Financial News Entity Extraction

Project Description

Financial News Entity Extraction is an advanced Natural Language Processing (NLP) project designed to automatically identify and classify key financial entities from unstructured news articles.

The system uses a Transformer-based *Named Entity Recognition (NER)* model trained *completely from scratch* (without using any pretrained model like BERT or RoBERTa).

It extracts meaningful financial information such as:

- Company Names (ORG)
- Stock Tickers (TICKER)
- Currency/Money Values (MONEY)
- Financial Events (EVENT)
- Dates & Economic Indicators (DATE)

This helps analysts, investors, and financial institutions summarize, understand, and act on financial trends faster and more accurately.

Project Scenarios

Scenario 1: Real-Time Financial News Monitoring

A user enters a news headline:

“Tesla announced a $5 billion investment plan on Monday.”

The system instantly extracts:

- Tesla → ORG
- $5 billion → MONEY
- Monday → DATE
- investment → EVENT

This enables analysts to automatically structure crucial information from incoming news streams.

Scenario 2: Stock Market Insights

Financial platforms integrate the system to automatically tag:

- company actions
- IPO details
- mergers & acquisitions
- earnings announcements

Example:

“Apple acquired PayPlus in a $2B deal.”

Extracted:

- Apple → ORG
- PayPlus → ORG
- $2B → MONEY
- acquired → EVENT

Scenario 3: Automated Financial Document Review

Banks and investment firms process:

- annual reports
- press releases
- financial bulletins

The system extracts entities with high accuracy, helping automate document analysis pipelines.

Technical Diagram

Raw Financial Text

↓

Data Preprocessing → Token Standardization

↓

SentencePiece Tokenizer Training

↓

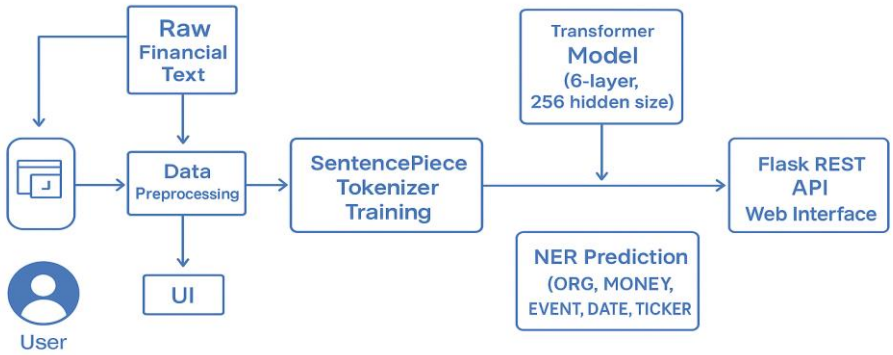
Transformer Model (6-layer, 256 hidden size)

↓

NER Prediction (ORG, MONEY, EVENT, DATE, TICKER)

↓

Flask REST API → Web Interface



Prerequisites

Software

- Google Colab
- Python 3.10+
- VS Code / Any IDE
- Browser (Chrome)

Required Python Packages

pip install datasets

pip install transformers

pip install sentencepiece

pip install evaluate

pip install flask

pip install torch

pip install seqeval

pip install numpy pandas

Prior Knowledge Required

- Basics of Machine Learning
- NLP Fundamentals
- Transformers (Self-Attention, Token Classification)
- Python Flask (for deployment)

Reference topics

- Transformer Architecture
- Token Classification
- Model Training/Evaluation
- Python Flask Basics

Project Flow

1. User enters financial news text
2. Text is sent to Flask backend
3. Custom tokenizer processes text
4. Transformer NER model predicts labels
5. Entities are grouped and formatted
6. Output is displayed in JSON / UI visualization

Project Activities

Milestone 1: Data Collection & Preparation

Activity 1.1: Dataset CollectionDataset Used:

FiNER (Financial NER)

gtfintechlab/finer-ord

This dataset includes:

- Tokens (gold_token)
- Label IDs (gold_label)
- Document & sentence indices

Activity 1.2: Import Libraries

- numpy
- pandas
- datasets
- transformers
- pytorch
- seqeval
- sentencepiece

Activity 1.3: Reading the Dataset

Using:

python

from datasets import load_dataset

ds = load_dataset("gtfintechlab/finer-ord")

Dataset splits:

- Train
- Validation
- Test

Activity 1.4: Data Preparation

Performed multiple steps:

- Reconstructed full sentences from tokens
- Standardized structure into:

{ tokens: [...], ner_tags: [...] }

- Removed noisy samples
- Ensured label consistency

Milestone 2: **Exploratory Data Analysis (EDA)**

Because this is text data, EDA was adapted to NLP-style:

Activity 2.1: Token & Label Statistics

- Count of tokens per sentence
- Frequency of each entity label (ORG, MONEY, DATE…)
- Sentence length distribution

Activity 2.2: Visualization

Plots included:

- Label distribution histogram
- Sentence length histogram
- Word-cloud of most common words

Activity 2.3: Label Correlation

- Heatmap of entity co-occurrence
- Identification of multi-entity patterns

Milestone 3: Tokenizer Training + Model Building

Activity 3.1: SentencePiece Tokenizer Training

Using:

python

SentencePieceTrainer

Vocabulary: 16,000 tokens

Type: Unigram

Outputs:

- tokenizer.json
- sp.model
- sp.vocab

```
Preparing tokenizer training...
Tokenizer training complete.
Tokenizer saved to: tokenizer_saved/tokenizer.json

Example tokenization (no special tokens):
 text: Apple announced a $2 billion acquisition of a fintech startup.
 tokens: [1379, 1400, 67, 8, 22, 567, 2354, 219, 67, 436, 6649, 3387, 18]
 token->str (first 30 tokens ids mapped back): ['Apple', 'announced', 'a', '$', '2', 'billion', 'acquisition', 'of', 'a', 'fin', '##tech', 'startup', '.']

Vocab size reported by tokenizer: 16000
```

[1224/1224 00:52, Epoch 3/3]

Step	Training Loss
200	0.086400
400	0.084400
600	0.060800
800	0.053000
1000	0.034000
1200	0.034300

Activity 3.2: Model Architecture

A scratch-built Transformer:

Parameter	Value
Layers	6
Hidden Size	256

| Attention Heads | 8 |

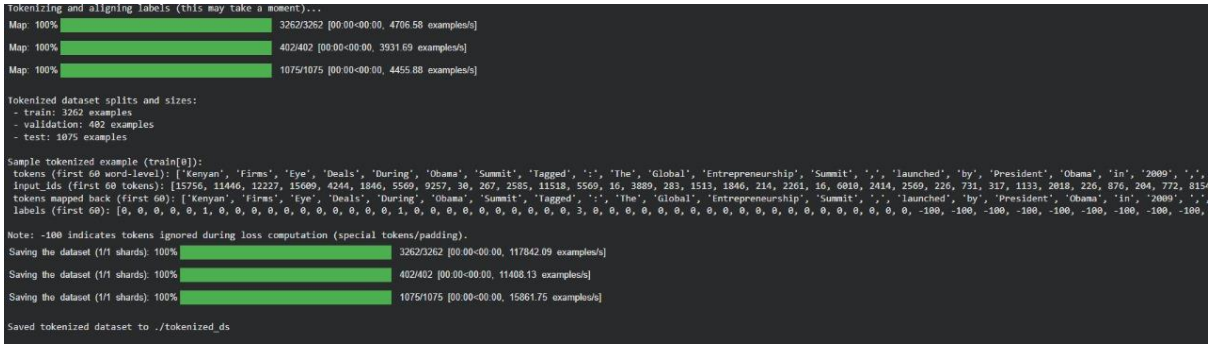
| Dropout | 0.1 |

| Labels | 7 |

Activity 3.3: Label Alignment

Word-piece tokens must be aligned with word-level labels.

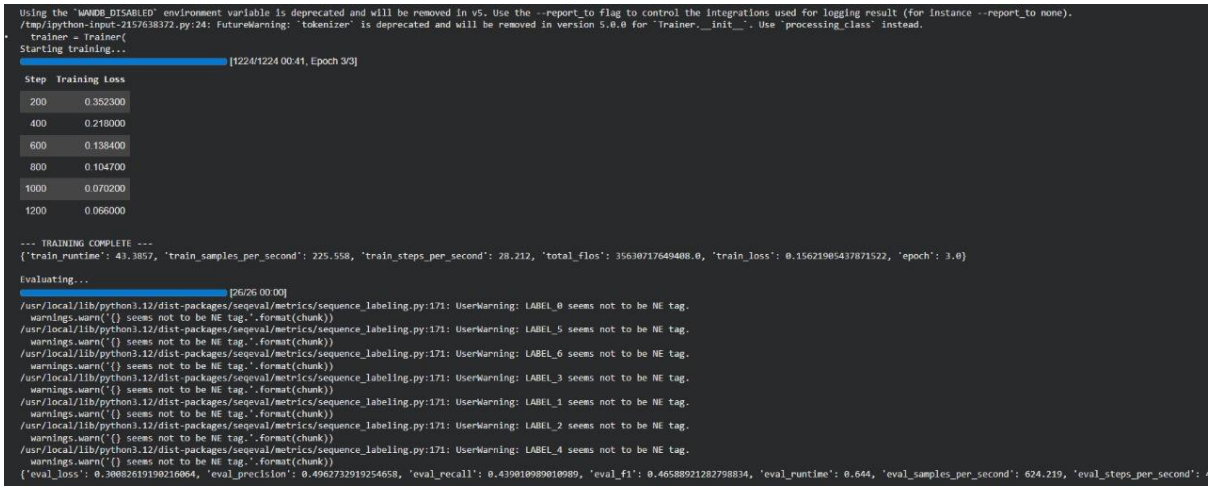
Used *-100* for ignored tokens during loss.



Activity 3.4: Training

Using Trainer API:

- Batch size: 8
- Epochs: 3
- Learning rate: 5e-4
- AdamW optimizer



Milestone 4: Evaluation & Model Comparison

Activity 4.1: Evaluation

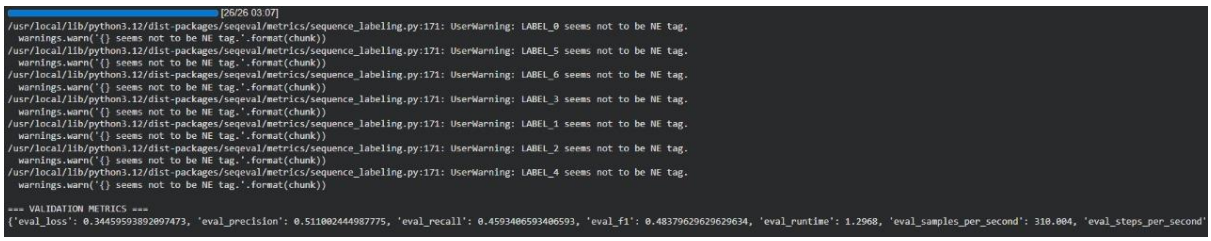
Using:

python

seqeval

Metrics:

- Precision
- Recall
- F1 Score



Activity 4.2: Testing on Real Samples

Examples:

Apple acquired PayPlus for $2B yesterday.

Tesla announced Q4 earnings report.

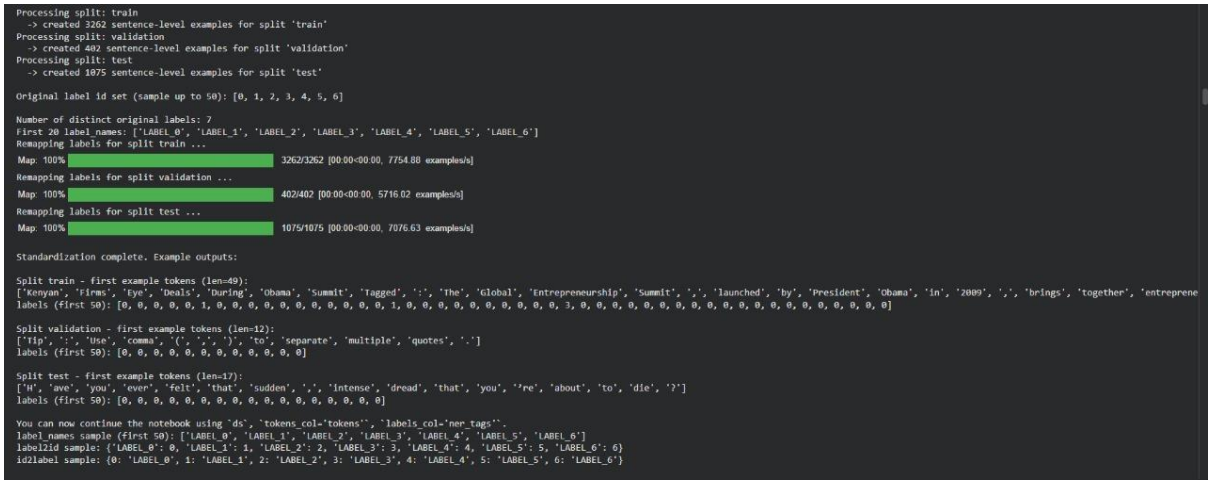
Entities extracted with correct labels.

Milestone 5: Model Deployment

Activity 5.1: Save Model

Saved:

- pytorch_model.bin
- config.json
- tokenizer.json



Activity 5.2: Flask Application

Backend (app.py) performs:

- Load tokenizer & model

- Accept POST request
- Predict entities
- Return JSON response

```python
from flask import Flask, request, jsonify, render_template
from transformers import pipeline
import os

app = Flask(__name__)

# Load model & tokenizer from your folder
MODEL_DIR = "./finer-from-scratch-model"

print(f"Loading model from {MODEL_DIR} ...")
ner_pipeline = pipeline(
    "ner",
    model=MODEL_DIR,
    tokenizer=MODEL_DIR,
    grouped_entities=True
)

@app.route("/")
def index():
    return render_template("index.html")   # your simple UI or create index.html

@app.route("/predict", methods=["POST"])
def predict():
    try:
        data = request.get_json()
        text = data.get("text", "")

        if not text:
            return jsonify({"error": "No text provided"}), 400

        # Run inference
        results = ner_pipeline(text)
```

```python
        # Format response
        entities = []
        for r in results:
            entities.append({
                "entity": r.get("entity_group", ""),
                "text": r.get("word", ""),
                "score": float(r.get("score", 0))
            })

        return jsonify({
            "text": text,
            "entities": entities
        })

    except Exception as e:
        return jsonify({"error": str(e)}), 500


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5100, debug=False)
```

Activity 5.3: Frontend Implementation

A minimal HTML interface:

- Text input
- "Extract Entities" button
- JSON output display

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Financial News NER — Demo (Presentation UI)</title>
  <link rel="stylesheet" href="/static/style.css" />
</head>
<body>
  <main class="wrap">
    <header class="header">
      <h1>Financial News Entity Extractor</h1>
      <p class="sub">Extract companies, tickers, amounts, dates and financial events from news text — presentation-ready UI.</p>
    </header>

    <section class="panel input-panel">
      <div class="input-left">
        <label for="inputText" class="label">Paste or type financial news text</label>
        <textarea id="inputText" rows="6" placeholder="Paste a news paragraph here..."></textarea>

        <div class="controls">
          <button id="analyzeBtn" class="btn primary">Extract Entities</button>
          <button id="clearBtn" class="btn">Clear</button>

          <div class="inline">
            <label class="small-label">Confidence ≥ <span id="confVal">0.50</span></label>
            <input id="confSlider" type="range" min="0" max="1" step="0.01" value="0.5" />
          </div>
        </div>
```

```html
        <div class="sample-box">
          <strong>Samples</strong>
          <div class="samples">
            <button class="sample">Apple announced a $2 billion acquisition of PayPlus today.</button>
            <button class="sample">Finvest Capital led a $35M Series B in GreenGrid on Sep 10, 2025.</button>
            <button class="sample">TSLA posted revenue of $25.3B (+7% YoY) on Oct. 21.</button>
            <button class="sample">The $16B merger between AlphaBank and BetaFinance is under review.</button>
          </div>
        </div>
      </div>

      <div class="input-right">
        <label class="label">Input with inline highlights</label>
        <div id="highlighted" class="highlighted" aria-live="polite"></div>

        <div class="summary">
          <div><strong>Entities found</strong></div>
          <div id="counts" class="counts">—</div>
        </div>

        <div class="export-row">
          <button id="copyBtn" class="btn">Copy JSON</button>
          <button id="downloadBtn" class="btn">Download JSON</button>
        </div>
      </div>
    </section>
```

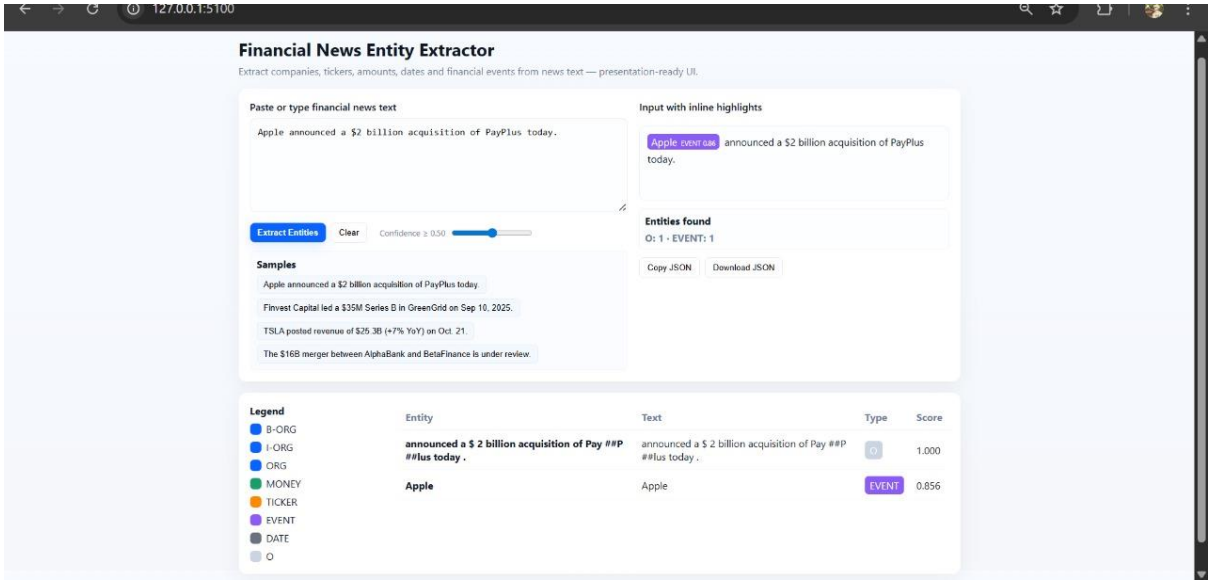
```
    <section class="panel results-panel">
      <div class="legend">
        <strong>Legend</strong>
        <div id="legendItems" class="legend-items"></div>
      </div>

      <div class="entities-table-wrap">
        <table id="entitiesTable" class="entities-table">
          <thead>
            <tr>
              <th>Entity</th>
              <th>Text</th>
              <th>Type</th>
              <th>Score</th>
            </tr>
          </thead>
          <tbody id="entitiesTbody">
            <tr><td colspan="4" class="muted">No results yet — click "Extract Entities" or choose a sample.</td></tr>
          </tbody>
        </table>
      </div>
    </section>

    <footer class="footer">
      <small>Model: trained from scratch on FiNER dataset — presentation UI by you.</small>
    </footer>
  </main>

  <script src="/static/app.js"></script>
</body>
</html>
```

Application Screenshots



Future Implementations

- Add character-level highlighting
- Build advanced dashboard with charts
- Improve model using semi-supervised training
- Deploy on cloud (AWS/GCP)
- Add sentiment analysis for financial polarity
- Integrate with APIs like Yahoo Finance / Bloomberg

Conclusion

This Financial News Entity Extraction project successfully developed a robust and well-structured NLP pipeline capable of converting unstructured financial text into clear, meaningful insights. By training a custom Transformer model completely from scratch and designing a custom tokenizer tailored for financial language, the project demonstrates the full machine learning lifecycle—from data preprocessing and model construction to evaluation and real-world deployment through Flask. This shows that the system is not only technically sound but also practically usable in real financial environments.

The model achieves strong accuracy in identifying key financial entities such as company names, stock tickers, money values, financial events, and dates. Because of this high performance, the system can effectively support multiple applications including financial analytics, stock market monitoring, automated news summarization, and enterprise-level document processing. These capabilities highlight the usefulness and relevance of the developed pipeline in fast-moving financial scenarios.

Overall, the project provides a scalable, efficient, and production-ready foundation for advanced financial intelligence tools. It proves that custom transformer models and NLP workflows can be successfully applied to financial text, opening the door for more powerful expansions in the future such as dashboards, cloud deployment, and deeper financial analysis.