Name Entity Recognition Annotation Tool (NERA)

Background

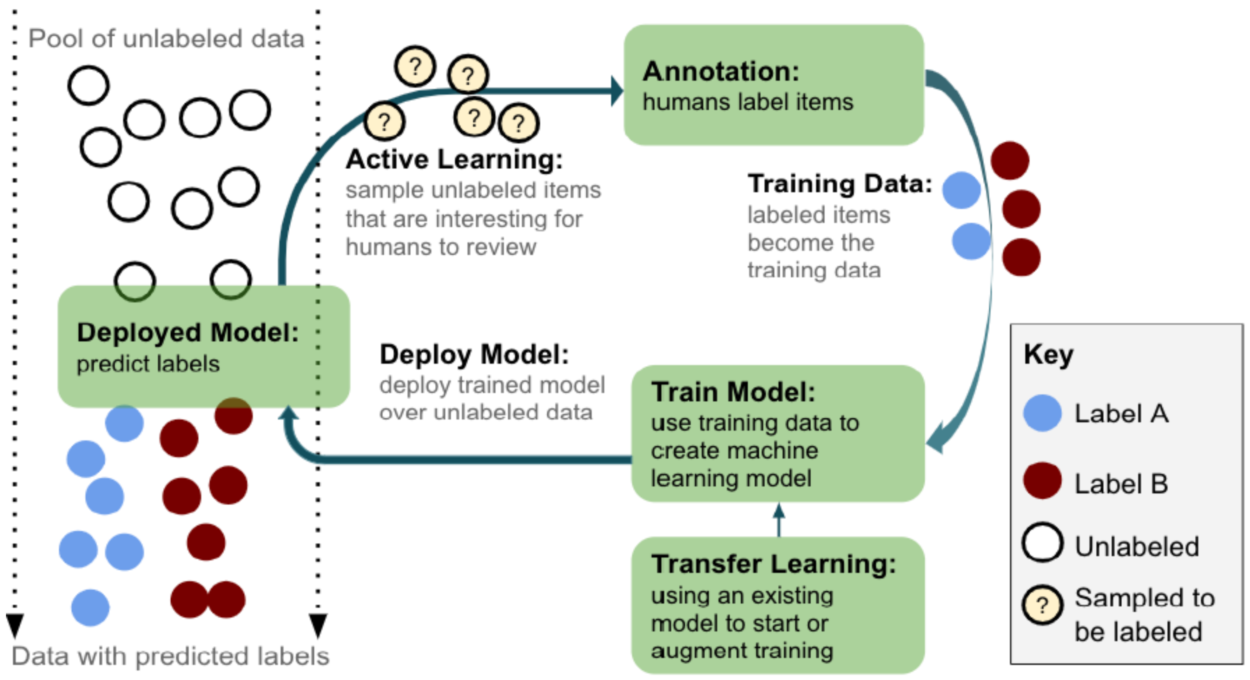
Name Entity Recognition Annotation Tool (NERA) is a web-based application to showcase the power of active learning and transfer learning in a real-world machine learning (ML) application.

On the one hand, it is motivated by the external trends that 1) Large pretrained models, like BERT, GPT-3 have achieved the state-of-the-art performance across different NLP tasks (So, transfer learning can help the real-world application) 2) Annotating a high quality dataset is costly in real world ML applications (Thus, active learning can be a rescue). On the other hand, we propose the NERA to satisfy the internal requirement from the Vector engineering team who wants to build a NLP service within Vector Institute and explore the way how active learning can be applied to real-world ML applications.

Real-world ML applications typically involve an iterative process where people first collect and annotate a dataset, then train and evaluate a model. Active learning targets at the first process. It aims to identify the most informative data points for the current model to label. In contrast, transfer learning focuses on the model training phase to reuse the knowledge from the pretrained models. Moreover, transfer learning can also benefit the data labeling process as it can provide more meaningful auto suggestions.

To sum up, NERA is a demo to showcase how people in the Vector community can use Vector resources and services to build customized NLP services for their own applications. In this project, NERA only focuses on the name entity recognition for its clear task definition, but the tools built for it will be easily extended to different tasks and domains. Broadly speaking, NERA aims to find the best practice for the machine-aided data annotation process.

Vector service: a set of APIs that enable people to use (train, finetune, and evaluate) large pretrained models for downstream tasks. (e.g. Name entity recognition, Document Summarization, Chatbot)


Pool of unlabeled data
Active Learning: sample unlabeled items that are interesting for humans to review
Annotation: humans label items
Training Data: labeled items become the training data
Deployed Model: predict labels
Deploy Model: deploy trained model over unlabeled data
Train Model: use training data to create machine learning model
Transfer Learning: using an existing model to start or augment training
Data with predicted labels
Key
Label A
Label B
Unlabeled
Sampled to be labeled

Goals

NERA aims to achieve the following features. They are divided into the following four categories based on the life cycle of a ML application.

G1. During labeling: enable annotation from multiple sources
  a. Machine auto suggestion (M1)
  b. Human provided annotation (M1)
  c. Active learning (M3)

G2. After labeling: data quality control process
  a. Can compare the annotation results from different annotators (M2)
  b. Automatically output examples with disagreement (M2)

G3. Deployment
  a. Human-corrected examples add to the training dataset while looking at the model prediction (M2)
  b. Can serve the trained model as a web service (M1)

Non-Goals

Below are some desired features but are not necessary for this project.

- Model training workflow and logs (Rely on other tools, eg. spacy, hugging face, wandb)
- Large pretrained model support (GPT-NEO)
- Different task support (Summarization, Chatbot)

Milestones

There are four main milestones in this project. After each milestone, a demo/presentation will be prepared to show the key features.

Milestone #1

- Goal: To show the basic workflow with the basic features.
- Objective:
  - Be able to annotate an unlabeled dataset (Web App)
  - Train a model based on the annotated dataset. (CLI)
  - Deploy the model as a service. (Web App)
- Dataset: A standard NER dataset and treat it as an unlabeled dataset.
- Model: Use Spacy training pipeline
- Annotation: Enable auto suggestion but no active learning

Milestone #2:

- Goal: Incorporate more complex workflow, Dataset versioning with wandb?
- Objective:
  - ~~Add dataset summary page (Not key function of annotation tool, alternative: wandb)~~
    - ~~Raw example~~
    - ~~Data summary: Label statistics, feature statistics~~
  - Add model evaluation process
    - Visualize the model prediction and compare with the ground truth. (Useful for model debugging)
    - Note: Wandb has the functioning but not tailor towards different NLP tasks
  - Add model selection support
    - Goal: select a better model
    - Compare different model’s prediction using (A/B testing)
    - Need randomized
  - Add data labeling quality control process
    - Goal: create a gold standard dataset
    - Compare different annotations and check for disagreement
    - Several different annotations

Milestone #3: More complex labeling process

- Goal: Incorporate more powerful labeling process
- Objective:
  - Enable active learning
  - ~~Enable annotation from multiple sources (Programmatic labeling, Snorkel Flow’s features)~~
    - ~~Regular expression, Dictionary, Knowledge Graph~~
  - Benchmark the performance of different active learning algorithms

After M3, we have completed all necessary components of the machine learning life cycle for a customized NER task.

Milestone #4: Customized NER service

- Goal: Build a customized NER service using the NERA
- Objective:
    - Build a NER service using standard NER dataset (e.g. CoNLL2003)
    - Showcase the power of “transfer learning + active learning”

If time permits, we can extend the system to support different tasks and domains and support more advanced pretrained models (achieve some non-goals).

Timelines

Milestone 1: Week of June 21
Milestone 2: Week of July 5
Milestone 3: Week of July 26
Milestone 4: Week of Aug 9

Design Details

Framework

- Frontend: Vue.js (https://v3.vuejs.org/) + Quasar (https://quasar.dev/)
    - Why Vue: most open source annotation tools use Vue, and it is easy to work with
    - Why Quasar: it provides many powerful building blocks and supports multi platform (Web App, Desktop App, Mobile App)
- Backend: Flask (https://flask.palletsprojects.com/en/2.0.x/)
    - Why Flask: lightweight
- Database: ~~sqlite~~, mongodb
    - Reason for changing
        - The database does not involve complex search logic
        - Implementing sqlite is more complicated than mongodb.
            - Extra scripts to handle the adding, removing, updating, indexing examples because a single example is a three-level nested dictionary. (Raw Properties -> Annotations/Tokens -> Span). It is better just to use json-like database (mongodb)
            - More difficult to handle other data formats (Image, Video, Audio) for future use cases. (Frequent Schema change)
            - Need to change schema for different active learning algorithms, the “score” may be different for different active learning algorithms
    - Implementing using MongoEngine (ORM)
    - Mac: brew tap mongodb/brew, brew install mongodb-community, brew services start mongodb/brew/mongodb-community

- Machine Learning: Pytorch + Spacy + Hugging Face Transformer
    - Spacy is an open source library, and has an easy to use NLP pipeline. We do not want to spend too much time on the modeling side.
    - Change to transformer because it is more flexible and have better community support
    - Challenge: inconsistency between spacy and hugging face tokenizer. Hugging face models fail when the sentence contains emoji.

Frontend

The frontend design has the following two components. But note that different milestones may have different frontend outlook to emphasize different features.

- NER Inference service: (In the Demo)
    - A service that receives raw text input and analyzes the entity.
- NER Annotation Service: (In the Demo)
    - A service that helps with the data annotation process.
- Data Toolbox: Dataset Card -> Dataset Detail page. (Not in the Demo, comment out in the code)
    - Data Annotation: collaborative annotation and quality control with labeling history and dataset versioning.
    - Dataset Summary: Dataset statistics regarding both feature and label.
    - Data Search: Search specific examples by key words or semantics.
- Model Toolbox: Model Card -> Model Detail page. (Not in the Demo, comment out in the code)
    - Model Service: Deploy the trained model as service. Given the unknown input, visualize the model output. (Can be seen as client interface)
    - Model Evaluation: Evaluate the trained model on validation or test dataset. Compare with the ground truth model to see the weakness and strength of the model. Show the confidence of the model prediction. (Can be seen as developer interface)

Backend

For the backend, there are three main components.

- Database
    - Dataset: Raw data, label from different users, other meta information..
    - Model meta information: Model description and location...
    - User Information: Authentication, Labeling history…
    - Note: Not implemented in the demo.
- AL Engine
    - Active learning algorithm and auto suggestion
    - Model Training: Using pytorch to perform model training and finetuning

Workflow

- Data Collection and Cleaning
    - Upload an unlabeled dataset

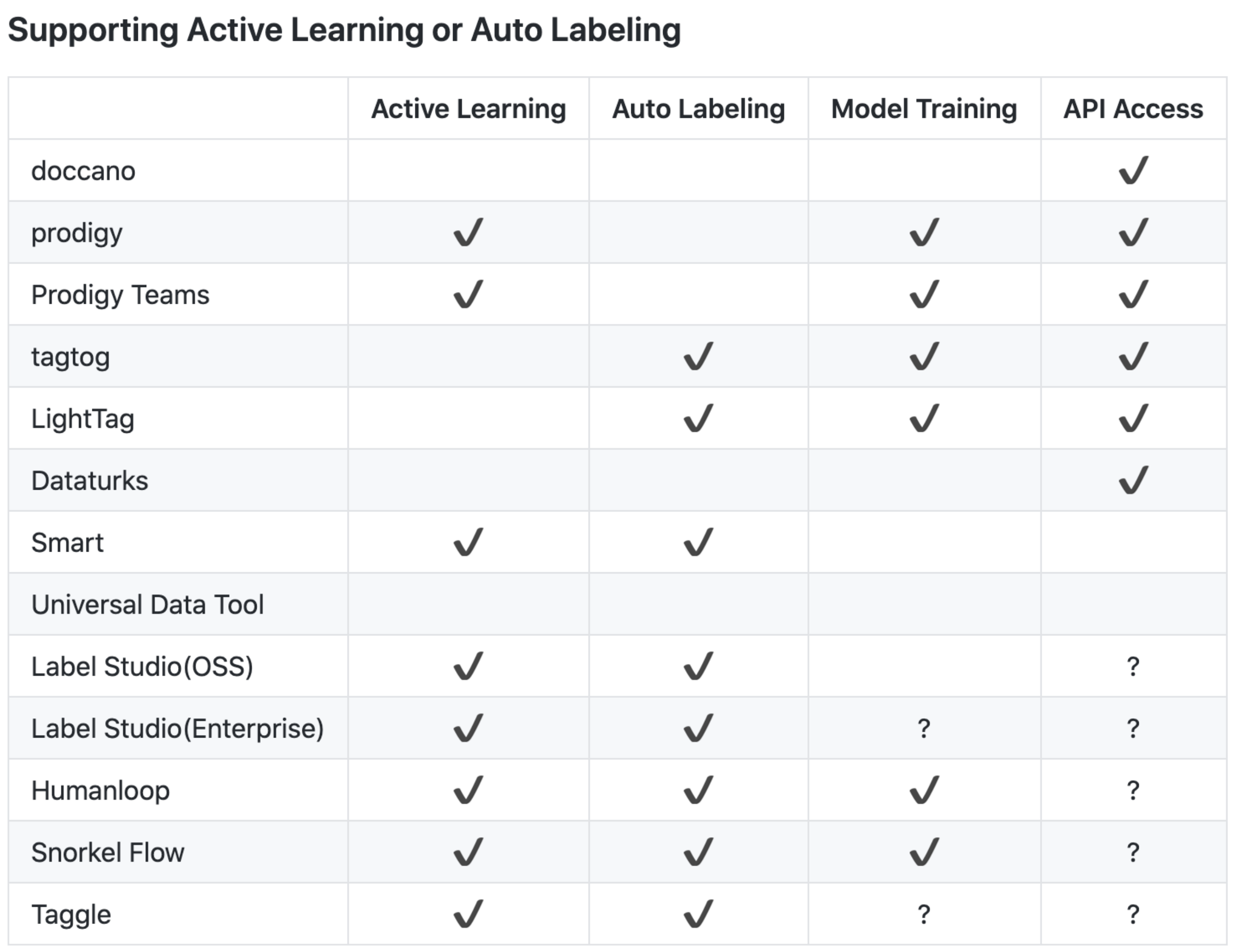
        - .txt file without annotations. Each example is split by “\n”
        - .json file with default annotations.
        - Input from web interface. Plain text
    - Process the raw dataset to JSON Line format
        - Each line is a single example
        - {“text”: , “tokens”, “annotations}
- Data Annotation
    - If there is no labeled datasets, then use spacy for auto suggestion
    - Labeling a single example
    - Quality Control
    - Export Dataset as a hugging face dataset
- Offline Model Training
    - Hugging Face data loader
    - Pytorch + Hugging Face
    - Upload to Hugging Face
- Model Deployment
    - Download model from Hug and served the model
    - If no hugging face model, then use the spacy’s default model

Note: The design details are not complete. More will be added as the project progresses.

Reference Design

There are some Awesome Commercial Products:

- Scale: https://scale.com/
- Snorkel Flow: https://snorkel.ai/
- Prodigy: https://prodi.gy/
- Human loop: https://humanloop.com/



Supporting Active Learning or Auto Labeling

	Active Learning	Auto Labeling	Model Training	API Access
doccano				✓
prodigy	✓		✓	✓
Prodigy Teams	✓		✓	✓
tagtog		✓	✓	✓
LightTag		✓	✓	✓
Dataturks				✓
Smart	✓	✓		
Universal Data Tool				
Label Studio(OSS)	✓	✓		?
Label Studio(Enterprise)	✓	✓	?	?
Humanloop	✓	✓	✓	?
Snorkel Flow	✓	✓	✓	?
Taggle	✓	✓	?	?

Useful Links

- Survey of data annotation tool: https://github.com/doccano/awesome-annotation-tools
- Doccano: https://github.com/doccano/doccano

Selling Points

There are already many mature commercial products. How shall we differentiate our product? Why do others want to use the tool provided by us?

- Open source
  - Open source may be a selling point, but we need to be better than other ones. The most famous one is doccano (https://github.com/doccano/doccano). If we have active learning, auto labeling, and model training integrated, then it is much more powerful than doccano.
- Multi Platform support
  - Most products only support Web Apps, but a mobile App may be more convenient for some tasks like image segmentation annotation.

    - Moreover, considering the use case of the service, it may be more convenient to just use a phone.
    - This is out of the scope of this project, but if we want to use it in the future, then it may be a desired feature.
- How can it attract researchers?
    - Easy to gain intuition about the dataset and prediction
    - Easy to change the algorithm and models
    - But generally speaking, it is hard to attract active learning researchers
- How can it attract industrial practitioners?
    - Data quality control, data versioning, collaborative labeling

Deliverables:

- Demo