Twitter Sentiment Analysis Using Classical & Neural Models

Final presentation Group 2

Data Science DHBW Mosbach - Nov 2025

Berriel Martins, Joaquin

Sengül, Emin

HUANG YEN WEI（Will）

From Discord Chat To Data Science Problem

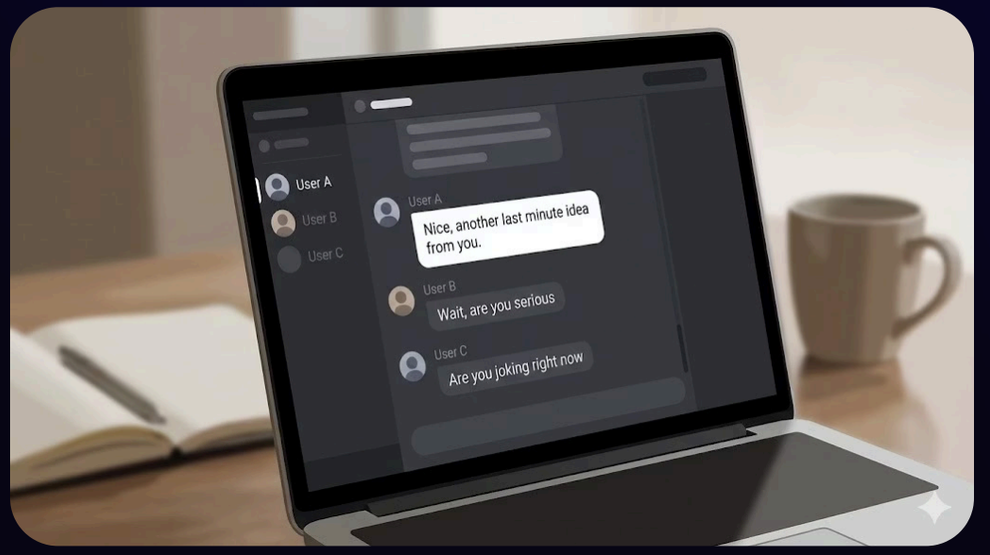
Brainstorming our project on Discord chat

Message like "*Nice, another last minute idea from you*" created confusion

We could not tell if it was a joke or real frustration

Realization: text alone often hides true emotion

How can we automatically understand sentiment in text at scale?



Why Sentiment Analysis Matters

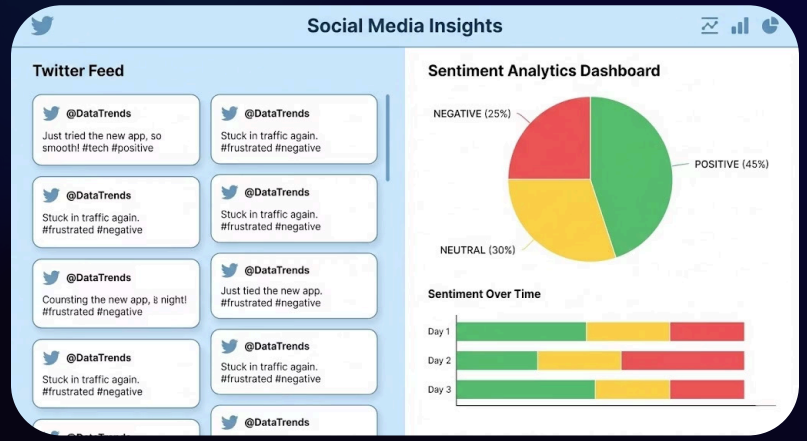
Social media platforms like Twitter produce millions of short messages

Companies and organizations watch these messages for customer opinion and crises

Manual reading is impossible at this scale

We focus on automatic classification of tweets into negative, neutral, positive

Goal: support monitoring and decision making with measurable sentiment



Research Questions And Objectives

RQ1

Which model performs best for multiclass Twitter sentiment classification ?

RQ2

How do TF IDF and Word2Vec features affect classical model performance ?

RQ3

Can a CNN LSTM neural model outperform classical ensembles ?

Objective: build a full pipeline from EDA to classical and neural models and compare them

Work Organization

Three feature pipelines

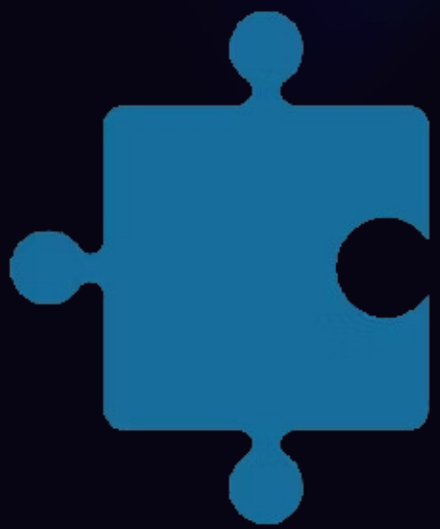
- *TF IDF* vectors on processed text
- *Word2Vec* average embeddings
- Tokenized and padded sequences for *Neural Network*

Classical models

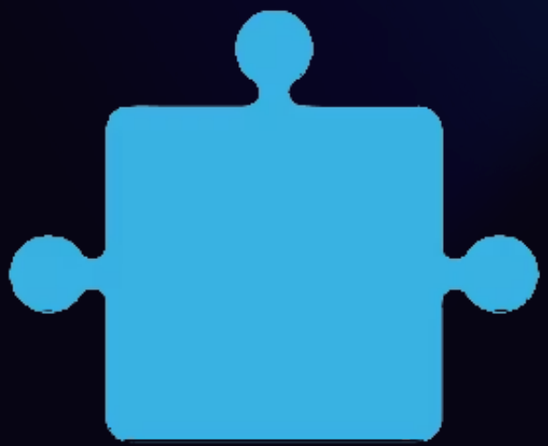
- *Decision Tree, KNN, Logistic Regression, Voting Classifier*

Neural model

- *CNN LSTM* hybrid for sequence modeling



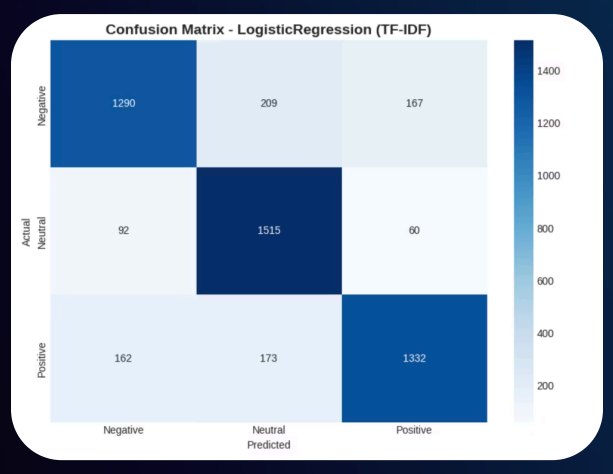
Joaquin **→ EDA, Dataset Preprocessing and Neural Network**



Emin **→TF IDF feature engineering**

Will **→Word2Vec feature engineering**

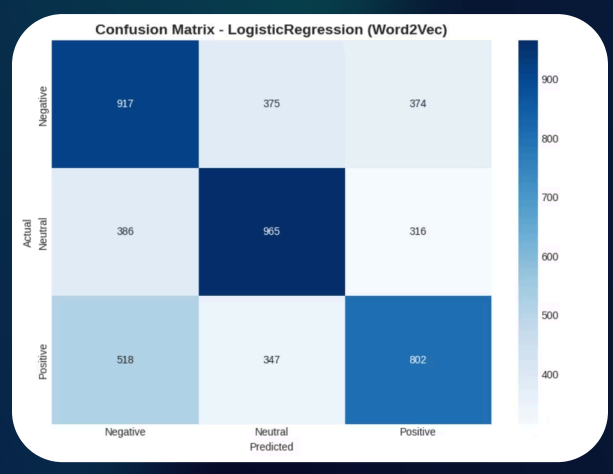
Feature Engineering And Models Overview



TF IDF And Classical Models

1. Build TF-IDF features (up to 5000 n-gram features)
2. Stratified train/test split
3. GridSearchCV for hyperparameter tuning, scoring with F1 macro
4. Train classical models: DT, KNN, Logistic Regression, Voting
5. Evaluate metrics and confusion matrices
6. Best TF-IDF model: Logistic Regression

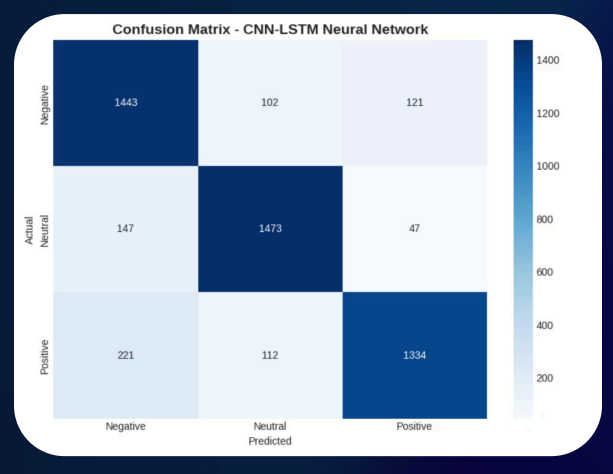
```
# Initialize TF-IDF Vectorizer
self.tfidf_vectorizer = TfidfVectorizer(
    max_features=5000,
    ngram_range=(1, 2), #bigrams like "not good"
    min_df=5,
    max_df=0.95
)
```



Word2Vec Models

1. Tokenize and clean text
2. Train Word2Vec (100-dim embeddings)
3. Average embeddings per tweet
4. Stratified train/test split
5. Train classical models (DT, KNN, LR, Voting)
6. Best model: Logistic Regression (F1 ≈ 0.53)

```
# Train Word2Vec model
print("Training Word2Vec model...")
self.word2vec_model = Word2Vec(
    sentences=tokenized_texts,
    vector_size=100,
    window=5,
    min_count=3,
    workers=4,
    seed=self.random_state
)
```



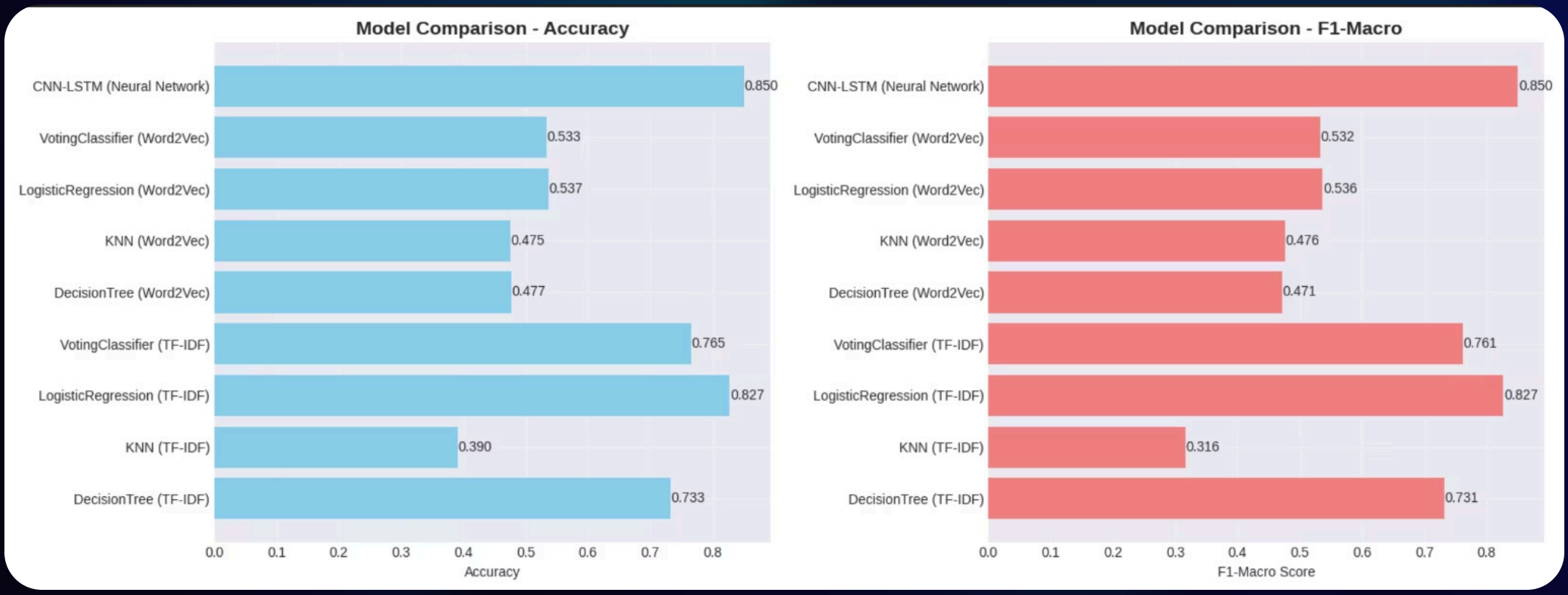
CNN-LSTM (Neural Network)

1. Tokenize text and pad sequences (max length = 100)
2. Convert labels to categorical (3-class)
3. Build hybrid model: Embedding → Conv1D → MaxPool → LSTM → Dense
4. Compile with Adam optimizer and cross-entropy loss
5. Train with callbacks (EarlyStopping, Checkpoint, LR scheduler)
6. Evaluate performance (accuracy, F1-macro, confusion matrix)

```
def build_cnn_lstm_model(self, max_words, max_len):  1 usage  berm
    """Build CNN-LSTM hybrid model..."""
    model = Sequential([
        # Embedding layer
        Embedding(max_words, output_dim: 128, input_length=max_len),

        # CNN layers (Conv1D)
        Conv1D(filters=64, kernel_size=3, activation='relu'),
        MaxPooling1D(pool_size=2),
        Dropout(0.3),  ...
        LSTM( units: 64, return_sequences=False,
            dropout=0.3, # incrised to reduce overfiting
            recurrent_dropout=0.3), ...
        Dense( units: 32, activation='relu'),
        BatchNormalization(),
        Dropout(0.4), ...
        Dense( units: 3, activation='softmax')
    ])
```

Model Comparison – TF-IDF vs Word2Vec vs CNN-LSTM



1	CNN-LSTM (Neural Network) achieves the best performance with Accuracy and F1-Macro around 0.85
2	TF-IDF with Logistic Regression and VotingClassifier is slightly behind and is our strongest classical approach
3	All Word2Vec models perform clearly worse than the best TF-IDF models
4	KNN with TF-IDF is the weakest model with the lowest Accuracy and F1-Macro

Conclusion:

feature representation and model family have a big impact – Neural Network > best TF-IDF models > Word2Vec models > KNN TF-IDF

Live Demo

Conclusion

1	*Which model performs best for multiclass Twitter sentiment classification?* The **CNN-LSTM** hybrid model achieved the best overall performance, demonstrating a robust F1-Macro score of **0.8663** for complex multiclass sentiment classification.
2	*How do TF IDF and Word2Vec features affect classical model performance?* For classical models, **TF-IDF** features were substantially superior to Word2Vec features, confirming TF-IDF's power in isolating discriminative keywords in social media text.
3	*CNN LSTM neural model outperform classical ensembles?* The deep learning (CNN-LSTM) approach outperformed all classical ensembles, proving that sequence models are necessary for capturing the nuances and context of Twitter data.

- Found that feature representation strongly influences model quality
- Future work will focus on interpretability and deployment in practical applications

Limitations

- Single dataset, one language, limited hyperparameter search
- Samples >25000 not possible to evaluate (directly affect Word2Vec Feature)
- Models do not consider emojis :) !!! ? ;)

Next steps

- Full Dataset Rerun
- Pre-trained Embeddings: Evaluate using pre-trained word embeddings (like GloVe or fastText)

Thank You!

Q&A

Berriel Martins, Joaquin
joa.berriel.24@lehre.mosbach.dhbw.de

Sengül, Emin
emi.senguel.24@lehre.mosbach.dhbw.de

HUANG YEN WEI (Will)
anin0919@gmail.com