Title

ScienceSearch: Enabling Search through Automatic Metadata Generation

Authors

Rodrigo, Gonzalo P

Henderson, Matt

Weber, Gunther H

et al.

Publication Date

2018-10-01

DOI

10.1109/escience.2018.00025

Peer reviewed

ScienceSearch: Enabling Search through Automatic Metadata Generation

Gonzalo P. Rodrigo, Matt Henderson, Gunther H. Weber, Colin Ophus, Katie Antypas, Lavanya Ramakrishnan
Lawrence Berkeley National Lab
Berkeley, CA 94720, USA
{gprodrigoalvarez, mhenderson, ghweber, clophus, kantypas, lramakrishnan}@lbl.gov

***Abstract*—Scientific facilities are increasingly generating and handling large amounts of data from experiments and simulations. Next-generation scientific discoveries rely on insights derived from data, especially across domain boundaries. Search capabilities are critical to enable scientists to discover datasets of interest. However, scientific datasets often lack the signals or metadata required for effective searches. Thus, we need formalized methods and systems to automatically annotate scientific datasets from the data and its surrounding context. Additionally, a search infrastructure needs to account for the scale and rate of application data volumes. In this paper, we present ScienceSearch, a system infrastructure that uses machine learning techniques to capture and learn the knowledge, context, and surrounding artifacts from data to generate metadata and enable search. Our current implementation is focused on a dataset from the National Center for Electron Microscopy (NCEM), an electron microscopy facility at Lawrence Berkeley National Laboratory sponsored by the Department of Energy which supports hundreds of users and stores millions of micrographs. In this paper, we describe a) our search infrastructure and model, b) methods for generating metadata using machine learning techniques, and c) optimizations to improve search latency, and deployment on an HPC system. We demonstrate that ScienceSearch is capable of producing valid metadata for NCEM's dataset and providing low-latency good quality search results over a scientific dataset.**

I. Introduction

Scientific discovery is becoming increasingly dependent on deriving insights from the growing amount of data collected at scientific facilities. While the data collected is valuable to the scientists running an experiment, the same data might also be useful to other scientists not involved in its production. Searching and sharing data can enable new scientific discoveries across disciplinary boundaries. Searching also provides access to historical data allowing users to compare it with data from a future experiment, that can sometimes reveal new insights from an old data set. For example, recent work [12] has shown that the existence of the Higgs Boson could have been confirmed through new analyses of previously collected LEP (Large Electron-Positron Collider [23]) data, which would have been significantly more cost-effective than building a new accelerator and running months of new experiments.

Search capabilities are fundamental to enable scientists to discover datasets of interest. However, datasets often lack the signals or metadata required for effective search since metadata generation is largely a manual and tedious task. The growth in scientific data volumes is making this even more tedious and complicated.

Our approach to automate scientific data exploration is inspired by the breakthroughs that enabled web search to transform our access to online information. Search engines, such as Google, rely on crawling and indexing large volumes of data in their cluster farms and use efficient algorithms to improve search for their end-users. Traditionally, Google relied mostly on algorithms such as PageRank, where the rules/signals were controlled by humans. However, more recently Google has started using machine learning (i.e., RankBrain [30]) to produce metadata and deliver search results.

In this paper, we present ScienceSearch, a generalized scalable search infrastructure that uses machine learning to capture metadata from data, context, and surrounding artifacts. Our current implementation focuses on the dataset from the National Center for Electron Microscopy (NCEM), a Department of Energy (DOE) facility at Lawrence Berkeley National Laboratory. NCEM's data contains millions of micrographs produced by hundreds of scientists. In this context, we identify multiple artifacts surrounding NCEM's data, including project proposals, publications, and the file system structure. ScienceSearch enables efficient search on NCEM data based on automatically generated metadata. Scientists can express their data needs as a text query in a web interface and receive a list of relevant micrographs, proposals, and publications within seconds. The internal search model aggregates metadata from multiple sources and balances their importance.

New methods to generate metadata and search results can only be improved with user verification and feedback. ScienceSearch incorporates user feedback by allowing users to explore the data and curate metadata—verifying existing data or providing new annotations that can be used to bootstrap the machine learning process. Additionally, user choices are recorded to improve search behavior.

The ScienceSearch infrastructure and methods are general and can be extended to apply to other domains or datasets. The novelty of ScienceSearch lies in its capacity to offer search for scientific datasets by analyzing the environment and artifacts surrounding them. Specifically, we make the following contributions:

- We describe our system architecture and a simple, yet powerful search model.
- We describe our methods for generating metadata using machine learning techniques from a number of artifacts including proposals, publications, file system structures, and features extracted from images.
- We describe our implementation, associated optimizations to improve search latency and its deployment on HPC systems. Our search infrastructure is deployed in production and allows users to search data produced by the TEAM I microscope at NCEM.

The remainder of this paper is organized as follows. First, we describe NCEM and its processes in Section II. We provide an overview over the ScienceSearch architecture in Section III, metadata generation methods in Section IV, and NCEM implementation details in Section V. We review related work in Section VI and finally present conclusions in Section VII.

II. National Center For Electron Microscopy

The National Center For Electron Microscopy (NCEM) is an electron microscopy user facility at the Molecular Foundry at Lawrence Berkeley National Lab. NCEM operates 11 electron microscopes, offering a wide range of capabilities for high resolution material characterization such as in-situ nanoindentation, spin-polarized low-energy microscopy, or tomography [4].

Scientists apply for microscope time at NCEM through a competitive process whereby a scientist writes a proposal describing research goals and experimental plans. The application is peer reviewed and, if accepted, the scientist is given a time slot during which he or she can use one of the microscopes to photograph samples. Images taken by a microscope are known as *micrographs* and they are the main type of data produced at NCEM. This data is stored on a network-connected file storage system and organized by the scientists utilizing the microscope time at NCEM. However, the micrograph files rarely include any metadata apart from microscope capture settings (e.g, exposure, contrast, signal voltage). Instead, users store metadata in diverse ways, e.g., in physical paper notebooks belonging to the individual scientists, elaborate naming conventions of files and directories, or simply by memory. This methodology (or lack of) does not facilitate the preservation of knowledge on the data. As a consequence, older micrographs slowly become more difficult to use due to the lack of information on the experiment. At the time of this publication, NCEM stores terabytes of micrographs in millions of files and a significant percentage of them lack metadata.

ScienceSearch aims to provide search over NCEM's data set to help users to find micrographs of interest. ScienceSearch generates new metadata by analyzing four main knowledge sources that we identify in NCEM: 1) storage file structure, capturing user's logic to organize data; 2) proposals and publications related to the images, which describe the purpose and goal of the micrographs; 3) image data, which can be modeled to identify common patterns between them; and 4) user feedback, which can add detailed information to a micrograph image. These four sources of information are selected since they provide valuable contextual information for search. Currently, only data that is public is included in ScienceSearch. ScienceSearch does not provide privacy or access controls. We assume data is only imported into ScienceSearch if it can be widely available.

III. ScienceSearch Architecture

ScienceSearch provides a generalized infrastructure to perform search over scientific data and artifacts. End users use a web-based interface to interact with the ScienceSearch infrastructure. They can search across data, publications and proposals, and provide feedback on the automatically generated tags.

ScienceSearch architecture is composed of four primary components that provide the functions required for data search (Figure 1). The four components include *data import or ingestion*, *metadata extraction*, *search engine* and *user feedback*. The *Data ingestion* component identifies the individual pieces of data and stores it in a database (center of the figure). There is a detailed representation that describes every data item in the database. Also, it includes metadata intrinsic to the data and a pointer to the file system where the data is located. The metadata required to enable search is produced by *metadata generation*. In the metadata generation, we use machine learning analysis on the data entries and artifacts (i.e., elements surrounding data that provide hints about it). We obtain text labels or tags that describe the purpose, scientific value, or production conditions of the data entries. The *user feedback* component gathers user input for the automatically-generated metadata tags. A user can curate and evaluate metadata produced by the generation process through a web interface. Users can explore data-entries, modify metadata tags, or provide new tags. User generated metadata also is used as a training dataset for the machine learning process that creates and refines the automated label generation. The *search* component provides search capabilities using a search model that relies on the generated metadata and crawled data entries.

A. Data ingest

The ScienceSearch infrastructure first curates the data that needs to be available through search. The data ingest component creates a database entry for each data item (e.g., an image) and captures information or metadata that is already accessible with the data. In the case of the NCEM dataset, an image is identified uniquely by its location on the NCEM file system. It is imported along with a few intrinsic metadata labels stored with the image file such as an acquisition timestamp, the dimensionality of the image, a json representation of the internal metadata generated by the microscope, and a URL pointing to a preview image of the micrograph. The exact data model in the database is domain-specific. Currently, we import SER, DM3, and DM4 formats from the micrograph files available at NCEM. We use parallelism to efficiently

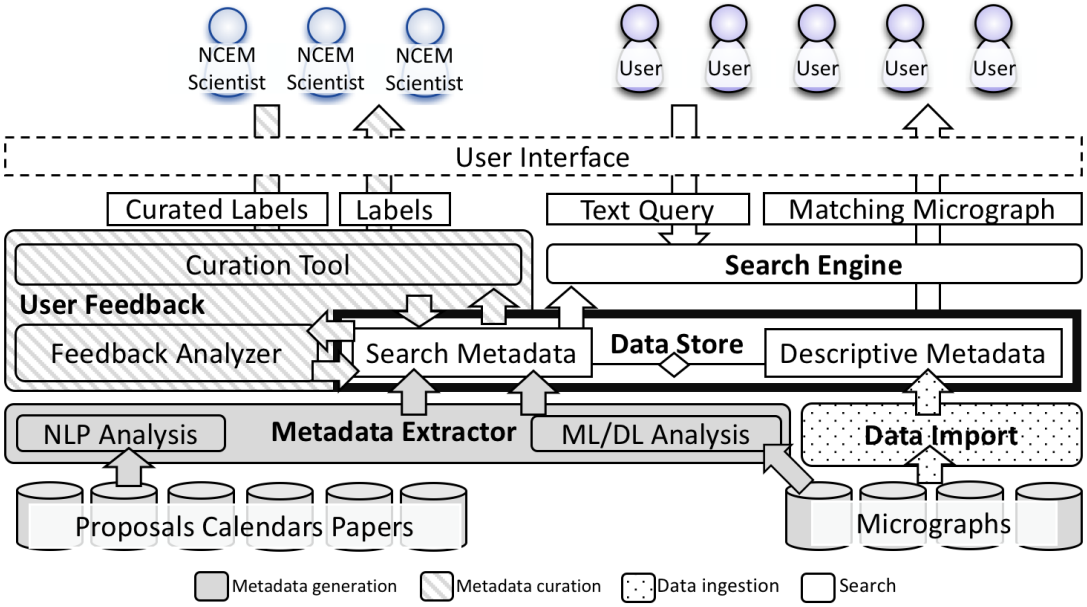

Fig. 1: ScienceSearch infrastructure includes components for data ingestion, metadata extraction, search, and user interface. Data entities are represented by square boxes, data flows by arrows, and components by round boxes.

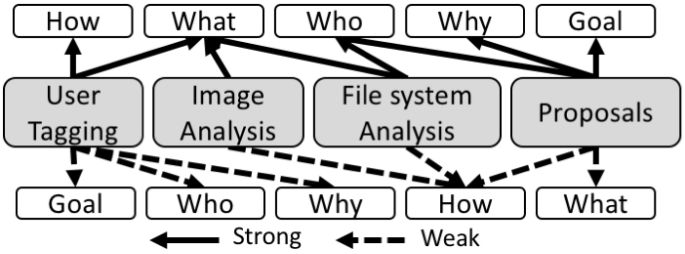

Fig. 2: Sources of metadata for NCEM's micrographs. Continuous arrows connect strong metadata sources with the semantics they express. Dotted arrows connect weak sources.

import NCEM's micrographs including exploring the file system structure, parsing the micrograph files, and creating the corresponding model entries.

Our current data model is unique to the NCEM dataset and must be recreated to support a different dataset. However, our implementation provides an easy reference point for other projects to easily add this component for other scientific datasets.

B. Metadata generation

The metadata generation tool extracts elements from the data and its environment that can provide hints about the data entries. We categorize these *metadata sources* in the following ways: 1) *artifacts*, any operational data that is part of the dataset creation workflow (including administrative); 2) *data store structural data*, logic and keywords from a scientist's effort to organize the dataset on the file system; 3) *data characteristics* that allows the data to be classified or grouped; and 4) *expert users*, that can provide direct knowledge on the data or curate metadata produced by other sources.

The metadata format produced by all methods is a tuple that contains (metadata_type, relevance_score, text_tag, pointed_object). The metadata_type indicates the source, pointed_object is the object (image, proposal, paper) associated with this metadata tuple, text_tag is a text string that describes the semantics of the metadata, and relevance_score is a real number expressing the relevance of text_tag to represent pointed_object. The elements of this tuple are used by the search engine process. Our generic metadata format allows us to easily plug-in other data sources and/or data from other domains into the ScienceSearch infrastructure.

The artifacts in the scientific ecosystem provide a rich context for the data. Each of the metadata sources may create tags or labels that are useful for searching the data. In particular, metadata sources help describe the content of data (*what*), the creation conditions (how), the owner or user (*who*), the research context (*why*), and scientific field (*goal*) of searchable data. The tags and labels created by each metadata source, may provide different degrees of certainty and accuracy in the context of search. For example, in NCEM, metadata tags derived from a proposal provide *strong* overall information about the project and context for the project micrographs. However, proposal descriptions are very general in their nature and tend to be a *weak* source of information on the actual contents of a particular image.

Figure 2 describes the sources that we identified in the context of the NCEM use case. We classify the sources by semantic content and strength based on information gathered from our users and early analyses of the data sources. The classification provides a high level guideline as to what can be expected from each data source. Quality or strength of individual artifacts might be variable. *NCEM's file system* structural information associates each micrograph with a file system path, which contains project and user names (who/strong), often the name of the sample (what/strong), and, in some cases, parameters of the image acquisition that are

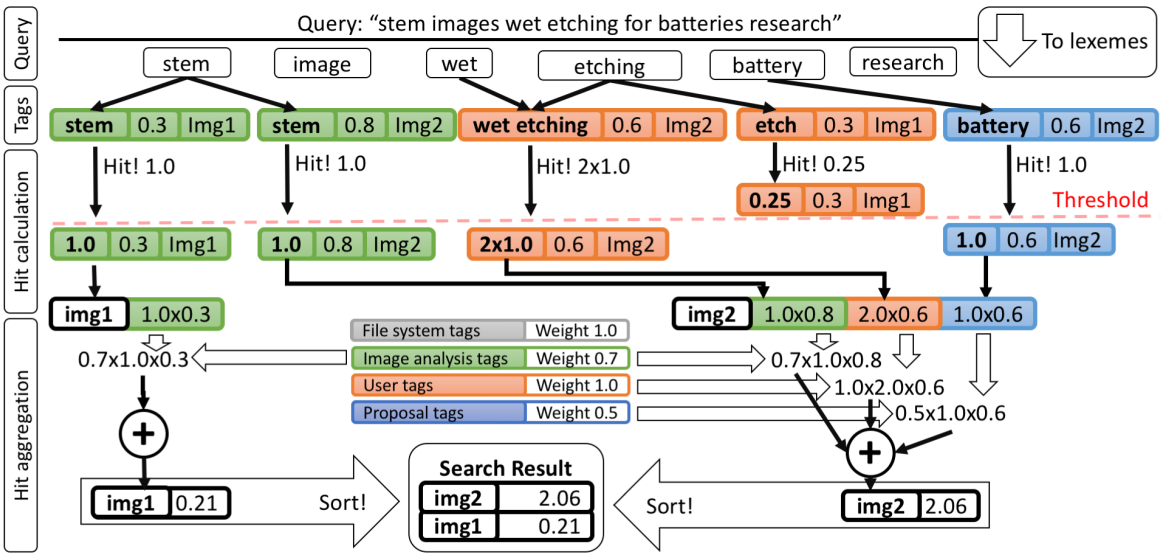

Fig. 3: ScienceSearch's search model: words in query are compared with metadata, producing "hits". Precise enough hits are grouped by image. Images search scores are the weighed sum of the score of their hits.

relevant (how/weak). *Application proposals and publications* are artifacts associated with images and describe the motivation (goal/strong) and research methodology (why/strong) to produce the micrographs. They might also contain some details on samples imaged (what/weak) and how they are captured (how/weak). *Feature detection analysis* models the relationship between known metadata and images. Feature models might capture the content of the images (what/strong) and characteristics of their capture (how/weak) such as the resolution or instrument configuration. Finally, *users* can provide feedback to curate existing metadata or provide new metadata. Users know data and describe its content (what/strong) and capture conditions (how/strong). Also, they might provide information on the overall methods (why) and purpose of the project (goal), but their knowledge on an particular image might be limited (weak). The metadata extraction methods are described in detail in Section IV.

C. Search

ScienceSearch utilizes the metadata extracted from the artifacts to provide search. In NCEM, search is provided over micrographs, facility proposals, and associated publications. Users submit a text query to a web interface, and the search engine returns a list of results ordered by their relevance.

Inspired by common search engine's architecture [29], we designed a model for ScienceSearch's search in three stages (Figure 3): 1) parsing user query; 2) comparison of the query with metadata; and 3) aggregation of the query comparison results. We describe the design and implementation of these stages in the context of ScienceSearch. Specifically, we describe it for the case of micrograph search, but they are also applicable to search on proposals and publications.

Query parsing. A user query is a string that is divided in words and each word is transformed to its lemma form. Transforming to a lemma form converts all syntactic variations of a word to the same string (i.e., the base or root form of a word). This simplifies the query term and metadata term comparison. For example, the lemma form of "looking" and "looked" is "look". Next, only nouns are retained, because the quality of the search results observed was higher if only nouns are taken into account in the queries and metadata. These steps are accomplished with the text annotation functions of the NLP library Spacy [7]. In Figure 3, the query string is transformed into *stem, image, wet* (as it can map to wetness), *etching, battery,* and *research*.

Hits calculation. Once the query is parsed, the resulting list noun lemmas (cardinality m) are compared with the text_tag of metadata entries in the database (cardinality n) producing a list of $m \times n$ hits. Each hit is a tuple of (hit_score, metadata_type, relevance_score, text_tag, pointed_image). In this tuple, hit_score is calculated using a lexicographic distance between the lemma of the the text_tag of the hit. This distance is a measure of the similarity of two strings. In particular, ScienceSearch uses the "Jaro-Winkler" [31] (J-K) lexicographic distance. The J-K similarity between two words is smaller if more characters need to be changed to transform one string into the another. However, character changes at the beginning of the string are considered more semantically important and make the similarity value even smaller than if they were positioned at the end. Figure 3, presents some of the comparisons between query terms and metadata in the *hit calculation* phase. For example, the query terms "wet' and "etching" produce two aggregated hits of similarity 1.0 with "wet etching" because now, the strings are identical. However, the hit of "etching" on "etch" produces a low hit score because three letters must be changed to transform one string into the other.

After the hit calculation, the engine keeps the hits that score higher than an empirically set threshold to filter out irrelevant metadata for the user's query. Following the example

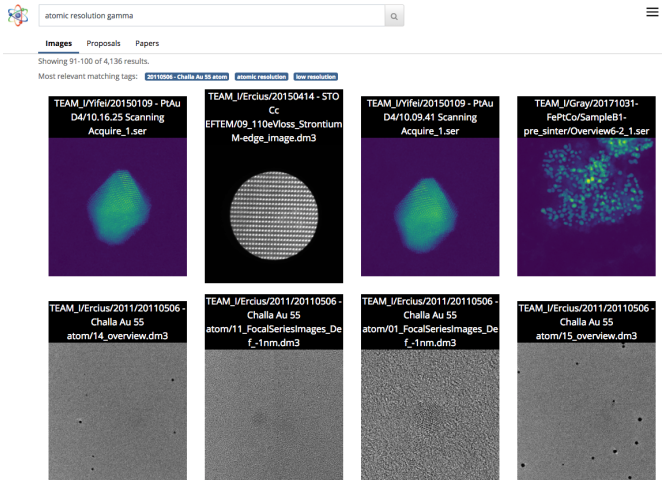

Fig. 4: Screen capture of the search function with a query searching for "atomic resolution gamma"

in Figure 3, only the hits for "stem", "wet etching", and "battery" are kept for the next phase.

Hits aggregation and search score calculation. The final score of each tuple is calculated as the product of hit_score, relevance_score, and a weight that depends on metadata_type. This calculation merges the significance of the hit_score (similarity between query and metadata) and relevance_score (large if the tag is important to the pointed object). Metadata weights are system configuration parameters to control the effect of the metadata types on the search results. In the example of Figure 3, user tags and file system tags have larger weights, so their tags score will be amplified in the image score calculation.

Once the final scores of the tuples are calculated, tuples are grouped by the micrograph they are pointing to (pointed_image). Finally, all the final scores in each group are summed to produce the search score of the pointed_image. The result of the search is the list of micrographs in descending order of their search scores.

Search quality adjustments based on user feedback. The search engine records which search results are selected by the users. This information is used to produce a report of how deep in the result list users explore to find their desired results. Search quality adjustments based on user feedback will be a long term research goal of the project. In the future, we propose to build a system that analyzes this feedback to infer: 1) a set of metadata_weights that could produce search results that would raise the position of the desired results; 2) if the quality of the metadata is good enough to serve the correct search results.

D. User Interaction

Users interact with ScienceSearch through a web interface[1] in two ways - a) they submit queries to search the data and b) they provide feedback on the metadata tags generated by the system.

[1]https://sciencesearch-ncem.lbl.gov

Search user interface. The search engine offers a web interface where users can submit queries and explore results. In the NCEM implementation, micrographs, proposals and papers can all be searched. Figure 4 presents the search results interface. On the top, the search box allows a user to input new text queries. Results can be explored in three tabs, one for each of the searchable data types: micrographs, facility proposals, and publications. Just under the search box (detail in Figure 5), the most relevant matching tags for a query are listed. The search results return a list of thumbnails and file system locations of the micrographs that best match the query. The user can click on any thumbnail image to explore a more detailed view. This view includes a higher resolution preview image, intrinsic file metadata, and hints that help the user understand why the search resulted in that image.

User validation of Metadata. ScienceSearch relies on automated machine learning methods that require feedback from users to improve and bootstrap learning. In commercial systems, user interactions with search results, although weak in semantics, are numerous and constitute enough feedback data to drive the learning [19]. Contrary to this, ScienceSearch's user base is small, but the users are experts and have the capability to provide highly semantic data that can generate additional metadata labels and provide feedback on existing labels. ScienceSearch includes a *tagging tool* to capture a user's high quality feedback. Before the users interact with ScienceSearch's tagging tool, an administrator loads the metadata set to to be evaluated. Tags are loaded under two categories: assigned (when the relationship between the metadata tag and data item is certain) or suggested (when the relationship between the metadata tag and the data item is less certain). Figure 6 shows the interface for NCEM data tagging. A user can select a data entry, such as an image or micrograph, to add new tags, validate a tag (confirm it as assigned or move a suggested tag to the assigned state), or invalidate an assigned tag. To simplify working with NCEM's large dataset, the ScienceSearch tool includes navigation features (route selector on top of the page) to limit the tagging and reviewing activities to a sub-folder of the file system. The navigation features are especially important from a user's perspective since images are typically grouped by experiment.

Once the metadata tags have been evaluated by users, the ScienceSearch tool produces new metadata tags that merge user feedback with existing metadata tags by removing tags invalidated by users, and adding new suggested ones. It also produces a report on which metadata tags have been rejected

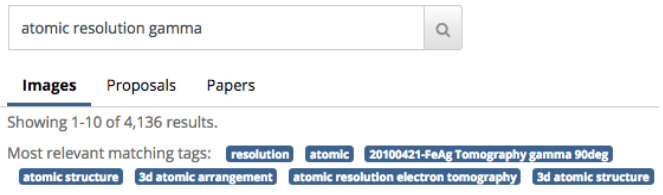

Fig. 5: ScienceSearch's search box. It presents the metadata that is relevant for the first results of the query.

Fig. 6: Screen capture of the user tagging tool

and from which source, such that the ScienceSearch learning models can be further refined.

IV. Metadata Extraction Methods

In this section, we present the methods used in ScienceSearch to extract formatted metadata for different types of artifacts. The extraction methods operate on specific data sources and outputs the metadata tuple format described in Section III-B. Each generated tag includes: 1) pointed_object, the object (image, proposal, paper) associated with this metadata tuple; 2) text_tag, a text string that describes the semantics of the metadata; and 3) relevance_score, a real number expressing the relevance of text_tag to represent pointed_object.

In this section, we describe the methods in ScienceSearch to extract metadata from various sources of data.

A. File system information

Scientific datasets are typically stored in shared file systems organized in folders and sub-directories. Scientists often encode metadata in the file names and the file system structure to help locate and find data. For example, in the following file path:
"TEAM_I/jdoe/20120523-STEM/05_FocalSeriesImages_Def_-4nm_-25eV_5eVslit.dm3" the folder directory names can be used as metadata text_tag for the image. From this directory structure (and hence metadata tags), we can infer that the pointed micrograph was taken at the TEAM_I microscope, was captured by the jdoe user (replaced for privacy), was taken on date 20120523, and used a technique called 'STEM' to capture the data. In this example, we also observe that higher folders correspond to the site organization and provide more general information (e.g., microscope, user). Meanwhile, deeper levels includes more specific semantic information that users chose to find and identify their own data (e.g., STEM, and capture parameters). The file name also includes rich information on the sample. For example, "05_FocalSeriesImages_Def_-4nm_-25eV_5eVslit.dm3" includes relevant terms such as the content of the image (FocalSeries), the order in some series (05), and parameters that would appear to be related to the capture parameters of the image (-4nm, -25eV, 5eVslit). To extract these semantics, the analysis module adds one tag for each the substrings in the filename split by common separators (e.g., "_", "-",) together with the complete filename (to support exact filename search). Similarly, other scientific collaborations often have specific formats for file structure to capture metadata. In its current implementation, relevance scores are set to the same value in all tags. In future implementations, we will explore how directory hierarchy affects relevance.

B. Text artifacts

Text artifacts, such as proposals and papers, can be associated with data, such as micrographs, to provide another set of important metadata tags. In the case of NCEM, users write proposals to gain access to the facility. Microscope access is organized in the form of a calendar with time slots assigned to proposals. Also, users write papers that include the produced images. ScienceSearch provides search over various text artifacts. The system includes models to import and capture the three text artifacts in NCEM: papers, proposals, and calendars. Many of these methods will directly apply to other domains, possibly including some domain-specific customization on context.

Proposals. The proposals data model includes title, submitter name, PI name, start date, end date, and staff contact. Also, it includes the proposal text which describes its significance, required NCEM capabilities, work tasks, and a project summary. The summary contains semantic information on why, how, and what the proposal is about. As a consequence, the proposal text is analyzed with Natural Language Processing methods to extract metadata tags that can describe image data related to a given proposal. Each proposal is uniquely identified by a serial number stored in the model. Proposals and their serial numbers are provided by NCEM in a formatted file that is imported into the ScienceSearch engine.

Calendar. Calendar entries map a user to a specific instrument, at a given time slot. This information is used to map an image capture timestamp with an instrument slot, and its corresponding proposal. This relationship allows the metadata tags generated from a proposal to be associated with the relevant micrograph images.

Papers. In ScienceSearch, papers are modeled to capture relevant information of the publication. The paper model includes title, authors, citations, publication venue, text body, and a URL pointing to the download page of the paper. Metadata on the paper is produced by analyzing the text body with NLP analysis, and papers can be connected to proposals (and thus images) by comparing their respective authors. NCEM provides a JSON-formatted list of publications

titles acknowledging the facility. A system was built to locate, download, parse, and import the articles in that list. Paper location and download was implemented over the "So Paper, So Easy" [34] (fetch) and Science Parse [9] (PDF parsing) libraries. Fetching functions extend browser-emulation fetchers that use Selenium [8] to interact with JavaScript based journal sites. Added fetchers work with the sites of *American Chemical Society, American Physical Society, IEEE, Institute of Physics, Nature, National Institutes of Health, Royal Society of Chemistry, Science Direct, Scitation.org*, and *Wiley online library*. To transform the PDFs into importable raw text, ScienceSearch utilizes "Science Parse" [9], a PDF parsing tool specific for scientific papers. Its output in JSON format is imported into ScienceSearch, together with the URL of the PDF. At this point, 2306 of the 3000 paper titles provided by NCEM have been located, downloaded, and imported. The metadata of a paper is extended to a proposal and its images if a correlation between authors and described work exist. This correlation analysis will be included in future deployment of ScienceSearch.

Natural Language Analyses. NLP analysis extracts and ranks the most relevant metadata in a paper or proposal. In each artifact, all text expressions are extracted and their relevance in the text ranked (i.e., how representative they are of the text). Then, ranks of the expression are adjusted, increasing the score of those more unique and decreasing the score of more common ones. This score correction promotes metadata that is better at differentiating data, which is a feature needed in metadata powering search. The NLP analysis employs the following techniques:

Individual ranked text expressions extraction is performed with the TextRank technique [22]. TextRank extracts a list of the most multi-word relevant expressions together with a score of their relevance in the text. This analysis is performed using the *pytextrank* library [25], which constructs a graph that connects words in expressions, expressions in sentences, and sentences into the whole text. Expressions are identified by their repeating appearance in the text, and their relevance is calculated similarly to Google's PageRank [27], i.e., the relevance of an expression is the sum of the relevance of the expressions pointing to it.

Context aware noun ranking is used to correct the expressions score in the context of a given artifact. Each text in an artifact (e.g. proposals) is parsed with a text analysis library (Spacy [7]) to extract each word and identify its function (verb, noun, adjective, adverb, or preposition). The method only keeps the nouns in their lemma form (e.g., for houses, the root form is house). Noun importance is calculated using a term frequency-inverse document frequency metric (Tf-idf). This metric expresses the capacity to differentiate a text piece within an artifact. Tf-Idf will be larger for more frequent terms, but smaller if the terms are frequent in all the analyzed artifacts. Tf-Idf's formula was adapted for its implementation in ScienceSearch to:

$$tfidf(w,i) = w_i * \frac{\log(1+n)}{1+\sum_{i=1}^{n} w_i} \quad (1)$$

where w is a word, i is a text id, w_i is the frequency of w in text i and n the number of texts in the corpus.

The final metadata for a particular expression tag is produced by correcting its score; by multiplying it by the average Td-Idf score of its nouns. This adjustment combines the power of TextRank to isolate expressions in a text and assess their differentiating importance within the text corpus.

C. Feature detection in images

In addition to the contextual information, we also extract metadata from the image data. Many scientific datasets are 2D images or can be visualized as such. In the current iteration, we use deep learning methods to suggest tags for new images, based on features learned from a manually labeled image subset, since they show promise.

Our automatic labeling approach considers each tag individually and trains a binary classifier to decide whether this tag applies to a new image or not. We use manually labeled images for which we know for certain whether that tag applies or not as training data. After training, we use the classifier to predict labels for all remaining images. The manual tagging tool presents these predictions as new, suggested labels, enabling users to confirm a prediction or to remove the label. User decisions are used to increase the size of the manually tagged training data set for subsequent automated labeling runs.

The classifier also returns its confidence in its label prediction, i.e., the estimated probability that the tag applies to the image. The search engine uses the predictions along with confidence information to identify and rank relevant images for a query containing this tag.

Creating an appropriately sized training data set. Manual labeling is a time-consuming process. Hence, we currently have only access to approximately 300 manually labeled images, an insufficient training data set size to choose an appropriate classifier or train all layers in a convolutional neural network (CNN). Due to operational procedures at NCEM, images using two operating modes of a microscope—transmission electron microscopy (TEM) and scanning transmission electron microscopy (STEM)—are saved using two different file formats. We can label images as TEM or STEM based on the file format. We use this method to generate a training and validation data set consisting of 500 TEM and 500 STEM images each, totaling 1000 images for each training and validation data set.

Implementing the classifier as convolutional neural network. We use a deep convolutional neural network (CNN) as binary classifier. To preserve important features in microscopy images, we cannot reduce the image resolution below 512×512 pixels. For deep learning, this is a relatively large input image size, compared, e.g., to ImageNet applications that pre-process images to 256×256 pixels. Due to the large input image size, we choose a simple CNN architecture, consisting of a few convolution layers, followed by a pooling layer and a few densely connected layers, to keep memory and processing power requirements for training at reasonable levels.

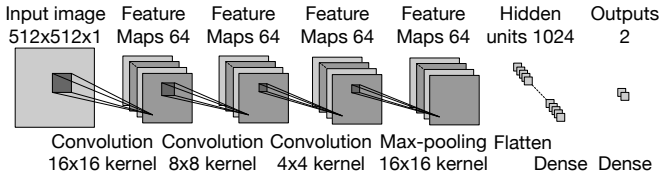

Fig. 7: Convolutional neural network architecture for transferring labels: Three convolution layers with kernel sizes of 16, 8 and 4 and a feature vector length of 64 are followed by a pooling layer with kernel size 16. The output of the pooling layer is flattened and used as input for a single densely connected layer comprised of 1024 neurons. The output of the network consists of two neurons for true and false.

In the current deployment of this work, we perform an extensive parameter search over networks with varying number of convolution and densely connected layers using an automatically generated training data set, to identify an appropriate CNN architecture.

Based on the results of this parameter search, we use the CNN architecture shown in Figure 7 for classification. We use a single dropout layer (dropout probability of 0.01) between the fully connected layer and the output layer for regularization and a batch size of 25 images. In the current deployment, for the TEM vs STEM classification task, we achieved a classification accuracy of approximately 80% on the separate validation dataset. Applying the resulting CNN to all images collected by the microscope achieved a similar accuracy.

Transfer learning. We use transfer learning to re-use the trained CNN in predicting other image tags. Specifically, we re-use the convolution layers of the CNN trained on TEM vs STEM classification and use the output values of the pooling layer as input of a new neural network that consists only of densely connected layers. If we use only a single densely connected layer with 1024 neurons, this approach is equivalent to "freezing" the values of the convolution layer and changing only weights of the densely connected layer. Using the output of the dropout layer allows us to pre-compute these output values for all images and subsequently train a much simpler neural network, reducing computational cost per label. Theoretically, this has the potential to reduce the minimum training set size required by the classifier. In future work that is outside the scope of this paper, the user feedback on the tags produced by this classifier will be analyzed to determine the quality of its predictions.

Estimating probabilities. To determine the probability that a given label applies to an image, we use the softmax function on the output of the CNN. The softmax function normalizes output values such that each value is in the range zero to one and that all entries add up to one. The softmax function is generally used to translate the output of a neural network into probabilities. Applied to the output of the binary classifier, using the vector value for an output of "true" this results in a probability estimate that a label applies to the given image.

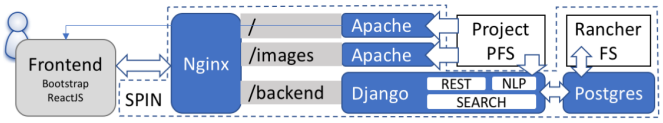

Fig. 8: ScienceSearch deployed in NERSC's Spin. Docker containers (round blue boxes) can access internal (rancher fs) and external HPC (project) storage.

V. Implementation and Optimizations

ScienceSearch is implemented as a web service wrapping a set of modules that can also be run independently. In this section, we describe the software and resources required to run ScienceSearch including details of its implementation and performance.

A. Software infrastructure

Each of ScienceSearch's components run within containers that provide portability, isolation, and scaling. For NCEM's ScienceSearch, containers are deployed in Spin, a service platform at the National Energy Research Scientific Computing (NERSC) Center that provides Docker container execution and has access to large-scale storage resources connected to NERSC supercomputers. Figure 8 shows the schema of the containers of ScienceSearch running in Spin (containers: round blue boxes; storage elements: white square boxes). When a user visits NCEM's ScienceSearch site, the browser connection lands in a container that hosts Nginx, a high performance load-balancer. This instance, provides SSL termination (certificates auto-generated with EFF's Letsencrypt) and balances the connections across the appropriate containers. During the first launch, the connection is redirected by Nginx to one of the Apache containers that serves the user interface files of ScienceSearch.

The user interface is implemented with ReactJS (v. 16) and Bootstrap (v. 4). ReactJS is a Javascript library that simplifies providing a smooth interactive user interface by enabling construction of reusable components, offering flexibility when incorporating additional libraries, and implementing a virtual DOM to quickly render updates to the browser [6]. The Bootstrap Javascript/HTML/CSS library simplifies templating, arranging, and styling of page content by offering pre-built UI components and a grid system for supporting responsive content [5]. The interface communicates with the backend container through HTTPS. This container is implemented over Django (v 2.) and exposes search and tagging functions through a REST API. Django is a Python-based framework that offers data models to abstract database representations, functions to develop REST APIs, and integration of external modules [1]. This container can import data from an external file system and is the only one with access to the PostreSQL container. PostreSQL (v.10) is used for data storage, since it is a reliable object-relational database that supports in-node parallelization [2]. This container uses Spin's internal storage, i.e., Rancher-fs, for its persistent files.

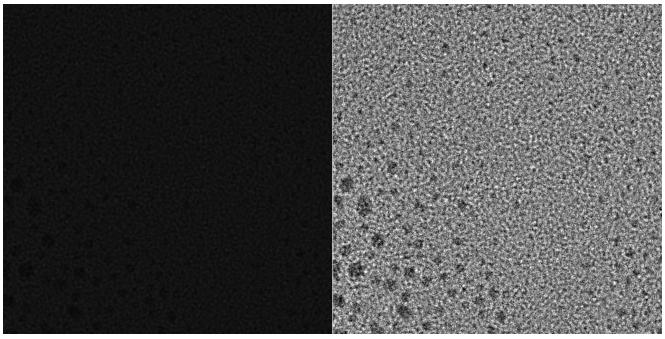

Fig. 9: Two preview images of the same micrograph of a carbon structure without (left) and with (right) correction.

B. Resources and scalability

Our current implementation of ScienceSearch in Spin utilizes compute and storage resources from NERSC. Copies and preview images of datasets is stored on NERSC's project file system. This GPFS [28] file system offers high capacity (50 TBytes quota), high read and write bandwidth (1̃30GB/s), and it is accessible from the facility's supercomputers, Spin infrastructure, and science gateways. Simple scalability is possible because all containers, except PostgreSQL, are stateless and can be replicated on demand. PostgreSQL supports concurrent access and, if needed, it can be easily replicated to serve read only queries.

C. Micrograph parsing and visualization

NCEM data is available in multiple formats including SER, EMI and DM3/DM4 and are represented as an array of numbers of variable dimensions. A library developed by NCEM (openNCEM [26]) was used to extract data. We extended the library to add memory caching to improve its performance by 100 times when reading from remote and parallel file systems.

For each imported micrograph, its internal metadata is captured, a human readable image (preview) is stored, and an entry for the image is inserted in the database. Generating previews for the micrographs requires different approaches depending on formats and data dimensions. One dimensional SER and DM3/DM4 micrographs (i.e., spectrum or sequence of numbers) are represented as a plot generated with a data analysis library (Hyperspy [13]). For data with two ore more dimensions, ScienceSearch creates a RGB bitmap preview accurate enough for a user to identify the micrograph. For two dimensional images, the preview is generated directly, However, micrographs with more than two dimensions (e.g., tomographies, movies) are sliced at middle points of each extra dimension. For example, if the micrograph captures the structure of cube (three dimensions), the preview image represents a horizontal section of the cube at half of its height.

Different image processing techniques were required for different file types. SER files are mapped on a bitmap directly because their value range is similar to the one in RGB bitmaps. However, DM3 and DM4 data have to be corrected because of the high range of its values. Otherwise, the data might be compressed and the resulting image might be hard to interpret due to poor contrast (Figure 9-left), To correct this, micrograph data is processed before mapping. First, the pixel value's mean and standard deviation are calculated. Then, the mean is subtracted from all the pixels and then divided by the standard deviation. Finally, values are capped to the range $[-3, 3]$ (i.e., no value diverges more than three times the standard dev. from the mean, skewing the mapping). Figure 9 shows a visually unrecognizable image if no correction is applied (left side), and how information reveals itself if applied (right side). These implementations highlight some of the challenges in working with scientific data of different formats. Our implementation is specific to NCEM's file formats and will be different for other sources of data. ScienceSearch's architecture separates the data ingest from the data format to enable the infrastructure to be used for multiple different domains with different file formats.

D. Efficient data ingestion

NCEM's dataset is large in size and number of files. For example, the micrographs from the TEAM I microscope occupy five terabytes of space in half a million files. We use parallelization and high performance computing resources from NERSC to handle the large dataset import. To parallelize this process, n instances of the ScienceSearch software are run. Each instance is instructed to import the files in a different location of the NCEM file system. All the instances access a single PostreSQL database that ensures data consistency while avoiding any blocking of the ScienceSearch instances.

Even if run in parallel, this process is very intense in terms of compute and I/O. We run it on Cori, a Cray XC40 supercomputer at NERSC with 64 hardware threads per node that has high speed access to the *project* file system where the dataset copies are stored. By using Shifter (NERSC's technology for HPC containers) [15], the Docker containers of ScienceSearch run in the supercomputer nodes without requiring major changes. After the processing is completed, new entries are loaded in bulk in the active instance of ScienceSearch in Spin. Combining both strategies, NCEM's TEAM I dataset can be ingested into ScienceSearch in less than 10 hours.

E. Search engine implementation optimizations

The search model's implementation is based on a module in the Django framework and its capabilities are exposed through a REST API. In its implementation, a number of optimizations were added to reduce search latency and resource requirements.

In a first approximation, the search model complexity is $O(n \times m)$, where n is the number of terms in the query and m is the number of metadata entries that the model searches over. This quadratic complexity usually presents small values of n (no more than ten), but large values of m (millions), so each search operation implies millions of data retrievals and comparisons. We details the other optimizations that aim to reduce n, m, and the cost of each individual operation.

Index of text_tag. To reduce m (metadata entries in search), a table with an index on metadata text_tag was created,

Fig. 10: Search metadata is sliced and processed by n workers that retrieve data, calculate hit scores, and image scores. Intermediate scores are aggregated to produce search results.

containing k entries, one for each unique text_tag. The hit calculation stage is performed against the index first, with a complexity of $O(n \times k)$. For NCEM micrographs, k is one order of magnitude smaller than m, as a consequence, since $k << m$, $O(n \times k) << O(n \times m)$. This optimization reduces significantly the number of operations in the hit calculation phase but increases the operations in the hit aggregation phase (Figure 3). Performance characterization experiments presented in Section V-E show that this secondary increase is not significant and search latency is greatly reduced with this optimization.

Multiprocess Parallelization. The hit calculation and hit aggregation operations are parallelized by dividing the input metadata in r slices processed by r worker processes running in the same node (Figure 10). A search worker reads its slice, calculates hit scores, and corresponding image scores for those which pass the threshold. When all workers end, intermediate image scores are aggregated to produce the final result. The search latency of this parallel approach is the runtime of the slowest worker plus the aggregation time. With r workers, the size of a metadata slice is $\frac{1}{r}$ of the total size, and the runtime of each worker should be at most $\frac{1}{r}$ of the runtime if $r = 1$. The resulting search latency is bound by $O(\frac{n \times k}{r})$ where r is the number of parallel processes employed.

Limited results and caching. Search query response might include tens of thousands of results, however the user interface can only present a limited number of results in a usable way. This allows for two optimizations. First, the search response returned through REST only includes a small number of results, reducing its size and the cost of micrographs data retrieval. Second, search results are cached in the engine for a period of time. If the same query is received while the cached copy is still valid, the cached result is served. This is especially important to support pagination, i.e., the interface requests new data each time users advance through the search results one page at a time. However, the complete search calculation is only executed the first time.

Search Performance. We measure search performance for a particular query to analyze the effects of the optimizations. The search is performed over 11,261,844 tags and 1,184,851 entries in the text index for 557,195 images. Experiments run over the production instance of ScienceSearch NCEM in Spin. The search engine and database run in the same compute SPIN node to avoid network bottlenecks. In the SPIN node, services shared two CPUs with 24 cores each, and two hardware threads per core (96 total) running over 256GB of RAM.

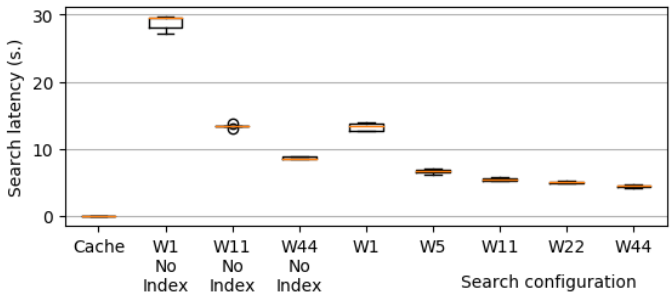

Fig. 11: Box-plot analysis of image search latencies for a query on a user name with different search optimizations in action. Cached queries are the fastest. Search latency is reduced by parallelization. Use of text index reduces search latency.

The test query is the name of an NCEM scientist who has authored many images. This query results in 4,968 micrographs which represents 1% of the total dataset (although all the metadata is utilized). Query latency is measured in five repetitions of the search for each optimization configuration. The load of the system is considered to be minimal at the time of experimentation. In Figure 11, we present a box-plot analysis of the observed query latencies. Results that are cached are returned within one tenth of a second. If results are not cached, the worst case (median 30s) is when no text indexing is used and only one worker calculates the search (W1, No Index). Increasing the number of workers reduces latency significantly. We observe that adding 10 workers, reduces the latency to values close to 14s (W11, No index). Increasing the number of workers, reduces the latency further until 44 workers. The use of the index over text_tag, reduces the latency too. For one worker, text indexing divides the latency by two (W1, No Index vs. W1). With text indexing, we also observe performance gains by increasing parallelism, again until 44 workers.

Finally, search latency over proposals (1,1650 tags for 45 proposals) and papers (297,768 tags for 2203 papers) is always under to 0.5 seconds.

VI. Related work

In this section we present work related to search models, metadata generation, and initiatives to organize scientific data to make it more accessible.

Search engines have organized and made internet data accessible for more than 20 years. In most engines and in ScienceSearch, search algorithms compare query terms to tags associated with search objects [29]. Classical models such as PageRank [27] analyze relationships between data objects to measure their relevance. However, current models rely on machine learning methods to assess this importance [30].

Metadata extraction on non-scientific data has also been extensively studied. Natural Language Processing (NLP) pro-

vides the capacity to understand [33] and summarize [14], human language. Methods like frequency analysis [18] and TextRank [22] identify and rank key expressions and terms in text. Machine learning provides the capacity of identifying and modeling data patterns (e.g., feature detection [17]). ScienceSearch relies on text tokenization [7], frequency analysis, TextRank analysis, and feature detection to create new metadata.

There has been numerous efforts to organize scientific data. For example, Mathematica [32] and Jupyter Notebooks [20] allow a user to attach runnable analysis methods, figures, and text to datasets. However, their interactivity with other systems (e.g., a search engine) is limited. Massive, human driven data annotation has been made possible by initiatives like Zooniverse [3], a crow-sourcing online platforms where anonymous users tag massive scientific datasets. Although powerful, scientific datasets often need expert user tagging and general crowd-sourcing techniques might not be sufficient.

There are notable efforts to build field-specific search engines. For example, the *Materials Project* [16] aims to accelerate materials discovery by providing search over individual material features. The *KBase* project [10] offers a unified search portal for DOE's biological data. Bioassayexpress [11] provides search over bioassays (experimental protocols in biology) and includes users as source the metadata and data curation. As a general effort, Apache Lucene [21] is an open-source search engine popular to build scientific portals. All these projects provide powerful search tools, however they are either narrow in their objective, focused on text-only data, and do not analyze the context of the data to infer metadata, limiting their capacity.

Finally, Google recently released Dataset Search, a general search engine for scientific data [24]. This engine provides dataset search across scientific fields and its algorithms are adapted to the keywords used in science. However, in contrast to ScienceSearch, Dataset Search does not include tools for semantic discovery that analyze data content or context. For a dataset to be included in this engine, scientists must manually separate each data set and provide metadata tags that describes it.

VII. Conclusions

In this work, we present the architecture of ScienceSearch, a generalized scalable infrastructure to provide search over scientific datasets. We use different methods including machine learning to extract metadata from four sources of metadata identified – proposals and publications, file system structure, image features, and user input in the context of NCEM.

Our performance evaluation shows that ScienceSearch is capable of executing simple queries on a single node, over 11 million metadata tags in less than five seconds. Early testing by expert users of NCEM indicate that result quality matched with the expected results for their test queries. Future work will improve our search model and explore methods to algorithmically assess search quality results.

Acknowledgment

This material is based on work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR) and resources used at the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility, supported by the Office of Science of the U.S. Department of Energy, both under Contract No. DE-AC02-05CH11231. We thank NCEM's staff for their help in providing data and insights that made this work possible. We also thank NERSC's Spin team for supporting the deployment of this work.

References

[1] Django project. *https://www.djangoproject.com/*.
[2] What is PostgreSQL? *https://www.postgresql.org/docs/current/static/intro-whatis.html*.
[3] Zooniverse: People-powered research.
[4] NCEM Capabilities & Tools, http://foundry.lbl.gov/facilities/ncem/expertise.html.
[5] Bootstrap · The most popular HTML, CSS, and JS library in the world, https://getbootstrap.com/.
[6] React - A Javascript library for building user interfaces, https://reactjs.org/.
[7] Industrial-Strength Natural Language Processing, https://spacy.io/.
[8] Selenium, browser automation, https://www.seleniumhq.org/docs/03_webdriver.jsp.
[9] Allen Institute for Artificial Inteligence. Science parse.
[10] A. P. Arkin, R. L. Stevens, R. W. Cottingham, S. Maslov, C. S. Henry, P. Dehal, D. Ware, F. Perez, N. L. Harris, S. Canon, et al. The doe systems biology knowledgebase (kbase). *bioRxiv*, page 096354, 2016.
[11] A. M. Clark, B. A. Bunin, N. K. Litterman, S. C. Schürer, and U. Visser. Fast and accurate semantic annotation of bioassays exploiting a hybrid of machine learning and user confirmation. *PeerJ*, 2:e524, 2014.
[12] K. Cranmer and I. Yavin. Recast—extending the impact of existing analyses. *Journal of High Energy Physics*, 2011(4):38, 2011.
[13] F. de la Peña et al. Hyperspy 1.3.
[14] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *journal of artificial intelligence research*, 22:457–479, 2004.
[15] L. Gerhardt, W. Bhimji, S. Canon, M. Fasel, D. Jacobsen, M. Mustafa, J. Porter, and V. Tsulaia. Shifter: Containers for hpc. In *Journal of Physics: Conference Series*, volume 898, page 082021. IOP Publishing, 2017.
[16] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *Apl Materials*, 1(1):011002, 2013.
[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
[18] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.
[19] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.
[20] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.
[21] M. McCandless, E. Hatcher, and O. Gospodnetic. *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co., 2010.
[22] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.
[23] S. Myers and E. Picasso. The design, construction and commissioning of the cern large electron–positron collider. *Contemporary Physics*, 31(6):387–403, 1990.
[24] G. Natasha Noy. Making it easier to discover datasets. https://www.blog.google/products/search/making-it-easier-discover-datasets/.

[25] P. Nathan. Pytextrank, a python implementation of textrank for text document nlp parsing and summarization. https://github.com/ceteri/pytextrank/, 2016.
[26] NCEM. openncem, https://github.com/ercius/openNCEM.
[27] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
[28] F. B. Schmuck and R. L. Haskin. Gpfs: A shared-disk file system for large computing clusters. In *FAST*, volume 2, 2002.
[29] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Citeseer, 2005.
[30] Wikipedia. Rankbrain, https://en.wikipedia.org/wiki/RankBrain.
[31] W. E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer, 1999.
[32] S. Wolfram. *The mathematica*. Cambridge university press Cambridge, 1999.
[33] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM, 2012.
[34] Y. Wu. So Paper, So Easy, https://github.com/ppwwyyxx/SoPaper.