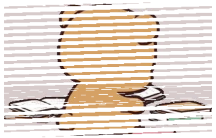
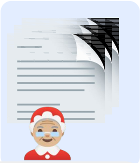
Federated Named-Entity Recognition

Jennifer Andersson, Lovisa Hagström, Kätriin Kukk, Thibault Marette

The Problem

- The santa association wants to detect vital information in text to develop an accurate naughty list.
- However, each country in the federation has little data and cannot share it.

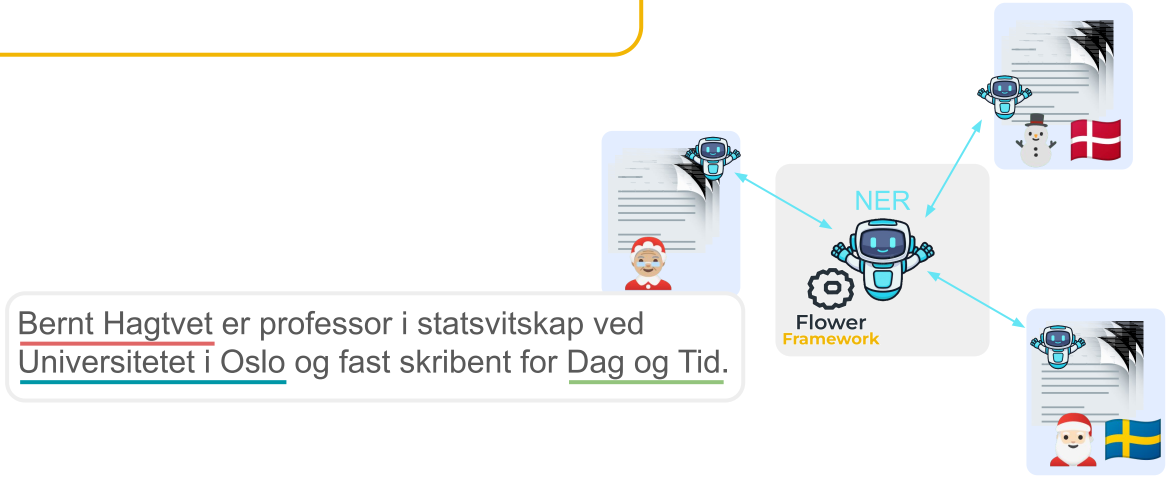




Bernt Hagtvet er professor i statsvitskap ved Universitetet i Oslo og fast skribent for Dag og Tid.

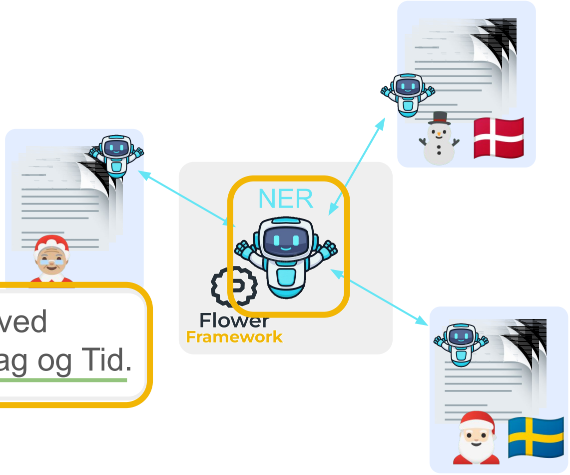
The Solution

- Use **Named-Entity Recognition** to automatically detect named entities in text.
- Leverage **federated learning** to share data resources.



What is Named-Entity Recognition?

Bernt Hagtvet er professor i statsvitskap ved Universitetet i Oslo og fast skribent for Dag og Tid.



What is Named-Entity Recognition?

Person real or fictional characters and animals

Organisation any collection of people

Location geographical places and buildings

Product artificially produced entities

Transformer encoder models are typically trained to detect named entities in some language.

Bernt Hagtvet er professor i statsvitskap ved Universitetet i Oslo og fast skribent for Dag og Tid.

Apple announced their new iCase.

Did you eat the last apple?

From NER to Federated NER

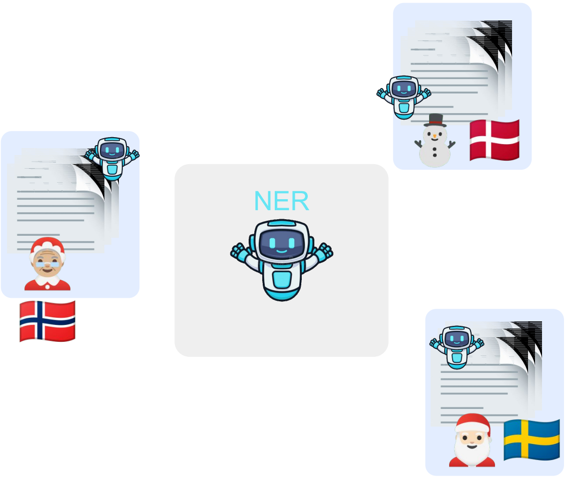
Problem: Each federation wants to collaborate but cannot share their own data.

We leveraged Federated Learning for NER.

From NER to Federated NER

Problem: Each federation wants to collaborate but cannot share their own data.

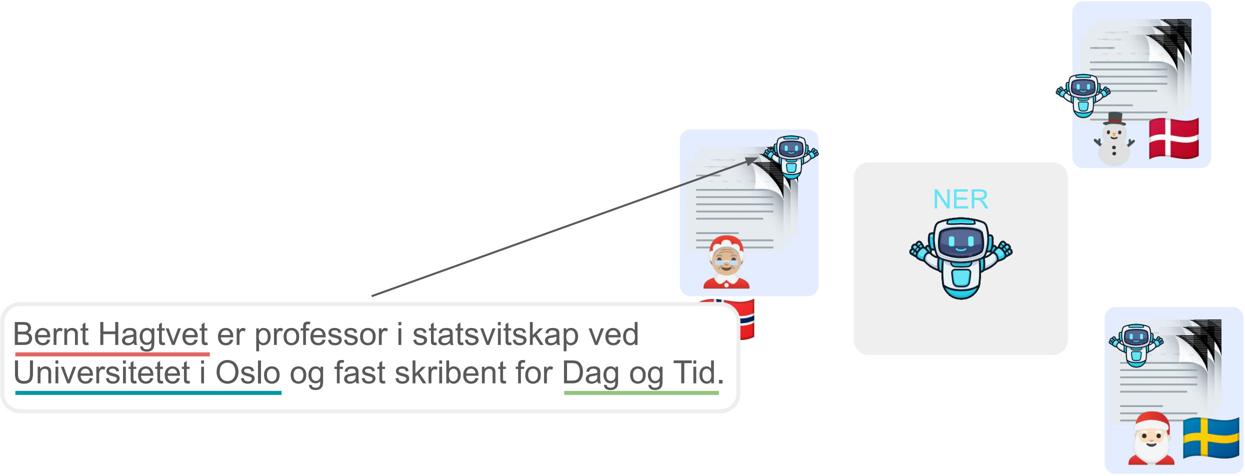
We leveraged Federated Learning for NER.



From NER to Federated NER

Problem: Each federation wants to collaborate but cannot share their own data.

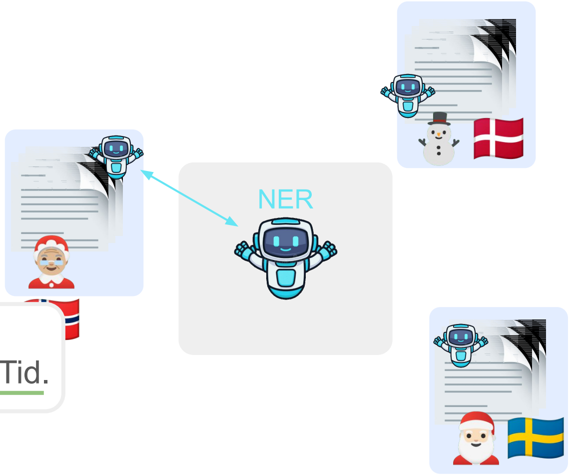
We leveraged Federated Learning for NER.



From NER to Federated NER

Problem: Each federation wants to collaborate but cannot share their own data.

We leveraged Federated Learning for NER.

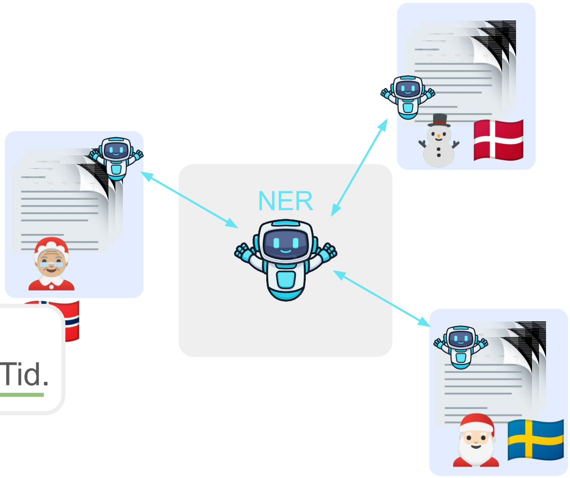


Bernt Hagtvet er professor i statsvitskap ved Universitetet i Oslo og fast skribent for Dag og Tid.

From NER to Federated NER

Problem: Each federation wants to collaborate but cannot share their own data.

We leveraged Federated Learning for NER.



Bernt Hagtvet er professor i statsvitskap ved Universitetet i Oslo og fast skribent for Dag og Tid.

From NER to Federated NER

Problem: Each country wants to collaborate but cannot share their own data.

We leveraged **Federated Learning** for NER.

In the real-world, we need a few practicalities:

- A dataset
- A model
- A machine to deploy our implementation
- A framework to simulate communication

UNER: a unified dataset corpora for NER tasks

We use new datasets from **UNER project**, a recent effort to develop **consistent** NER benchmarks in many languages.

Dataset features:

- 3 tags: Person Organisation and Location.
- Solving in and inter dataset ambiguities using **clear** and **common annotation guidelines.**
- **Heterogeneous** data to simulate the federated setting.

Dataset	Language	#sentences	Content
sv_talbanken	Swedish	4 303	Professional prose
da_ddt	Danish	5 512	Fiction, nonfiction, news, Speech
nob_norne	Norwegian (bokmål)	16 309	Literature, newspapers, government publications
nno_norne	Norwegian (nynorsk)	14 878	Literature, newspapers, government publications

Base model overview

We used Facebook’s **XLM-RoBERTa** as a pre-trained model.

- Transformer-based language model pre-trained on 100 different languages
- Multilingual model without sacrificing per-language performance
- Can determine language on the input

Fine-tuning the model for our project

- Setting hyperparameters based on the one reported in the research paper
- Warm Up phase
- Perform **early stopping** based on validation loss

Model deployment

- Requested access to national NAISS ressource **Alvis**
  - Deployed our implementation on a **small cluster** (4 NVIDIA Tesla A40 GPUs with 48GB RAM)
  - Each client runs on a **separate GPU** within the node.

Federated Learning with Flower

- **Flower** → open-source framework for federated learning

Flower
Framework

Federated Learning with Flower

- **Flower** → open-source framework for federated learning
- **Easy** to transition from centralized ML to federated ML

Flower Framework

- framework agnostic → builds on top of existing DL frameworks
- compatible with Python, Android, iOS, C++ etc

Federated Learning with Flower

Flower
Framework

- **Flower** → open-source framework for federated learning
- **Easy** to transition from centralized ML to federated ML

- framework agnostic → builds on top of existing DL frameworks
- compatible with Python, Android, iOS, C++ etc

- **Scalable** and **flexible** → move seamlessly from prototyping to production

- Prototyping in simulation mode (on your laptop or GPUs in a cluster)
- Smooth transition to production mode, with little to no changes in your code

Federated Learning with Flower

- **Flower** → open-source framework for federated learning
- **Easy** to transition from centralized ML to federated ML

- framework agnostic → builds on top of existing DL frameworks
- compatible with Python, Android, iOS, C++ etc

- **Scalable** and **flexible** → move seamlessly from prototyping to production

- Prototyping in simulation mode (on your laptop or GPUs in a cluster)
- Smooth transition to production mode, with little to no changes in your code

- **Flower Datasets** → benchmarking made easy

Federated Learning with Flower

- **Client handles:**

- Each client trains model based on its individual data partition (i.e. language)

Federated Learning with Flower

- **Client handles:**

- Each client trains model based on its individual data partition (i.e. language)

- **Flower handles**:

- Coordinating communication between server and clients
- Combining client model updates into a global model
- Orchestrating simulation mode

Federated Learning with Flower

- **Client handles:**
  - Each client trains model based on its individual data partition (i.e. language)

- **Flower handles**:
  - Coordinating communication between server and clients
  - Combining client model updates into a global model
  - Orchestrating simulation mode

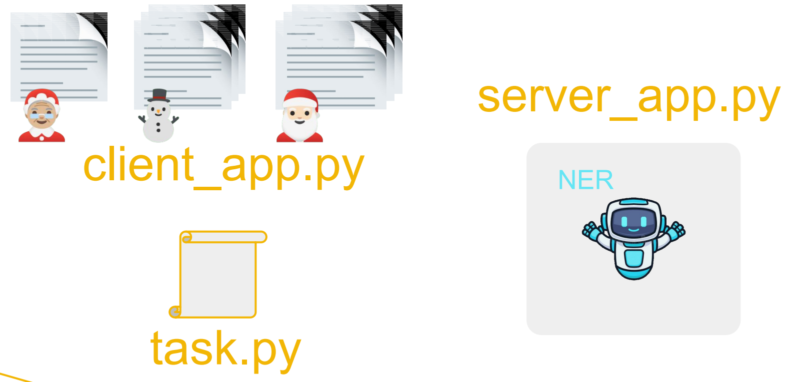
- **Code structure:**
  - Server app, client app and task.py

Federated Learning with Flower

- **Client handles:**
  - Each client trains model based on its individual data partition (i.e. language)

- **Flower handles**:
  - Coordinating communication between server and clients
  - Combining client model updates into a global model
  - Orchestrating simulation mode

- **Code structure:**
  - Server app, client app and task.py

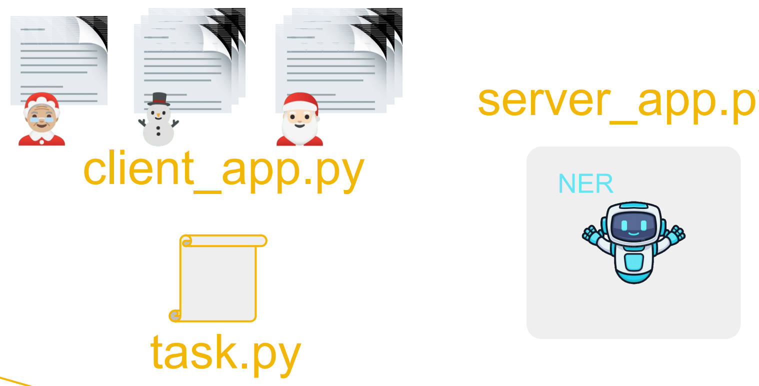


Federated Learning with Flower

- **Client handles:**
  - Each client trains model based on its individual data partition (i.e. language)

- **Flower handles**:
  - Coordinating communication between server and clients
  - Combining client model updates into a global model
  - Orchestrating simulation mode

- **Code structure:**
  - Server app, client app and task.py



- We use **PyTorch Lightning** and the **FedAvg** aggregation strategy

Results

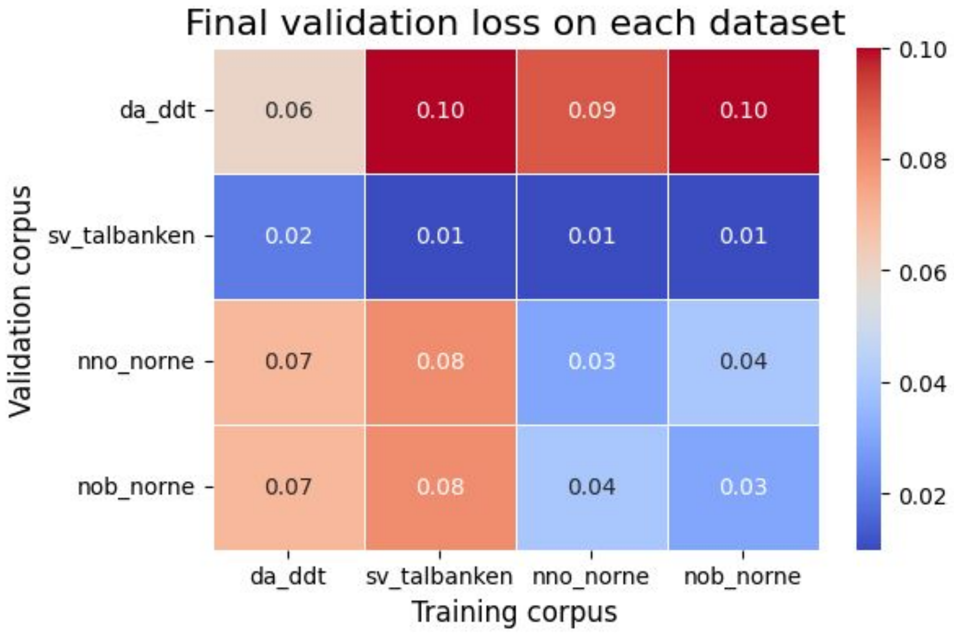
- **Baseline**
  - Trained and evaluated on a single dataset
  - Monitored validation loss on all 4 datasets
  - 10 epochs, limited number of steps per epoch, early stopping
  - Evaluated using precision, recall and F1

- **Federated learning**
  - Each client was training and evaluating on their own dataset
  - 10 rounds, 1 epoch per round, limited number of steps per epoch
  - At the end of each round the global model is updated
  - Evaluated using precision, recall and F1

S1	S2	S3
da_ddt + sv_talbanken	S1 + nno_norne	S2 + nob_norne

Results: Baseline

Dataset	B: Precision	B: Recall	B: F1-score
da_ddt	0.80	0.82	0.81
sv_talbanken	0.66	0.86	0.75
nno_norne	0.83	0.89	0.86
nob_norne	0.85	0.87	0.86

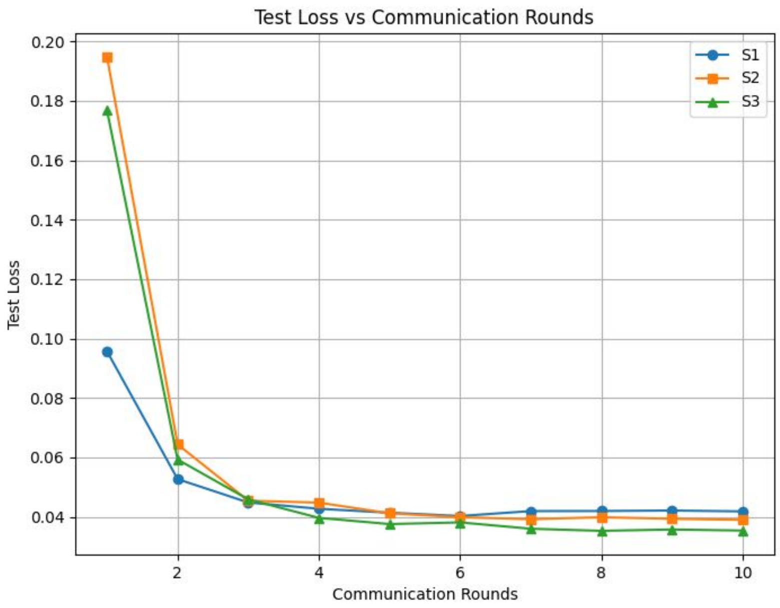


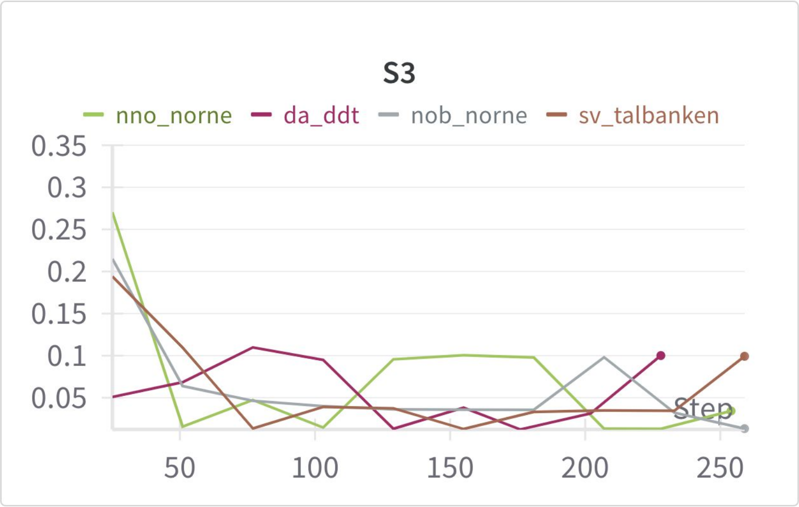
Results: Baseline (above) vs Federated learning (below)

Dataset	B: Precision	B: Recall	B: F1-score
da_ddt	0.80	0.82	0.81
sv_talbanken	0.66	0.86	0.75
nno_norne	0.83	0.89	0.86
nob_norne	0.85	0.87	0.86

Dataset	S1: Precision	S1: Recall	S1: F1-score	S2: Precision	S2: Recall	S2: F1-score	S3: Precision	S3: Recall	S3: F1-score
da_ddt	0.84	0.81	0.82	0.84	0.87	**0.86**	0.86	0.75	0.80
sv_talbanken	0.78	0.90	**0.84**	0.85	0.76	0.80	0.86	0.75	0.80
nno_norne				0.85	0.76	0.80	0.87	0.91	**0.89**
nob_norne							0.82	0.85	**0.84**
average performace	0.81	0.86	0.83	0.85	0.80	0.82	0.85	0.82	0.83

Results: Test loss aggregated (left) and per dataset (right)

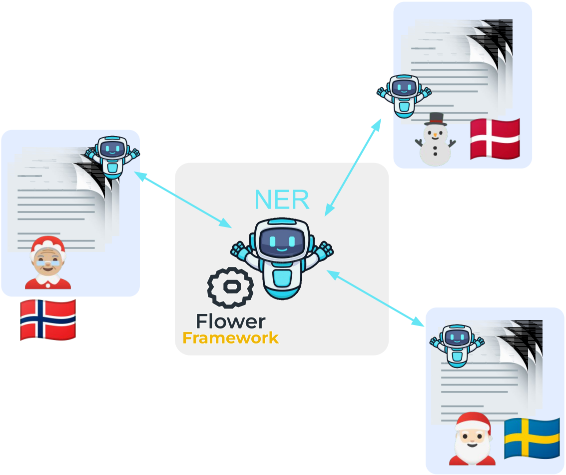




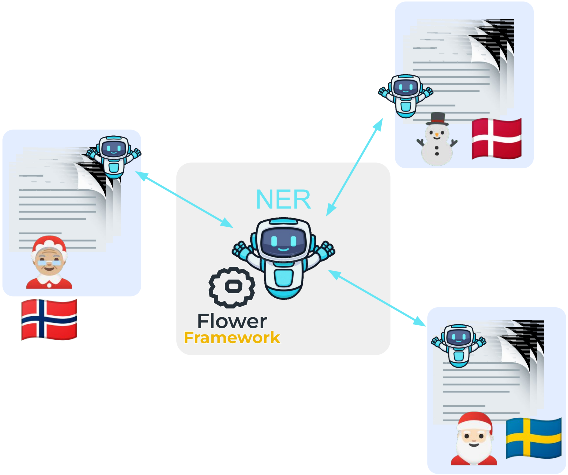
Discussion

- In 3 cases out of 4, the best F1-score was achieved in a federated setup
- Federated learning seems more beneficial for small datasets than for large ones
- Results are sensitive to dataset composition
- Competition between datasets
- Communication issues when sending model parameters back and forth
- Limitations
  - Used one small model
  - Only tried one federated setup

What’s Next?

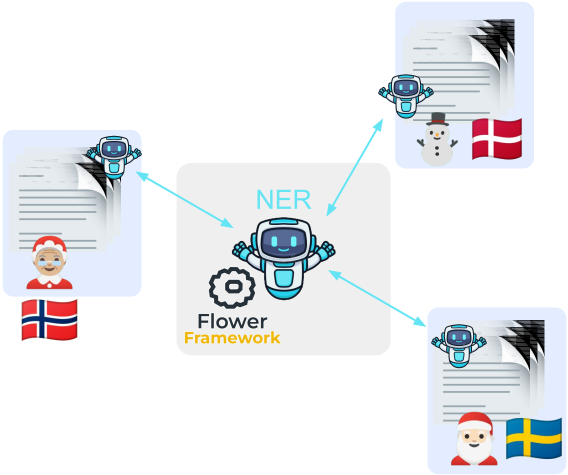


What’s Next?



- Investigate different aggregation strategies and possibility of using federated learning to enhance performance on rare language datasets
- Scale up experiments further and investigate federated performance using a large number of clients in a truly federated setting
- Explore integration with FEDn

What's Next?



- Investigate different aggregation strategies and possibility of using federated learning to enhance performance on rare language datasets
- Scale up experiments further and investigate federated performance using a large number of clients in a truly federated setting
- Explore integration with FEDn

Thank you for listening!