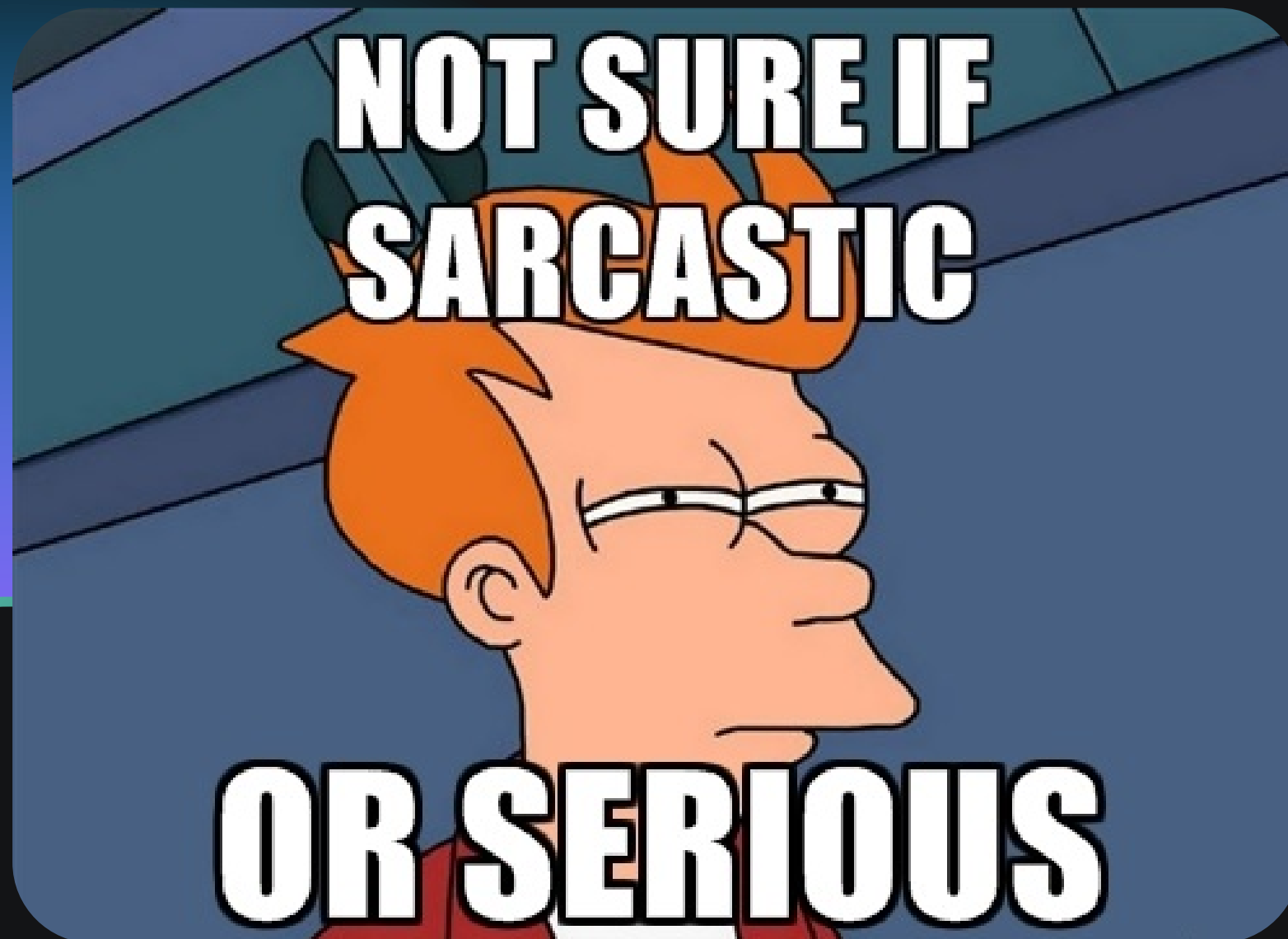


Sarcasm Detector

Natural Language Processing



Presented by Danel Kanbakova,
Zhanel Aldan and Aruzhan Satybaldy

Our Team



Kanbakova Danel



Aldan Zhanel



Satybaldy Aruzhan



CONTENT

1

INTRODUCTION

Project is about detecting sarcasm in text, which is tricky because sarcasm can be subtle. The goal is to build a model that can spot sarcastic comments, especially in social media posts.

2

THEORETICAL PART

We look at the basics of natural language processing and machine learning, and how they can help us understand sarcasm in text.

3

PRACTICAL PART

We clean the data and build a machine learning model using Python to tell sarcastic sentences apart from regular ones.

4

RESULTS

The model was tested on real data and demonstrated a strong ability to detect sarcasm, providing valuable insights for future enhancements.

5

SUMMARY

This project shows how NLP and machine learning can work together to detect sarcasm, and points to ways we could improve in the future.



NLP

Natural Language Processing

01

Abstract

Detecting sarcasm in text is one of the more difficult challenges in Natural Language Processing. In this project, we aimed to develop and compare several machine learning and deep learning models for sarcasm detection, using a balanced dataset from Kaggle.

Our workflow included data exploration, preprocessing, feature extraction, model training, and evaluation. Transformer-based models, such as RoBERTa, demonstrated significantly better performance than traditional and recurrent neural network approaches.

Introduction

1

Aim

The main goal of our project is to design and evaluate different models capable of identifying sarcasm in text data, thereby enhancing the automated understanding of nuanced human language in digital communication.

2

Scope

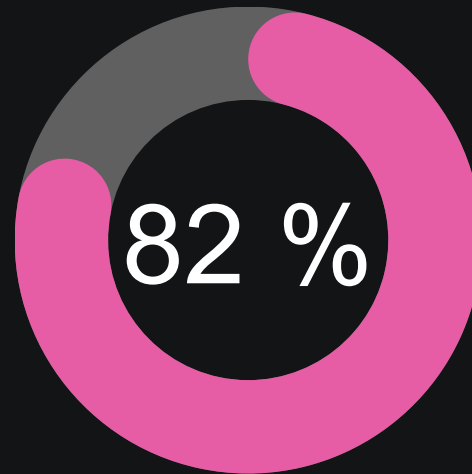
This project covers several stages: acquiring a public sarcasm detection dataset, performing text preprocessing, building models using supervised learning, neural networks, and transformer-based methods, and finally evaluating these models using appropriate performance metrics.

3

Methodology

We implemented this project in Python using Jupyter Notebook. The libraries we used include Pandas, NLTK, Scikit-learn, TensorFlow/Keras, and Hugging Face Transformers. The workflow involved data cleaning, feature extraction, model training, and thorough evaluation using cross-validation and classification metrics.

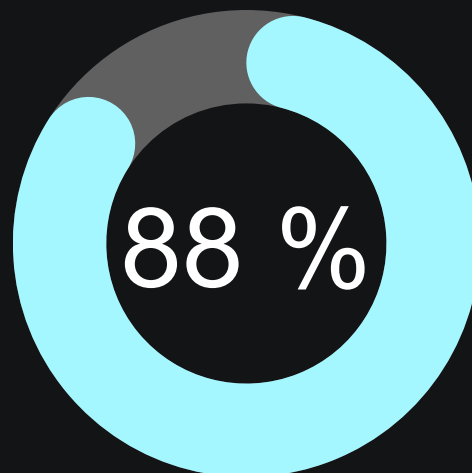
I. Theoretical Part



Supervised Learning Algorithms

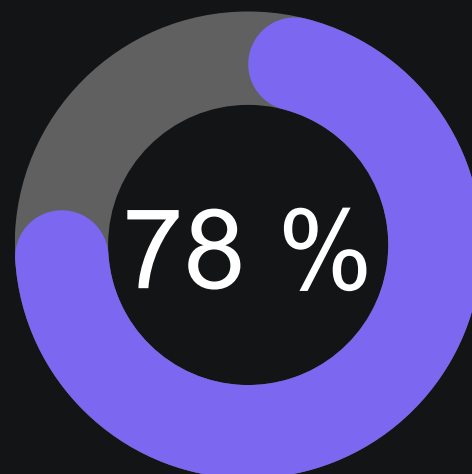
We applied two supervised algorithms:

- Logistic Regression: A simple, interpretable linear classifier with fast training time.
- Gradient Boosting: An ensemble learning technique that combines multiple weak learners to build a more accurate and robust model, especially for handling non-linear patterns.



Neural Network Models

- LSTM (Long Short-Term Memory): A type of recurrent neural network effective for sequential data and capable of capturing long-term dependencies in text.
- GRU (Gated Recurrent Unit): A simplified variant of LSTM, offering similar advantages with fewer parameters and faster training.



Transformer-Based Models

RoBERTa (A Robustly Optimized BERT Pretraining Approach): A state-of-the-art pretrained transformer model that excels in understanding the context and nuances of natural language.

II. Practical Part



**Model
Evaluation**



**Model
Implementation**



**Data
Preprocessing**



**Data
Collection**

Data Collection

We used a dataset sourced from Kaggle, containing approximately 1.3 million text samples labeled as sarcastic or non-sarcastic. The data was provided in CSV format and was already balanced between both classes.

Data Preprocessing

Text preprocessing included several steps:

- Removing URLs, punctuation, numbers, and stopwords
- Applying tokenization and lemmatization
- Conducting exploratory analysis to understand class distribution and text characteristics

Model Implementation

1

- Logistic Regression with TF-IDF vectorization
- Gradient Boosting Classifier

2

- LSTM and GRU neural networks using TensorFlow/Keras
- RoBERTa transformer model using Hugging Face Transformers

An example:

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(max_features=10000, ngram_range=(1,2))),
    ('clf', LogisticRegression(max_iter=100))
])
pipeline.fit(X_train, y_train)
```


80%

Accuracy

0,78

F1-score

Model Evaluation

01

Accuracy

02

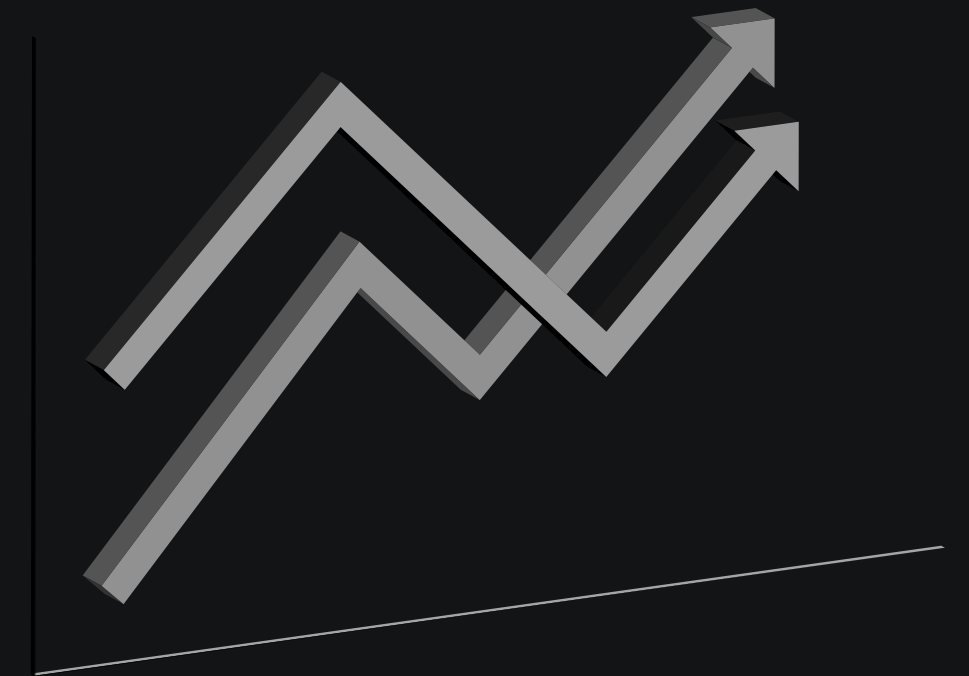
Precision

03

Recall

04

F1-Score



These metrics help us understand how accurately the model detects sarcasm and balances between false positives and false negatives.

Summary



GOALS ACHIEVED

Successfully applied NLP methods to detect sarcasm in text.



BEST MODEL

Transformer-based models (especially RoBERTa) gave the best results.



OTHER MODELS

- LSTM / GRU: Promising, but slow training and overfitting issues
- Traditional ML: Simple, but solid baseline



FUTURE WORK

- Use larger datasets
- Fine-tune transformer hyperparameters
- Try newer models like GPT or LLaMA

Thank You