

Build a Data Annotation Tool using AL and TL

Final Presentation

Thursday, August 19, 2021

Presented by: Yongchao Zhou

Motivation

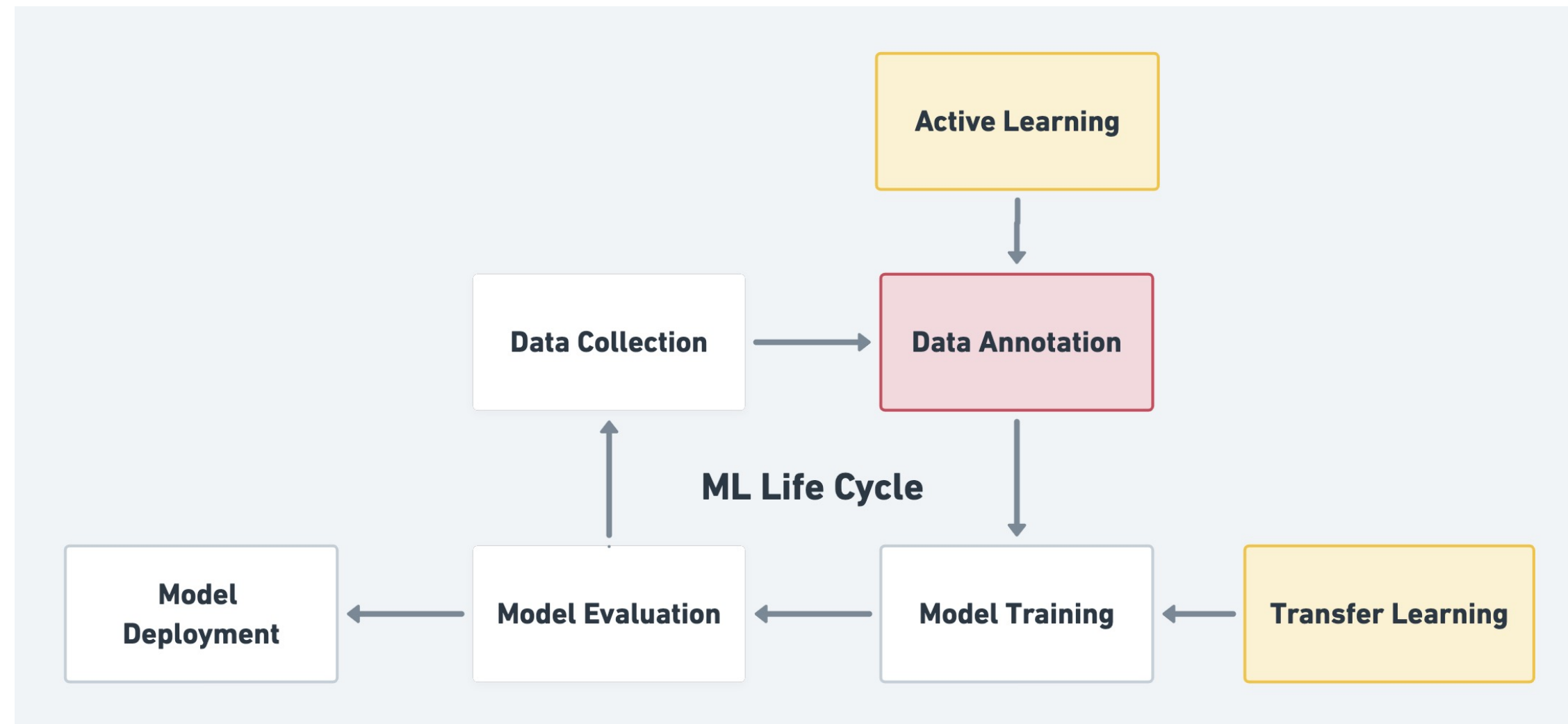


Figure 1: Machine Learning Life Cycle

Motivation

How to build a label-efficient ML System?

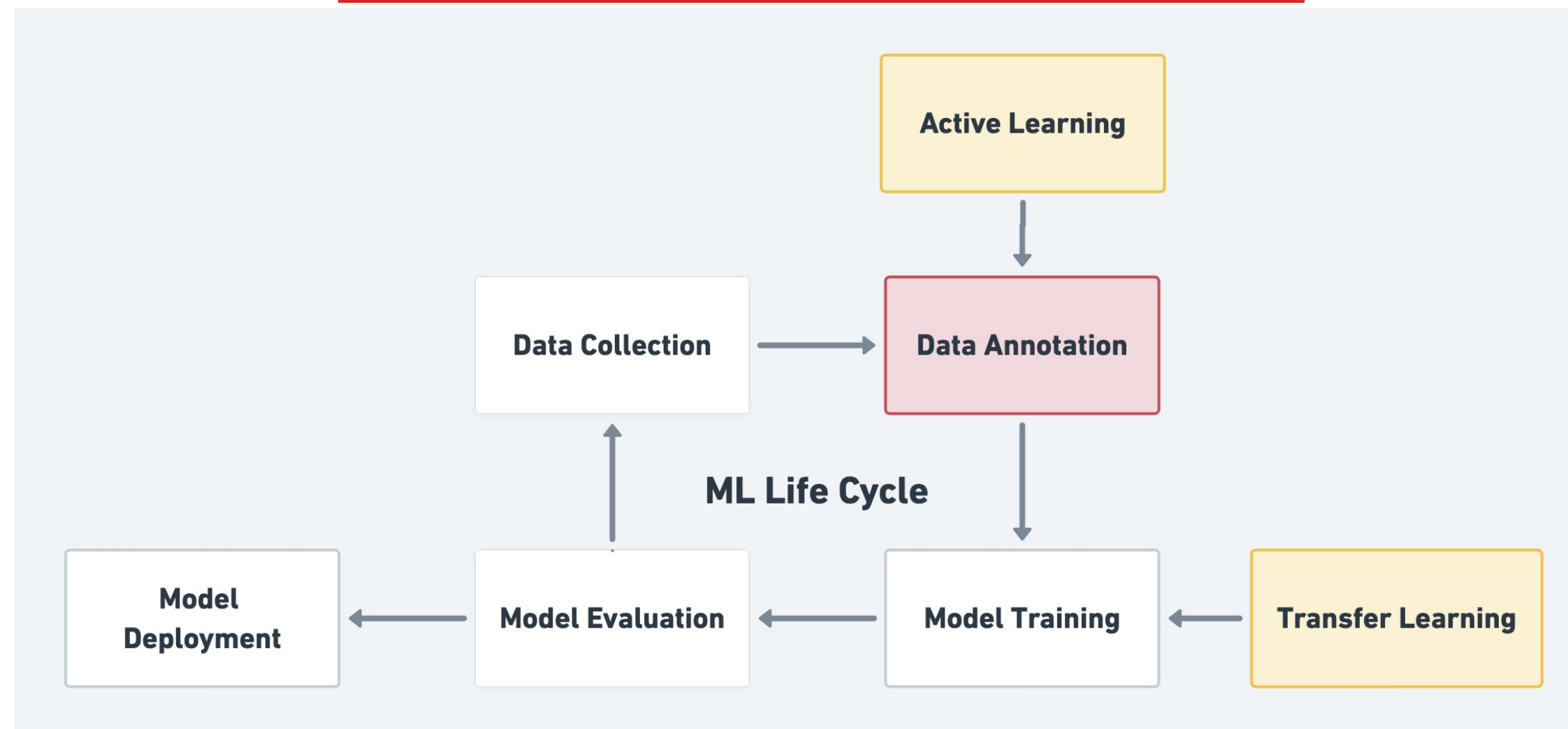


Figure 1: Machine Learning Life Cycle

Motivation

How to build a label-efficient ML System?

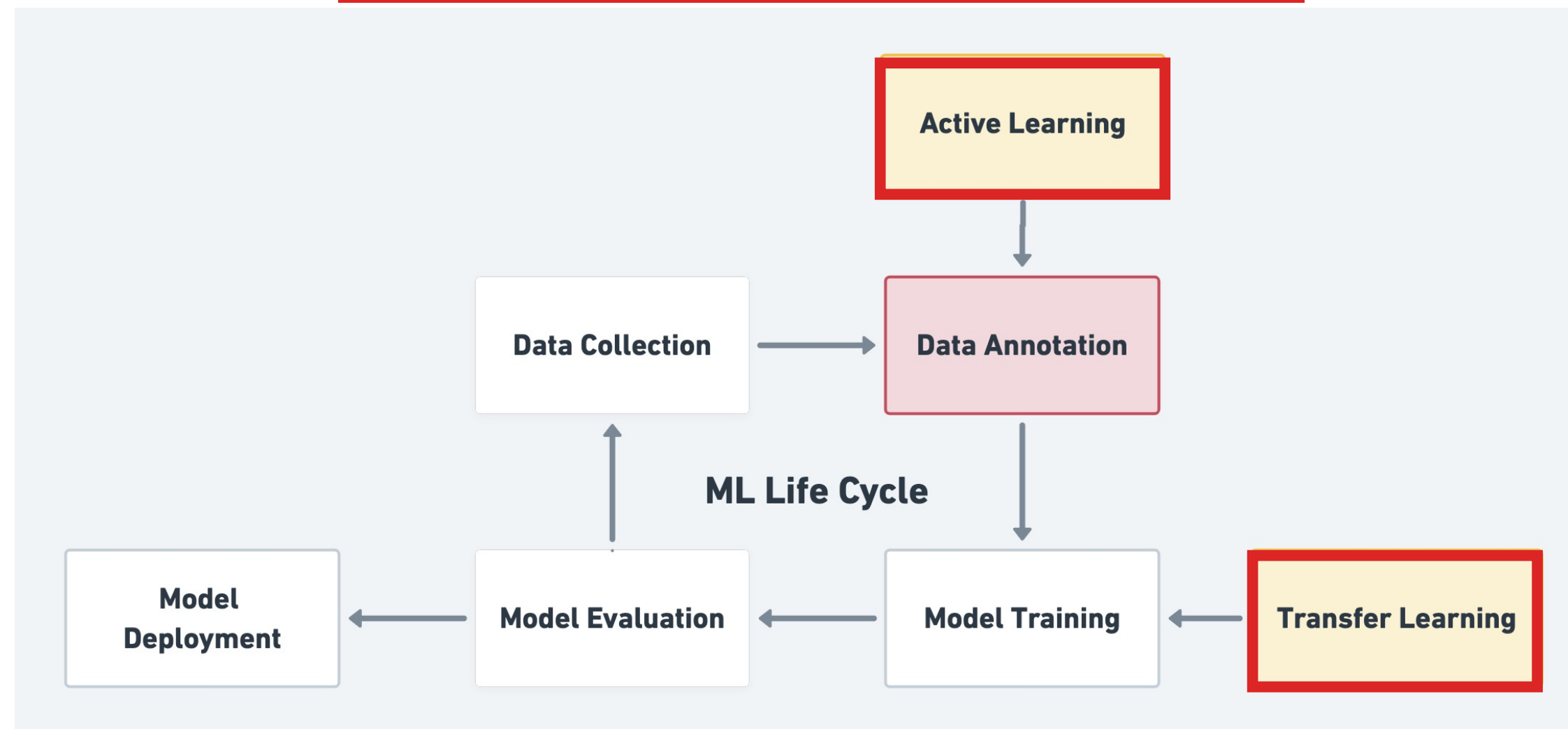


Figure 1: Machine Learning Life Cycle

Why AL + TL ?

Table 1: Comparison of Active Learning and Transfer Learning

| | Active Learning (AL) | Transfer Learning (TL) |
|--------|---------------------------------|---|
| Goal | Select most informative example | Reuse knowlege learned elsewhere |
| Effect | Reduce number of labeled data | Reduce model training time Reduce number of labeled data |
| Target | Data | Model |

Live Demo - DANER

Named Entity Recognition Service

☒ ORG

☒ PRODUCT

☒ GPE

☒ LOC

☒ PERSON

☐ MISC

☐ NORP

☐ FACILITY

☐ EVENT

☐ LAW

☐ LANGUAGE

☐ ART

☐ DATE

☐ TIME

☐ MONEY

☐ QUANTITY

☐ ORDINAL

☐ CARDINAL

☐ PERCENT

SELECT ALL

DEFAULT

RESET

Model

en_core_web_sm

Hinton is viewed as a leading figure in the deep learning community. The dramatic image-recognition milestone of the AlexNet designed in collaboration with his students Alex Krizhevsky and Ilya Sutskever for the ImageNet challenge 2012 was a breakthrough in the field of computer vision.

Max characters 1500

287

ANALYZE

RESET

Hinton **ORG** is viewed as a leading figure in the deep learning community. The dramatic image-recognition milestone of the AlexNet **ORG** designed in collaboration with his students Alex Krizhevsky **PERSON** and Ilya Sutskever **PERSON** for the ImageNet **ORG** challenge 2012 was a breakthrough in the field of computer vision.

Figure 2: NER Inference Service

Annotation Configuration

Datasets

CONLL2003

Model

hf_distillbert

☒ ORG

☒ LOC

☒ PER

☒ MISC

☐ PRODUCT

☐ DATE

☐ TIME

☐ MONEY

☐ QUANTITY

SELECT ALL

DEFAULT

RESET

Show Confidence

Confidence Level: 0

100

Active Learning

AL Algorithm

MNLP

START

Early Phase (Demo Only)

Annotation

<

>

✓

✕

☒ ORG

☐ LOC

☐ PER

☐ MISC

Canadian **MISC** bonds opened softer on Friday , pulled lower by a sinking U.S. **LOC** market , but outperformed U.S. **LOC** bonds on positive Canadian **LOC** economic data , analysts said .

Figure 3: NER Annotation Service

How to implement?

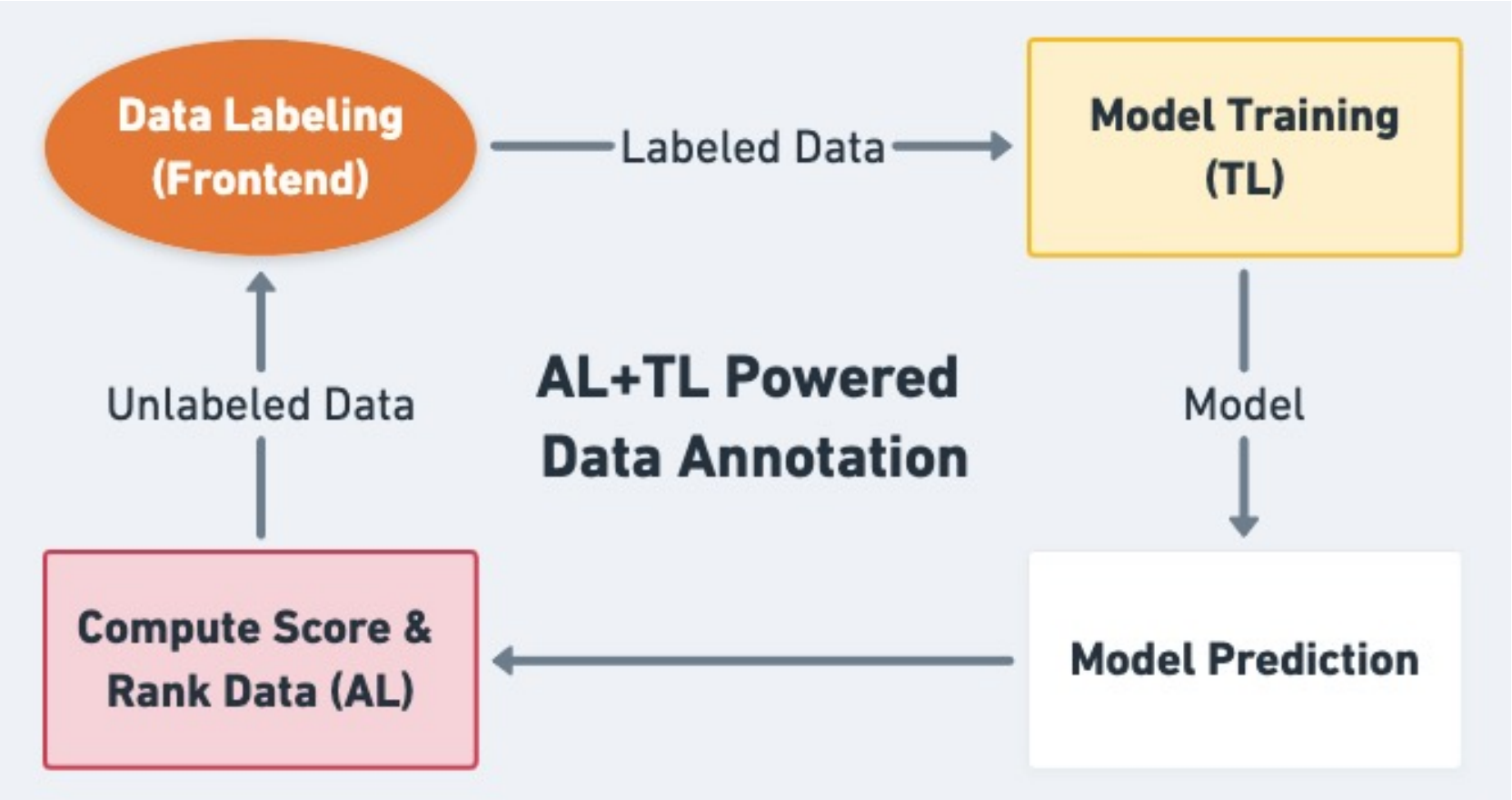


Figure 4: Data Annotation Process

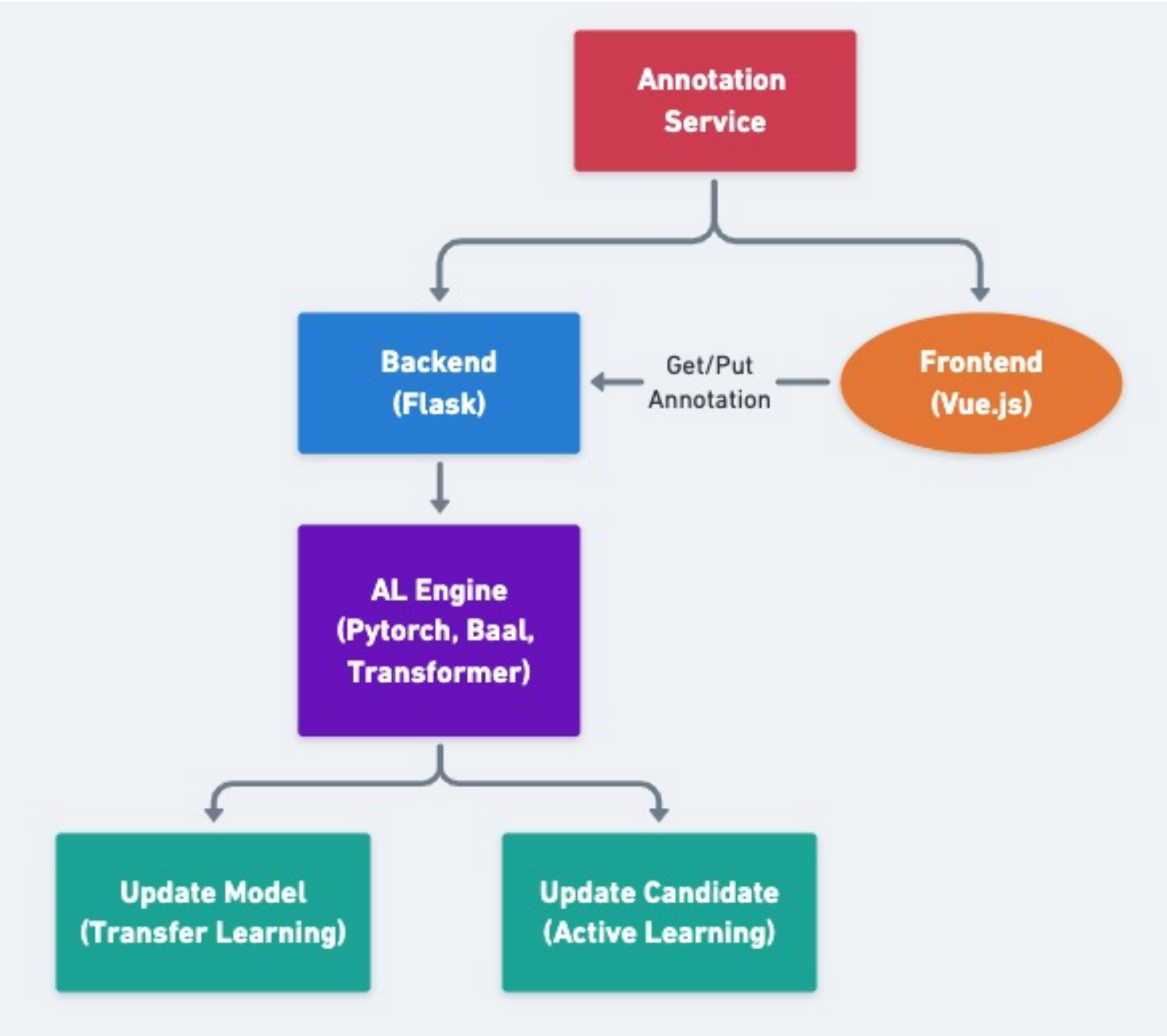


Figure 5: Annotation Service Structure Overview

How to implement?

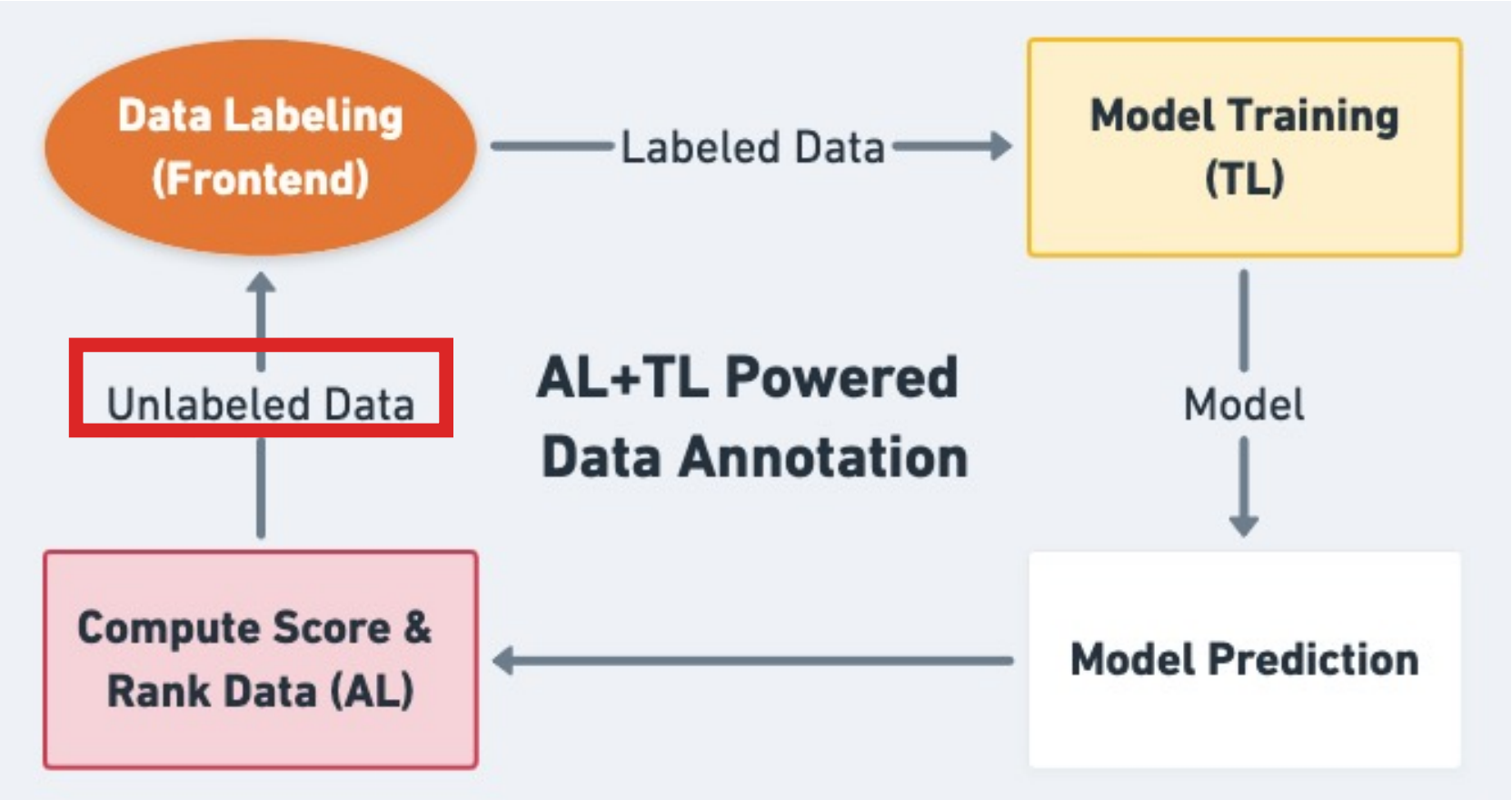


Figure 4: Data Annotation Process

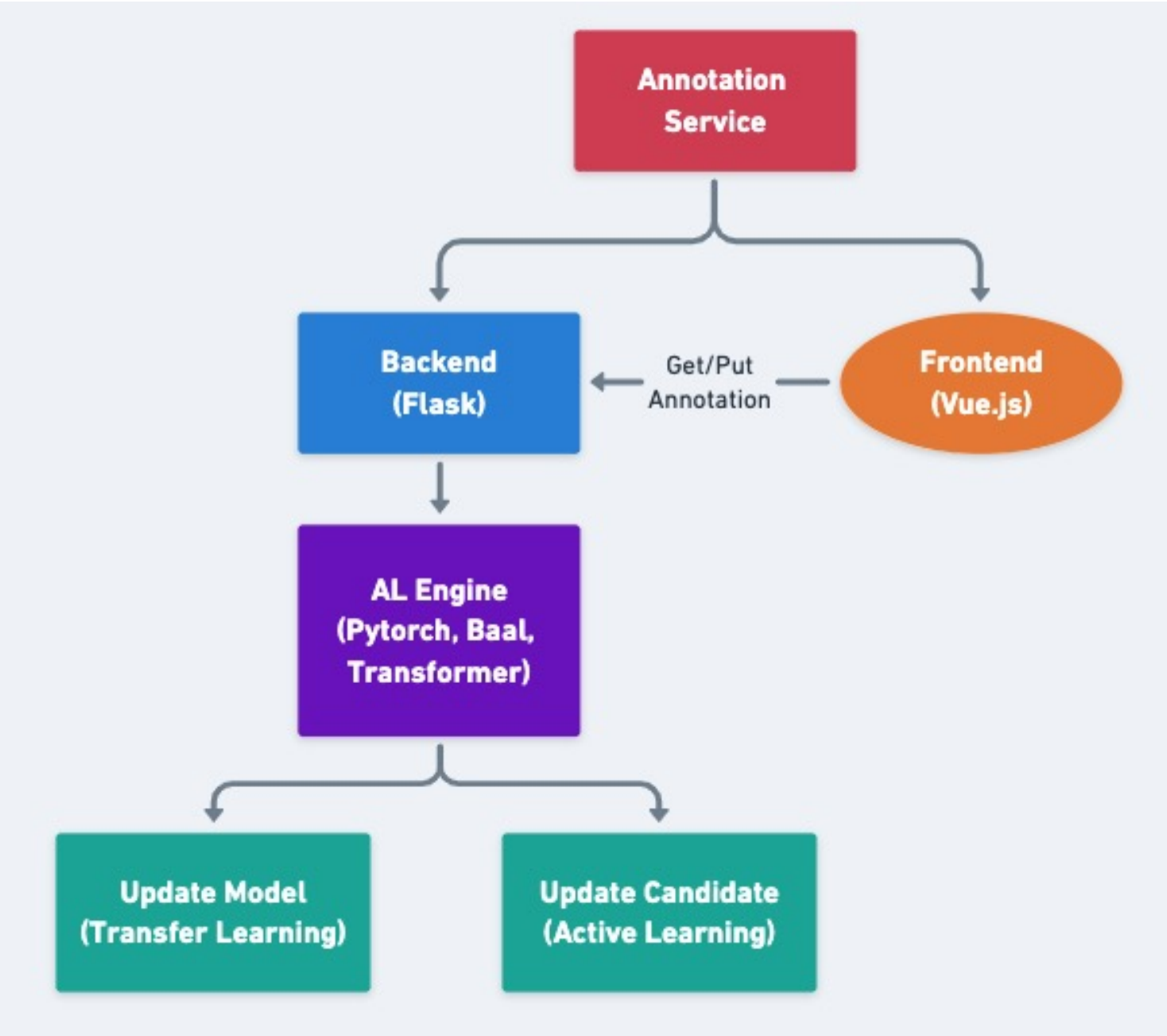


Figure 5: Annotation Service Structure Overview

How to implement?

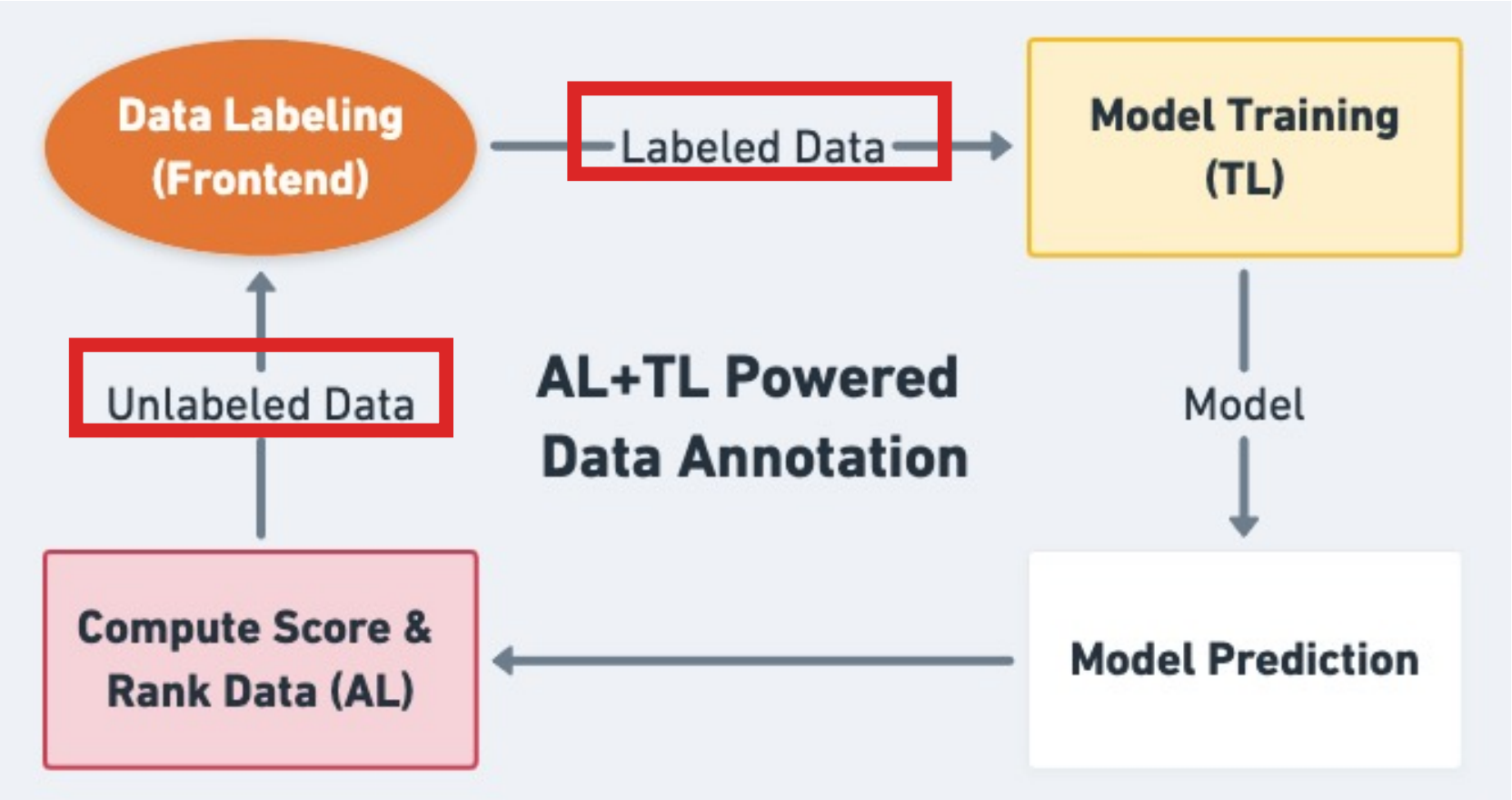


Figure 4: Data Annotation Process

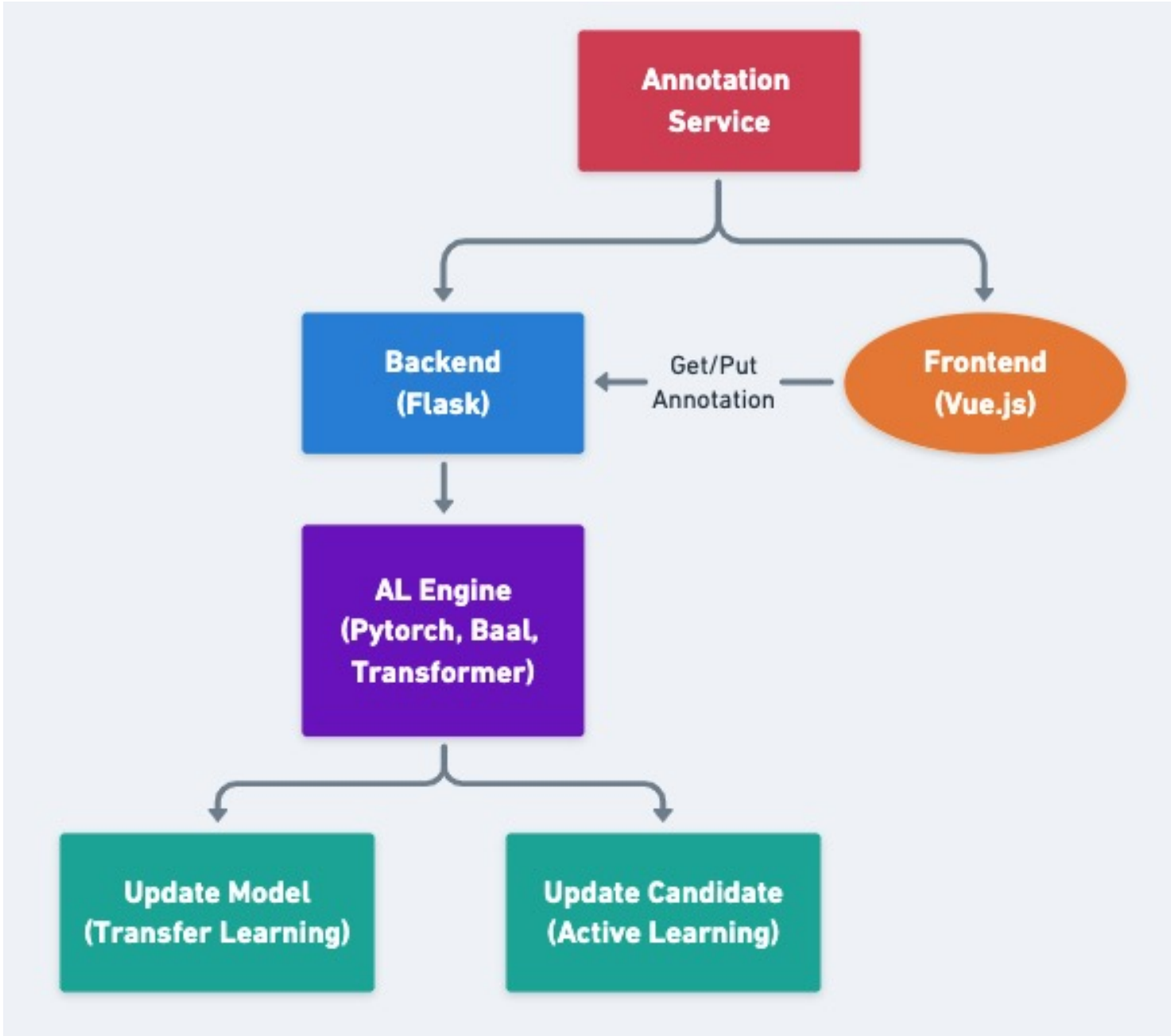


Figure 5: Annotation Service Structure Overview

Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64,  # batch size for evaluation
    weight_decay=0.01,             # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,                # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,           # the tokenizer that is compatible with model
    args=training_args,           # training arguments, defined above
    train_dataset=active_set,      # AL dataset
)

trainer.train()
```

Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64,  # batch size for evaluation
    weight_decay=0.01,             # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,                # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,           # the tokenizer that is compatible with model
    args=training_args,            # training arguments, defined above
    train_dataset=active_set,      # AL dataset
)

trainer.train()
```

Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64,  # batch size for evaluation
    weight_decay=0.01,             # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,                # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,           # the tokenizer that is compatible with model
    args=training_args,           # training arguments, defined above
    train_dataset=active_set,      # AL dataset
)

trainer.train()
```

Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64, # batch size for evaluation
    weight_decay=0.01,           # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,               # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,          # the tokenizer that is compatible with model
    args=training_args,          # training arguments, defined above
    train_dataset=active_set,     # AL dataset
)

trainer.train()
```

Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64,  # batch size for evaluation
    weight_decay=0.01,             # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,                # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,           # the tokenizer that is compatible with model
    args=training_args,           # training arguments, defined above
    train_dataset=active_set,      # AL dataset
)

trainer.train()
```

Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64,  # batch size for evaluation
    weight_decay=0.01,             # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,                # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,           # the tokenizer that is compatible with model
    args=training_args,           # training arguments, defined above
    train_dataset=active_set,      # AL dataset
)

trainer.train()
```


Update Model - Transfer Learning

```
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer, TrainingArguments

training_args = TrainingArguments(
    max_steps=MAX_STEPS,          # total number of training steps per AL step
    per_device_train_batch_size=32, # batch size per device during training
    per_device_eval_batch_size=64,  # batch size for evaluation
    weight_decay=0.01,             # strength of weight decay
)

tokenizer = AutoTokenizer.from_pretrained(model_name)
hf_model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(label_list))

trainer = Trainer(
    model=hf_model,               # the instantiated 🤗 Transformers model to be trained
    tokenizer=tokenizer,          # the tokenizer that is compatible with model
    args=training_args,          # training arguments, defined above
    train_dataset=active_set,     # AL dataset
)

trainer.train()
```

Update Candidate - Active Learning

- Maximum Normalized Log-Probability (MNLP)
 - Intuition: select the most uncertain points.
 - How to represent the uncertainty?
 - Let \mathbf{x}_i be the i^{th} data point with n tokens.
 - The probability of model current prediction: $\max_{y_1, \dots, y_n} \mathbb{P}[y_1, \dots, y_n \mid \mathbf{x}_i]$
 - The normalized log probability of model current prediction on \mathbf{x}_i is computed as,

$$MNLP(\mathbf{x}_i) = \max_{y^*1, \dots, y^*n} \frac{1}{n} \sum *j = 1^n \log \mathbb{P}[y^*j \mid \mathbf{x}_i]$$

- AL Data Candidate $\{\mathbf{x}\} = \arg \min_{\mathbf{x}_i} MNLP(\mathbf{x}_i)$

Update Candidate - Active Learning

- Maximum Normalized Log-Probability (MNLP)
 - Intuition: select the most uncertain points.
 - How to represent the uncertainty?
 - Let \mathbf{x}_i be the i^{th} data point with n tokens.
 - The probability of model current prediction: $\max_{y_1, \dots, y_n} \mathbb{P}[y_1, \dots, y_n \mid \mathbf{x}_i]$
 - The normalized log probability of model current prediction on \mathbf{x}_i is computed as,

$$MNLP(\mathbf{x}_i) = \max_{y^*1, \dots, y^*n} \frac{1}{n} \sum^* j = 1^n \log \mathbb{P}[y^* j \mid \mathbf{x}_i]$$

- AL Data Candidate $\{\mathbf{x}\} = \arg \min_{\mathbf{x}_i} MNLP(\mathbf{x}_i)$

Update Candidate - Active Learning

- Maximum Normalized Log-Probability (MNLP)
 - Intuition: select the most uncertain points.
 - How to represent the uncertainty?
 - Let \mathbf{x}_i be the i^{th} data point with n tokens.
 - The probability of model current prediction: $\max_{y_1, \dots, y_n} \mathbb{P}[y_1, \dots, y_n \mid \mathbf{x}_i]$
 - The normalized log probability of model current prediction on \mathbf{x}_i is computed as,

$$MNLP(\mathbf{x}_i) = \max_{y^*1, \dots, y^*n} \frac{1}{n} \sum^* j = 1^n \log \mathbb{P}[y^* j \mid \mathbf{x}_i]$$

- AL Data Candidate $\{\mathbf{x}\} = \arg \min_{\mathbf{x}_i} MNLP(\mathbf{x}_i)$

Update Candidate - Active Learning

- Maximum Normalized Log-Probability (MNLP)

- Intuition: select the most uncertain points.

- How to represent the uncertainty?

- Let \mathbf{x}_i be the i^{th} data point with n tokens.

- The probability of model current prediction: $\max_{y_1, \dots, y_n} \mathbb{P}[y_1, \dots, y_n \mid \mathbf{x}_i]$

- The normalized log probability of model current prediction on \mathbf{x}_i is computed as,

$$MNLP(\mathbf{x}_i) = \max_{y^*1, \dots, y^*n} \frac{1}{n} \sum *j = 1^n \log \mathbb{P}[y^*j \mid \mathbf{x}_i]$$

- AL Data Candidate $\{\mathbf{x}\} = \arg \min_{\mathbf{x}_i} MNLP(\mathbf{x}_i)$

Select data with small MNLP score!

Small score = Low Confidence = High Uncertainty

Limitation and Next Step

- Active Learning Performance Gap
 - Dataset? Architecture?
 - Transfer Learning?
 - More robust AL Algorithm is needed
- Noisy Human Label
 - Human label may not be reliable

User 1

Geoffrey Everest Hinton **PERSON** (born **6** **CARDINAL** **December 1947** **DATE**) is a **British** **NORP** - Canadian cognitive psychologist and computer scientist , most noted for his work on artificial neural networks .

User 2

Geoffrey Everest Hinton (born **6 December 1947** **DATE**) is a **British** **NORP** - Canadian cognitive psychologist and computer scientist , most noted for his work on artificial neural networks .

- Data Quality Control

Limitation and Next Step

- Active Learning Performance Gap
 - Dataset? Architecture?
 - Transfer Learning?
 - More robust AL Algorithm is needed
- Noisy Human Label
 - Human label may not be reliable

User 1

Geoffrey Everest Hinton PERSON (born 6 CARDINAL December 1947 DATE) is a British NORP - Canadian cognitive psychologist and computer scientist , most noted for his work on artificial neural networks .

User 2

Geoffrey Everest Hinton (born 6 December 1947 DATE) is a British NORP - Canadian cognitive psychologist and computer scientist , most noted for his work on artificial neural networks .

- Data Quality Control

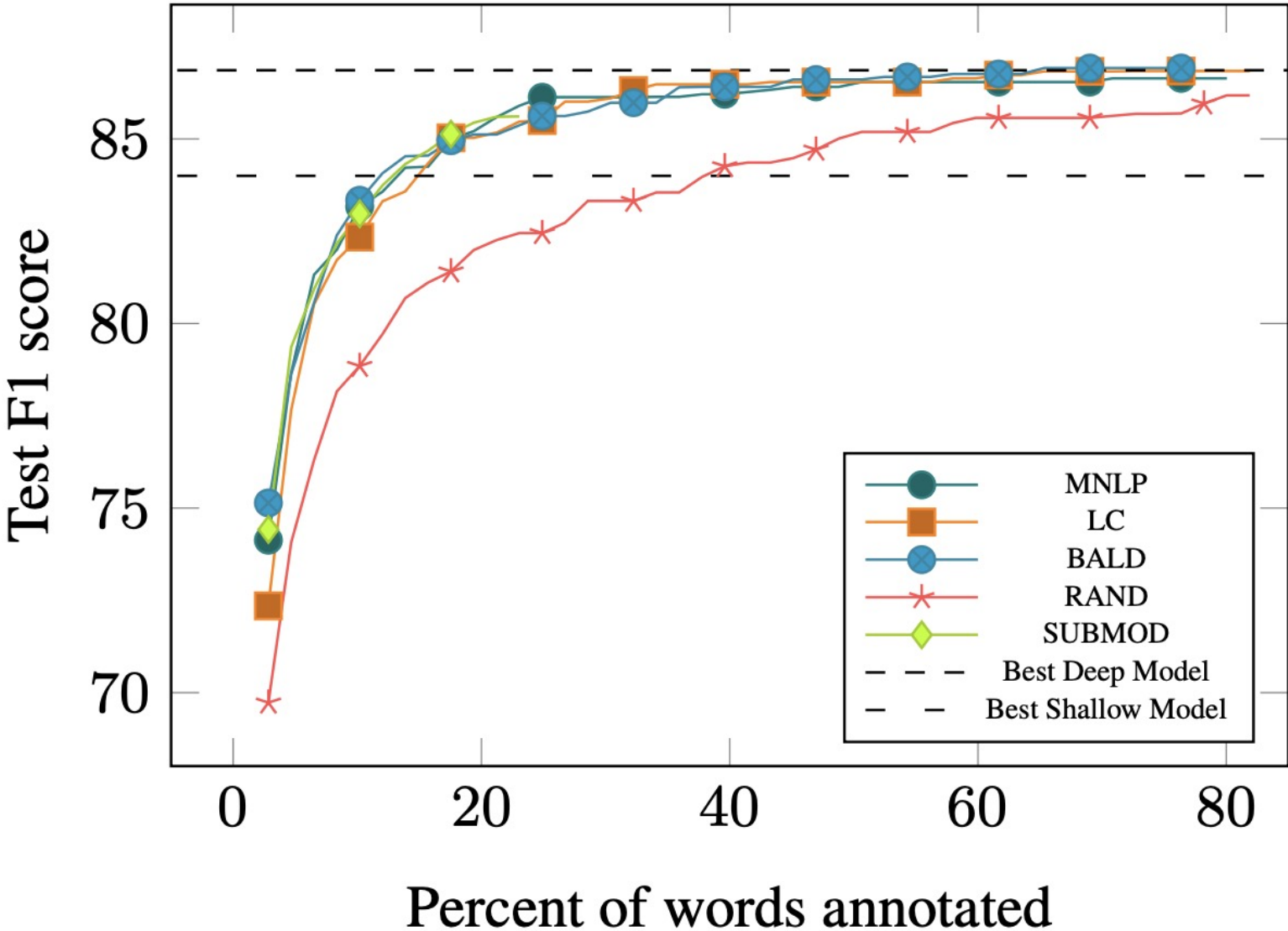


Figure 6: AL Performance from Literature [1]

[1] Deep Active Learning for Named Entity Recognition (ICLR2018)

Take away

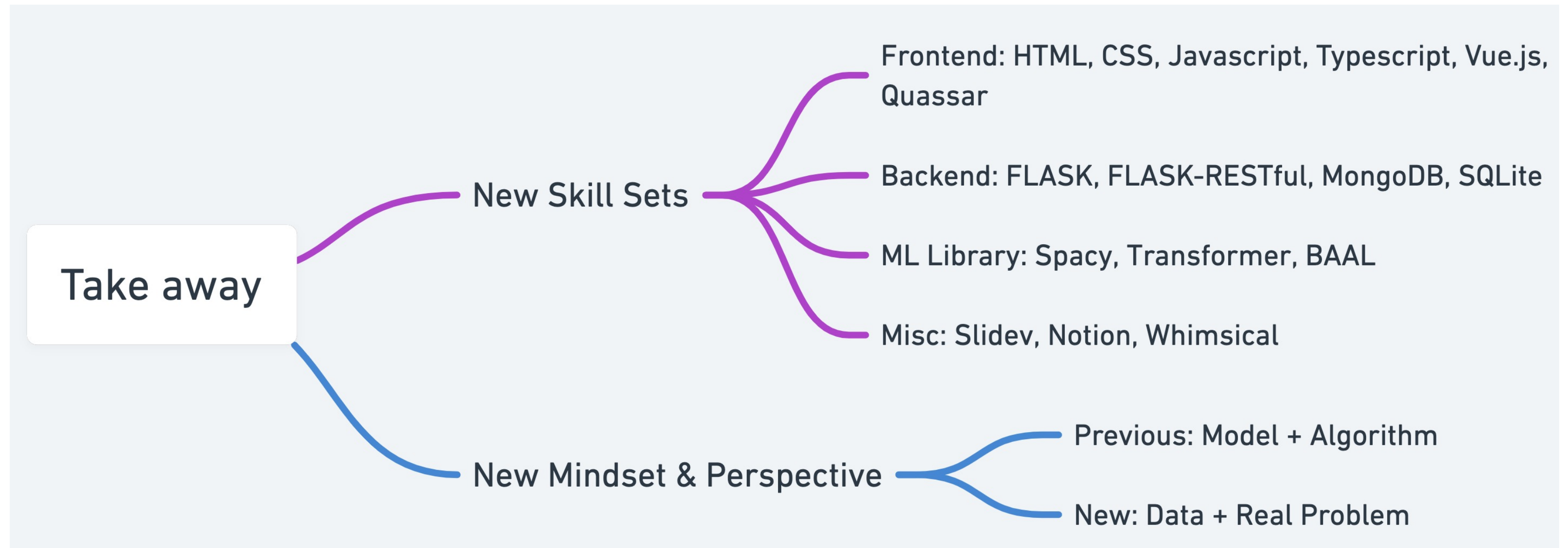


Figure 7: Internship Take Away