On Model Discovery for Hosted Data Science Projects

HUI MIAO, ANG LI, LARRY S. DAVIS, AMOL DESHPANDE
UNIVERSITY OF MARYLAND, COLLEGE PARK
Use of the term “data science” is increasingly common, as is “big data.”

*From “Data Science and Prediction” CACM 2013:*

“Data science is different from statistics and other existing disciplines in several important ways:

- Data is increasingly *heterogeneous* and *unstructured*.
- The concern is finding *interesting and robust patterns* that satisfy the data.”
Background
All about building models

From complex models in big companies, now data science has become a profession in our society

- Numerus online/offline courses/degree programs,
- Panels on data science education in recent database conferences
Background
Landscape of current system research for data science

Models are typically derived iteratively in a lifecycle

A line of lifecycle management systems are proposed to address different aspects of it, in terms of predictive models

This talk focuses on: how to learn from others
new problems in identifying reference models from shared projects
Outline

Data Science Project Sharing in Practice & Research

- Rationale of project sharing
- Repository sharing in the real world
- System research facilitating sharing

Model Discovery, a missing piece in lifecycle management
Project Repository Sharing
Many reasons to share: practice & research

Reproducibility
  ◦ ReproZip, Codalab

Reusability
  ◦ Links to files in authors' pages
  ◦ Community wiki/repo

Collaboration
  ◦ Github, Notebooks on the web
  ◦ Enterprise systems, e.g. Labbook, GOODS

Enabled by New Systems
  ◦ CADS
  ◦ Datahub & Modelhub
Project Repository Sharing

Real world status: General purpose hosting services

GROWTH OF GITHUB REPOSITORIES USING PYTHON NOTEBOOKS

Data Source:
Github Weekly Dump @ Google Big Query
Check out the model zoo documentation for details.

To acquire a model:

1. download the model gist by `./scripts/download_model_from_gist.sh <gist_id> <dirname>` to load the model metadata, architecture, solver configuration, and so on. (`<dirname>` is optional and defaults to caffe/models).
2. download the model weights by `./scripts/download_model_binary.py <model_dir>` where `<model_dir>` is the gist directory from the first step.

or visit the model zoo documentation for complete instructions.

Table of Contents

- Berkeley-trained models
- Network in Network model
- Models from the BMVC-2014 paper "Return of the Devil in the Details: Delving Deep into Convolutional Nets"
- Models used by the VGG team in ILSVRC-2014
- Places-CNN model from MIT.
- GoogLeNet GPU implementation from Princeton.
- Fully Convolutional Networks for Semantic Segmentation (FCNs)
TensorFlow Models

This repository contains machine learning models implemented in TensorFlow. The models are maintained by their respective authors. To propose a model for inclusion, please submit a pull request.

Currently, the models are compatible with TensorFlow 1.0 or later. If you are running TensorFlow 0.12 or earlier, please upgrade your installation.

Models

- **adversarial_crypto**: protecting communications with adversarial neural cryptography.
- **adversarial_text**: semi-supervised sequence learning with adversarial training.
- **attention_ocr**: a model for real-world image text extraction.
- **autoencoder**: various autoencoders.
- **cognitive_mapping_and_planning**: implementation of a spatial memory based mapping and planning architecture for visual navigation.
- **compression**: compressing and decompressing images using a pre-trained Residual CNN network.
Project Repository Sharing
Ongoing research: Datahub

DataHub: Collaborative data science platform

- Joint work among database groups in MIT, UIUC and UMD [CIDR’15]
  - A dataset management system, import/search/query/analyze a large number of public datasets
- Techniques for managing a large number of datasets, their versions over time, and derived data products [VLDB’15, VLDB’16]
  - Exploit overlap to reduce storage while keeping retrieval costs low
- Query language & execution engine for supporting higher-level introspection of datasets, provenance, and workflows [TaPP’15, SIGMOD’17, HILDA’17]
- Lifecycle management for ML models such as deep learning models [ICDE’17]
# Project Repository Sharing

Repository in research: modeling lifecycle management

## Main Components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLV</td>
<td>Domain-specific version control system</td>
</tr>
<tr>
<td>DQL</td>
<td>Declarative language to interact and enumerate managed models</td>
</tr>
<tr>
<td>PAS</td>
<td>Versioned weight parameter archival store</td>
</tr>
<tr>
<td>Hosting Service</td>
<td>Online service for learning from others</td>
</tr>
</tbody>
</table>

---

**ModelHub System Architecture**

[ICDE’17, ICDE’17 demo, HILDA’17]
Project Repository Sharing
Repository in research: modeling lifecycle management

Domain-specific VCS

Store modeling artifacts:
- network architecture
- parameters
- hyperparameters
- data references

Rich query facilities:
- saved model exploration
- new model enumeration
- repository sharing

Decorative Network Creation:
- select, slice, construct, evaluate
Project Repository Sharing
Repository in research: modeling lifecycle management
Project Repository Sharing
Repository in research: modeling lifecycle management

DQL Query

EVALUATE m FROM (SELECT m1 where m1.name like "model-0")
WITH config="model-0/solver.prototxt"
VARY config.base_lr IN [0.1, 0.01, 0.001]
KEEP TOP(1, m["loss"], 200)

Model Candidates 3
Temporary Directory dql_ab5c36f/model1, dql_ab5c36f/model2, dql_ab5c36f/model3

Status Running

Progress

Result

<table>
<thead>
<tr>
<th>model</th>
<th>iter</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>model3</td>
<td>0</td>
<td>2.30259</td>
</tr>
<tr>
<td>model3</td>
<td>10</td>
<td>2.30235</td>
</tr>
<tr>
<td>model3</td>
<td>20</td>
<td>2.30211</td>
</tr>
<tr>
<td>model3</td>
<td>30</td>
<td>2.30123</td>
</tr>
<tr>
<td>model3</td>
<td>40</td>
<td>2.29978</td>
</tr>
<tr>
<td>model3</td>
<td>50</td>
<td>2.29869</td>
</tr>
<tr>
<td>model3</td>
<td>60</td>
<td>2.29835</td>
</tr>
</tbody>
</table>
Project Repository Sharing
Metadata/Provenance enrichment of modeling artifacts

In the lifecycle, metadata capturing, provenance management is an important topic in the lifecycle.

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical file</td>
<td>Data file properties, schema, size, author, content; script AST, runtime profiling, lib dependencies, etc.</td>
</tr>
<tr>
<td>Logical artifact</td>
<td>Model type, metric value, hyperparameters of a model, user annotations on a model, etc.</td>
</tr>
<tr>
<td>Steps in a physical file</td>
<td>System IO calls, library methods (e.g. logistic regression, pandas dataframe), DNN network architecture, and their dependencies, etc.</td>
</tr>
<tr>
<td>Steps in an logical artifact</td>
<td>Analysis step type (data loading, cleaning, feature engineering), step input, output, transitions, etc.</td>
</tr>
<tr>
<td>Relations among physical files</td>
<td>File versions, file dependencies in the execution history, system library dependency, etc.</td>
</tr>
<tr>
<td>Relations among logical artifacts</td>
<td>Logical artifact versions, e.g. earlier version for a model, lineages from modeling branches, etc.</td>
</tr>
<tr>
<td>Metadata/Provenance type of interests in lifecycle management research</td>
<td>Users' contributions, communications, notes, comments, cognitive annotations, etc.</td>
</tr>
</tbody>
</table>

Data model for a project repository

Next step: Search hosted repositories and find related projects.
Outline

Data Science Project Sharing in Practice & Research

Model Discovery, a missing piece in lifecycle management
  ◦ Query type of interest
  ◦ ModelHub Discovery, an IR approach
  ◦ Compare & rank matched projects
  ◦ Advanced queries for matched projects
Discover Related Projects

Query Type of Interest

Given a set of hosted projects, for data science practitioners, we wish to find related projects according to discovery queries:

<table>
<thead>
<tr>
<th>Search Criteria</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>I have a dataset, e.g. U.S. census</td>
</tr>
<tr>
<td>Project Goal</td>
<td>I want to predict the mortgage default rate</td>
</tr>
<tr>
<td>Modeling Method</td>
<td>Show me results using general linear models, or deep learning</td>
</tr>
<tr>
<td>Programming Language</td>
<td>Written in Python, or Jupyter notebooks</td>
</tr>
<tr>
<td>Rel. with other artifacts</td>
<td>Better with plots</td>
</tr>
</tbody>
</table>
Discover Related Projects
Find related projects is difficult in current systems

e.g. Predicting mortgage default rate using linear regression in python

General web search treats each URL separately & does not index repo as a whole

For code/commit/issue/wiki, it only looks at individual files too

Opportunities for System Research
ModelHub Discovery
An ongoing system research: Architecture & Design Decisions

**Query Interface:**
natural language + metadata conditions & datasets

**Match:**
Information Retrieval approach on keywords from the whole repo e.g. doc, src, dataset, metadata catalog

---

**Compare and Rank:**
Project similarities Deduplications

---

**Composite & Constrain:**
Select returned models Model ensembles Quality constraints
Currently exploring traditional keyword based IR approach, and use metadata/provenance as matching filters because:

- Most of the public repo currently include many keyword tokens
  - Documents (readme, notes)
  - iPython/Jupyter/R notebooks
  - Source code (APIs methods, imported packages, comments)
- Structured metadata often does not have consistent meaning across heterogeneous projects even if they use the same modeling method
  - Validation accuracy, testing loss, training time

Collected Github repos with ipynbs and used Lucene as a first step

Future problems

- Using datasets to explore projects
  - Query: explore projects which used a dataset like this ..
  - Challenges: dataset transformations (discuss next)
Compare & Rank Projects

Potential Problem: Matched repositories may be similar

Popular hosted projects are often forked many times. Returning similar models hurts the discovery experience.

How to determine two repositories are similar?
Compare & Rank Projects

Potential Problem: Similar artifacts may be transformed

Data may be changed due to

- feature engineering,
- validation set splits
- data wrangling
  - Add / Drop / Split / Merge columns
  - Transform columns by applying rules (i.e. normalizations), fixing discrepancies

Models may be updated as well.

- e.g. Fine tuning is a common practice in deep learning

1. Change prediction labels
2. Fix weight for some layers
   2.1. update hyperparameters
   2.2. retrain the model
3. Goto 2

Figures from Pandas data wrangling document
Compare & Rank Projects
Proposed approach: Align multidimensional sequences

Notice that various artifacts in a repository can be viewed as tabular data, or more generally, multi-dimensional arrays (may be categorical)
  ◦ Input data
  ◦ Learned model parameters
  ◦ Result table (predicting results)

Get the following high level idea:

Given a pair of artifacts in two projects, which may be transformed, we wish to revert the transformations as much as possible, and use a proper distance on the aligned pair as a similarity measure.
Compare & Rank Projects

Proposed approach: Align multidimensional sequences

Artifacts pair \((A, B)\): Parameters (vectors, matrices, tensors ..)
Transformation \(T\) & Revert \(R\): Feature engineering, finetuning, etc.
Concrete Problem:

\[
\text{argmin}_X \ D(A - R(B, X))
\]

Best vector alignment is easy (bipartite matching), but when number of dimensions increases, they are surprisingly difficult problems

\[
\begin{bmatrix}
8 & 2 & 5 & 3 \\
4 & 1 & 2 & 2
\end{bmatrix} - \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0
\end{bmatrix} = 13
\]

\[\mathcal{D}(\mathcal{B}, X) := P_1 B P_2, \quad X := (P_1, P_2)\]

Solution involves two coupled permutations. NP-hard problem.
(reduction from graph edit distance problem)
Compare & Rank Projects

Preliminary Evaluation (nonconvex models, finetuned VGG):
Correlation with cosine distance on actual prediction results

**Dataset**
finetuned VGG models, last layer to 1000 labels stopped at 10k iterations

**Prediction result table (Y)**
Last layer model outputs

Both axes are *normalized*

**corr = 0.8224**
Compare & Rank Projects

Results & Advantages of the approach & Open problems

If distance $D$ is a metric, the alignment distance is also a metric.

Possible to embed models into low dimensional vector space w.r.t. best alignment distance among all model pairs, which enables

- learning better similarity functions
- learning to rank given user clicks

Lead to interesting concrete problems when considering other artifacts

- e.g. categorical value (string distance), tabular data, ...

Future problems:

- Evaluate the proposed method extensively
- Revisit data science project similarity problem by looking at related domains (schema alignment, data cleaning, etc.)
- Consider the workflow and other artifacts (e.g. scripts, notebooks)
When returned models are solving a similar task (e.g., competitions, classic problems using standard datasets, many models tried in an enterprise application for the same task)

- Matched projects can be compositied, a.k.a. *Model Ensembles*

A open secret to win competitions, e.g. Netflix, Kaggle, etc.

- Methods exist with/without retraining the models and the ensembles
- Apart from method, people started to look at systems for model ensembles from a library of models decade ago [ICML’04]

**Popular practice of model ensembles**

- Voting, Bagging, Trees, AdaBoost, GBRTBoost, ...

**Systems tradeoffs**

- Quality: Order of the models and selection of ensemble method
- Efficiency: Amount of computation / query time
Process & Ensemble Projects

Advanced query: composite & constraint returned models

When returned models are solving a similar task (e.g., competitions, classic problems using standard datasets, many models tried in an enterprise application for the same task)

- Matched projects can be compositied, a.k.a. Model Ensembles
- User may have different Constraints for Models

Interesting constraints seen in research in recent years

- User gives a precision-recall curve for the compositied models [NIPS’12]
- Due to the disparate datasets, fairness constraints for the classifiers [KDD’15]
- Limited resources, e.g., deliver a model to mobile environment [ICLR’16]

Systems issues

- Unified query processing formalism for processing such constraints
- Efficiency issues when retraining multiple alternatives
Conclusion

We study the model discovery step in the data science lifecycle

After examining the status of current system offering, we argue the space is a rich field for future system research

Introduce Modelhub Discovery, highlights:

- Use an IR approach and treat the projects as first class citizens
- Study the project similarity problem, which is a key issue in the IR pipeline
- Propose a general similarity framework for project artifacts, and study a concrete case
- From user practice, we look at the model ensemble, w.o./w interesting constraints, as a post processing step
Thank you!