Data Annotation for Contemporary ML Challenges

Innovation
1. Data is the backbone of machine learning models, yet real-world data is messy: data annotations format this data in a way that a model can learn from.
2. In the past, we had used and attempted to extend on open-source annotation tool: cdQA-annotator; however, there were a number of pitfalls:
   a. No centralised storage: annotations had to be manually downloaded after each annotation session and uploaded to Sharepoint.
   b. Difficult to implement new features: the tool was no longer maintained, and the code base’s infrastructure did not support pluggable features.
   c. Lack of a project structure and workflow.
3. As such, our project’s specifications focused around these three missing features.

Why Another Annotation Tool?
1. Distinct from many open-source annotators, a priority was making our annotator pluggable for future tools and software contributors.
2. Similarly, despite a number of paid annotation tool services that offer the features we were looking for, we wanted to create a tool that could annotate data for the lowest $0.24 challenge.
   a. One such example is NL2SQL: no annotation service offers a tool for natural language to SQL annotations (depicted in the diagram below).

Annotating Workflow

Project Management
1. Organization of data annotations becomes increasingly important as annotation projects get larger, this call for:
   a. Store and retrieve functionality for data and annotations.
   b. Project and task structure to distribute annotation workload.
   c. Workflows to push annotations to other storages automatically.

Question-Answer Systems, Classification, and SQL Annotators

Generic Annotation Workflow
1. The end-user uploads a supported file type (e.g., .csv, .txt, .md) directly to the annotator or pulls a previously uploaded file from a project.
2. The user creates an annotation, specific to the annotator being used:
   a. For QAS, the user writes a question.
   b. For SQL-Annotators, the user writes a SQL question and a corresponding SQL query.
   c. For Classification, the user drag and drop labels onto table rows, classifying them into categories.
3. The annotations are sent to the database for additional validation and bleaching.
4. The annotations are used for training and project statistics are updated.

Future Technical Goals:
- Create better collaboration between the frontend design and the backend logic by using a REST in a GraphQL framework.
- Refine the frontend design development by using React or another JavaScript framework instead of Vanilla JavaScript.
- Add more annotation tools to the MAT suite.
- Multiple choice image labeling.
- Audio transcriptions.
- Named entity labeling.
- Implement integrations with other data services, such as Pure Storage.

References & Acknowledgments

Conclusions & Future Goals

Conclusions:
- The learning curve for a full stack of technologies was larger than initially anticipated.
- MAT’s Postgres cluster has distinct 8 microservices in its stack.
- Deployment time for seemingly small features increases drastically as they can be integrated with existing features.
- Necessity of a separation between frontend and backend.

Acknowledgments:
- We would like to thank our:
  - Corporate Partner Mentors: Sarah Redenek and Justin Gould
  - Student Mentor: Robin Kapil
  - And the entire Ford team for giving us this opportunity, letting us access their internal data and systems, and guiding us as we created and refined this project.
- We would also like to thank: Dr. Mark Daniel Ward, Ellen Gendebich, and Maggie Ann Betz for supporting us during the Academic Year and providing us with the resources for the successful completion of our project.

References:
- SQL Query

Acknowledgments:
- We would like to thank our:
  - Corporate Partner Mentors: Sarah Redenek and Justin Gould
  - Student Mentor: Robin Kapil
  - And the entire Ford team for giving us this opportunity, letting us access their internal data and systems, and guiding us as we created and refined this project.
- We would also like to thank: Dr. Mark Daniel Ward, Ellen Gendebich, and Maggie Ann Betz for supporting us during the Academic Year and providing us with the resources for the successful completion of our project.

Project and task structure to distribute annotation workload.

Workflows to push annotations to other storages automatically.

Store and retrieve functionality for data and annotations.

Caching frequently visited files, databases, and project statistics are updated.

Annotating Workflow

1. A user creates a project, inviting other members if necessary.
2. Any team member uploads data following the supported formats.
3. As annotations are created, project statistics are updated and at any point all annotations for an project can be downloaded.
4. Optionally, annotations can be periodically sent to an AWS or GCP storage.

Database & Cache:
- PostgreSQL stores annotations.
- Redis caches frequently visited files, databases, and project statistics are updated.

A user creates a project, inviting other members if necessary.

As annotations are created, project statistics are updated and at any point all annotations for a project can be downloaded.

Project and task structure to distribute annotation workload.

Workflows to push annotations to other storages automatically.

Store and retrieve functionality for data and annotations.

Conclusions & Future Goals

Conclusions:
- The learning curve for a full stack of technologies was larger than initially anticipated.
- MAT’s Postgres cluster has distinct 8 microservices in its stack.
- Deployment time for seemingly small features increases drastically as they can be integrated with existing features.
- Necessity of a separation between frontend and backend.

Future Technical Goals:
- Create better collaboration between the frontend design and the backend logic by using a REST in a GraphQL framework.
- Refine the frontend design development by using React or another JavaScript framework instead of Vanilla JavaScript.
- Add more annotation tools to the MAT suite.
- Multiple choice image labeling.
- Audio transcriptions.
- Named entity labeling.
- Implement integrations with other data services, such as Pure Storage.