

A Scalable Architecture for Extracting, Aligning, Linking, and Visualizing Multi-Int Data

Craig A. Knoblock and Pedro Szekely

University of Southern California, Information Sciences Institute, Marina del Rey, CA, USA;

ABSTRACT

An analyst today has a tremendous amount of data available, but each of the various data sources typically exists in their own silos, so an analyst has limited ability to see an integrated view of the data and has little or no access to contextual information that could help in understanding the data. We have developed the Domain-Insight Graph (DIG) system, an innovative architecture for extracting, aligning, linking, and visualizing massive amounts of domain-specific content from unstructured sources. Under the DARPA Memex program we have already successfully applied this architecture to multiple application domains, including the enormous international problem of human trafficking, where we extracted, aligned and linked data from 50 million online Web pages. DIG builds on our Karma data integration toolkit, which makes it easy to rapidly integrate structured data from a variety of sources, including databases, spreadsheets, XML, JSON, and Web services. The ability to integrate Web services allows Karma to pull in live data from the various social media sites, such as Twitter, Instagram, and OpenStreetMaps. DIG then indexes the integrated data and provides an easy to use interface for query, visualization, and analysis.

Keywords: big data, information integration, analysis, visualization

1. INTRODUCTION

There is a huge variety of data available that can be exploited to understand what is going on in a given area or to answer specific intelligence questions. The available data includes GEOINT data (e.g., satellite imagery, motion imagery, mapping data, etc.), MASINT data (e.g., LIDAR, SAR, hyperspectral data, etc.), SIGINT data (e.g., ELINT, COMINT, etc.), HUMINT data (e.g., intelligence reports, documents, etc.), and OSINT (e.g., news articles, photos, videos, telephone books, web sites, publications, social media, domain registrations, etc). The challenge is to allow an analyst to easily integrate or fuse the various sources available to answer specific intelligence questions or to detect anomalies.

To address this problem, we developed the Domain-specific Insight Graph (DIG) system, an innovative architecture for extracting, aligning, linking, and visualizing massive amounts of domain-specific content from both structured and unstructured sources. The DIG system harnesses state-of-the-art open source software combined with an open architecture and flexible set of APIs to facilitate the integration of a variety of extraction and analysis tools. DIG builds on rich models of a domain that support fine-grained data collection, organization, and analysis. DIG builds a graph of the entities and relationships within a domain using scalable extraction and linking technologies. DIG also includes a faceted content search interface for users to query the data and visualize information on maps, timelines, and tables. DIG is designed to be scalable by building on open-source cloud-based infrastructure (i.e., HDFS, Hadoop, Elastic Search, etc.), supports a diversity of source types, and is rapidly re-targetable to new domains of interest. We have already successfully applied this architecture to multiple application domains, including human trafficking, where we extracted, aligned and linked data from 50 million online web advertisements. We have also applied it on publications and patents in material science to understand the research trends in that area.

Consider the case where one wants to determine the nuclear know-how of a given country by analyzing the open source data for the universities within a country, which would include the people, publications, and affiliations of the physicists within the universities. In this paper, we describe how we would build a specific domain-specific insight graph (DIG) by analyzing the open-source data available to build a highly detailed

For additional information, contact Craig Knoblock: knoblock@isi.edu

Seyed Mohammad
 Professor, **Chair**
 Research Interests: Thin Films, Nanophysics, Optoelectronics, (...)
 Email: (...)@sharif[dot]edu
[more ...](#)
 Homepage: <http://phys.sharif.edu/web/...>

Abdollah
 Professor, **Vice-chair for education**
 Research Interests: Strongly Correlated Electron Systems
 Email: (...)@sharif[dot]edu
[more ...](#)
 Homepage: <http://sharif.edu/...>

Department of Physics **Seyyed Mohammad** **Information**

Web Content Display

Contact
 Room 435
 P.O.Box: 11155-9161
 Tel: +98 21 (...)
 Fax: +98 21 (...)
 E-Mail: (...)@sharif.edu (...)@sharif.edu
 Personal Page: <http://sina.sharif.edu/...>

Recent Publications

1. Ranjbar M., Fardindoost S., (...) S. M., Iraj-zad A. & Tahmasebi N., "PdCl₂: characterization and gasochromic properties", Solar Energy Materials
2. S. Fardindoost, A. Iraj Zad, S. M. (...) "Hydrogen Sensor Based on Sur Engineering and Technoloav 76. (2011) 639-642.

Figure 1. Sample pages with researcher profiles and publications

knowledge base about the scientists within a country (we are using this application to illustrate the process and have not built this particular application.) Such a knowledge base would include all of the open source information that is available about such scientists, including their education histories, organizational affiliations, connections to other scientists, publications, areas of research, and so on. The data about such scientists is spread over a huge number of sources, including university web pages, individual home pages, social media web sites, and online databases. Figure 1 shows two sample pages with data that we would like to acquire and integrate into the DIG knowledge base. Creating this DIG requires identifying the relevant web sites to crawl, creating domain-specific feature extractors, and building a domain vocabulary to align the data in this domain. Once that initial application work is done, the DIG architecture will be able to extract the data from the aggregated pages, align the data to the domain vocabulary, link the scientists to organizations and other scientists, and provide a visualization of the results. The DIG architecture makes it possible to perform this task on the millions of available web pages, rapidly creating a comprehensive knowledge base that is not available today. In addition, DIG provides an analysis dashboard that enables users to query the database and discover connections between individuals, academic institutions, research topics, conferences, publications and other concepts as they evolve over geography and time. In the remainder of this paper, we describe each step in the process to create the domain-specific insight graph about scientists within a given country.

2. CRAWLING

The DIG crawling capability, illustrated in Figure 2, is designed to download just the relevant to an application domain. A key challenge is to enable data scientists building a DIG to configure the crawling capability to download only those pages that contain relevant information. Restricting a crawler to remain within a domain (e.g., usc.edu) is not restrictive enough. In our sample application, the crawler would download all pages of a university rather than just the pages from the physics department.

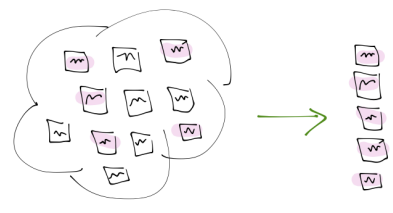


Figure 2. Crawling acquires data from the web

DIG uses the Apache Nutch framework to support crawling at scale, enabling the construction of integrated datasets consisting of millions of pages. Nutch offers a RESTful configuration interface that makes it easy to specify the URL patterns to be crawled, to monitor crawling progress and to define revisit cycles to re-crawl periodically, downloading revisions to already crawled pages. To provide further control over the crawled pages, we are integrating a semi-structured content extractor into Nutch. The extractor can identify specific elements within a page, such as the researcher home pages in Figure 1, and direct Nutch to follow only those links within a page. Without the extractor, Nutch would follow links that point to the university home page and start downloading all pages in a university.

In addition to providing access to the shallow (or surface) web, DIG also provides capabilities to access the deep web, i.e., web pages residing behind web forms and REST services, which are inaccessible to crawlers that acquire pages by following links. To do so, DIG uses Karma, which provides a scripting language that enables data scientists to easily compose RESTful URLs based on previously downloaded data, and the ability to subsequently invoke these URLs to download data. Together, Nutch and Karma provide the ability to download relevant data for a domain.

3. EXTRACTING

After crawling, the next step is to extract features from the harvested data to produce a structured representation that can be used for indexing and linking in the next step of the DIG construction pipeline.

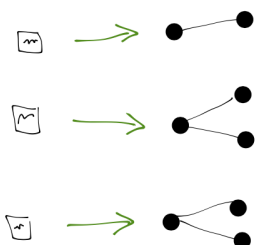


Figure 3. Extraction identifies useful data in web pages and constructs a structured representation of it

Given the wide variety of pages and data on the web, it is infeasible to develop data extraction technology that works for any page and any type of data. DIG provides an open architecture that makes it easy to integrate a wide range of extraction technologies, so that data scientists can select the extraction technology most appropriate for the pages in their application domain. As illustrated in Figure 3, an extractor is a component that takes as input a web page or a text document and produces as output a structured data representation (JSON, XML, CSV, etc.) Data scientists can configure the DIG workflows to include a wide variety of data extractors as long as they comply with the simple input/output conventions described above. In addition to providing an open data extraction architecture, DIG also provides components for extracting data from semi-structured pages and from text.

The DIG semi-structured page extractor, called the *landmark extractor*, identifies elements in a page using landmarks defined with regular expressions. The key feature of this extractor is that it can identify lists of elements (e.g., all the publications in Figure 1, it can identify records within lists (e.g., each paper consists of authors, title, etc.) and it can recurse to identify lists within lists (e.g., the individual authors in each publication). For example, the extractor to extract the metadata for scientific publications in Journal of Artificial Intelligence is defined using five simple rules. The rule that extracts the page numbers of an article in a journal uses two landmarks, as shown below. The desired data is between the text “ , pages ” and “</cite>”:

```
{
  "name": "pages",
  "rule_type": "RegexRule",
  "begin_regex": ", pages ",
  "end_regex": "</cite>"
}
```

DIG provides a learning component that automatically infers rules for the landmark extractor from examples. To train an extractor, the data scientist provides a collection of pages and corresponding extractions (e.g., a persons home page containing a list of publications and the data elements that should be extracted from the page). Using a handful of examples, the learning component automatically creates a landmark extractor that extracts data from similar pages.

To support extraction from text, DIG offers a capability to enable data scientists to easily train extractors specialized to an application domain. For example, we used DIG in the human trafficking domain and needed to extract data elements such as eye-color, hair type and color, and ethnicity from escort advertisements in web sites. To train a new extractor for a text corpus, a data scientist highlights the desired data elements in small number of sample sentences or short paragraphs, selected from the corpus. For example, in the sentence “Perfect Green eyes Long curly black hair Im a Irish, Armenian and Filipino”, the data scientists highlights “Green eyes” and “Long curly black hair”. After the data scientists designates a text corpus and defines the examples, DIG automatically constructs thousands of tasks to acquire additional annotations for the desired

data elements on randomly selected paragraphs from the corpus. DIG uses the examples provided by the data scientist as instructions for workers, and automatically deploys thousands of annotation tasks on the Amazon Mechanical Turk crowd sourcing platform. By default, DIG requests three annotations on each task and uses the majority answer when workers provide conflicting answers.

After a few days, once workers complete the tasks, DIG automatically downloads the responses, and uses the annotations as training data to train an automatic text labeler. Once trained, the text labeler can take as input any text from the corpus, automatically annotate it, and produce the annotated text as output. For example, after training a text labeler on annotations for eye color and hair characteristics, the labeler can automatically extract these characteristics from new text. Our learning component uses a Conditional Random Field model and simple domain independent features. In our approach, data scientists do not need to understand the machine learning technology or worry about defining features for the learning algorithms.

We evaluated our approach on news wire text, training it to extract persons, organizations and places, and comparing it with the Stanford NER.¹ Our approach achieves 78% precision, 3 percentage points within the performance of the Stanford NER, but our approach is easily trainable on new corpora and for arbitrary domain-specific attributes.

4. CLEANING

Since much of the data that we use to build a DIG comes from online sources, a significant problem is dealing with noisy data. As shown in Figure 4, the tools for creating a DIG must support this data cleaning task, which often involves incorrect extractions, unwanted extractions, or other types of inconsistent data. Consider the publications shown in Figure 1. In the first publication, one of the co-authors is listed as "Iraji-zad A." and the second publication lists one of the co-authors as "A. Iraji-zad". If we can correct these types of inconsistencies, it will improve the accuracy of the entity resolution described in a later step.

In previous work we developed Karma, a tool to semi-automatically clean and integrate data from a variety of sources.² Within Karma, we have developed several methods to support the cleaning and normalization of data.

- **Visualization:** Karma provides a histogram for each attribute, which shows the distribution of data values for a particular attribute. This allows a user to quickly see if there are irregularities in the data that might indicate a problem. For example, one can immediately see if there are null values where none are expected or if there are duplicate entries when all entries are expected to be unique.
- **Learned Transformations:** Karma provides a capability to learn transformation rules by example.³ A user simply provides examples of the desired format for a few examples and the system generates a transformation rule, applies it to the relevant data, and displays the results. The user can then confirm the transformation or provide additional examples to refine the learned transformation rule.
- **Cleaning programs:** Some types of data cleaning can be quite complex, so Karma also allows one to write data cleaning scripts in Python, which will support arbitrarily complex transformations on the data. This makes it possible to filter rows, which can be used to eliminate bad records, or specific rows in a nested table. Cleaning scripts can be used to remove html from the extractions or to remove data that does not parse correctly.

5. INTEGRATION

After cleaning, the DIG knowledge base will contain a large number of structured data records. Some of these records come from databases or other structured representations such as XML, JSON or CSV. Many of the records are the output of a variety of extractors that extract data items from web pages or text documents.

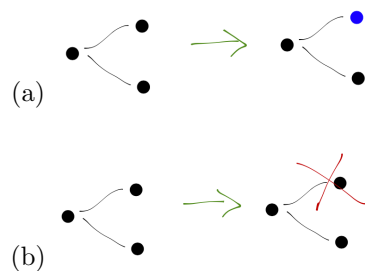


Figure 4. Data cleaning corrects incorrect extractions (a) and removes unwanted extractions (b)

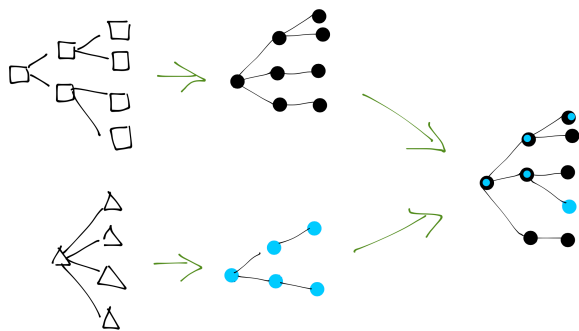


Figure 5. Schema alignment followed by data integration

At this stage of processing, the records consist of data items represented in different schemas (different shapes in Figure 5), and the only connections between data items are those that were present in the page from where data items were extracted. The goal of the integration step is to convert all records to a uniform schema defined by a domain ontology, and to unify records that contain unique identifiers such as an email address (Figure 5).

We also use Karma to support the integration of data across sources. One of Karma's unique capabilities is that it learns semantic descriptions of the sources.⁴⁻⁶ These descriptions define a mapping from the contents of a source (e.g., the extracted features or the columns in a spreadsheet) to a domain ontology

that the user specifies. Karma uses these descriptions to integrate data from multiple sources, merging fields only when their meaning is the same, and publishing the integrated data in any of a variety of formats, including XML, RDF, and JSON.

Karma plays an important role in DIG by providing the mapping of both the extracted data as well as structured data sources into shared domain ontology. This makes it possible to bring data together from various web sources, databases, spreadsheets, etc. and align all of the data using the same vocabulary. This mapping directly supports further analysis and reasoning on the data.

Figure 6 shows a screenshot of Karma with the data extracted from a web page containing publications and includes the authors, titles, abstracts, countries of affiliation, and so on. The bottom part of the figure contains

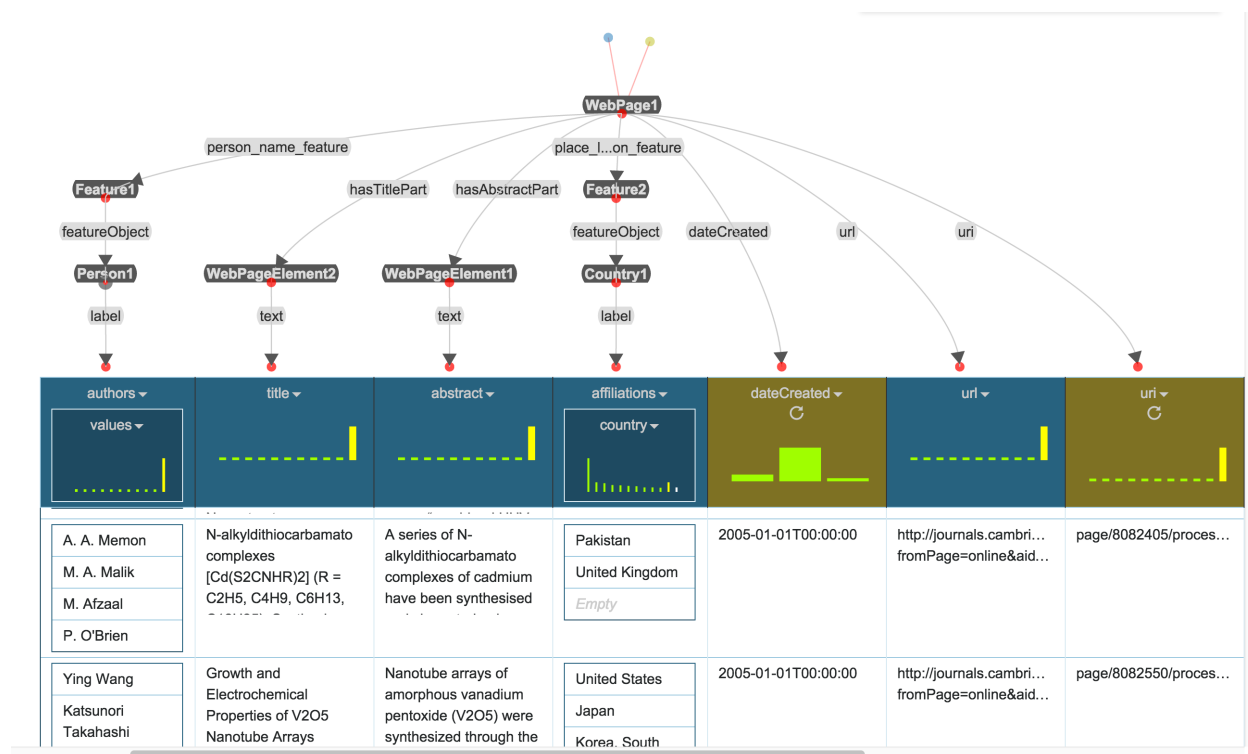


Figure 6. Screenshot of Karma showing the model of the publication data

the extracted data fields, which can contain multi-valued attributes. The histogram for data cleaning are shown immediately above the columns of data. And the source model is shown by the graph at the top of the figure. The model provides the mapping from the data to the domain ontology and gives the precise relationships between the sources. In the model shown in the figure, the dark ovals are the classes of information and the links labeled with gray rectangles are the relations between the classes. For example, the overall class is a `WebPage` and it has a set of relations including `hasTitlePart` and `hasAbstractPart`, which in turn have `WebPageElements` that link to the actual text. An initial source model is proposed by the system using machine learning techniques and then the user refines the model using an easy-to-use interactive graphical user interface (GUI).

The integration of the data is performed by creating the source model for each of the different types of extracted data. In the example from Figure 1, there would be one model for the pages containing the faculty members and another model for the pages containing publications. After modeling each of these sources, Karma can then generate the data in a standard language called JSON-LD (`json-ld.org`) using a the consistent set of terms from the domain ontology. We then use this aligned data in the following steps of the DIG pipeline.

6. SIMILARITY

After the integration step, data will be represented in the schema of the domain ontology and records containing unique identifiers are merged. Most pages and records do not contain identifiers, so the knowledge base at this stage of processing consists of a large number of small data islands represented in a homogeneous schema. The goal of this stage of processing is to augment the knowledge base with links between similar data items (Figure 7).

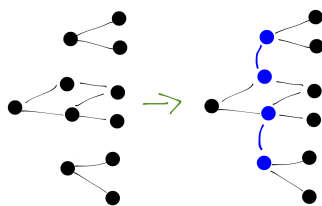


Figure 7. Similarity analysis

DIG provides capabilities to compute similarity for images and for text data. DIG’s image similarity capability uses DeepSentiBank, a deep convolutional neural networks approach.⁷ The approach extracts over 2,000 features from each image and computes hashcodes that can be used to retrieve similar images. An important benefit of this approach is that there is no need to train the similarity algorithms with images in the domain of interest. In our human trafficking application we used this approach with a database of 20 million images. The system is able to find identical and near duplicate images for an unseen image in less than 2 seconds. For example, given a photo of a person, the system can find other photos of a person taken in similar settings (e.g., in the same room) even if the person is in a different pose.

DIG uses Minhash/LSH algorithms⁸ to compute similarity on text data, as these algorithms can scale to large datasets containing hundreds of millions of documents. These algorithms work by computing random hashing functions on the tokens of a document, and can find pairs of similar items in a large dataset in $O(n * \log(n))$ time. Minhash/LSH computes an approximation of Jaccard similarity, defined as the ratio of tokens two documents have in common over the combined number of tokens in the two documents.

To use these algorithms, DIG needs to associate a document with each data record, and then run the Minhash/LSH algorithms over the associated documents. DIG provides a library of tokenization methods to compute the tokens that form the document associated with a data record. If a data record contains sentences or larger texts, then the document can be formed using the words in the document, or word n-grams (sequences of several words). If the data records contain small values such as names of people or geographic locations, then the document can be formed using character n-grams (sequences of several characters). These n-grams are useful because they allow the algorithm to find similar items when they use slightly different spellings for words.

To find similar data items, a data scientist first uses the tokenization library to define appropriate tokenization schemes for the data items. Then, the data scientist designates a property in the ontology to record data item similarity and adds a step to the data workflows to compute the similarity over the complete data set. It is common to compute similarities over different data items. For example, in a scientist domain we may compute similarity of research profiles to identify scientists working on very similar topics; we may compute similarity between publication records to identify multiple copies of publications; etc.

7. ENTITY RESOLUTION

The next step in the DIG pipeline is to find the matching entities (often called entity resolution). Consider our running example with the data extracted from web pages about scientists. The entities are scientists, organizations, and publications and each of these entities has a set of properties associated with them. The task in this step is to determine which data corresponds to the same entities. For example, if the same scientist is extracted from multiple pages, we would want to determine that all of the associated information corresponds to the same scientist. Or when publications list multiple authors, we would want to find the connection to the researchers that might be listed on other university faculty pages. The output of this step would be a set of explicit links between entities extracted from different sources. The objective, illustrated in Figure 8 is to discover the relevant entities in the data (blue circles in the figure), and to associate the appropriate entities with the relevant records.

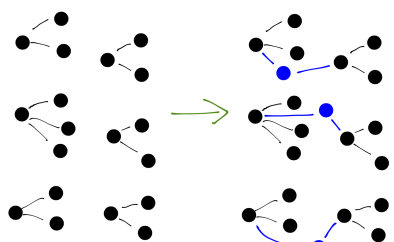


Figure 8. Entity resolution determines the matching entities

DIG addresses two variations of the problem. The easier case is when there is an appropriate reference dataset that contains all the relevant entities. For example, GeoNames (geonames.org) is a comprehensive geographical database containing over 2.8 million populated places, so it can be used as a reference set for cities. In DIG, we use GeoNames as a reference set for populated places, so entity resolution for cities becomes the problem of mapping mentions of cities to the appropriate entity in GeoNames (e.g., mapping the string “Los Angeles, CA” to the record identifier for the city of Los Angeles in California).

To solve this variant of the entity resolution problem, a data scientist first uses Karma to map the reference dataset to the ontology being used. Then, the data scientist uses the similarity analysis discussed in the previous section to compute similarities between records in the reference dataset and other records in the knowledge base. The output of the similarity matching step is a small number of candidate entities for each entity mention in the knowledge base. Often, to identify entities accurately, it is necessary to define matching algorithms tuned to the entity type. Data scientists can define custom matching algorithms or use classifiers such as support vector machines (SVM) to define custom matching components to determine whether a mention should be matched with an entity. This final step is not subject to the same scalability requirements as the previous similarity step. As mentioned in the previous section, the similarity step is highly scalable and yields a small number of similar items for each entry. Consequently, the final custom matching step is allowed to be quadratic without affecting overall scalability.

The second variant of the entity resolution problem addresses the case when there is no reference set for the entities of interest. For example, there is no reference set for the set of physics researchers. For these cases it is necessary to infer the set of entities from their mentions. DIG represents each entity as a set of features (e.g., a person can be represented by a name feature, birth-place and work-place features, a research interests feature, etc.) To define entities, DIG first converts each mention to a set of features and defines a new entity for each mention. This first step creates redundant entities given that multiple mentions may refer to the same entity. The second step eliminates the redundant entities using a clustering approach similar to Swoosh.⁹ Using the techniques described in the previous section, a data scientist directs DIG to compute similarities between potentially redundant entities. When the similarity between entities is above a certain threshold, DIG combines the entities by defining a new entity whose set of features is a combination of the set of features of the component entities. This technique works well with strict thresholds that lead to a conservative policy for combining entities. Strict thresholds lead to small, but correct clusters (high precision). The down-side is that entities that should be combined do not get combined (e.g., DIG may have two entities for a single person). Less strict thresholds leads to a higher likelihood of correctly combining all the entities that should be combined, but also leads to situations where entities are combined incorrectly (higher recall, lower precision). The downside of lower thresholds, is that the feature sets of the combined entities grow and make it more likely for an increasing number of entities to be incorrectly combined. For this reason we encourage data scientists to use strict thresholds as this increases the likelihood of creating correct entities.

Entity resolution is a critical step in the analysis process since it provides a way of finding the data that refers to the same entities, which in turn allows us to find the interesting connections between entities. In the case of integrating the data about scientists, the entity resolution combines the data about the individual scientists with their publications and co-authors on those publications. This allows us to build a graph of the connections between scientists, organizations and co-authors, which can be used to find the clusters of scientists that work together and especially those groups of scientists that have specific areas of expertise, such as those required to create nuclear weapons.

8. GRAPH CONSTRUCTION

The steps described in the previous sections operate on datasets that can have hundreds of millions of records. In the DIG approach, the crawling and extraction steps consume and produce heterogeneous data representations. The data integration step converts all data to a homogeneous schema as defined in a domain ontology, and converts all data to a homogeneous representation as JSON-LD documents. The similarity and entity resolution steps that follow consume and produce JSON-LD documents.

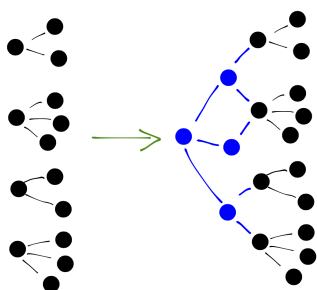


Figure 9. All data is integrated into a graph that can be queried and analyzed

DIG stores crawled data in the Hadoop File System (HDFS), and uses the Oozie workflow system (oozie.apache.org) to automate all processing steps. The output of the entity resolution steps are a collection of JSON-LD document, one per entity (depicted on the left side of Figure 9) Naturally, the number of documents can be in the hundreds of millions.

DIG can deploy the data in multiple formats, tuned to multiple uses:

HDFS: the integrated dataset is available as a collection of sequence files in HDFS. This representation is appropriate for applications that will to run additional analyses using Hadoop.

ElasticSearch: a highly scalable document store supporting keyword and structured query (elastic.co).

Graph databases: including RDF stores such as OpenLink Virtuoso (virtuoso.openlinksw.com) and native graph databases such as Titan (thinkaurelius.com).

DIG supports bulk processing in Hadoop, enabling the processing of large amounts of data (e.g., 50 million documents in 2 hours in a 20-node Hadoop cluster). This processing pipeline enables data scientists to experiment with different ontologies, different similarity and entity resolution approaches, and to run large scale experiments in a reasonable amount of time. In addition, DIG supports incremental processing using the same Oozie workflows, updating the deployed datasets periodically (e.g., every hour). The DIG components are also designed to run in Apache Storm (storm.apache.org) to support real-time incorporation of crawled data. Data scientists can define appropriate Storm topologies to support real-time processing.

9. QUERY, ANALYSIS, & VISUALIZATION

Supporting efficient querying of large datasets is challenging. A query, represented as a graph pattern over the green arrow in Figure 10, selects subgraphs that match the pattern. The choice of data store determines the types of queries that can be answered efficiently as well as a variety of other trade-offs relating to loading time, replication, transactions, and a range of other issues beyond the scope of this paper. As mentioned in the previous section, DIG supports deployment of the integrated dataset in multiple kinds of data stores to give developers options to navigate the trade-off space.

For interactive query, DIG deploys the dataset in ElasticSearch, which provides efficient support for keyword querying, for faceted browsing, and aggregation queries.

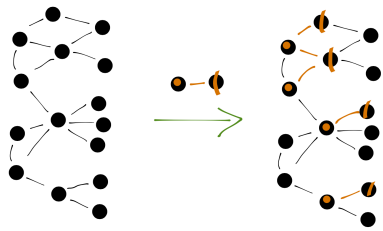


Figure 10. Queries are graph patterns that selects subgraphs of a larger graph supports a variety of visualizations types including bar charts, line charts, tag clouds, maps and social network displays.

Figure 11 shows a screenshot of the DIG user interface. The user is looking for publications from Iran that contain the word “carbon” in the abstract. The interface paradigm is similar to that of popular web sites such as Amazon (amazon.com). The interface provides a box for keyword search at the top, shows ranked results in the main panel below the search box, provides facets on the left to enable users to filter the results by category. This interface paradigm makes DIG easy to use given that this paradigm is familiar to millions of users who routinely purchase goods on the Internet.

The visualization at the top of Figure 11 shows a timeline of all publications that match the search criteria. The visualization changes dynamically as the user refines the query. In addition to time lines, DIG

10. DISCUSSION

In this paper we described the DIG system and discussed how it could be applied to the problem of assessing the nuclear know-how of a country based on open source data. DIG can pull data from a combination of web pages and databases, extract and clean the data from those sources, integrate the data across sources, link the entities within the sources, build a graph of all of the data, and then query the data to solve specific analytical problems. DIG is not limited to this specific application and has already been applied to the problem of combating human and labor trafficking and understanding the research trends in the field of material science. The underlying tools and technology are widely applicable and can be applied to many other applications, such as integrating and analyzing multi-INT data

In future work, we plan to refine the tools and technology to make it easier and faster to build new applications. In particular, we are developing improved tools for rapidly training information extractors for new applications domains. We are working on highly scalable entity-linking and entity resolution techniques, which will allow us to both find entities within the set of extracted data as well as to resolve the entities with known entities in another source. And we are working on improving the machine learning techniques for automatically modeling new sources, which will improve the accuracy and speed of integrating new sources of data.

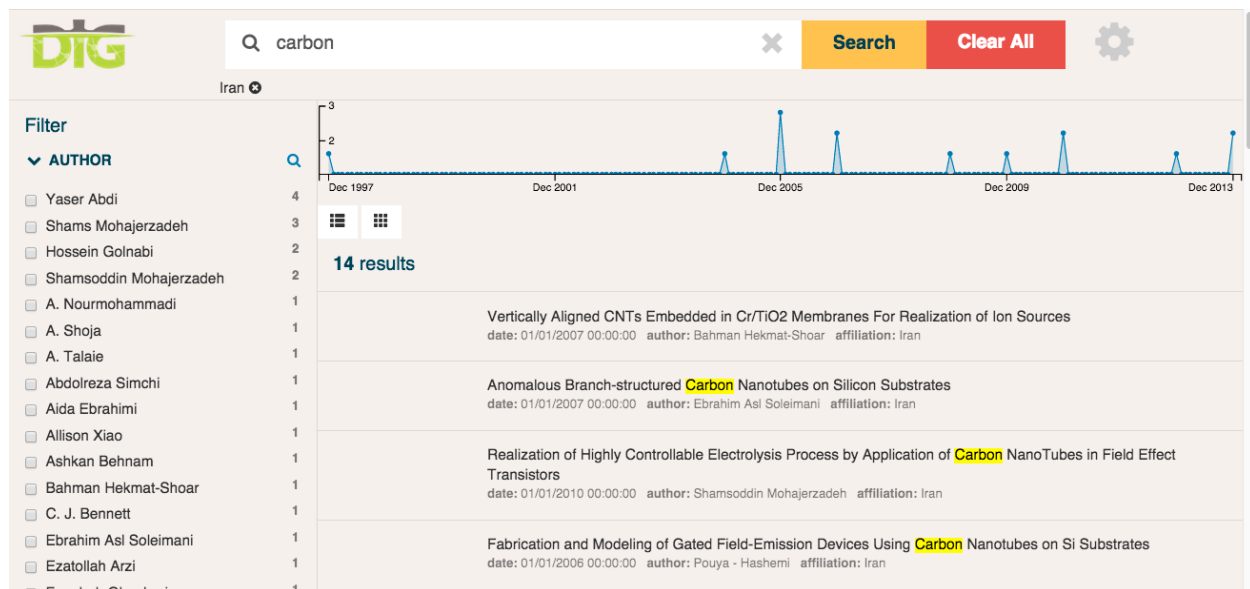


Figure 11. Screenshot of DIG query interface showing results for keyword “carbon” and authors from “Iran”

ACKNOWLEDGMENTS

This research is supported in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract number FA8750-14-C-0240, and in part by the National Science Foundation under Grant No. 1117913. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, NSF, or the U.S. Government.

REFERENCES

- [1] Finkel, J. R., Grenager, T., and Manning, C., “Incorporating non-local information into information extraction systems by gibbs sampling,” in [*Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*], 363–370, Association for Computational Linguistics (2005).
- [2] Knoblock, C. A. and Szekely, P., “Exploiting semantics for big data integration,” *AI Magazine* (2015).
- [3] Wu, B. and Knoblock, C. A., “Iteratively learning conditional statements in transforming data by example,” in [*Proceedings of the First Workshop on Data Integration and Application at the 2014 IEEE International Conference on Data Mining*], 1105–1112, IEEE (2014).
- [4] Knoblock, C. A., Szekely, P., Ambite, J. L., , Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyani, M., and Mallick, P., “Semi-automatically mapping structured sources into the semantic web,” in [*Proceedings of the Extended Semantic Web Conference*], (2012).
- [5] Taheriyani, M., Knoblock, C. A., Szekely, P., and Ambite, J. L., “A graph-based approach to learn semantic descriptions of data sources,” in [*Proceedings of the 12th International Semantic Web Conference (ISWC 2013)*], (2013).
- [6] Taheriyani, M., Knoblock, C. A., Szekely, P., and Ambite, J. L., “A Scalable Approach to Learn Semantic Models of Structured Sources,” in [*Proceedings of the 8th IEEE International Conference on Semantic Computing (ICSC 2014)*], (2014).
- [7] Chen, T., Borth, D., Darrell, T., and Chang, S., “Deepsentibank: Visual sentiment concept classification with deep convolutional neural networks,” *CoRR* **abs/1410.8586** (2014).
- [8] Leskovec, J., Rajaraman, A., and Ullman, J. D., [*Mining of massive datasets*], Cambridge University Press (2014).
- [9] Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., and Widom, J., “Swoosh: A generic approach to entity resolution,” *The VLDB Journal* **18**, 255–276 (Jan. 2009).