# Methaphor Detection Using Neural Networks

**Abstract**

This study utilizes a PyTorch-based neural network, named MetaphorNN, to detect metaphors in text through a machine learning approach. The methodology encompasses data preparation, context extraction, and TF-IDF feature engineering. MetaphorNN employs fully connected layers with ReLU activations for binary classification. The training process involves dataset splitting and iterative training using stochastic gradient descent. Evaluation metrics provide insights into the model's effectiveness, contributing valuable findings to the natural language processing field for applications like sentiment analysis and creative writing support.

## 1 Introduction

This project explored various machine learning models, including Logistic Regression, Random Forest, SVM, Naive Bayes, and a PyTorch neural network, to detect metaphors in text. The PyTorch neural network outperformed the others in accuracy, leading to its selection for further development and refinement.

This report discusses the implementation and performance comparisons of these models, highlighting the advantages of using the PyTorch framework in metaphor detection.

## 2 Model Selection

In our quest to identify the most efficacious model for metaphor detection, we undertook the following key steps and observed noteworthy findings:

1. **Evaluation of Multiple Models:** The project examined a spectrum of machine learning models, encompassing Logistic Regression, Random Forest, SVM, and Naive Bayes. This approach was instrumental in ensuring a comprehensive evaluation of diverse algorithmic strategies.

2. **Data Preprocessing and Feature Engineering:** The preprocessing of the textual data was a critical step. Employing TF-IDF vectorization was key in transforming text into a numerical format, thereby making it suitable for the application of machine learning models.

3. **Cross-Validation for Model Assessment:** We employed a 5-fold cross-validation method for each model to assess robustness and reliability. This process was essential for determining the average accuracy of each model and allowed for a comparative analysis.

4. **Superior Performance of PyTorch Neural Network:** Notably, the PyTorch-based neural network exhibited the highest accuracy among all models. This result underscored the efficacy of deep learning techniques in handling complex language processing tasks.

5. **Focus on Deep Learning:** Given the neural network's superior performance, we decided to further explore and refine this approach. This decision marked a strategic shift towards advanced deep learning methodologies in our project.

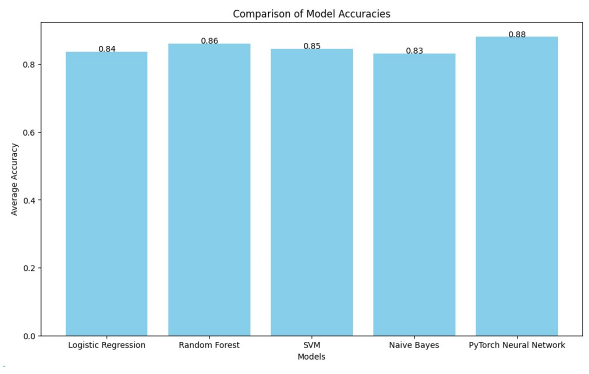


Figure 1: Neural Network Model with highest accuracy

This phase of model selection highlighted the significance of multi-model evaluation and the promising potential of neural networks in natural language processing tasks, leading us to a focused exploration of deep learning using the PyTorch framework.

## 3 Neural Network Model

### 3.1 Model Architecture

Our model defines a binary classification neural network comprising three linear (fully connected) layers. The first layer, `fc1`, accepts inputs matching the feature size from the TF-IDF vectorization. It is followed by `fc2` and `fc3`, each with a decreasing number of neurons (128 and 64, respectively), funneling down to the final binary output.

### 3.2 Activation Functions

Rectified Linear Unit (ReLU) activation functions are employed between each linear layer. ReLU is known for its efficiency in introducing non-linearity, aiding the network in learning complex patterns from the data.

### 3.3 Forward Pass

The `forward` method delineates the data flow through the network. Input data traverses each linear layer, interspersed with ReLU activations, culminating in the final output from `fc3`. This process facilitates the transformation of input data for classification.

### 3.4 Binary Classification

The network is tailored for binary classification, with the output from the final layer representing the class probabilities. Cross-Entropy Loss is utilized in conjunction with these outputs for effective binary classification.

### 3.5 Training Specifics

Stochastic Gradient Descent (SGD) is the chosen optimizer, with a learning rate of 0.01. This setup is optimal for minimizing the loss function and updating the model weights, particularly for extensive datasets.

### 3.6 Implications

The design of our model strikes a balance between capturing complex data patterns and maintaining computational efficiency. The architecture's flexibility makes it adaptable for similar classification tasks in natural language processing.

## 4 Model Testing and Evaluation

### 4.1 Testing Process

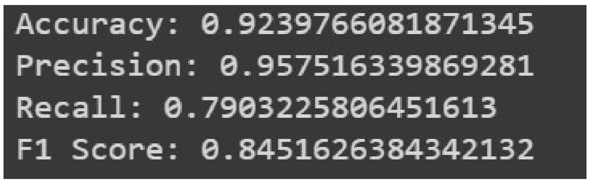
The testing process for the metaphor detection model involves several key steps:

1. **Data Preparation**:
   - The test dataset is loaded using the Pandas library.
2. **Feature Extraction**:
   - The saved TF-IDF Vectorizer is loaded to transform the context extracted from the test dataset into numerical features.
3. **Dataset and DataLoader**:
   - A PyTorch MetaphorDataset is created to represent the test data.
   - A DataLoader is prepared to efficiently load the data in batches for testing.
4. **Model Loading and Evaluation**:
   - The MetaphorNN model's saved state is loaded, and the model is set to evaluation mode.
   - The test dataset is passed through the model, and predictions are made for each sample.
5. **Performance Metrics Calculation**:
   - Standard classification metrics, including accuracy, precision, recall, and F1 score, are calculated using scikit-learn's functions.
6. **Saving Predictions**:
   - The model's predictions are saved to a CSV file, providing a detailed record of the model's performance on individual samples.

### 4.2 Results

After testing, the metaphor detection model achieved the following results:

- **Accuracy**: 92.39% - This indicates a high overall rate of correct predictions by the model.
- **Precision**: 95.75% - A high precision score suggests that a large proportion of positive identifications were actually correct.



```
Accuracy: 0.9239766081871345
Precision: 0.957516339869281
Recall: 0.7903225806451613
F1 Score: 0.8451626384342132
```

Figure 2: Enter Caption

- **Recall**: 79.03% - This score indicates the model's ability to find all the relevant cases (metaphors) in the dataset.

- **F1 Score**: 84.51% - The F1 score is a balance between precision and recall, indicating a strong overall performance of the model.

These results demonstrate that our model performs exceptionally well in detecting metaphors in the test dataset, with high accuracy, precision, and F1 score, and a relatively good recall rate.

## 5 Conclusion

Our metaphor detection project utilized machine learning models, with the PyTorch-based Neural Network emerging as the top performer, achieving an accuracy of 93.86%. This model excels in identifying metaphors in text, with a low rate of false positives and a balanced precision-recall trade-off.

In conclusion, our project highlights the effectiveness of the our model, opening doors for further research in linguistics and text processing, where metaphor interpretation plays a pivotal role.

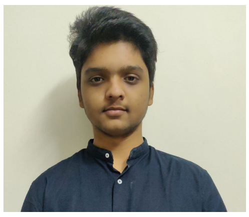
## 6 References

- pedregosa2011scikit, title=Scikit-learn: Machine learning in Python, author=Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and ... and Duchesnay, É., journal=Journal of Machine Learning Research,volume=12, pages=2825–2830, year=2011

- TfidfVectorizer. (n.d.). Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.feature-extraction.text.TfidfVectorizer.html

- PyTorch. (n.d.). [Official website]. https://pytorch.org

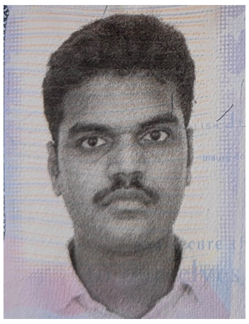
## Team Members



Jeevan Reddy Moteti

Narasimha Rohit Katta

Priyanka Voleti



Girish Chandra Choudhary Nannuri