GeoDeepDive: Statistical Inference using Familiar Data-Processing Languages [*]

Ce Zhang Vidhya Govindaraju Jackson Borchardt Tim Foltz
Christopher Ré Shanan Peters

University of Wisconsin-Madison, USA
{czhang@cs., vidhya@cs., jsborchardt@, tlfoltz@, chrisre@cs., peters@geology.} wisc.edu

ABSTRACT

We describe our proposed demonstration of GeoDeepDive, a system that helps geoscientists discover information and knowledge buried in the text, tables, and figures of geology journal articles. This requires solving a host of classical data management challenges including data acquisition (e.g., from scanned documents), data extraction, and data integration. SIGMOD attendees will see demonstrations of three aspects of our system: (1) *an end-to-end system* that is of a high enough quality to perform novel geological science, but is written by a small enough team so that each aspect can be manageably explained; (2) a *simple feature engineering system* that allows a user to write in familiar SQL or Python; and (3) *the effect of different sources of feedback* on result quality including expert labeling, distant supervision, traditional rules, and crowd-sourced data.

Our prototype builds on our work integrating statistical inference and learning tools into traditional database systems [3, 2]. If successful, our demonstration will allow attendees to see that data processing systems that use machine learning contain many familiar data processing problems such as efficient querying, indexing, and supporting tools for database-backed websites, none of which are machine-learning problems, *per se*.

Categories and Subject Descriptors

H.0 [**Information Systems**]: General

Keywords

Statistical inference; demonstration; geoscience

1. INTRODUCTION

There has been an explosion of data sources that contain valuable information but are difficult for an application developer to use. This difficulty stems from several characteristics of these new data sources: the structure of the data may be unknown to the developer, the data may be less structured than traditional relational data, or it may be in a non-ASCII format, e.g., scanned PDFs. In a rough analogy with *dark matter* in physics, which is the unseen mass of the universe, some have coined the term *dark data* to describe this wealth of data that sits beyond the reach of application developers. While each of the problems surrounding dark data have been studied in isolation (in some instances for decades), an application writer faces the problem of acquiring, extracting, and integrating data in a holistic way.

We propose to demonstrate GeoDeepDive, our ongoing effort to build a dark-data extraction system to support geoscience. Our system is currently being built in collaboration with geologists. Our goal is to support novel geological science, e.g., understanding the carbon cycle and characterizing the organic carbon content of Earth's crust. Currently, geoscience research is conducted at the microscopic level, i.e., one can find out information about the handful of formations that a research group can read about or personally visit. Building on Shanan Peters' Macrostrat database[1], a PostgreSQL-based database of geological facts about North America, our hope is that GeoDeepDive can enhance this database to provide one of the most comprehensive, data-backed *macroscopic* views of the Earth. Similar macroscopic views have been assembled by hand by our collaborators – at great expense – have yielded remarkable insights about the Sulfur cycle [1]. Our hope is that a system like GeoDeepDive can help accelerate this type of science.

We build on our group's recent work on scalable statistical learning, inference, and acquisition techniques [7, 6]. Our goal is to demonstrate three aspects of our system.

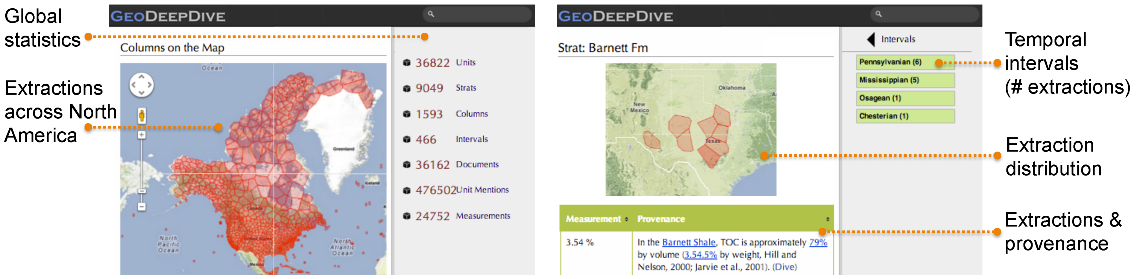
(Aspect 1) End-to-end System. We demonstrate an end-to-end dark data system that delivers results that are of a high enough quality to be used in geological science. However, as the system was built by a small team, it is small enough that a SIGMOD attendee can see the pipeline end to end.

(Aspect 2) Simpler Feature Engineering. Our intended users are scientists, not computer scientists. To support them, we tried to abstract the messy, but routine, details of machine learning as possible. We developed a framework that allows one to express features in popular scripting and programming languages, e.g., Python

[*] A live demo of GeoDeepDive is available at `http://hazy.cs.wisc.edu/demo/geo`. A video walkthrough of GeoDeepDive can be found at `http://www.youtube.com/watch?v=X8uhs28O3eA`.

[1] `http://macrostrat.org/`

SIGMOD'13, June 22–27, 2013, New York, New York, USA.
Copyright 2013 ACM 978-1-4503-2037-5/13/06 ...$15.00.



36822 Units
9049 Strats
1593 Columns
466 Intervals
36162 Documents
476502 Unit Mentions
24752 Measurements

Measurement	Provenance
3.54 %	In the Barnett Shale, TOC is approximately 79% by volume (3.54.5% by weight, Hill and Nelson, 2000; Jarvie et al., 2001). (Dive)

(a) GeoDeepDive homepage

(b) Temporal view of TOC extractions in Barnett Formation

Figure 1: GeoDeepDive Front-end System.

and SQL. We believe it will be interesting to many SIGMOD attendees that some statistical inference and learning tasks do not require specialized languages.

(Aspect 3) Different Sources of Feedback. Our system is able to take several types of feedback [6]: crowd-sourced labels, human expert labels, distant supervision, and traditional hard rules or constraints. One aspect of our demonstration is understanding how each type of feedback contributes to various quality metrics. We will provide sliders where attendees can use differing amounts of each type of feedback and see the results compared to a geologist-provided gold standard.

The remainder of this paper is organized as follows: Section 2 walks through the GeoDeepDive front end and back end. Section 3 describes our intended demonstration scenarios in detail.

2. SYSTEM WALK-THROUGH

We introduce both the front-end and back-end systems of GeoDeepDive.

2.1 GeoDeepDive Front-end System

Figure 1 shows the front-end system of GeoDeepDive. Geologists whose intention is to use GeoDeepDive as a research tool are the main consumers of GeoDeepDive's front-end system. A geologist uses the front-end system to learn about different entities, e.g., rock formations, locations, temporal intervals, etc. When designing the front-end system, we aggregate and expose the extractions to geologists to support their research.

Figure 1(a) shows the homepage, which gives an overview of all extractions in GeoDeepDive. GeoDeepDive extracts mentions for each type of entity, and relations among entities. As shown in the global statistics column, we have about 500K extractions for rock units and 24K measurements for rock formations from 122K geology journal articles.[2] These extractions are distributed across the entire U.S. and Canada. The map consists of a set of polygons, each of which represents a geological area. The opacity of each polygon represents the number of extractions in that area.

Geoscientists can dive into a single entity to see extractions aggregated by different dimensions, e.g., temporal interval, location, etc. For example, Figure 1(b) shows all Total Organic Carbon (TOC) extractions for Barnett Formation organized by temporal intervals. In this example, GeoDeepDive shows extractions for four temporal intervals: Pennsylvanian, Mississippian, Osagean, and Chesterian. Geoscientists can click on a temporal interval to see extractions and their provenance.

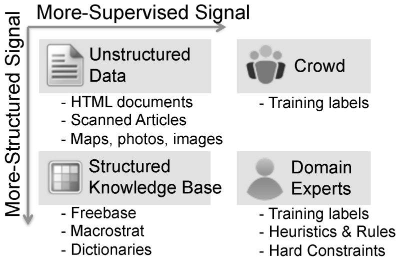


Figure 2: Signals that GeoDeepDive uses

2.2 GeoDeepDive Back-end System

The back-end system of GeoDeepDive extracts mentions of entities and relations among entities from geology journal articles for the front-end system. To achieve this with high quality, GeoDeepDive takes a diverse set of signals, as shown in Figure 2. The design goal of GeoDeepDive's back-end system is to make the integration of these resources easy, and abstract as much of the messy machine learning details as possible.

GeoDeepDive uses Elementary [7], our machine reading system, to extract mentions and relations. The same framework has been applied to other domains, e.g., the framework built DeepDive [4] for web-scale relation extraction from news articles. In DeepDive, recall was important, but precision was less important. However, in GeoDeepDive the results need to be used for science so high precision (95%+) is required. We briefly describe Elementary, and how it was used to develop GeoDeepDive.

Data Model. Elementary has a very simple data model. However, as we will see later, this data model can still support sophisticated statistical learning and inference.

Elementary uses a relational data model, i.e., relation-in-relation-out. The input of the system is a corpus of documents D and a set of relations $\bar{V}$. Each document $d_i \in D$ consists of a set of possibly overlapping *text spans* (e.g., tokens, phrases, or sentences) $T(d_i)$. Define $\mathcal{T}(D) = \cup_{d_i \in D} T(d_i)$. The relations in $\bar{V}$ may contain training ex-

[2] This number is as of April 11, 2013.

amples, text representation of PDFs, dictionaries of place names, or complete databases, e.g., the entire Macrostrat relational database. The output of our system is a set of relations that contains extracted and integrated data.

To represent the target knowledge base, we adopt the classic Entity-Relationship (ER) model: the schema of the target knowledge base is specified by an ER graph $G = (\bar{E}, \bar{R})$ where $\bar{E}$ is one or more sets of entities, and $\bar{R}$ is one or more relationships. Define $\mathcal{E}(G) = \cup_{E \in \bar{E}} E$, i.e., the set of known entities. To specify an extraction task to ELEMENTARY, one provides the schema G and the text corpus as text spans $\mathcal{T}(D)$. Our goal is to populate the following tables:

1. Entity-mention tables $M_E(E, \mathcal{T}(D))$ for each entity type $E \in \bar{E}$.
2. Relationship-mention tables $M_{R_i} \subseteq \mathcal{T}(D)^{k+1}$ for each $R_i \in \bar{R}$, where k is the arity of R_i, the first k attributes are entity mentions, and the last attribute is a relationship mention.
3. Relationship tables $R_i \in \bar{R}$.

Both M_E and M_{R_i} provide provenance that connects the knowledge base back to the documents supporting each fact. We call the process of populating M_E *entity linking*, and the process of populating M_{R_i} *relation extraction*. Intuitively, the goal is to produce an instance J of these tables that is as large as possible (high recall) and as correct as possible (high precision).

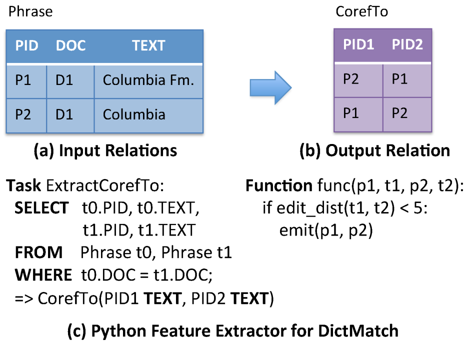
To create and populate new relations, ELEMENTARY goes through a standard three-phase pipeline: feature extraction, statistical learning, and statistical inference. We briefly introduce these three phases, and the reader can consult [7] for details.

Feature Extraction. In this phase, ELEMENTARY produces feature relations based on evidence relations and other previously extracted features. A feature relation is a standard database table with an arbitrary schema. A feature extractor is a Python function or SQL query provided by users.

Figure 3 shows an example feature extractor for coreference, in which we decide which mentions are coreferent. The input relation `Phrase` contains a list of phrases. The output relation `CorefTo` maps each phrase to the phrase that is coreferent with it. To write a feature extractor, a developer needs to provide two pieces of information.

1. **Schema:** A developer needs to define the input relation and the schema of the output relation. In Figure 3, the input relation is specified by an SQL query, which produces phrase pairs that appears in the same document; the output relation is `CorefTo` the output of which is a set of pairs of phrases that are coreferent.
2. **Function:** A developer then provides a Python function that populates the output relation. In Figure 3(c), the Python function processes each row of the SQL results, and outputs (emit) a tuple into `CorefTo` if the edit distance (edit_dist) between two phrases is smaller than 5.

Given a set of feature extraction functions, ELEMENTARY will populate the specified relations by running Python functions as UDFs and SQL over the input relations. ELEMENTARY uses standard database techniques to run this computation in parallel.



Phrase

PID	DOC	TEXT
P1	D1	Columbia Fm.
P2	D1	Columbia

CorefTo

PID1	PID2
P2	P1
P1	P2

(a) Input Relations

(b) Output Relation

```
Task ExtractCorefTo:
 SELECT  t0.PID, t0.TEXT,
         t1.PID, t1.TEXT
 FROM    Phrase t0, Phrase t1
 WHERE  t0.DOC = t1.DOC;
 => CorefTo(PID1 TEXT, PID2 TEXT)
```

```
Function func(p1, t1, p2, t2):
  if edit_dist(t1, t2) < 5:
    emit(p1, p2)
```

(c) Python Feature Extractor for DictMatch

Figure 3: An Example Feature Extractor

Statistical Learning and Inference. After the feature extraction phase, ELEMENTARY populates a set of relations, called *query relations*, which contain predictions, e.g., that two mentions are coreferent. This stage performs statistical inference and learning, but ELEMENTARY's goal is to hide that from developers. Here, ELEMENTARY does two tasks: (1) *weight learning*, in which ELEMENTARY's goal is to assign weights to the rules written in the previous step, e.g., $func$ in Figure 3, and (2) *inference*, in which we use the learned weights to infer the contents of the query relations.

To perform step (1), ELEMENTARY uses the evidence tuples provided by experts and crowd workers as labeled examples, e.g., `CorefTo`. The rule in Figure 3 is not a high-precision rule, and therefore, a smaller weight will be assigned by ELEMENTARY automatically. Then, ELEMENTARY performs step (2) and runs statistical inference to populate the query relation (here, `CorefTo`) using our group's recent techniques [3, 7].

3. DEMONSTRATION SETTING

The SIGMOD attendees will participate in a *live* demonstration of all three aspects of our system: (1) the end-to-end system, (2) our simple feature engineering system, and (3) the ability to explore different sources of feedback. Also, a SIGMOD attendee will discover new geological facts and write extractors to improve GEODEEPDIVE *on her own*.

3.1 Aspect 1: End-to-end System

The first aspect shows how GEODEEPDIVE delivers high-quality results and helps geoscientists discover interesting facts about our Earth. We also show the end-to-end process of how GEODEEPDIVE's back-end system runs to generate the front-end system.

Front-end System. Analyzing changes and singular points in geological measurements can provide evidence and implications about geoscience phenomena such as the Cambrian explosion [5]. One measurement that is interesting to our geoscience collaborators is Total Organic Carbon (TOC). Our collaborators try to understand how TOC changes across different epochs and basins. This is not an easy task for them, because there are more than 10K journal articles, 3K tables, and 800K web pages related to TOC. With GEODEEPDIVE, they are able to access knowledge in these resources without reading and aggregating extractions by themselves.

SIGMOD attendees will see how to use GEODEEPDIVE to discover these facts. The interaction will be based on real queries that SIGMOD attendees are interested in. But we have a set of prepared questions. We describe one example.

Suppose that a SIGMOD attendee wants to understand how TOC changes in Barnett Formation across different epochs. Without GeoDeepDive, she needs to go through papers related to Barnett Formation, manually record TOC reports in different epochs, and aggregate them together.

With GeoDeepDive, we can search "Barnett Formation." GeoDeepDive will report 13 TOC extractions for Barnett Formation in 36K geoscience journal articles. These extractions are grouped by different views, one of which is *time*. We then click the time view to see the results organized in different temporal intervals. For the Barnett Formation, GeoDeepDive has extractions for four temporal intervals: Pennsylvanian, Mississippian, Osagean, and Chesterian. These extractions span across Texas. From these extractions, we discover that the average TOC predicted by GeoDeepDive for the Barnett Formation is around 6%.

Back-end System. We demonstrate an end-to-end execution of the back-end system to process one document. SIGMOD attendees will see the input and output of (1) feature extraction, (2) statistical learning, and (3) statistical inference. The result will be loaded into the front-end system to create a version of GeoDeepDive.

3.2 Aspect 2: Simpler Feature Engineering

The second aspect shows how simple Python and SQL code can be used to enable statistical processing using our back-end system, described in Section 2.2. SIGMOD attendees will write new extractors by themselves on site. The interaction will be based on real GeoDeepDive errors that are discovered by SIGMOD attendees. For example:

1. We browse the system and find some errors made by GeoDeepDive. For example, in the location entity linking task, we may find that GeoDeepDive mistakenly links the word "Madison" to "Madison, Wisconsin" instead of "Madison, Alabama," although the whole sentence refers to "Alabama." These errors happen in the query relation `EntityLinking`(mid, eid), which contains, for each mention, the entity it refers to.
2. We come up with a heuristic to fix these errors. In the Madison example, one possible heuristic is *Detect State/County in the same sentence, and use them to guide location entity linking.*
3. We write feature extractors using Python to generate new feature relations. In the Madison example, we first write a Python function (<10 lines) to produce a relation `ContainedIn`$(eid1, eid2)$. A tuple $(eid1, eid2) \in$ `ContainedIn` if entity $eid1$ is contained in entity $eid2$ (e.g., Madison and Wisconsin). We will also reuse a feature relation called `ELC`$(mid, sent, eid)$, which contains, for each mention, its sentence ID and the candidate entity that it refers to.
4. We write an SQL query to integrate our heuristic.

```
SELECT elc1.mid, elc1.eid
FROM  ELC elc1, ELC elc2, ContainedIn c
WHERE elc1.eid=c.eid1 AND elc2.eid=c.eid2
      AND elc1.sent=elc2.sent;
```

Given this input, GeoDeepDive automatically runs statistical learning to learn the weight, and runs statistical inference to produce a new instance of GeoDeepDive. We will see this error fixed in our updated instance.

3.3 Aspect 3: Different Sources of Feedback

The third aspect demonstrates that taking different types of feedback leads to the high quality of GeoDeepDive. SIGMOD attendees will see how each type of feedback affects various quality metrics.

We provide a slider for each type of feedback, such as (1) number of human expert labels, (2) number of crowd source labels, (3) number of expert-provided rules, etc. By sliding these sliders, SIGMOD attendees will see different instances of GeoDeepDive, along with quality metrics like precision, recall, etc. By comparing different instances intuitively, and comparing quality numbers quantitatively, SIGMOD attendees will get a sense of the relevant impact of each type of feedback. For example, if we decrease the number of expert-provided rules, the precision of the top-200 extractions for temporal interval attachment decreases from 80% to 12%.

4. CONCLUSION

This demonstration describes GeoDeepDive, our preliminary effort to help geoscientists discover knowledge that is buried in geology journal articles. This problem requires us to solve the problems of data acquisition, extraction, and integration in a single framework. We show that it is possible to build such a system using familiar database processing languages. We demonstrate this in GeoDeepDive by showing that (1) an end-to-end system can be built with essentially just these familiar tools, and (2) a simple feature engineering system can be built using Python and SQL. We show that these aspects can be accomplished using our recent results to integrate statistical inference into traditional data processing systems. Attendees will interactively see the effect of different sources of feedback on quality.

5. ACKNOWLEDGEMENT

We gratefully acknowledge the support of the National Science Foundation under EAR-1242902. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of NSF, or the US government.

6. REFERENCES

[1] I. Halevy et al. Sulfate burial constraints on the Phanerozoic sulfur cycle. *Science*, 2012.

[2] J. Hellerstein et al. The MADlib analytics library or MAD skills, the SQL. In *PVLDB*, 2012.

[3] F. Niu et al. Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. *PVLDB*, 2011.

[4] F. Niu et al. DeepDive: Web-scale knowledge-base construction using statistical learning and inference. In *VLDS*, 2012.

[5] S. Peters et al. Formation of the 'Great Unconformity' as a trigger for the Cambrian explosion. *Nature*, 2012.

[6] C. Zhang et al. Big data versus the crowd: Looking for relationships in all the right places. In *ACL*, 2012.

[7] C. Zhang et al. Towards high-throughput Gibbs sampling at scale: A study across storage managers. *SIGMOD*, 2013.