We gratefully acknowledge the support from Toshiba, Google, DARPA DEFT and XDATA, ONR, and NSF. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of Toshiba, Google, DARPA, ONR, NSF, or the US government.

DeepDive: A DeepDive Application

Statistical Inference using Familiar Data-Processing Languages

Executive Summary

Takeaways

- DeepDive enables macroscopic science by building a "dark data" extraction system.
- Developers should think about features, not algorithms. It is possible to abstract probabilistic inference for use by domain scientists using standard SQL and Python.
- We can achieve comparable (or better) quality to human volunteers.

PaleoDeepDive

How does climate change impact biodiversity?

Extract biodiversity-related relations from journal articles.

PaleoDeepDive supports different high-level languages to specify a factor graph, e.g., Markov logic network, WinBUGS, etc. PaleoDeepDive is built with a combination of Python and SQL.

The DeepDive Approach

The central task of PaleoDeepDive

PaleoDeepDive achieves comparable (or better) quality with human volunteers, in a cheaper way.

The Architecture of DeepDive

DeepDive is the underlying framework

DeepDive supports different high-level languages to specify a factor graph, e.g., Markov logic network, WinBUGS, etc. PaleoDeepDive is built with a combination of Python and SQL.

Feature extraction with Python & SQL

DeepDive supports different high-level languages to specify a factor graph, e.g., Markov logic network, WinBUGS, etc. PaleoDeepDive is built with a combination of Python and SQL.

Input relations (e.g. Coref Task)

Example Extractor

Phrase

Phrase A is coreferent to phrase B if the edit distance between A and B is smaller than 5 and they appear in the same document.

Python

We write an SQL query to generate all phrase pairs that appear in the same document, and pair it with a python function.

```
SELECT t0.PID, t0.TEXT, t1.PID, t1.TEXT
FROM Phrase t0, Phrase t1
WHERE t0.DOC=t1.DOC
USEPYTHON pyfunc
```

We write a Python function to process all phrase pairs and make predictions.

```
def pyfunc(p1, t1, p2, t2):
    if edit_dist(t1, t2) < 5:
        emit("Coref", p1, p2)
DeepDive will learn the weight automatically
```

DeepDive has three features!

- **Rigorous Probabilistic Framework**
  - DeepDive uses a joint probability model that enables rigorous probabilistic interpretation
  - **Accuracy**
    - We expect that 8 of 10 with probability 0.8 will be correct
  - **Extractors**
    - # Excer$$ors$$
      - Candidates for improvement
      - Output to users
    - Goal
      - Output Probability

- **Simpler Feature Engineering**
  - DeepDive supports an "E3 loop" for feature engineering
  - **Application**
    - PaleoDeepDive
  - **Explore**
    - Error analysis
  - **Extract**
    - Extractor
    - Write/Improve extractors

- **Different Sources of Signals**
  - DeepDive is able to integrate a diverse set of signals & feedback
  - The more signals we use, the better quality we can expect!
  - **More-Supervised Signal**
    - Unstructured Data
    - Crowd
    - Structured Data
    - Training labels
    - Domain Knowledge
    - Heuristics & Rules
    - Hard Constraints

PaleoDeepDive: A DeepDive Application

PaleoDeepDive is able to extract relations from unstructured text automatically.

PaleoDeepDive achieves comparable (or better) quality with human volunteers, in a cheaper way.

PaleoDeepDive is built with a combination of Python and SQL.

Input relations (e.g. Coref Task)

Example Extractor

Phrase

Phrase A is coreferent to phrase B if the edit distance between A and B is smaller than 5 and they appear in the same document.

Python

We write an SQL query to generate all phrase pairs that appear in the same document, and pair it with a python function.

```
SELECT t0.PID, t0.TEXT, t1.PID, t1.TEXT
FROM Phrase t0, Phrase t1
WHERE t0.DOC=t1.DOC
USEPYTHON pyfunc
```

We write a Python function to process all phrase pairs and make predictions.

```
def pyfunc(p1, t1, p2, t2):
    if edit_dist(t1, t2) < 5:
        emit("Coref", p1, p2)
DeepDive will learn the weight automatically
```

DeepDive has three features!

- **Rigorous Probabilistic Framework**
  - DeepDive uses a joint probability model that enables rigorous probabilistic interpretation
  - **Accuracy**
    - We expect that 8 of 10 with probability 0.8 will be correct
  - **Extractors**
    - # Excer$$ors$$
      - Candidates for improvement
      - Output to users
    - Goal
      - Output Probability

- **Simpler Feature Engineering**
  - DeepDive supports an "E3 loop" for feature engineering
  - **Application**
    - PaleoDeepDive
  - **Explore**
    - Error analysis
  - **Extract**
    - Extractor
    - Write/Improve extractors

- **Different Sources of Signals**
  - DeepDive is able to integrate a diverse set of signals & feedback
  - The more signals we use, the better quality we can expect!
  - **More-Supervised Signal**
    - Unstructured Data
    - Crowd
    - Structured Data
    - Training labels
    - Domain Knowledge
    - Heuristics & Rules
    - Hard Constraints