Submitted Model (Roberta)*$_{0.70}$, *Submitted Model (SciBERT)*$_{0.70}$ (Nguyen & Nguyen, 2021) , and *Retrained Model (RoBERTa)*$_{0.70}$ (this work), described in Section 4.5 and Figure 1, randomly sample from precomputed vector-embeddings to compute classification scores for each token. The results of these approaches can be highly

skewed, and the distribution was not readily amenable to tabular summarization. The performance of this approach was inferior to the others and is not included in Section 6. For completeness, we graphically display the results of running these approaches on the validation set in Figure 2, which shows that the performance of the Submitted models heavily skews toward zero for both precision and recall.

6. Applying Models at Scale

In this section, we describe the application of our proposed models (2, 4, 5 from Table 1) on a large subset of articles from Elsevier's Scopus database of articles, hereafter called the Scopus Search Corpus. The process for creating the Scopus search corpus is described in detail in a further paper in this special edition (Emecz et al, 2024). In summary, the process involves identifying a subset of records from the Scopus database where licenses from the publishers permit a full text search and where the records are consistent with the date range and subject areas of specific interest.

In our analysis, we focus on a subset of these articles relevant to data sets of interest to the National Center for Science and Engineering Statistics (NCSES). The Scopus Search Corpis consisted of $\sim 500,000$ documents reflecting the topics and data parameters collected collaboratively defined with NCSES. The proposed methods are run on the filtered collection of $500,000$ documents. For our analysis, the documents are further filtered by including the documents for which at least one detection from any of the models is matched, via fuzzy matching, to a target data set provided by the NCSES[8]. Fuzzy matching is done using the `token_sort_ratio` function from the `fuzzywuzzy` [9] package with a threshold of 85/100. The resulting document set used for analysis here is $\sim 50,000$ documents. Finally, due to text encoding issues, some documents had improperly encoded text, which produced incoherent results. Those results are filtered out by filtering out model predictions that contain either a back slash '\' or a forward slash '/'.

Table 2. Number of unique detections per document and detection length.

Model	Average Unique Detections per Document	Average Detection Length (characters)
Regex Enhanced Match	1.11 ± 0.39	7.47 ± 7.83
Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor	4.55 ± 3.95	37.17 ± 15.96
NER $(\text{SciBERT})_{0.70}$	1.88 ± 1.23	19.44 ± 12.04
$\text{Ensemble}_{(2)+(4)+(5)}$	1.19 ± 0.98	9.49 ± 11.24

Note. For each proposed model, the average number of unique detections per document for all documents where that model detects a data set and average length of detections in characters. For each model for each measurement single standard deviation is also reported.

We examine the model outputs in four ways. First, in Table 2 we show, for all documents where a model has at least one detection, the average number of unique detections and average length of the detections in characters. Interestingly, our Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor tends to produce the highest number of detections in documents where a detection is made. Unsurprisingly, the Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor model produces the longest detections on average. The flexibility of the regular expression described in Section 4.4 gives the entity classification model the ability to capture long chains of text. The Regex Enhanced model returns the shortest detections on average, which seems also in line with expectations because the targeted list of data sets includes acronyms, and the regular expression will consistently find those kinds of data sets. The NER model detects the least number of data sets per document on average and a typical detection is about 19 characters long.

Second, in our analysis we look at the most common detections by each model in Table 3. In Table 3 we see that the most common detection for the Regex Enhanced Match model are almost all acronyms. This seems intuitive because it is common practice to introduce an entity with its full name and an acronym and then refer to it by acronym only for the remainder of the text. The sixth entry for the Regex Enhanced model is interesting as well because though the entity "Fields of Study" is indicated as a data set to look for, it is likely to also be associated with plenty of false positives. The results for the Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor seem to be very promising. The regular expression described in Section 4.4 is very flexible and matches many non–data set entities (i.e., United States of America, Johnny and Jenny), so the quality of the outputs seen in the table seem to indicate that the classifier is filtering out poor-quality candidates extracted by the regular expression extractor. The NER model generally produces detections that seem close to right, but there are noticeable false positives like the first entry "Higher Education" and the eighth entry "All." No post-processing or heuristics are performed on model outputs, which may help limit some of the false positives shown.

Table 3. Most common reported detections on the Scopus Search Corpus.

Rank	Regex Enhanced Match	Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor	NER (SciBERT)$_{0.70}$
(1)	R&D intensity	Science and Engineering Indicators	Higher Education
(2)	SED	Web of Science	Survey of Earned Doctorates
(3)	FAC	Survey of Earned Doctorates	Survey of Doctorate Recipients
(4)	SDR	National Survey of College Graduates	in Science and Engineering

(5)	FSS	Survey of Doctorate Recipients	Survey of College Graduates
(6)	fields of study	American Community Survey	National Survey of College Graduates
(7)	GSS	National Survey of College Graduates (NSCG)	Survey of Graduate Students and Postdoctorates in Science and Engineering
(8)	STEM education	American Community Survey (ACS)	All
(9)	FFS	Journal of Economic Surveys	Survey
(10)	BERD	Survey of Earned Doctorates (SED)	Survey of Research and Development

Note: For each of the proposed models, the top 10 detected data sets are reported. The output classifications are not modified with any spelling errors and premature text truncations preserved. See (4) and (9) in the NER column for examples.

Third, in our analysis, we examine the least commonly detected data sets in Table 4. As expected, the Regex Enhanced Match model produces reasonable results due to its high precision. The Retrained $\text{Model}_{0.90}$ + Regex Entity Extractor model returns detections that also seem reasonable indicating that the pairing of a less-restricted regular expression with a strong classifier seems to consistently produce good results. A weakness of the NER model can be seen in the results in Table 4. Without being paired with some heuristics, the NER model produces some easy to identify false positives such as “and” and interesting incomplete predictions such as “Higher Education R.”

Table 4. Least common model detections on the Scopus Search Corpus.

Rank	**Regex Enhanced Match**	**Retrained Model**$_{0.90}$ + Regex Entity Extractor	**NER (SciBERT)**$_{0.70}$
(1)	S&E indicators	British Labour Force Survey (BLFS)	and
(2)	fall enrollment survey	Turkish Academic Career Survey	Higher Education Data
(3)	Science and Engineering Labor Force	Summer Undergraduate Research Experiences (SURE)	National Postdoctoral Association

(4)	Academic Research and Development	Survey on Science and Technology Research	International STEM Graduate Student in the
(5)	The State of U.S. Science and Engineering	IIP Patent Database	Longitudinal
(6)	Business Enterprise Research and Development Survey	NISTEP Corporate Name Dictionary	Higher Education R
(7)	Survey of Federal Science and Engineering Support	The Survey of Research and Development	Funds for Research and Development
(8)	IPEDS Enrollment	Large Research Infrastructures OECD Global Science Forum Homepage	Higher Education Research Institute
(9)	Women, minorities, and Persons with Disabilities	Trends in Citation	Higher Education and
(10)	National center for science and engineering statistics	Library Genesis Project	Trends in Education

Note: For each of the proposed models, the bottom 10 detected data sets are reported. The model outputs are not modified with any spelling errors and premature text truncation are preserved. See (1) and (6) in the NER column.

Finally, we quantitatively examine the correctness of the proposed method outputs by calculating an approximate precision value between the predicted data sets with previously identified data sets on a subset of the Scopus Search Corpus. The previously identified data sets were found using the methods submitted to the Kaggle competition and validated by a group of experts that viewed the prediction with some surrounding text. A total of 551 documents were used for the analysis. The precision is approximate because the set of documents is not comprehensively reviewed such that the list of given identified data sets is known to be exhaustive. Therefore, some detections could be erroneously assigned false positive, but were actually yet to be detected data sets. Table 5 shows the approximate precision and number of detections for each method and the ensemble. Table 6 shows the most common false positives for each method. The approximate precision for each model is lower than the reported precision in Table 1. Some of the difference can be attributed to some data sets that are marked as false positives are actually yet to be discovered data sets in the text.

The Regex Enhanced Match model had a notable decrease in precision, from 1.0 on the Kaggle test set to an approximate precision of 0.70. Some of the false positives, seen in Table 6, seem to be yet to be discovered data sets, however, the top 3 found false positives (“fields of study,” “STEM education,” “R&D intensity”) could be because these keywords passed to the model to search for are commonly referred to in contexts outside data set mentions.

The Retrained Model$_{0.90}$ + Regex Entity Extractor records the lowest approximate precision and the highest number of detections. This, though, may also be misleading because the top false positives in Table 6 perhaps are also yet to be identified data sets.

The NER (SciBERT)$_{0.70}$ model's precision also decreased from 0.86 on the Kaggle test set to an approximate precision of 0.63. The false positives in Table 6 seem to be a mix of plausible new detections and clearly false positives (e.g. "current" and "in").

Table 5. Precision and number of detections on the Scopus Search Corpus.

Model	Approximate Precision	# of Detections
Regex Enhanced Match	0.70	833
Retrained Model$_{0.90}$ + Regex Entity Extractor	0.33	1,102
NER (SciBERT)$_{0.70}$	0.63	262
Ensemble$_{(2)+(4)+(5)}$	0.51	2,005

Note: The number of detections and approximate precision for the proposed models on the subset of the Scopus Search Corpus for which we some validated detections exist. The precision is approximate because the documents used for analysis may include incorrect false-positives. See Section 6 for more details on the subset of the Scopus Search Corpus documents used for analysis. See Table 6 for examples of false positives.

Table 6. Top false positives on the Scopus Search Corpus.

Rank	Regex Enhanced Match	Retrained Model$_{0.90}$ + Regex Entity Extractor	NER (SciBERT)$_{0.70}$
(1)	fields of study [57]	Web of Science [23]	Higher Education [9]
(2)	STEM education [45]	American Community Survey [17]	Current [4]
(3)	R&D intensity [30]	American Community Survey (ACS) [11]	Current Population Survey [4]
(4)	SED [25]	Current Population Survey [8]	in Science and Engineering [4]
(5)	FAC [19]	World Development Indicators [6]	in [4]

(6)	STEM Education [16]	National Longitudinal Survey of Youth [5]	in Science [3]
(7)	BERD [11]	Current Population Survey (CPS) [4]	Study [3]
(8)	GSS [6]	Science Citation Index Expanded [4]	Education Longitudinal Study [2]
(9)	higher education R&D [6]	Integrated Postsecondary Education Data System [4]	National Study of [2]
(10)	R&D employment [6]	Web of Science Core Collection [4]	Baccalaureate and Beyond Longitudinal Study [2]

Note: Top 10 false positives for each of the proposed methods on the subset of the Scopus Search Corpus for which some data sets have been identified recorded as "Name of Dataset [Count]". Some of the false positives seem to be previously unidentified data sets. For consistency, these are all recorded as false positives in the approximate precision calculation in Table 5. See Section 6 for more information about the subset of documents used for analysis.

7. Conclusion

In this work, we extracted and improved upon the core methodologies submitted to the COL22 competition. We have developed an open source framework for developing and evaluating new models on the same data. We further described how the methods are applied at scale to uncover the usage of government data sets in the scientific community. Model evaluation results are described in detail in Table 1. Interestingly, the strongest recall values, outside the *Ensemble* category, are found in the *Entity Classification* category. Unsurprisingly, the strongest precision values are found in the *String Matching* category. An ensemble approach comprised of *String Matching*, *Entity Classification*, and *Token Classification* yields the best overall F_1 score and recall score. Though we found in Table 5 a decrease in apparent precision, Table 6 seems to show that some portion of false positives are likely yet to be discovered true positives. Table 3 and Table 4 seem to indicate that each of the developed methods finds a different variety or kind of data set mention reinforcing the idea that using the models in an ensemble will leverage the strengths of each separate method.

In future work, we propose improving the methodologies presented here in a few areas. The regular expression used in Section 4.3 can be further tested and improved for unknown corner cases. Other approaches to entity extraction (Section 4.4), for example, considering sentence classification for extraction and NER classification as discussed in Section 2 could be a fruitful avenue to explore. For token classification–based models (Section 4.5), a closer and more thorough evaluation of all of the generated tags and data set locations would improve the quality of the training data.

Next, a closer look at the COL22 competition inputs and labels suggests that the labels might benefit from a more unified set of metadata that connects common data set acronyms back to a 'parent' data set definition, for example, analogous to Istrate et al. (2022). Additionally, a rich taxonomy of labels would likely help improve model analysis by revealing which fields of study have data set labels that are more or less likely to be discovered by an automated algorithm. Future improvements in supervised approaches, like the ones explored and proposed here, would necessitate the development of a true gold standard data set. Though some work on a gold standard has been done (see Section 2), a comprehensive and agreed upon benchmark has yet to be defined.

Further, large language models like ChatGPT (OpenAI, 2023), Falcon LLM (Penedo et al., 2023), and Llama 2 (Touvron et al., 2023) could be applied to this data set and may further yield improved results. Finally, this work focused on supervised classification of known data sets, and future work could explore unsupervised data set discovery.

Disclosure Statement

RH and HA acknowledge funding from the National Center for Science and Engineering Statistics grant number 49100422C0028 and the Patrick J. McGovern Foundation. RH acknowledges funding from Schmidt Futures. The authors acknowledge that this research makes use of SciServer, a resource developed and operated by the Institute for Data Intensive Engineering and Science at The Johns Hopkins University.

References

Bassinet, A., Bracco, L., L'Hôte, A., Jeangirard, E., Lopez, P., & Romary, L. (2023). *Large-scale machine-learning analysis* [Preprint]. https://hal.science/hal-04121339

Beltagy, I., Lo, K., & Cohan, A. (2019, November). SciBERT: A pretrained language model for scientific text. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (pp. 3615–3620). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1371

Duck, G., Nenadic, G., Brass, A., Robertson, D. L., & Stevens, R. (2013). Bionerds: Exploring bioinformatics' database and software use through literature mining. *BMC Bioinformatics*, *14*(1), Article 194. https://doi.org/10.1186/1471-2105-14-194

Emecz, A., Mitschang, A., Zdawczyk, C., Dahan, M., Baas, J., & Lemson, G. (2024). Turning visions into reality: Lessons learned from building a search and discovery platform. *Harvard Data Science Review*, (Special Issue 4). https://doi.org/10.1162/99608f92.d8a3742f

Fan, L., Lafia, S ., Wofford, M. , Thomer, A. , Yakel, E. , & Hemphill, L. (2023). Mining semantic relations in data references to understand the roles of research data in academic literature. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 215–227). IEEE. https://doi.org/10.1109/JCDL57899.2023.00039

Ghavimi, B., Mayr, P., Lange, C., Vahdati, S., & Auer, S. (2016). A semi-automatic approach for detecting dataset references in social science texts. *Information Services & Use*, *36*(3-4), 171–187. https://doi.org/10.3233/ISU-160816

Ghavimi, B., Mayr, P., Vahdati, S., & Lange, C. (2016). *Identifying and improving dataset references in social sciences full texts*. ArXiv. https://doi.org/10.48550/arXiv.1603.01774

Heddes, J., Meerdink, P., Pieters, M., & Marx, M. (2021). The automatic detection of dataset names in scientific articles. *Data*, *6*(8), Article 84. https://doi.org/10.3390/data6080084

Hou, Y., Jochim, C., Gleize, M., Bonin, F., & Ganguly, D. (2019). Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 5203–5213). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1513

Hou, Y., Jochim, C., Gleize, M., Bonin, F., & Ganguly, D. (2021). TDMSci: A specialized corpus for scientific literature entity tagging of tasks datasets and metrics. In P. Merlo, J. Tiedemann, & R. Tsarfaty (Eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (pp. 707–714). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.59

Istrate, A.-M., Li, D., Taraborelli, D., Torkar, M., Veytsman, B., & Williams, I. (2022). *A large dataset of software mentions in the biomedical literature*. ArXiv. https://doi.org/10.48550/arXiv.2209.00693

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist, 11*(2), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Lane, J., Gimeno, E., Levitskaya, E., Zhang, Z., & Zigoni, A. (2022). Data inventories for the modern age? Using data science to open government data. *Harvard Data Science Review*, *4*(2). https://doi.org/10.1162/99608f92.8a3f2336

Lee, C. M. (2021). *Model summary*. GitHub. https://github.com/Coleridge-Initiative/rc-kaggle-models/blob/main/2nd%20Chun%20Ming%20Lee/Model%20Summary.pdf

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady, 10*(8), 707–710.

Li, P., Liu, Q., Cheng, Q., & Lu, W. (2021). Data set entity recognition based on distant supervision. *The Electronic Library, 39*(3), 435–449. https://doi.org/10.1108/EL-10-2020-0301

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach*. ArXiv. https://doi.org/10.48550/arXiv.1907.11692

Luan, Y., He, L., Ostendorf, M., & Hajishirzi, H. (2018). Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 3219–3232). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1360

Mikhail, A. (2021). *Strong, careful, simple: Solution report*. GitHub. https://github.com/Coleridge-Initiative/rc-kaggle-models/blob/main/3rd%20Mikhail%20Arkhipov/Coleridge_Report_Mikhail_Arkhipov_3d_place.pdf

Montani, I., Honnibal, M., Honnibal, M., Landeghem, S. V., Boyd, A., Peters, H., Samsonov, M., McCann, P. O., Geovedi, J., Regan, J., Orosz, G., Altinok, D., Kristiansen, S. L., Roman, Bot, E., Fiedler, L., Howard, G., Phatthiyaphaibun, W., Tamura, Y., . . . Henry, W. (2021). *explosion/spaCy: v3.1.2: Improved spancat component and various bugfixes (Version v3.1.2)*. Zenodo. https://doi.org/10.5281/zenodo.5226955

Nguyen, K., & Nguyen, M. (2021). *Model summary*. GitHub. https://github.com/Coleridge-Initiative/rc-kaggle-models/blob/main/1st%20ZALO%20FTW/MODEL_SUMMARY.pdf

OpenAI. (2023). *GPT-4 Technical Report*. ArXiv. https://doi.org/10.48550/arXiv.2303.08774

Otto, W., Zloch, M., Gan, L., Karmakar, S., & Dietze, S. (2023). *GSAP-NER: A novel task, corpus, and baseline for scholarly entity extraction focused on machine learning models and datasets*. ArXiv. https://doi.org/10.48550/arXiv.2311.09860

Pan, H., Zhang, Q., Dragut, E., Caragea, C., & Latecki, L. J. (2023). DMDD: A large-scale dataset for dataset mentions detection. *Transactions of the Association for Computational Linguistics, 11*, 1132–1146. https://doi.org/10.1162/tacl_a_00592

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., & Launay, J. (2023). *The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only*. ArXiv. https://doi.org/10.48550/arXiv.2306.01116

Polikar, R. (2012). Ensemble learning. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 1–34). Springer New York. https://doi.org/10.1007/978-1-4419-9326-7_1

Prasad, A., Si, C., & Kan, M.-Y. (2019). Dataset mention extraction and classification. In V. Nastase, B. Roth, L. Dietz, & A. McCallum (Eds.), *Proceedings of the Workshop on Extracting Structured Knowledge* (pp. 31–36). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-2604

Schwartz, A. S., & Hearst, M. A. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. In R. B. Altman, A. K. Dunker, L. Hunter, T. A. Jung, & T. E. Klein (Eds.), *Pacific Symposium on Biocomputing 2003* (pp. 451–462). IEEE. https://doi.org/10.1142/9789812776303_0042

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., . . . Scialom, T. (2023). *Llama 2: Open foundation and fine-tuned chat models*. ArXiv. https://doi.org/10.48550/arXiv.2307.09288

Yao, R., Hou, L., Ye, Y., Zhang, J., & Wu, J. (2019). Method and dataset mining in scientific papers. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 6260–6262). https://doi.org/10.1109/BigData47090.2019.9006262

Younes, Y., & Scherp, A. (2023). Question answering versus named entity recognition for extracting unknown datasets. *IEEE Access, 11*, 92775–92787. https://doi.org/10.1109/ACCESS.2023.3309148

Zhao, H., Luo, Z., Feng, C., Zheng, A., & Liu, X. (2019). A context-based framework for modeling the role and function of on-line resource citations in scientific literature. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language* (pp. 5206–5215). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1524

Appendix

$$\operatorname{lev}(d_y, d_{\hat{y}}) = \begin{cases} |d_y| & \text{if } |d_{\hat{y}}| = 0, \\ |d_{\hat{y}}| & \text{if } |d_y| = 0, \\ \operatorname{lev}(\operatorname{tail}(d_y), \operatorname{tail}(d_{\hat{y}})) & \text{if } d_y[0] = d_{\hat{y}}[0], \\ 1 + \min \begin{cases} \operatorname{lev}(\operatorname{tail}(d_y), d_{\hat{y}}) \\ \operatorname{lev}(d_y, \operatorname{tail}(d_{\hat{y}})) \end{cases} & \text{otherwise.} \end{cases} \quad (A.1)$$

The Levenshtein distance equation, adapted from https://en.wikipedia.org/wiki/Levenshtein_distance.

Algorithm 1. Submitted String Search Algorithm. The algorithm takes as input a list of target data sets D. Both the text and the data sets are converted to lowercase characters. The text is converted into a sequence of sentences by splitting the text on the period character [.]. If a target data set is found within a sentence, then the text immediately after is searched for a

pair of parentheses. If a detected pair of parentheses contains a sequence of at least two uppercase characters and that group is either the only text or is the first in the list when the text is split by either a semicolon or comma, then include that grouping as an additional detection.

```
Input: ASCII encoded text document T, List of target datasets D
Output: | delimited string containing datasets t
O ← set()
for d dataset in D do
    d ←LOWER(d)
    for s in t.split('.') do
        s_l ← LOWER(s)                          /* LOWER converts text to lowercase */
        if d in s_l then
            O ← O ∪ {d}
            if PARENSAFTER(d, s_l) then
                p ← TEXTINPARENS(d, s)
                if ALLCAPS(p) and LENGTH(p) > 2 then
                    O ← O ∪ {p}
                end
            end
        end
    end
end
t ←""
for o in O do
    t ← t + o+"|"                               /* + indicates string concatenation */
end
return t
```

Algorithm 2: Regular Expression Enhanced String Search Algorithm. The algorithm takes as input a list of target data sets D. These are data sets that we are actively searching for in the text. Each data set is converted into a regular expression. Depending on the number of words in the data set a different procedure is executed. If the data set is a single word and is in all caps, then the text is left as is because this is likely an acronym. If it is only a single word but not all caps, then any combination of upper and lowercase characters are searched for. This helps to handle mixed-case acronyms. Otherwise, the first letter of each word in a data set name is changed to search for either case.