Case Study NER

-

ABHINAV GUPTA

Contents:

1. About NER

- **Named Entity Recognition (NER)** is a technique in **natural language processing (NLP)** that focuses on identifying and classifying entities. The purpose of NER is to automatically extract structured information from unstructured text, enabling machines to understand and categorize entities in a meaningful manner for various applications like text summarization, building knowledge graphs, question answering, and knowledge graph construction.

- Example: [Abhinav]$_{\text{PERSON}}$ is working at [Sapient]$_{\text{ORGANIZATION ,}}$ which is in [Noida] $_{\text{LOCATION.}}$

- **Applications:**
- Information Retrieval
- Chatbots & Virtual Assistants
- Sentiment Analysis
- Business Intelligence

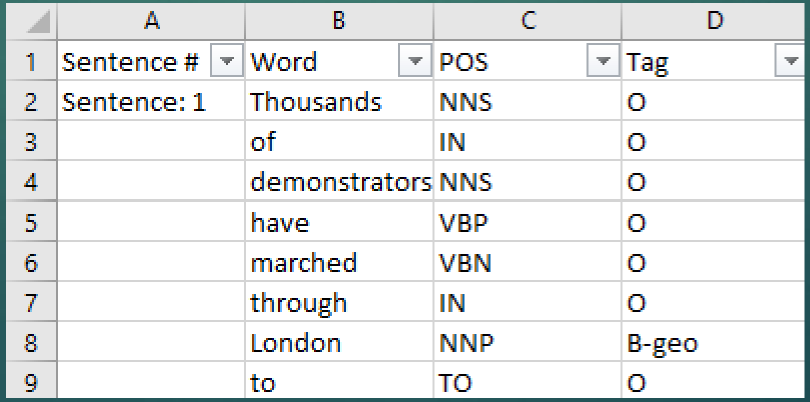
1. About NER cont.

- NER is a sequence modeling problem at it's core. It is more related to classification class of problems where in we need a labeled dataset to train a classifier. It uses IOB technique for labeled classifier described as below:

- IOB is a common tagging format for tagging tokens like:
  - **I- prefix** before a tag indicates that the tag is inside a chunk.
  - **B- prefix** before a tag indicates that the tag is the beginning of a chunk.
  - **O- tag** indicates that a token belongs to no chunk (outside).
- The tags in this dataset are explained as follows:
  - **geo** = Geographical Entity
  - **org** = Organization
  - **per** = Person
  - **gpe** = Geopolitical Entity
  - **tim** = Time indicator
  - **art** = Artifact
  - **eve** = Event
  - **nat** = Natural Phenomenon
- Anything outside these classes is termed as other, denoted as O.

2. NER_Dataset.csv

- Dataset Columns: Sentence #, Word, POS (to be ignored), Tag
- Sentence #: Sentence grouping
- Word: Tokenized word
- POS: Part-of-Speech tag (ignored for this case study)
- Tag: Named Entity tag (B/I/O scheme)
- We have 47959 sentences that contain 35178 unique words.
- These sentences have a total of 42 unique POS tags and 17 unique NER tags in total.
- Sentence – [1,2,3,…4,7959] part of a multiple news articles based on politics, foreign affairs, sports, events etc.



	A	B	C	D
1	Sentence #	Word	POS	Tag
2	Sentence: 1	Thousands	NNS	O
3		of	IN	O
4		demonstrators	NNS	O
5		have	VBP	O
6		marched	VBN	O
7		through	IN	O
8		London	NNP	B-geo
9		to	TO	O

3. Scikit-learn libraries & Dataset Split

- Libraries Used:
  - pandas for data handling
  - sklearn-crfsuite for sequence labeling
  - sklearn for evaluation metrics
  - CRFsuite: A fast implementation of Conditional Random Fields (CRFs), a machine learning algorithm for sequence labeling tasks.
- Dataset Split: Initially , dataset is split with train/test with (80/20)% ratio for Baseline Model and further train/val (initial train data) dataset with (80/20)% ratio for Advanced model (CRF) run.

4. Baseline Model

- Rule-based approach for feature extraction of training dataset and tested it on test dataset. Steps include:
  - Frequency-based Lookup: Build a dictionary mapping each word (from training data) to its most frequent NER tag.
  - Mapping words with their most frequent tags.
  - Define a **‘baseline_predict’ function** that Predicts NER tags for a list of words using the frequency-based baseline. If a word is unseen, it returns 'O' (outside).
  - Evaluate the baseline on the test dataset.
  - Print Classification Report with baseline Accuracy of 69% without outside tag ‘O’

- **Baseline Model Evaluation Metrics**
  - **Accuracy:** Percentage of correct predictions
  - **Precision:** (TP / (TP + FP)) → How many predicted entities are correct?
  - **Recall:** (TP / (TP + FN)) → How many actual entities are correctly predicted?
  - **F1-Score:** Harmonic mean of Precision and Recall
  - **Confusion Matrix:** Identifies misclassifications

5. Baseline Model Evaluation Metrics

	precision	recall	f1-score	support
B-art	0.41	0.13	0.2	82
B-eve	0.54	0.31	0.4	67
B-geo	0.8	0.85	0.82	7674
B-gpe	0.95	0.94	0.94	3151
B-nat	0.56	0.41	0.47	37
B-org	0.73	0.52	0.61	3990
B-per	0.77	0.68	0.72	3397
B-tim	0.88	0.78	0.83	4057
I-art	0	0	0	56
I-eve	0.42	0.16	0.23	51
I-geo	0.74	0.56	0.64	1559
I-gpe	0.56	0.56	0.56	34
I-nat	0	0	0	5
I-org	0.76	0.55	0.64	3265
I-per	0.74	0.67	0.7	3449
I-tim	0.8	0.15	0.25	1310
accuracy			0.69	32184
macro avg	0.57	0.43	0.47	32184
weighted avg	0.79	0.69	0.73	32184

6. Shortcomings of Baseline Model

- Context Ignorance: The baseline only uses the word itself and ignores its surrounding context, which is crucial for disambiguating tags.
- Unable to Handle Ambiguities: Fails on similar words with different meanings
- No Sequential Context Capturing: Does not learn dependencies between words
- Handling Unseen Words: Defaulting to O for unknown words can lead to a high number of false negatives.
- No Sequence Modeling: The model does not take into account dependencies between tags (e.g., a tag following B-PER is more likely to be I-PER).
- Limited to Surface Features: Does not utilize any additional features (such as capitalization, word shape, suffixes/prefixes) that could help in better prediction.
- No Learning of Semantic or Syntactic Features: The model does not learn features such as word embeddings, capitalization patterns, or neighboring words that are important for robust NER.

7. Advanced Model

- NER with Conditional Random Field: Conditional Random Field (CRF) is a sequence labelling technique that can be used to perform NER tagging. CRF is a type of probabilistic model that uses conditional probabilities to predict outcomes (entity types like PERSON, ORG, LOC). Unlike simpler models, CRF takes into account the context of surrounding words when making predictions, allowing it to capture dependencies between labels.
- CRF trains upon sequence of input data to learn transitions from one state (label) to another. To enable such an algorithm, we need to define features which take into account different transitions. In the **function, 'word2features()'**, it transforms each word into a feature dictionary depicting the following attributes or features:
- lower case of word
- prefix containing first 3 characters
- suffix containing last 3 characters
- flags to determine upper-case, title-case, numeric data, etc.
- It also attaches attributes related to previous and next words or tags to determine beginning of sentence (BOS) or end of sentence (EOS)

8. Code Function Logic of Advanced Model

- Let's break down each parameter and argument used in this CRF model:
  - algorithm='lbfgs': This specifies the optimization algorithm used for training the CRF model.L-BFGS stands for Limited-memory Broyden-Fletcher-Goldfarb-Shanno, which is an efficient algorithm for parameter estimation in large-scale machine learning problems. It helps find the optimal parameters for the model by minimizing the objective function.
  - c1=0.1: This is the coefficient for L1 regularization.L1 regularization helps prevent overfitting by penalizing the absolute values of the model parameters, effectively encouraging sparsity (i.e., some parameters become zero, making the model simpler).
  - c2=0.1: This is the coefficient for L2 regularization.L2 regularization helps prevent overfitting by penalizing the squared values of the model parameters. It discourages the model from fitting to noise in the training data by shrinking the parameters towards zero, but not exactly zero.
  - max_iterations=100: This sets the maximum number of iterations for the optimizer to run. The optimizer will perform at most 100 iterations to find the optimal model parameters. If the model converges before reaching this limit, the training process will stop earlier.
  - all_possible_transitions=False: This parameter specifies whether to include all possible transitions between labels in the model. Setting it to ‘False’ means that the CRF will only consider transitions that are present in the training data. This can make the model more efficient and avoid learning invalid transitions.

9. Advanced Model Evaluation Metrics

	precision	recall	f1-score	support
B-art	0.33	0.11	0.16	75
B-eve	0.48	0.29	0.36	49
B-geo	0.86	0.91	0.88	6040
B-gpe	0.96	0.94	0.95	2491
B-nat	0.8	0.5	0.62	32
B-org	0.8	0.72	0.76	3232
B-per	0.84	0.83	0.83	2661
B-tim	0.92	0.87	0.9	3234
I-art	0.15	0.03	0.05	60
I-eve	0.31	0.22	0.26	41
I-geo	0.78	0.79	0.79	1142
I-gpe	0.87	0.68	0.76	40
I-nat	0.8	0.44	0.57	9
I-org	0.81	0.78	0.79	2760
I-per	0.85	0.89	0.87	2732
I-tim	0.85	0.73	0.79	1086
accuracy	0.85		0.97	25684
macro avg	0.71	0.61	0.65	25684
weighted avg	0.85	0.84	0.84	25684

10. Comparison: Baseline Model vs Advanced Model

Model	Precision	Recall	F1-Score	Accuracy
Baseline Model	0.79	0.69	0.73	0.69
Advanced CRF Model	0.85	0.84	0.84	0.84

- Here, Advanced model removes the shortcomings of Baseline model and provides:

- Improved Precision, Recall, and F1-Score
- Better handling of Named Entities
- Increased robustness to unseen data

11. System Design Tasks

- 1. Design System Architecture to Deploy ML Model in Production: To deploy an ML model in production, it needs a robust system architecture that ensures scalability, reliability, and maintainability. Here are the key components:
  - Model Serving: Use a model serving framework like TensorFlow Serving, Flask, or FastAPI to expose your model as a REST API.
  - Load Balancing: Implement load balancers to distribute incoming traffic across multiple instances of your model server.
  - Data Pipeline: Ensure a reliable data pipeline for real-time or batch data ingestion, preprocessing, and feature extraction.
  - Monitoring and Logging: Set up monitoring and logging to track model performance, errors, and usage metrics.
  - Security: Implement security measures like authentication, authorization, and encryption to protect sensitive data and model endpoints.
- 2. How to Perform Canary Build: Canary builds involve rolling out a new version of your application to a small subset of users before deploying it to everyone. Here's how to do it:
  - Deploy the Canary: Deploy the new version to a small percentage of your user base (e.g., 5%).
  - Monitor Performance: Monitor the performance, errors, and user feedback for the canary version.
  - Evaluate Results: If everything goes well, gradually increase the percentage of users on the new version.
  - Rollback if Needed: If issues are detected, roll back to the previous version and fix the problems.

11. System Design Tasks cont.

- 3. Strategy for ML Model Monitoring: ML model monitoring involves continuously tracking the performance and health of the model in production. Key strategies include:
- Performance Metrics: Track metrics like accuracy, precision, recall, and F1-score.
- Data Drift Detection: Monitor for changes in the distribution of input data over time.
- Model Drift Detection: Detect when the model's performance degrades.
- Anomaly Detection: Identify unusual patterns in predictions or input data.
- Alerts and Dashboards: Set up alerts for critical issues and use dashboards for real-time monitoring.

- 4. Load and Stress Testing: Load and stress testing ensure that ML model can handle high traffic and perform under stress. Steps include:
- Define Test Scenarios: Create realistic scenarios that simulate high traffic and stress conditions.
- Use Testing Tools: Tools like Locust, JMeter, or Gatling can simulate user requests and measure performance.
- Analyze Results: Evaluate the results to identify bottlenecks and performance issues.
- Optimize: Make necessary optimizations to improve performance and scalability.

11. System Design Tasks cont.

- 5. Track, Monitor, and Audit ML Training: Tracking and auditing ML training involves keeping a record of training processes, data used, and model versions. Key practices include:
  - Version Control: Use version control systems like Git to track changes in code and data.
  - Experiment Tracking: Use tools like MLflow or TensorBoard to log experiments, parameters, and results.
  - Audit Logs: Maintain detailed logs of training runs, including timestamps, parameters, and outcomes.
  - Compliance: Ensure compliance with data privacy and ethical guidelines.
- 6. Design Framework for Continuous Delivery and Automation: A continuous delivery framework automates the deployment and management of ML models. Key components include:
  - CI/CD Pipelines: Set up CI/CD pipelines using tools like Jenkins, GitLab CI, or GitHub Actions to automate testing and deployment.
  - Automated Testing: Implement automated tests for code quality, performance, and functionality.
  - Infrastructure as Code: Use tools like Terraform or CloudFormation to manage infrastructure configurations.
  - Model Registry: Maintain a model registry to manage different versions of your models.
  - Automated Rollbacks: Implement automated rollback mechanisms in case of deployment failures.

12. Conclusion

- The advanced CRF model significantly improves upon the baseline model by incorporating contextual features, character-based information, and hyper-parameter tuning, leading to a higher F1-score and better entity recognition accuracy.
- While the baseline model struggles with complex and ambiguous entity recognition, the optimized CRF model mitigates these issues through enhanced feature engineering and better generalization to unseen data.
- Deploying the advanced model into an ML pipeline involves steps such as data preprocessing, feature extraction, model training, validation, and real-time inference monitoring.
- The model can be further enhanced by integrating deep learning-based methods like LSTMs, BiLSTM-CRF or Transformers, which can capture long-range dependencies in the text.
- Future work includes fine-tuning the model with domain-specific datasets, handling multilingual entity recognition, and improving computational efficiency for large-scale deployment.

-

Thank You