Towards a Flexible System Architecture for Automated Knowledge Base Construction Frameworks

Osman Din
MIT

Knowledge Bases are ubiquitous

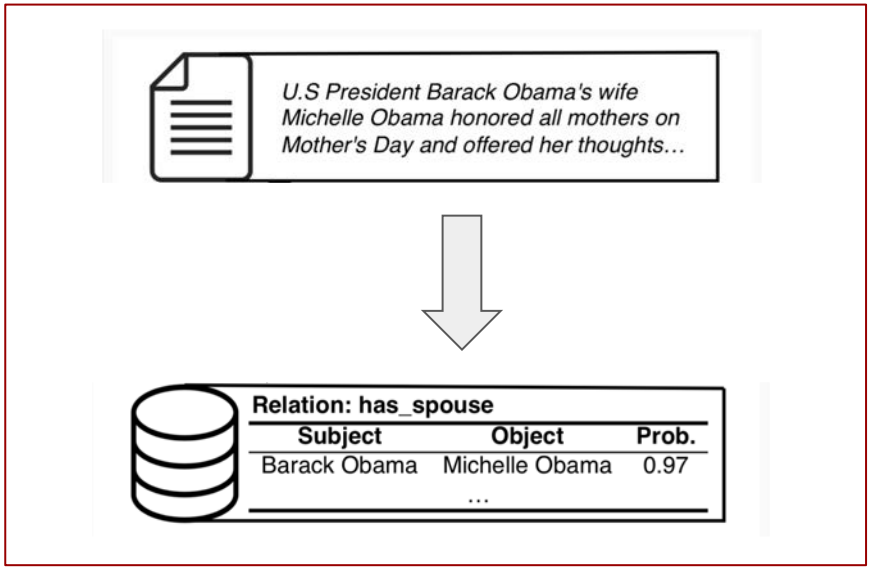
- Academic and commercial knowledge bases in the wild:
  - Freebase, Google's Knowledge Graph, YAGO2, IBM Watson, and specialized domain knowledge bases such as PaleoDB, etc.
- KBs used for many tasks:
  - Entity enrichment, semantic annotation, etc.
  - Information extraction from HTML tables in vertical search engines (ebay or alibaba)
  - Drug discovery and precision medicine
- Used in many specialized domains including LAM:
  - Projects by U.S. National Library of Medicine, UK National Archives, National Archives of France, etc.

Problem: Building specialized knowledge bases by hand is resource intensive

- PaleoDB (complex KB containing information about fossil records) and GWAS Catalog (data of genome wide association studies) has taken 10+ years and hundreds of curators.
- These knowledge bases are (still) domain specific.

Solution: Automate knowledge base construction

Knowledge base construction is the process of taking unstructured information as input and converting it into a structured knowledge base as output.



Source: DeepDive: A Data Management System for Automatic Knowledge Base Construction.

Automated frameworks applications

- Reasonable success in meeting a number of technical challenges
  - Accuracy
  - Completeness
  - Performance and flexibility:
    - Deep learning is resource intensive
    - Creating features by hand for training purposes is expensive.
- Have been successful applied to PaleoDB and GWAS Catalog, with competitive accuracy and coverage.
- Evolving, active area of research

Questions faced by implementers with use cases for a knowledge base

- What does an ideal automated KB framework look like?
- What are the some of the features to consider?
- What are some of the limitations in current solutions?

Adopted research approach

- Gather core requirements
- Evaluate automated knowledge base frameworks against these constraints (and against the stated goals of these frameworks)
- Analysis of gaps and limitations
- Contributions – research and technical

Core requirements

Functional

- Handling multiple varieties of data
- Support for storage and search
- Flexible feature extraction
- Domain-specific features
- Support for human feedback for error analysis

Non-functional

- Performance
- Scaling
- Usability
- Transparency and fairness

Evaluation

Selected KB machine learning based systems (as opposed to rule based systems)

Evaluated Alexandria, Founduer, and DeepDive (relatively mature frameworks)

Inadequate support for:

- Flexibility in modifying the processing pipelines
- Scalability - vertical scaling only
- Capability to add domain features
- Usability, often requiring scripting knowledge or knowledge of the internals

Contributions

- System architecture for a knowledge base construction framework
- Extensions to a specific knowledge base framework to validate the architecture

Design guidelines for the architecture

- Underlying functionality should be exposed through APIs for extensibility and scalability.
- Functionality should be composed into components or services. Middleware should be used to connect components instead of using point-to-point connections.
- Transparency and fairness aspects should be considered at the design stage.
- Open source frameworks (as opposed to vendor specific platforms) should be preferred.

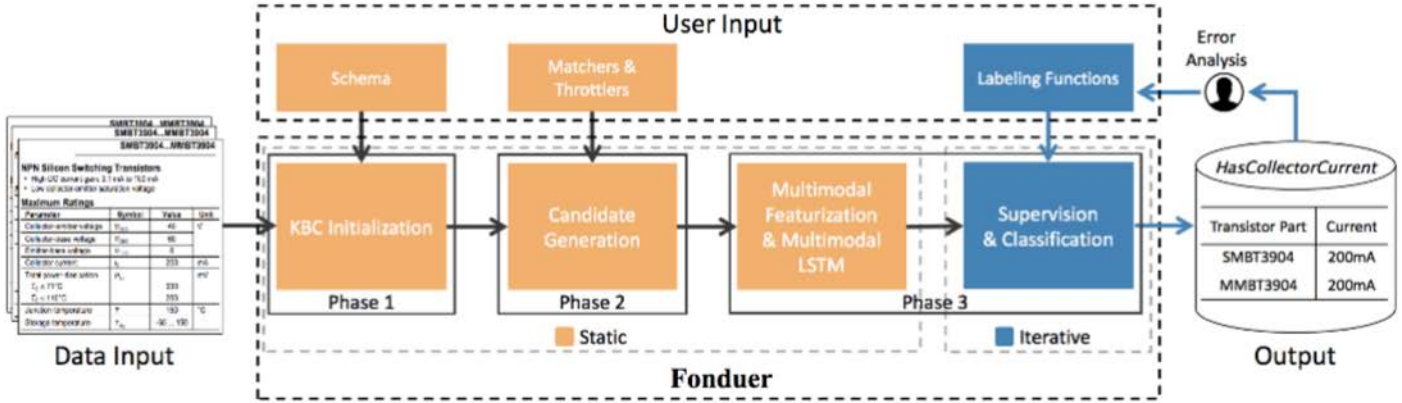
System Architecture

- Knowledge base engine (Fonduer)
- ML distributed middleware for training and candidate feature generation: Snorkel and TensorFlow
- Persistence (relational database and tripestore)
- Graphical user interface
- External workflow Engine to automate and externalize ingest processes (Optional)

Fonduer Knowledge Base Framework

- Relies on multiple clues (textual, structural, visual, and tabular) in addition to textual signals. Better match for the archives domain due to occurrence of richly formatted data (present in tables, figures, etc., in scholarly literature).
- Uses weak supervision to generate training data using Snorkel (data denoising framework), eliminating the need to create large sets of data for by hand for training models.
- Uses deep learning (LSTM), with augmented features from multiple signals in a document as input.

Fonduer Internal Pipeline

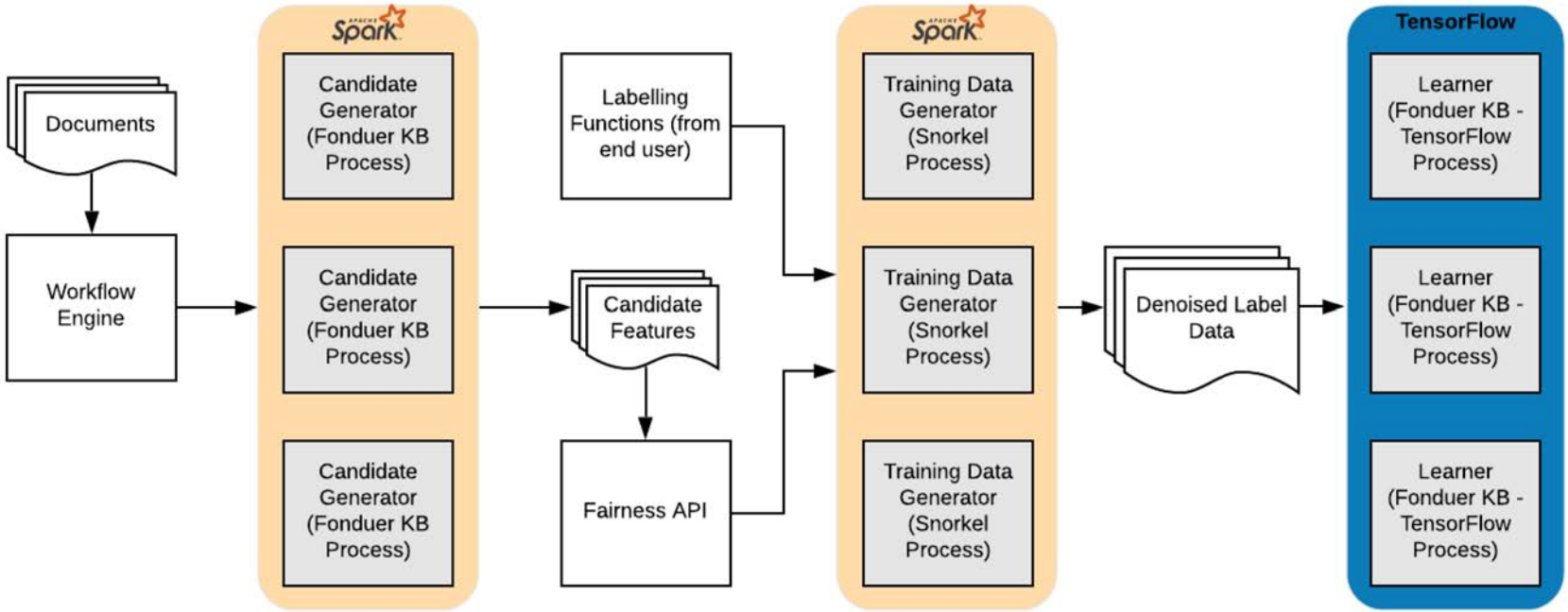


Credits:
Fonduer: Knowledge Base Construction from Richly Formatted Data (SIGMOD '18)

Extensions to Fonduer

- Addition of APIs as building blocks
- Distributed deep learning through TensorFlow
- Distributed weak supervision through Snorkel and Spark
- Integration with an experimental API that flags potentially biased features

Distributed Architecture Overview



System Pipeline Overview

Current Limitations

- Work in progress – validation is needed
- Need to discover domain features particular to material or target workload relevant to archives (to increase the accuracy and coverage of the knowledge base)
- Providing a user interface, instead of making users enter rules programmatically. (Also provide debugging support through a GUI.)
- Extracting data using multiple processing engines using the pipeline, thereby extracting data from scholarly artifacts such as plots or figures

Conclusions

- Identifying a set of desired properties of an AKBC framework and some of the design constraints
- Architecture guidelines for automated knowledge base frameworks that could be useful to framework designers in meeting the end goal
- Awareness of the potential (and challenges) of AKBC frameworks for computational archives use cases
- Plan to share experience report with CAS community and also share experiences with AKBC researchers, and ideally form a symbiotic relationship

Thanks!

“Towards a Flexible System Architecture for Automated Knowledge Base Construction Frameworks”

https://github.com/sagekb/sagekb (available soon)

Osman Din

osmandin@mit.edu