# Named entity recognition system

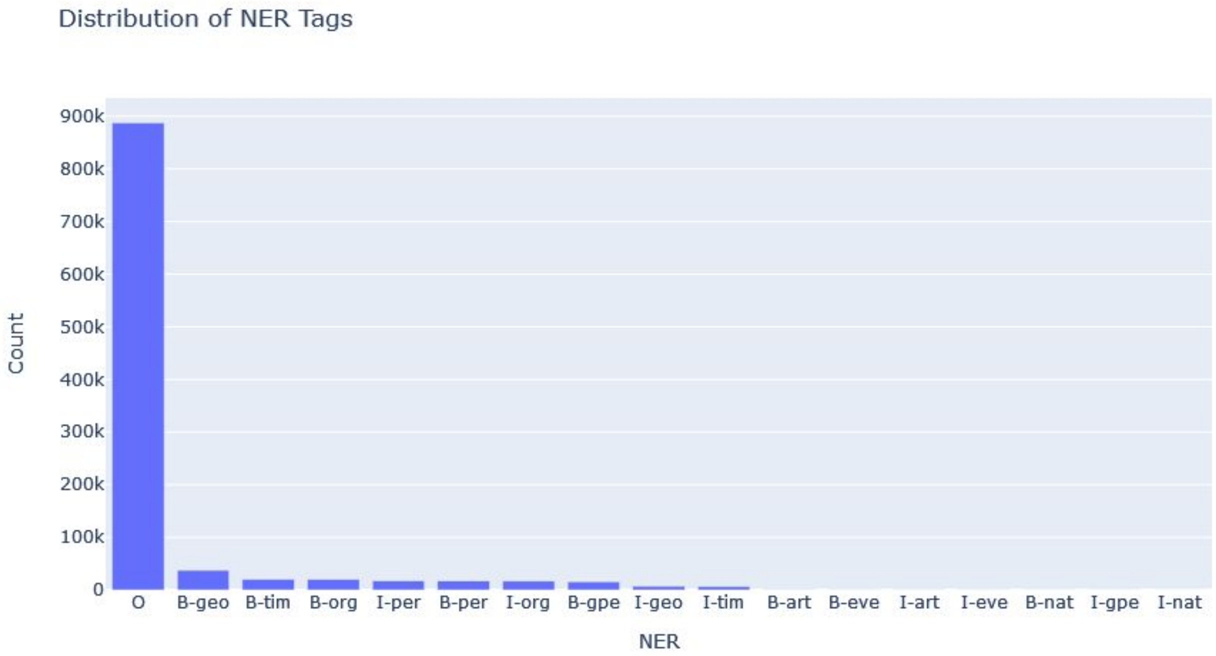
# Objective

The goal of case study is to develop a NER system which recognizes entities from the given text sentence.

# 1. <u>Exploratory data analysis</u>

# Exploratory analysis

## NER Tags distribution

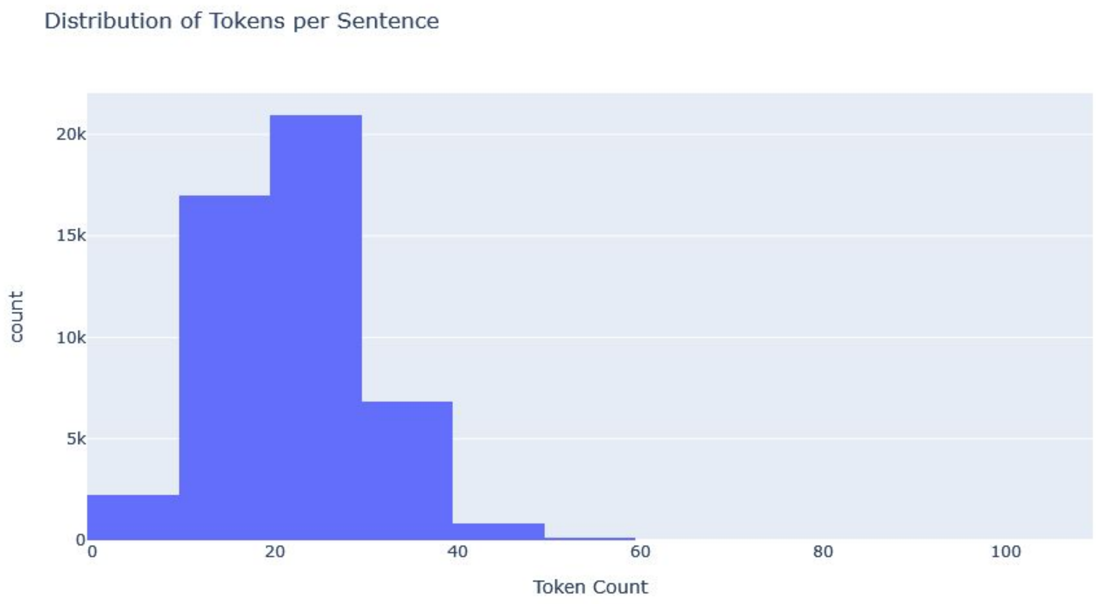


# Exploratory analysis

| Rank | Label | Count |
|---|---|---|
| 0 | O | 887,889 |
| 1 | B-geo | 37,644 |
| 2 | B-tim | 20,333 |
| 3 | B-org | 20,143 |
| 4 | I-per | 17,251 |
| 5 | B-per | 16,990 |
| 6 | I-org | 16,783 |

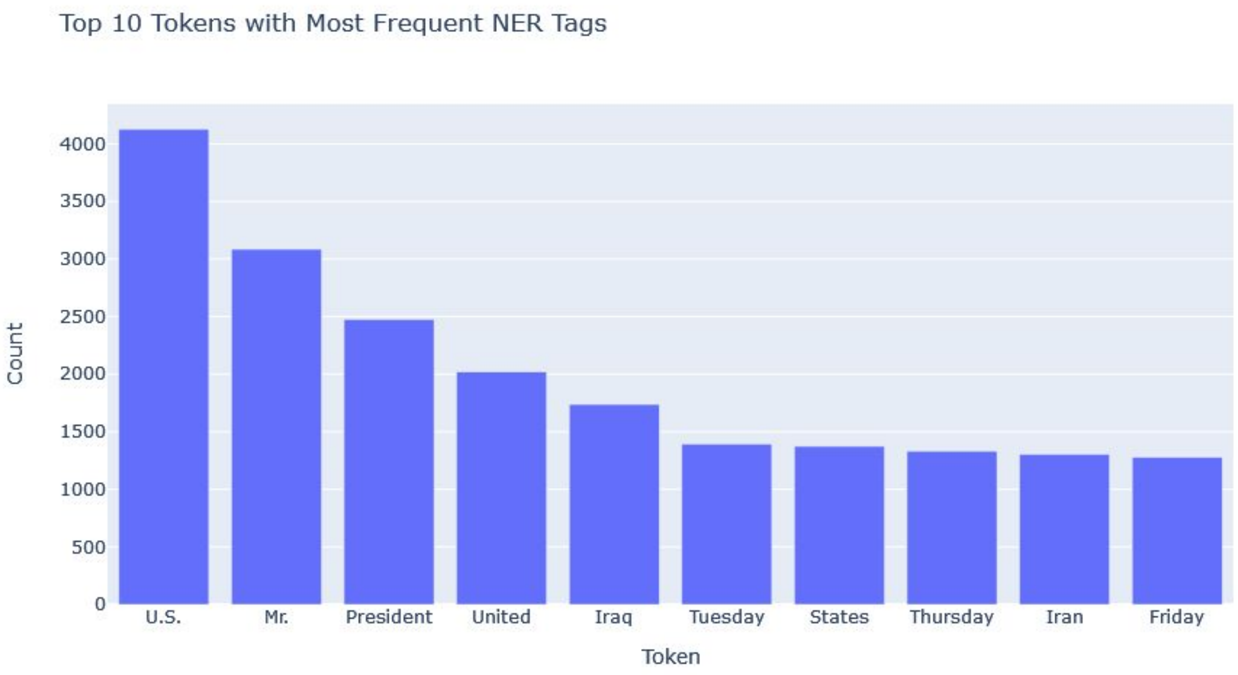
# Exploratory analysis

Number of tokens(words) per sentence - Avg words are between (21-35)



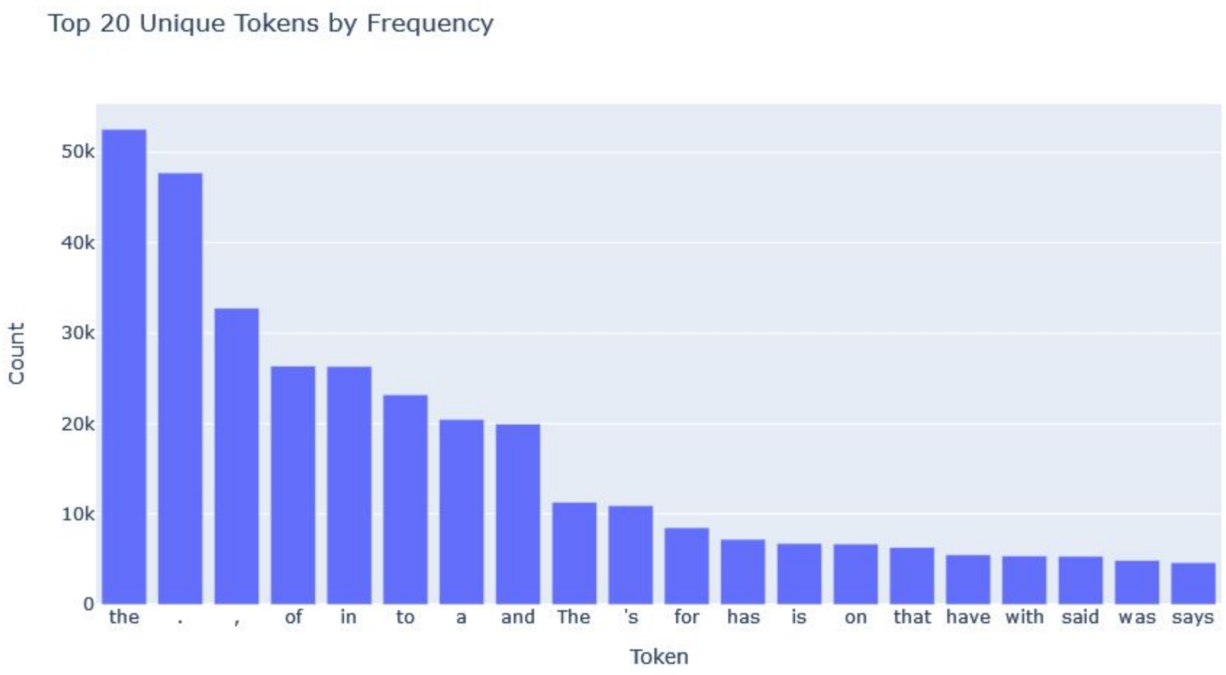
# Exploratory analysis

Top 10 tokens with frequent tags



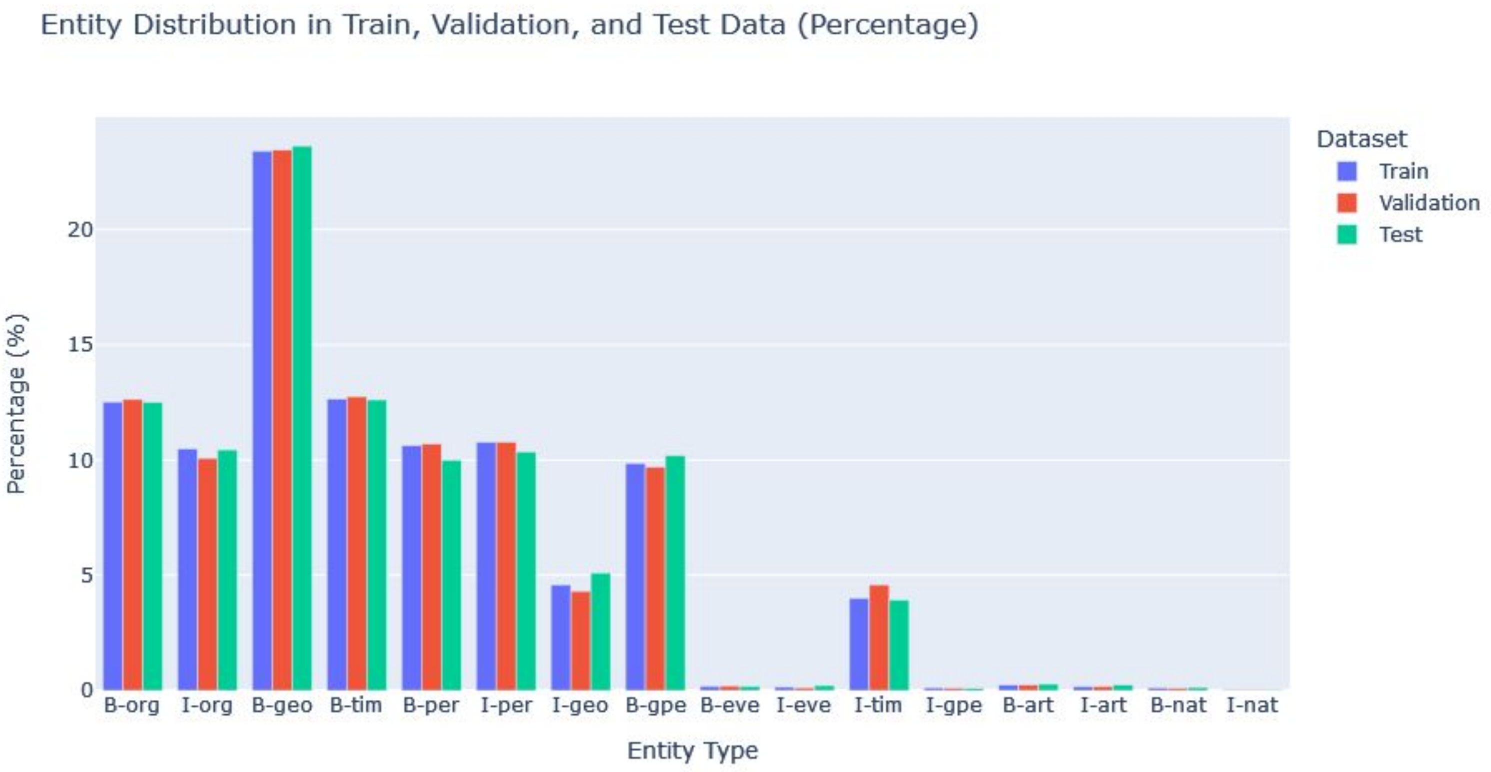
# Exploratory analysis

Top 20 tokens by frequency



# Exploratory analysis

Train/val/test distribution



# Model train & inference logs

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2400 | 1407.45 | 8595.59 | 83.94 | 83.49 | 84.40 | 0.84 |
| 1 | 2600 | 1321.73 | 8351.43 | 84.04 | 85.13 | 82.98 | 0.84 |
| 1 | 2800 | 1308.59 | 8498.99 | 84.59 | 85.76 | 83.46 | 0.85 |
| 1 | 3000 | 1324.84 | 8109.55 | 84.45 | 85.29 | 83.63 | 0.84 |
| 2 | 3200 | 1355.47 | 8036.71 | 84.13 | 83.96 | 84.30 | 0.84 |
| 2 | 3400 | 2613.49 | 7377.70 | 84.43 | 84.75 | 84.11 | 0.84 |
| 2 | 3600 | 1395.91 | 7465.91 | 84.84 | 85.50 | 84.19 | 0.85 |
| 2 | 3800 | 1418.90 | 7483.58 | 84.40 | 85.24 | 83.58 | 0.84 |
| 2 | 4000 | 1433.12 | 7604.89 | 85.12 | 85.58 | 84.67 | 0.85 |
| 3 | 4200 | 1487.28 | 6635.40 | 84.53 | 85.34 | 83.73 | 0.85 |
| 3 | 4400 | 1549.75 | 6915.25 | 84.57 | 85.06 | 84.10 | 0.85 |
| 3 | 4600 | 1534.06 | 7054.32 | 84.83 | 85.44 | 84.24 | 0.85 |
| 3 | 4800 | 1530.11 | 7140.91 | 84.58 | 84.89 | 84.27 | 0.85 |
| 4 | 5000 | 1546.30 | 6307.02 | 84.77 | 84.67 | 84.86 | 0.85 |
| 4 | 5200 | 1619.10 | 6182.98 | 84.81 | 85.45 | 84.18 | 0.85 |
| 4 | 5400 | 1685.61 | 6348.10 | 84.52 | 84.82 | 84.21 | 0.85 |
| 4 | 5600 | 1720.74 | 6530.99 | 84.94 | 84.96 | 84.92 | 0.85 |

```
✓ Saved pipeline to output directory
/content/drive/MyDrive/sapient/model-last
```

# Model train & inference logs

```
================================ NER (per type) ================================

               P        R        F
B-geo      86.78    90.11    88.41
B-gpe      96.24    93.62    94.92
B-org      78.98    75.47    77.18
I-geo      80.76    81.95    81.35
B-per      83.19    83.71    83.45
I-per      83.83    90.98    87.26
I-org      84.22    75.47    79.60
B-tim      92.22    90.19    91.19
I-tim      82.57    75.40    78.82
I-art       0.00     0.00     0.00
B-art     100.00     2.22     4.35
I-gpe      84.62    73.33    78.57
B-eve      36.36    13.79    20.00
I-eve      61.54    22.86    33.33
I-nat       0.00     0.00     0.00
B-nat      58.82    45.45    51.28
```

# Observation & Findings

**Overall Performance**:

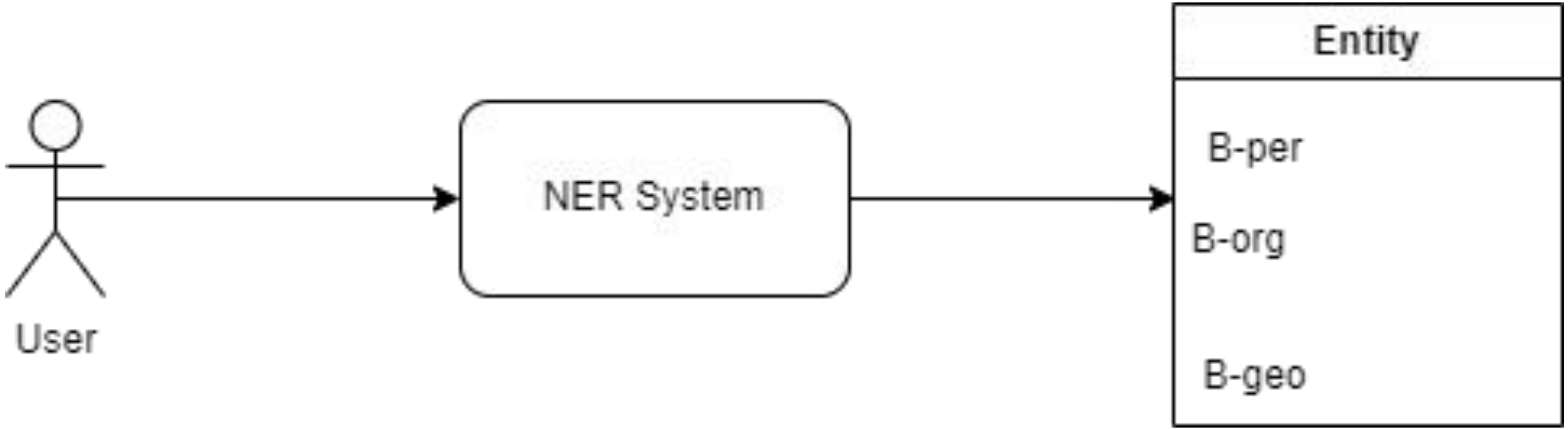
- The model shows good performance on common entity types such as `B-geo`, `B-gpe`, `B-org`, `B-per`, `I-per`, `B-tim`, and `I-tim`, with F1-scores above 75.
- The highest performance is observed in `B-gpe` (F1-score 94.92) and `B-geo` (F1-score 88.41).

# Observation & Findings

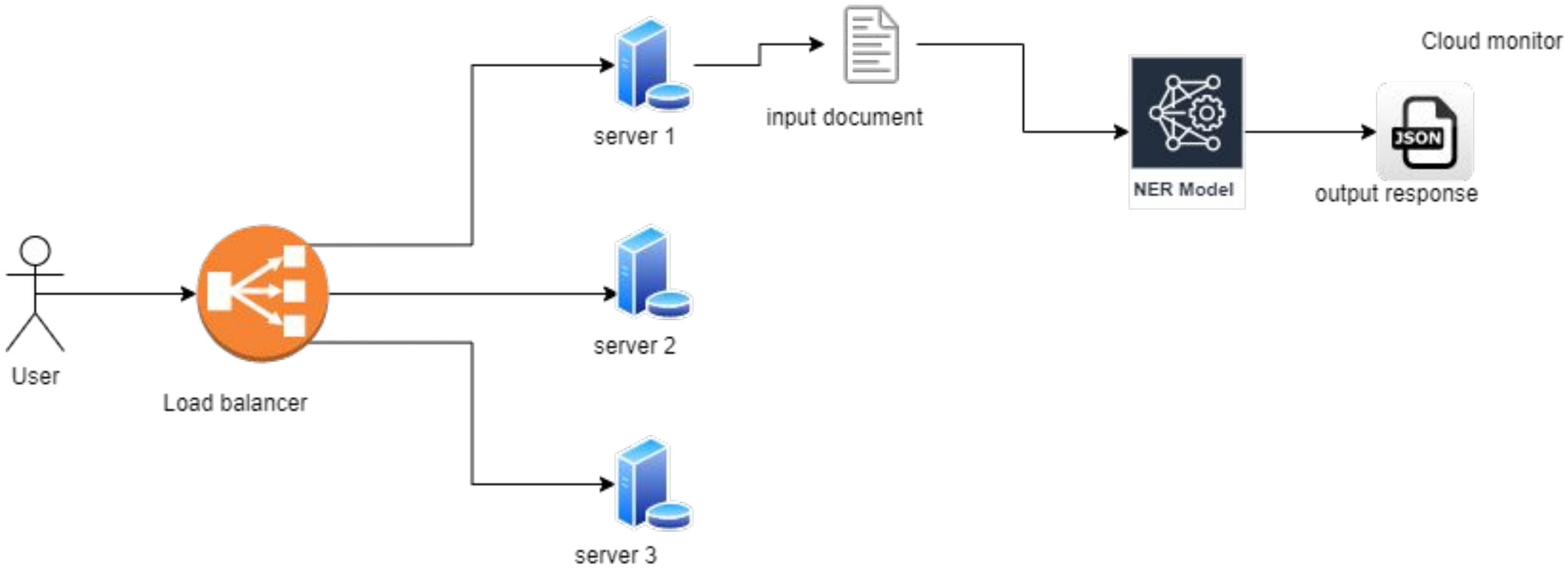
`I-art`, `B-art`, `B-eve`, `I-eve`, `I-nat`, and `B-nat` have very low F1-scores.

For `I-art` and `I-nat`, the model failed to predict any correct entities (`P=0.00`, `R=0.00`).

# System architecture - high level



# System architecture



# How do you perform canary build?

**Deploy the new NER model to a subset of servers**.

**Monitor the performance** of the new NER model.

**Gradually increase the traffic** to the new model if no issues are found.

**Rollback** if any issues are detected.

# Strategy for monitoring

**Strategy:**

1. **Track performance metrics** (e.g.,precision, recall, F1-score) for each NER tag.
2. **Monitor input data** for data drift.
3. **Log predictions** and compare with actual GT.

# CI/CD

**Kubeflow**: For orchestration.

**GitHub Actions**: For CI/CD pipelines.

**Docker:** For containerization

# Other Alternatives

- We can try using open(BERT)/closed source LLM - Trade off is cost Vs performance. Latency in prediction. Context length.

# To explore

- Experiment with ensemble methods or deep learning architectures for improved performance.

Improvements & optimization

1) Hyperparam tuning instead of default params of model.
2) Gather more hard samples to check the performance.

# Conclusion

1) **Hyperparam tuning to get the best model.**
2) **Inference result - ensemble of different models .**
3) **Other strategy to train - Instead of ML models, can try with BERT or decoder only model by attaching classification head and fine-tuning them(compute,time cost increase, decrease in explainability)**

Any other questions/suggestions - Feel free to reach out on vivek.mail2022@gmail.com