Metaphor Detection in Text Analysis




Ayush Manojkumar Lodha amlodha@iu.edu
Sameer Hussain samehuss@iu.edu
Aditya Gaitonde adigaito@iu.edu
Ravi Teja Seera raseer@iu.edu

Abstract

Metaphors are a fascinating aspect of language, offering a way to understand and describe complex ideas, emotions, and relationships in a more relatable and vivid manner. Metaphors add richness to language but detecting them using computers is challenging due to their varied meanings and reliance on context. This research undertakes the task of metaphor detection through machine learning, utilizing a dataset comprising 1,870 training and 800 test examples. Our methodology integrates comprehensive data preprocessing, the employment of ensemble learning techniques, and extensive hyperparameter tuning to refine our models. The endeavor goes beyond mere text analysis, offering insights into the computational interpretation of figurative language. Although this is an academic class project for the course, it serves as a valuable learning example for understanding further applications such as sentiment analysis, enabling more empathetic and accurate interpretations of text across various applications.

1 Introduction

The interpretation of metaphors is a complex yet essential task in understanding human language, as it often conveys deeper meanings than the literal words suggest. Metaphors serve as a bridge to express emotions, and concepts that might be otherwise difficult to articulate. However, for computational systems, this poses a significant challenge. Consider the sentence from our dataset:
"I don't know how to live without fear right now. I have just finished chemo waiting to have a mastectomy. There have been so many waits and turns on this journey. It started out as a small blimp in the ***road*** *according to the Surgeon, well then it became a big bump."*
Here, the word **"road"** is not merely a physical pathway but a metaphor for the patient's challenging health journey. This project is focused on creating a machine-learning model capable of detecting these metaphorical uses of language. With a dataset highlighting such phrases, our model is trained to differentiate between metaphorical and literal usage.

2 Dataset Description

The dataset comprises 1,870 instances, featuring two independent variables and one dependent variable. You can see the first 5 samples in Fig. 1 The first independent variable, 'metaphorID', is categorical, mapping to various metaphor types via integers, with seven distinct categories such as 'road', 'candle', 'light', etc. The second independent variable, 'text', contains textual data in the form of sentences or paragraphs. The dependent variable, "label_boolean", is a binary indicator denoting the presence of a metaphor within the text. The dataset is imbalanced, with 1,432 instances labeled 'True' and 438 labeled 'False' for the presence of a metaphor. Class imbalance can be visually seen in Fig. 2a and Fig. 2b and for detailed number and percentage-wise overview can be seen in the Table 1

```
df.head()
✓ 0.0s
```

	metaphorID	label_boolean	text
0	0	True	Hey , Karen !!!! I was told that on the day of...
1	2	False	Hi Ladies ... my last chemo was Feb 17/09 , ra...
2	2	False	I have just come form my consult with a lovely...
3	4	False	I also still question taking Tamox for stage 1...
4	2	False	Just checking in to say hello ladies . I had a...

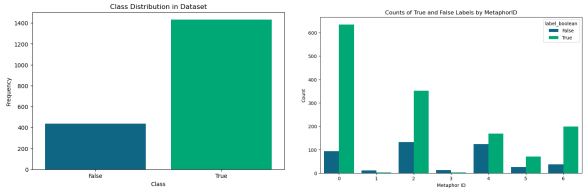
Figure 1: First 5 samples in Dataframe.

3 Data Preprocessing

Before proceeding with the modelling strategies, we complete the data preprocessing stage in which tokenization, oversampling of the miniority class are the crucial steps.

Table 1: Frequency and Percentage of Metaphorical vs. Non-Metaphorical Usage of Selected Words

Metaphor Name	Metaphor ID	True (Metaphor)		False (Non-Metaphor)	
		N	%	N	%
"road"	0	635	87.11	94	12.89
"candle"	1	3	21.43	11	78.57
"light"	2	352	72.58	133	27.42
"spice"	3	3	18.75	13	81.25
"ride"	4	169	57.68	124	42.32
"train"	5	71	73.20	26	26.80
"boat"	6	199	84.32	37	15.68
all		1432	76.58	438	23.42



(a) Overall (b) Metaphor wise

Figure 2: Dataset samples and class distribution

3.1 Tokenization

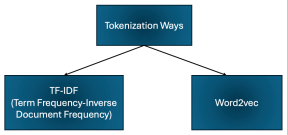
There are two ways of tokenization that we explored, which we can see in Fig. 3a

- TF-IDF (Term Frequency- Inverse Document Frequency) evaluates word relevance in a document, balancing term frequency with its commonness across documents.
- Word2Vec analyzes word meanings through usage in context, learning from text corpus to produce vector representations of words that reflect linguistic relationships.

It's critical to apply TF-IDF to training and validation datasets separately to avoid data leakage and ensure model generalization. While both methods are valuable, TF-IDF may be more beneficial for our specific task in model building.

3.2 Oversampling

To address the class imbalance in our dataset, where the minority class (label 0) is underrepresented, we implement oversampling techniques. This helps balance the class distribution, avoiding model bias towards the majority class and enhancing training robustness and generalization. Such a strategy aligns with machine learning best practices for fair and unbiased model performance. Updated data distribution after oversampling can be in Fig. 3b




```
Original Class Distribution:
 label_boolean
1    1146
0     350
Name: count, dtype: int64
Resampled Class Distribution:
 label_boolean
1    1146
0    1146
Name: count, dtype: int64
Number of Samples Added to Each Class:
 label_boolean
1      0
0    796
Name: count, dtype: int64
```

(a) Tokenization of Words (b) Oversampling Distribution

Figure 3: Preprocessing of Dataset

4 Evaluation Metrics

In classification report, we prioritizes the following metrics:

- **Precision**: Assesses the model's accuracy for both classes, aiming for a low rate of false positives.
- **Recall**: Especially important for the minority class (label 0), it measures the model's ability to detect all non-metaphorical instances, emphasizing the reduction of false negatives.
- **F1-Score**: Provides a balanced view of precision and recall, valuable for our imbalanced class distribution.
- **Balanced Accuracy**: Reflects the average accuracy across both classes, ensuring equal consideration for label 0 and label 1 performance.

5 Modelling

5.1 LSTM

We implemented LSTM, but we did not explore it further because it was out of the scope of the class.

5.2 Bagging

Bagging, or Bootstrap Aggregating, reduces variance and enhances model generalization by training multiple models on bootstrapped data subsets and aggregating their predictions, a technique we've studied in class.

- **Random Forest**: It uses multiple decision trees on varied data subsets, averaging their outputs to enhance accuracy and curb overfitting. Results of Gridsearched Hypertuned parameters of Random Forest can be seen in Fig. 4a. Its high recall for metaphorical instances (label 1) underscores its effectiveness.

- **Extra Trees**: This variant of Random Forest adds more randomness in selecting thresholds, which can quicken training and further diminish variance.Results of Gridsearched Hypertuned parameters of Extratrees can be seen in Fig. 4b. It yielded a balanced performance, evidenced by an improved F1-score for non-metaphorical instances (label 0).

Model: Random Forest

	precision	recall	f1-score	support
0	0.73	0.36	0.48	88
1	0.83	0.96	0.89	286
accuracy			0.82	374
macro avg	0.78	0.66	0.69	374
weighted avg	0.81	0.82	0.79	374

Accuracy: 0.8181818181818182

(a) RandomForest Model

Model: Extra Trees

	precision	recall	f1-score	support
0	0.78	0.45	0.58	88
1	0.85	0.96	0.90	286
accuracy			0.84	374
macro avg	0.82	0.71	0.74	374
weighted avg	0.84	0.84	0.83	374

Accuracy: 0.8422459893048129

(b) Extratrees Model

Figure 4: Results of Bagging Model

5.3 Boosting

Boosting, a sequential ensemble method, to bolster model accuracy. This technique emphasizes correcting misclassifications from previous iterations, with algorithms training models in sequence. Following are the boosting models which we used for our analysis:

- **Gradient Boosting**: Gradient Boosting is a machine learning technique that sequentially builds models by correcting the errors of previous models. Results of Gridsearched Hypertuned parameters of Gradient Boosting can be seen in Fig. 5a. Notable for its high recall for the metaphor class, it's adept at identifying a broad range of metaphorical expressions.

- **XGBoost**: XGBoost is an optimized distributed gradient boosting library designed for speed and performance. Results of Gridsearched Hypertuned parameters of XGBoost can be seen in Fig. 5b. It achieves a balance between precision and recall, evidenced by uniform f1-scores, indicating its overall robustness.

- **AdaBoost**:AdaBoost (Adaptive Boosting) is a boosting algorithm that adjusts the weights of weak learners to form a strong predictive model. Results of Gridsearched Hypertuned parameters of AdaBoost can be seen in Fig. 5c. While precision is marginally lower for non-metaphorical instances, it maintains high recall for metaphorical ones, capturing subtle metaphorical nuances. We applied this technique because we have extensively learned about its pseudocode in class.

Model: Gradient Boosting

	precision	recall	f1-score	support
0	0.74	0.58	0.65	88
1	0.88	0.94	0.91	286
accuracy			0.85	374
macro avg	0.81	0.76	0.78	374
weighted avg	0.85	0.85	0.85	374

Accuracy: 0.8529411764705882

(a) Gradient Boosting Model

Model: XGBoost

	precision	recall	f1-score	support
0	0.66	0.57	0.61	88
1	0.87	0.91	0.89	286
accuracy			0.83	374
macro avg	0.77	0.74	0.75	374
weighted avg	0.82	0.83	0.82	374

Accuracy: 0.8288770053475936

(b) XGBoost Model

Model: AdaBoost

	precision	recall	f1-score	support
0	0.49	0.39	0.43	88
1	0.82	0.87	0.85	286
accuracy			0.76	374
macro avg	0.65	0.63	0.64	374
weighted avg	0.74	0.76	0.75	374

Accuracy: 0.7593582887700535

(c) Adaboost Model

Figure 5: Results of Boosting Models

5.4 Ensemble Modelling - Averaging

As in Fig 6, we've employed an ensemble modeling approach, leveraging both bagging and boosting techniques to mitigate variance and bias, respectively. Bagging models like Random Forest and ExtraTrees operate in parallel, reducing variance by averaging multiple estimators. Boosting models, including Gradient Boosting, AdaBoost, and XGBoost, build sequentially to address bias, enhancing predictive accuracy. Hyperparameter tuning was conducted via GridSearch as discussed previously in results of each of the models, fine-tuning our models to optimize performance. The culmination of our efforts is an averaging ensemble method, synthesizing predictions to bolster accuracy and robustness in metaphor detection. Results of Averaging Ensemble Model can be found 7

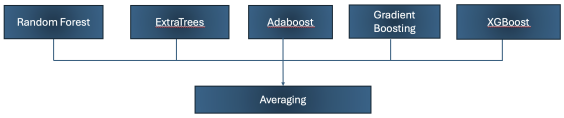


Figure 6: Ensemble Model

Ensemble Model Performance:

	precision	recall	f1-score	support
0	0.74	0.52	0.61	88
1	0.87	0.94	0.90	286
accuracy			0.84	374
macro avg	0.80	0.73	0.76	374
weighted avg	0.84	0.84	0.83	374

Accuracy: 0.8449197860962567

Figure 7: Results of Ensemble Averaging Model - GridSearchCV

6 Hyperparameter Tuning

Hyperparameter tuning is a pivotal aspect of our metaphor detection project, as it significantly enhances model performance.

6.1 Grid Search CV

Utilizing GridSearchCV, we meticulously optimized the hyperparameters for each classifier, ensuring our ensemble approach is fine-tuned for accuracy. For Random Forest, Extra Trees, AdaBoost, Gradient Boosting, and XGBoost, we adjusted parameters such as the number of estimators, learning rate, and tree depth. We have already discussed its results in previous sections and in Fig. 7

6.2 Random Search CV

We have tried to use the RandomSearchCV, to explore further hypertuning the parameters. It gives better performance but utilizes more time and computational resources. Hence results from hypertuned parameters of Gridsearch is submitted. Results of the Averaging Ensemble Model in which each model is hyper-tuned by the Random Search CV can be found in Fig. 8

7 Cross Validation

To ensure the robustness and reliability of our models, we implemented 5-fold cross-validation. This method helped us assess the generalization capability of the models across different subsets of the dataset.

Ensemble Model Performance:

	precision	recall	f1-score	support
0	0.74	0.52	0.61	88
1	0.87	0.94	0.90	286
accuracy			0.84	374
macro avg	0.80	0.73	0.76	374
weighted avg	0.84	0.84	0.83	374

Accuracy: 0.8449197860962567

Figure 8: Results of Ensemble Averaging Model - RandomSearchCV

8 Results and Discussion

The results are derived from a train/validation split of 80/20 on the train.csv file. We conducted hyperparameter tuning via GridSearchCV and RandomSearchCV for models such as Random Forest, Extra Trees, AdaBoost, Gradient Boosting, and XGBoost, each fine-tuned with specific parameters like the number of estimators and learning rates for enhanced accuracy and reduced variance. For example, Random Forest was adjusted for balanced class weights, while Extra Trees parameters included learning rate and estimator count adjustments.

Focusing on the precision and recall for label 0, we implemented an averaging ensemble modeling technique, a strategy validated in the later part of our course. This approach, supported by Cross Validation, has demonstrated efficacy in reducing variance and ensuring the robustness and generalizability of our model.

9 Conclusion

In summary, our project marks a significant stride in metaphor detection, harnessing an ensemble of machine learning models fine-tuned through rigorous hyperparameter optimization to discern metaphorical from literal language. This approach, rooted in a combination of bagging and boosting techniques, has yielded a robust system with a promising balance of precision and recall. The outcomes underscore the potential of computational methods in interpreting complex linguistic constructs and set the stage for future advancements that could further narrow the gap between human linguistic competence and machine understanding.

References

Lecture of Statistical Machine Learning Course
by Prof. Hyeju Jang

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html