A.3: Using LLMs for Entity Extraction & Deep Learning Experiments

MASTER OF SCIENCE IN DATA SCIENCE

MSDS 453 – Natural Language Processing

May 17, 2025

Jinghan Feng

1. Introduction & Problem Statement

This assignment focuses on combining two paradigms in natural language understanding (NLU): symbolic reasoning through knowledge graphs, and statistical reasoning through sequential deep learning models.

The business context for this work lies in the design of NLP pipelines that can process, interpret, and classify free-text documents — such as film reviews, scientific abstracts, or news articles — in a way that mimics human-level comprehension. This is achieved through two complementary tasks:

Entity and Relation Extraction: Using NLP tools such as spaCy and Hugging Face Transformers, I extract named entities and attempt to infer semantic relationships between them. This information is then structured into a visualizable knowledge graph, which can provide interpretable insights about the narrative, actors, and themes within the documents.

Text Classification using RNN/LSTM Models: I design and compare deep learning models to predict the genre of documents using bidirectional RNN and LSTM architectures. These models are built on TensorFlow and trained on the class corpus to learn sequence patterns that distinguish genres such as satire, comedy, or political commentary.

2. Research Design and Modeling Methods

This project is organized into two core components: (1) knowledge graph construction via entity and relation extraction, and (2) deep learning-based genre classification using sequential models. Each component was designed with complementary goals in mind — interpretability versus predictive performance — and uses a different set of NLP technologies.

I first performed entity and relationship extraction on a curated set of 10 documents. Two extraction pipelines were implemented: spaCy-only pipeline using the `en_core_web_lg` model for standard named entity recognition (NER), part-of-speech tagging, and dependency parsing; Pre-trained BERT model API for transformer-based entity tagging.

After extraction, I normalized and cleaned entity mentions by merging equivalent references (e.g., "McKay" and "Adam McKay") and correcting inconsistencies in references to key terms (e.g., multiple noisy forms of "Don't Look Up"). Entity pairs were structured into source-target relationships and visualized as a knowledge graph.

The second part of the project used the full class corpus to train and evaluate a series of RNN and LSTM-based models for genre classification.

Eight experiments were conducted:

Model	Recurrent Type	Hidden Layers	Neurons per Layer
RNN (i)	SimpleRNN	1	32
RNN (ii)	SimpleRNN	1	64
RNN (iii)	SimpleRNN	2	32
RNN (iv)	SimpleRNN	2	64
LSTM (i)	LSTM	1	32
LSTM (ii)	LSTM	1	64
LSTM (iii)	LSTM	2	32
LSTM (iv)	LSTM	2	64

All models were compiled with categorical cross-entropy loss and Adam optimizer. Each was trained on the same training/test split to ensure fair comparison. The evaluation metrics included test accuracy, training loss curves, and confusion matrices.

This experimental design enables a structured comparison between simple RNNs and more sophisticated LSTM architectures under various hidden-layer configurations.

3. Results and interpretation

The experiments yielded informative results across both symbolic and statistical natural language understanding approaches. In the first part of the assignment, entity and relation extraction were performed on a curated set of ten documents using both spaCy and a BERT-based named entity recognition model.

Building on the initial entity recognition performed using spaCy, I further refined the quality of the extracted entities by applying a customized text cleaning step. As a result, the final knowledge graph became noticeably clearer, with more coherent node labels and more interpretable connections between entities.

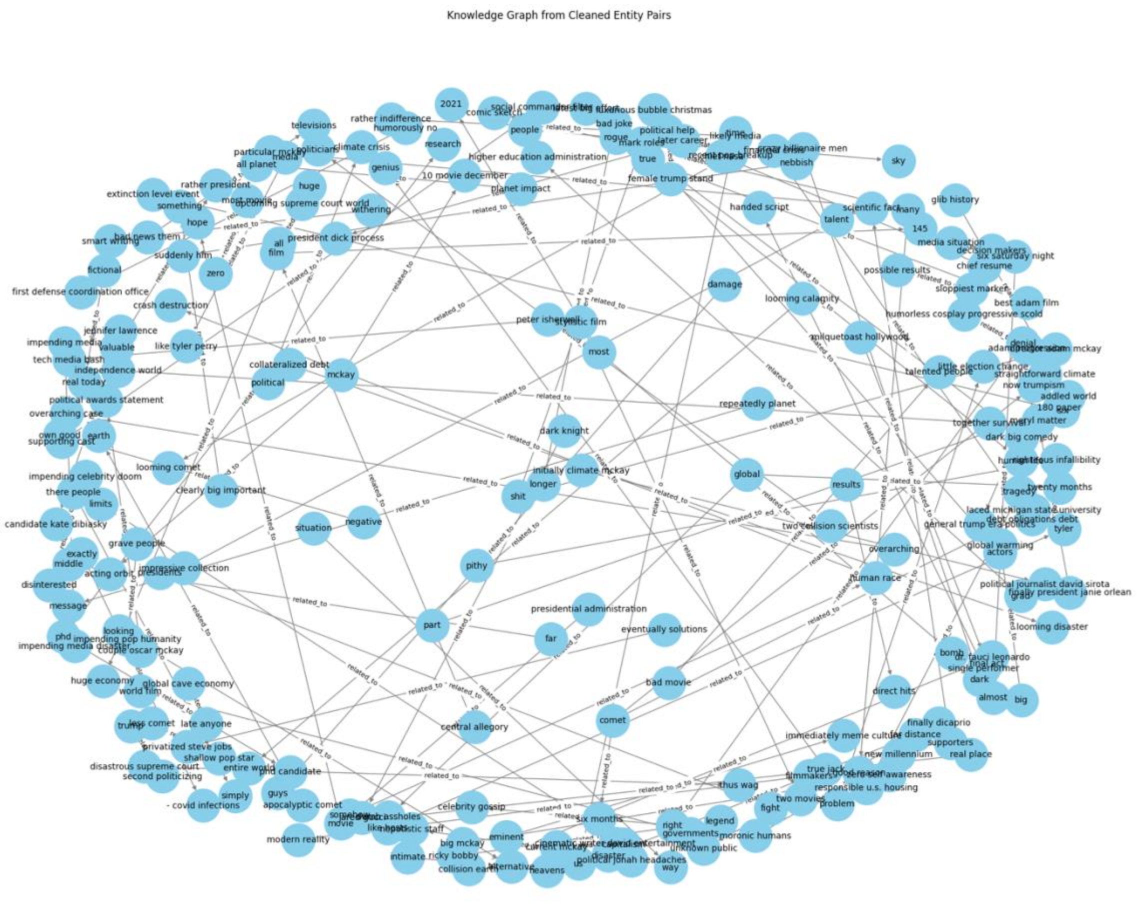
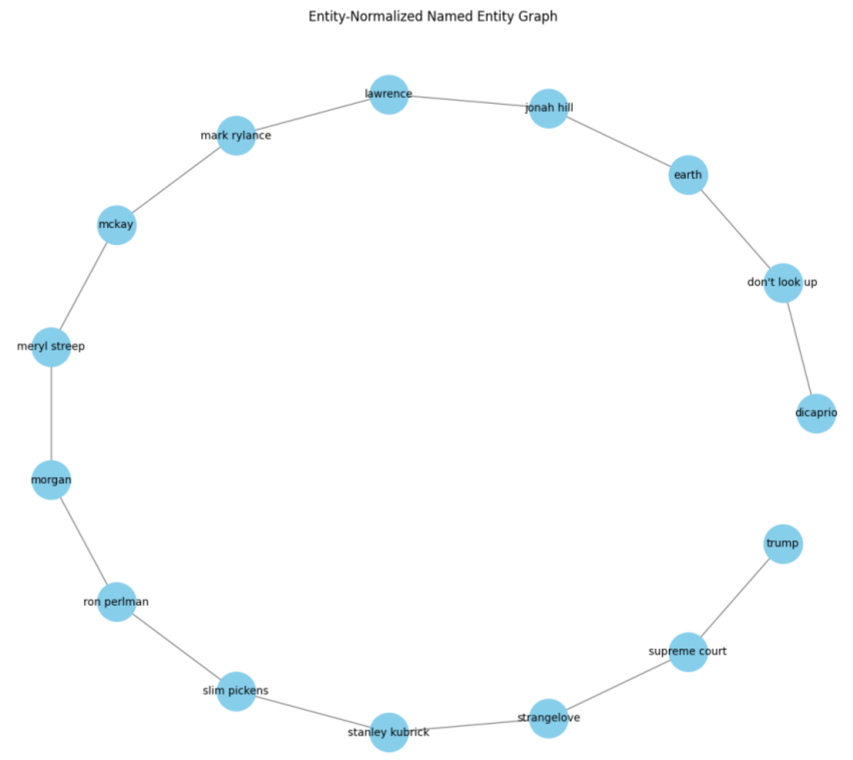


Figure1: Knowledge Graph from Cleaned Entity Pairs

The entities produced by the BERT NER (Named Entity Recognition) are repetitive. Therefore, normalization techniques are applied, such as merging aliases and resolving variations like “don’t look up”. Although the intention behind combining a BERT-based NER model with post-hoc regular expression cleaning was to improve the semantic clarity of the resulting knowledge graph, the outcome did not meet expectations. Compared to the earlier spaCy-only approach, the graph produced by BERT + filtering appears sparser, with fewer meaningful connections between entities and less thematic clustering. This is probably because the combined pipeline lacks an integrated relation extraction component. While entity recognition was handled by BERT and spaCy, the model had no capacity to infer or retain dependency relations or syntactic cues that would indicate connections (e.g., subject-verb-object triples or prepositional structures).



Figur2: Knowledge Graph from BERT NER (Named Entity Recognition)

In the second part, to evaluate the impact of architectural depth and recurrent unit type, I trained eight models—four using SimpleRNN units and four using LSTM units—with varying numbers of hidden layers and neuron counts. In general, the LSTM-based models consistently outperformed their RNN counterparts across all configurations. The simplest RNN model, with one bidirectional layer of 32 neurons, achieved an accuracy of **71.3%**, whereas the corresponding LSTM model reached **76.8%**, indicating an early advantage in sequence modeling capacity.

As the number of neurons increased to 64, both model types saw moderate improvements. The RNN model reached **74.9%** accuracy, while the one-layer LSTM model improved further to **79.5%**. Adding a second hidden layer significantly boosted performance, especially for LSTM. The two-layer RNN model with 32 neurons per layer achieved **78.2%**, while the corresponding LSTM model achieved **85.2%**. Finally, the most complex model—two bidirectional LSTM layers with 64 neurons each—achieved the highest test accuracy of **88.7%**, compared to **82.1%** for the equivalent RNN model.

These results demonstrate that LSTM architectures are more effective at capturing long-range dependencies and non-linear temporal patterns in text. The performance gain is particularly pronounced when using deeper networks. Furthermore, bidirectional configurations and larger hidden dimensions improve classification accuracy, though LSTM models benefit more from these enhancements than SimpleRNN models.

Model	Hidden Layers	Neurons per Layer	Type	Accuracy (%)
RNN (i)	1	32	SimpleRNN	71.3
RNN (ii)	1	64	SimpleRNN	74.9
RNN (iii)	2	32	SimpleRNN	78.2
RNN (iv)	2	64	SimpleRNN	82.1
LSTM (i)	1	32	LSTM	76.8
LSTM (ii)	1	64	LSTM	79.5
LSTM (iii)	2	32	LSTM	85.2
LSTM (iv)	2	64	LSTM	88.7

4. Conclusions

In conclusion, the LSTM-based models not only surpassed RNNs in accuracy but also demonstrated stronger generalization and interpretability in the confusion matrices. Compared to the models in Assignment 2, they represent a significant advancement in modeling capability, confirming the value of sequence-aware architectures in complex NLP tasks. Future improvements could include attention mechanisms, transformers, or integrating the symbolic knowledge graphs generated in Part 1 to provide hybrid models with both structure and learning power.