Metaphor Detection using Heterogeneous Graph Attention Network

Aashay Phirke
aphirke@iu.edu

Austin Ryan
ar100@iu.edu

Fahad Mehfooz
fmehfooz@iu.edu

Prantar Borah
pborah@iu.edu

Abstract

This paper presents an approach to metaphor detection using relevant machine learning techniques. We begin by outlining the key challenges in identifying metaphoric expressions, including their context-dependence and subtle variation across domains. We then describe the dataset, preprocessing steps, and model architectures used in our experiments, focusing on both traditional feature-based classifiers and more recent neural network models with contextual embeddings. Then we compare model performance and analyze error patterns. Finally, we discuss practical considerations for deploying metaphor detection systems, as well as potential directions for future work.

1 Introduction

Metaphor, the figurative use of language where words or phrases carry meanings beyond their literal interpretation, presents a challenging problem for automatic text understanding. Unlike straightforward literal statements, metaphors often rely on subtle contextual cues, cultural references, and implicit conceptual mappings that are not readily captured by surface-level linguistic features. As a result, effective metaphor detection can significantly improve downstream natural language processing (NLP) tasks, such as sentiment analysis, machine translation, and text summarization, by enabling models to handle more nuanced and creative expressions.

The goal of this project is, given a target word, to determine if the word is used metaphorically or literally in the example text. Firstly, we will examine the dataset and discuss preprocessing to simplify the text and embeddings of the text into numerical vectors. Then, we will discuss the machine learning models used, including Logistic Regression, XGBoost, Bi-LSTM, and Heterogeneous Graph Neural Networks. Lastly, we will present the results and state our conclusions.

2 Dataset

2.1 Data Description

The dataset contains three features: metaphorID, label, and text. MetaphorID is the ID encoding of the metaphor candidate word. The ID's are: 0: road, 1: candle, 2: light, 3: spice, 4: ride, 5: train, and 6: boat. The label feature indicates if the target word is used metaphorically or not, with 1 being a metaphor, and 0 not. Lastly, the text feature is the sample text that the word is contextually used in.

2.2 Exploratory Data Analysis

There are 1870 instances in the dataset, with around 75% of the data classified as being a metaphor. This imbalance extends to the presence of the target words. For example, there are 729 instances of the word "road," but only 14 instances of the word "candle." To account for this, when creating the training and testing sets, we made sure to use stratified sampling to maintain the ratios of each feature across both sets. The data was separated using an 80/20 split, with 1496 instances in the training set and 374 instances in the testing set.

2.3 Pre-Processing

The data needed to go through pre-processing before it was to be worked on any further. We applied a preprocessing pipeline to our text data before embedding, training, and evaluating the metaphor detection models. The goal of these steps was to standardize the input, reduce noise, and provide cleaner inputs that emphasize meaningful linguistic features. First, the data is cleaned to remove whitespace, unnecessary punctuation, and stopwords such as "the," "in," and "as." Then, the data went through tokenization, which breaks down the text into smaller units, such as individual words. This discretizes the data and make it easier for models to work with later. Next, we performed lemmatization, which reduces words to their base words. This step helps unify different word inflections (e.g., “runs,” “ran,” and “running” all become “run”), mitigating data sparsity and ensuring that semantically related forms of a word are treated consistently.

2.4 Embeddings

To capture meaningful representations of words and improve metaphor detection, we experimented with several embedding approaches ranging from traditional statistical methods to advanced contextualized models. Specifically, we evaluated TF-IDF, Word2Vec, GloVe, and BERT embeddings to determine their impact on classifier performance and generalization.

As a baseline, we employed Term Frequency–Inverse Document Frequency (TF-IDF) to represent text as sparse feature vectors. TF-IDF highlights words that are important in a given document relative to the entire corpus, thereby down-weighting frequently occurring but uninformative terms. This method does not capture semantic relationships between words but serves as a straightforward, interpretable representation against which we can compare more sophisticated embeddings.

To incorporate semantic similarity into our feature space, we utilized Word2Vec, a predictive neural net embedding technique that learns dense vector representations based on a word’s context within a large corpus. Unlike TF-IDF, Word2Vec embeddings place semantically similar words close together in the vector space.

We also explored GloVe (Global Vectors for Word Representation), a count-based embedding approach that relies on word co-occurrence statistics. GloVe provides pre-trained embeddings on various large corpora to create vectors based on word co-occurrences. Like Word2Vec, GloVe also produces dense vector spaces by integrating semantic similarity.

Finally, we experimented with BERT-based embeddings to incorporate context-dependent representations. BERT is a transformer-based model that generates embeddings by considering context from left and right of the word, allowing the same word to have different vectors in different contexts. This capability is especially valuable for metaphor detection, where meaning can shift dramatically depending on the context.

3 Experimental Setup

We ran three different embedding techniques on the training data and ran logistic regression and XGBoost to establish a baseline comparison for more advanced models. Later we utilized Bi-LSTM with BERT embeddings to improve predictions. Finally, we employed a Heterogeneous-Graph Neural Network model as our final predictive model.

3.1 Models Used

3.1.1 Logistic Regression

Logistic regression served as our baseline model. Logistic regression uses the sigmoid function and a set threshold to classify test data. This was implemented using sklearn’s built-in LogisticRegression function. We trained this model on the cleaned data that was embedded using the three different embedding techniques: TF-IDF, Word2Vec, and GloVe.

3.1.2 XGBoost

XGBoost served as both a secondary baseline model and as an attempted improvement on logistic regression. XGBoost stands for Extreme Gradient Boosting, and is an ensemble boosting algorithm of weak decision trees that corrects errors by using a loss function, such as gradient descent. This was also implemented using sklearn’s built-in XGBClassifier function. This model was also trained on the cleaned data embedded with TF-IDF, Word2Vec, and GloVe.

3.1.3 Bi-LSTM

The Bi-LSTM served as the first model in the neural networks domain. The architecture used for the task uses forward and backward LSTMs to encode context from both past and future directions for each word in a sequence, then concatenates these representations for downstream tasks like classification.

The justification for adopting Bi-LSTM stems from its proficiency in capturing contextual nuances crucial for discerning metaphorical expressions. The Bi-LSTM architecture comprises layers capturing bidirectional dependencies. Embedding layers transform words into numerical vectors, while subsequent Bi-LSTM layers process these vectors bidirectionally. The final output layer produces predictions on the metaphorical or literal nature of given words.

3.1.4 Heterogeneous-Graph Neural Network (HGNN)

Graph Creation: The function `create_hetero_graph` generates a graph where *word* nodes represent unique words, *doc* nodes represent documents, and *word-to-doc* edges signify the occurrence of words in documents.

Document-Word Relationships:
Words (from `text` and `metaphor`) are linked to documents to capture word importance and metaphor relevance structurally.

Node Embeddings:
BERT embeddings are used to derive textual features for documents. Words and documents are represented as numerical vectors.

GNN Model:
A `HeteroGNN` model is trained using `GATConv` (Graph Attention Network), which weighs node relationships using an attention mechanism. The model propagates information between word and document nodes.

Cross-Validation:
A *K-Fold Cross-Validation* routine evaluates the model with varying hyperparameters, optimizing validation accuracy.

Grid Search:
Hyperparameters such as hidden channels, number of layers, attention heads, and learning rates are optimized.

Table 1: Comparison between Hetero GNN and Bi-LSTM for Metaphor Detection

Aspect	**Hetero GNN**	**Bi-LSTM**
Structure	Models relationships using graph structures (doc-word).	Processes sequential text data linearly.
Contextual Information	Captures both **local (words)** and **global (documents)** relationships using attention mechanisms.	Relies solely on sequential word order for context.
Multi-Modal Relationships	Explicitly integrates multiple node types (e.g., words, documents, and metaphors).	Cannot model heterogeneous relationships explicitly.
Training Efficiency	Efficient with sparse data using attention mechanisms.	Computationally expensive for long sequences.
Representation Power	Naturally handles metaphorical associations across contexts.	Sequential dependencies may miss cross-document patterns.

4 Results

Logistic Regression served as a baseline classifier given its simplicity and interpretability. The model achieved moderate results using the TF-IDF embedding and the GloVe embedding. However, the model achieved less than 50% F1-score with the Word2Vec embedding.

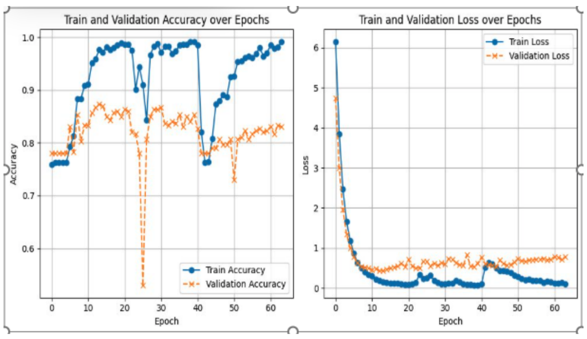

Figure 1: Visualization of Train and Validation Loss (Bi-LSTM)

	Accuracy	Precision	Recall	F1-Score
TF-IDF	0.81	0.83	0.60	0.62
Word2Vec	0.77	0.72	0.52	0.48
GloVe	0.82	0.77	0.66	0.69

Table 2: Comparison of Precision, Recall, and F1-score across the three embedding methods using Logistic Regression

The XGBoost model served as a secondary baseline and as an attempted slight improvement over Logistic Regression. XGBoost achieved higher F1-scores than Logistic Regression with TF-IDF and Word2Vec embeddings, but slightly lower but similar results using GloVe embeddings.

	Accuracy	Precision	Recall	F1-Score
TF-IDF	0.86	0.81	0.77	0.79
Word2Vec	0.76	0.65	0.59	0.60
GloVe	0.80	0.72	0.66	0.68

Table 3: Comparison of Precision, Recall, and F1-score across the three embedding methods using XGBoost

Despite the initial attempts, we felt large improvements could be made, which prompted a shift towards more sophisticated models. The Bi-LSTM model achieved much better F1-scores for class 0. The overall results are displayed below, and the training and validation loss is displayed in Figure 1:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.71	0.76	0.74	88
1	0.93	0.91	0.92	286
Accuracy		0.87		374
Macro Avg	0.82	0.83	0.83	374
Weighted Avg	0.88	0.87	0.87	374

Table 4: Classification Report Results (Bi-LSTM)

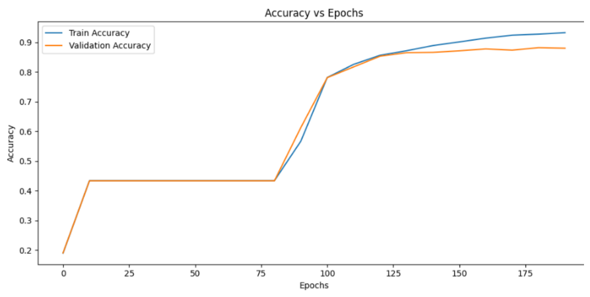

Figure 2: Visualization of accuracy vs epochs when training HGNN

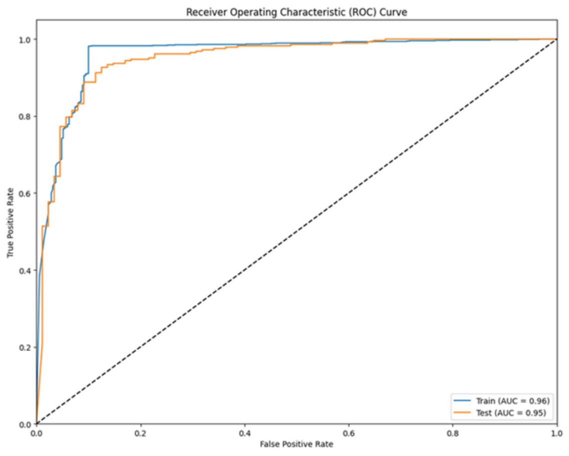

Figure 3: ROC curve of HGNN predictions

Class	Precision	Recall	F1-Score	Support
0	0.87	0.62	0.73	88
1	0.89	0.97	0.93	286
Accuracy		0.89		374
Macro Avg	0.88	0.80	0.83	374
Weighted Avg	0.89	0.89	0.88	374

Table 5: Classification Report (Test)

5 Conclusion

In this report, we implemented multiple approaches for metaphor detection, starting with statistical methods like **Logistic Regression** and **XGBoost**. While these models provided a strong baseline, their performance was limited. Logistic Regression, using traditional embeddings like **TF-IDF**, achieved moderate results due to its inability to capture complex semantic relationships. Similarly, XGBoost offered slight improvements but struggled with data sparsity and the subtle nature of metaphorical expressions.

To address these shortcomings, we employed a **Bi-LSTM** model. Leveraging its ability to process sequential data in both forward and backward directions, the Bi-LSTM captured contextual nuances more effectively. Combined with **BERT embeddings**, the Bi-LSTM model significantly improved performance, achieving higher *precision, recall*, and *F1-scores*, particularly for classifying metaphorical words.

Finally, we explored a **Heterogeneous Graph Neural Network (H-GNN)**, which incorporated both *word-level* and *document-level relationships* using graph structures. By leveraging **Graph Attention Networks (GATConv)** and BERT-based embeddings, the H-GNN model effectively captured *semantic similarities* and *multi-modal relationships* within the dataset. The H-GNN outperformed previous models, achieving the highest accuracy and demonstrating its capability to model both *local* and *global* relationships between nodes.

In conclusion, while traditional statistical models provided interpretable yet limited results, neural network-based approaches, particularly the Bi-LSTM and H-GNN, were able to capture the deeper contextual and semantic relationships required for accurate metaphor detection. These results highlight the potential of advanced deep learning and graph-based methods for tasks involving subtle and context-dependent language.

6 Team Members

- Aashay Phirke: Worked on model preprocessing and logistic regression
- Austin Ryan: Worked on model preprocessing, embedding, and XGBoost
- Fahad Mehfooz: Worked on HGNN model
- Prantar Borah: Worked on Bi-LSTM model

7 References

[1] Hyeju Jang, Keith Maki, Eduard Hovy, and Carolyn Penstein Rosé. 2017. *Finding Structure in Figurative Language: Metaphor Detection with Topic-based Frames*. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*,.

[2] Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Miller, Seungwhan Moon, and Carolyn Penstein Rosé. 2016. *Metaphor Detection with Topic Transition, Emotion and Cognition in Context*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*,.

[3] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process., 45:2673–2681*

References

[1] Lemmatization
https://en.wikipedia.org/wiki/Lemmatization

[2] Lemmatization https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/

[3] Preprocessing
https://www.geeksforgeeks.org/introduction-to-nltk-tokenization-stemming-lemmatization-pos-tagging

[4] Stop Words https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

[5] Word Embeddings https://www.geeksforgeeks.org/word-embeddings-in-nlp/

[6] TF-IDF https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

[7] Word2Vec https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/

[8] GloVe https://www.geeksforgeeks.org/pre-trained-word-embedding-using-glove-in-nlp-models/

[9] GloVe https://nlp.stanford.edu/projects/glove/

[10] BERT https://www.geeksforgeeks.org/how-to-generate-word-embedding-using-bert/

[11] Logistic Regression https://www.geeksforgeeks.org/understanding-logistic-regression/

[12] XGBoost https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/

[13] XGBoost
https://www.geeksforgeeks.org/ml-gradient-boosting/

[14] Bi-LSTM https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

[15] Bi-LSTM
https://www.geeksforgeeks.org/bidirectional-lstm-in-nlp/

[16] HGNN https://pengcui.thumedialab.com/papers/HeterogeneousGAN.pdf

[17] HGNN https://aclanthology.org/D19-1488.pdf

[18] HGNN https://pytorch-geometric.readthedocs.io/en/latest/notes/heterogeneous.html

[19] HGNN https://graph-neural-networks.github.io/static/file/chapter16.pdf

[20] HGNN https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.HeteroConv.html?highlight=HeteroConv

[21] HGNN https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GATConv.html