Project 3: NLP Challenge

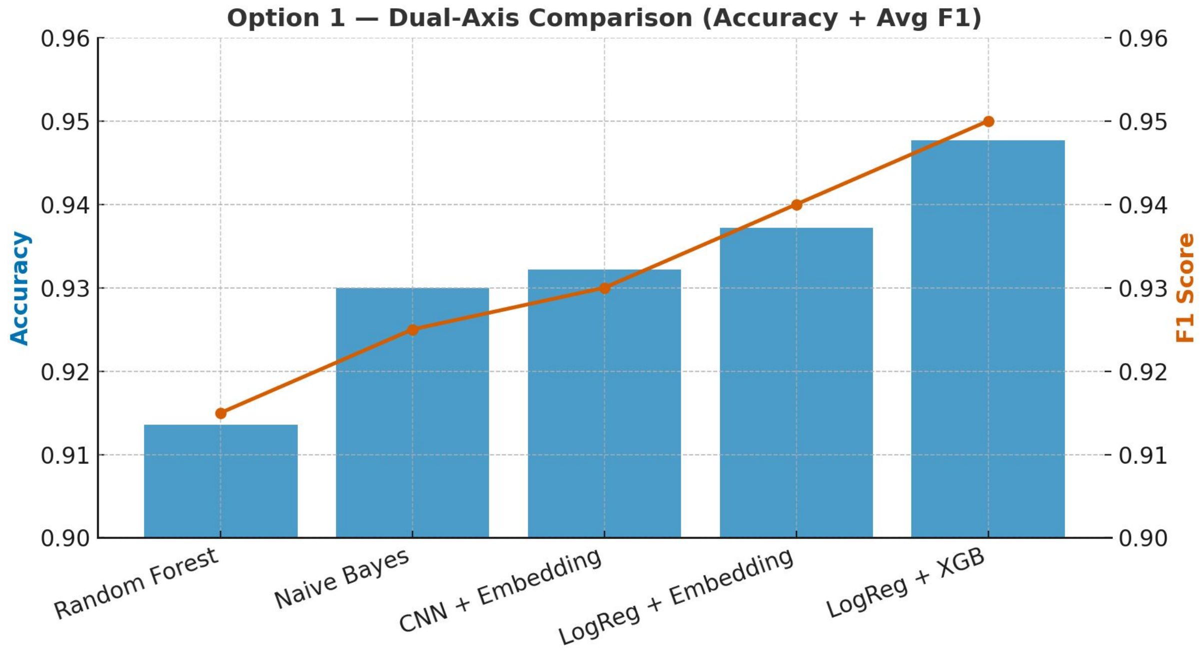
Maria Eduarda Oliveira, Adrián A. H., Matheus Viana, Kira Redberg

Executive summary

Final result: accuracy of 94.77%
Model: LogisticRegression + XGB

Other alternatives:

- Naive Bayes Model
- Random Forest Classifier
- Convolutional Neural Network (CNN) + Embedding
- LogisticRegression + Embedding

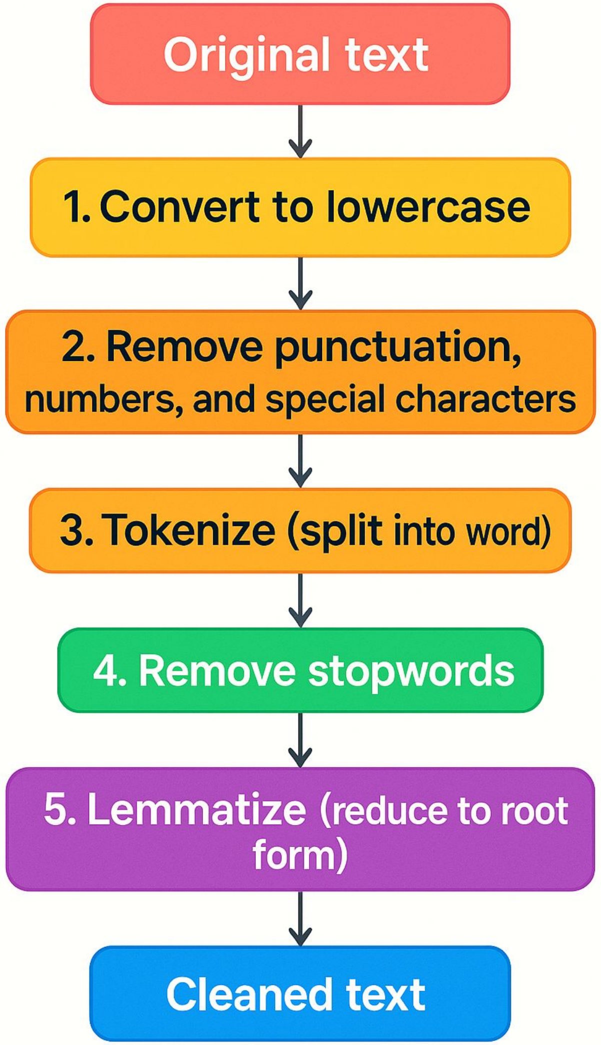


Methods (preprocessing)

2- Load and check the data

```python
# Reads the CSV file using "\" as separator
df = pd.read_csv("training_data_lowercase.csv", sep='\t', engine='python', names=['label', 'title'])

# Show first few rows of the dataframe
print(df.head())
```

```
   label                                              title
0      0  donald trump sends out embarrassing new year,s...
1      0  drunk bragging trump staffer started russian c...
2      0  sheriff david clarke becomes an internet joke ...
3      0  trump is so obsessed he even has obama,s name ...
4      0  pope francis just called out donald trump duri...
```



Methods (preprocessing)

1. Load & Split Data
Load train/test datasets and split the training data into train/validation sets.

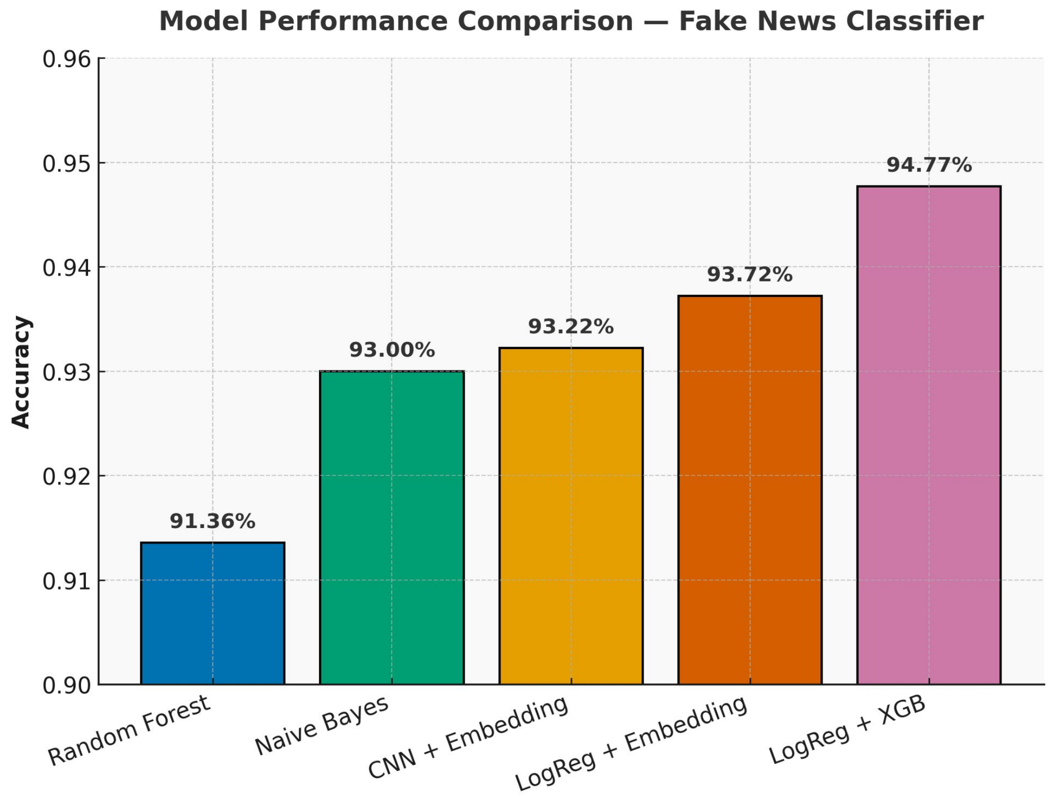
2. Text Features
Extract sentiment, subjectivity, and clickbait indicators from headlines.

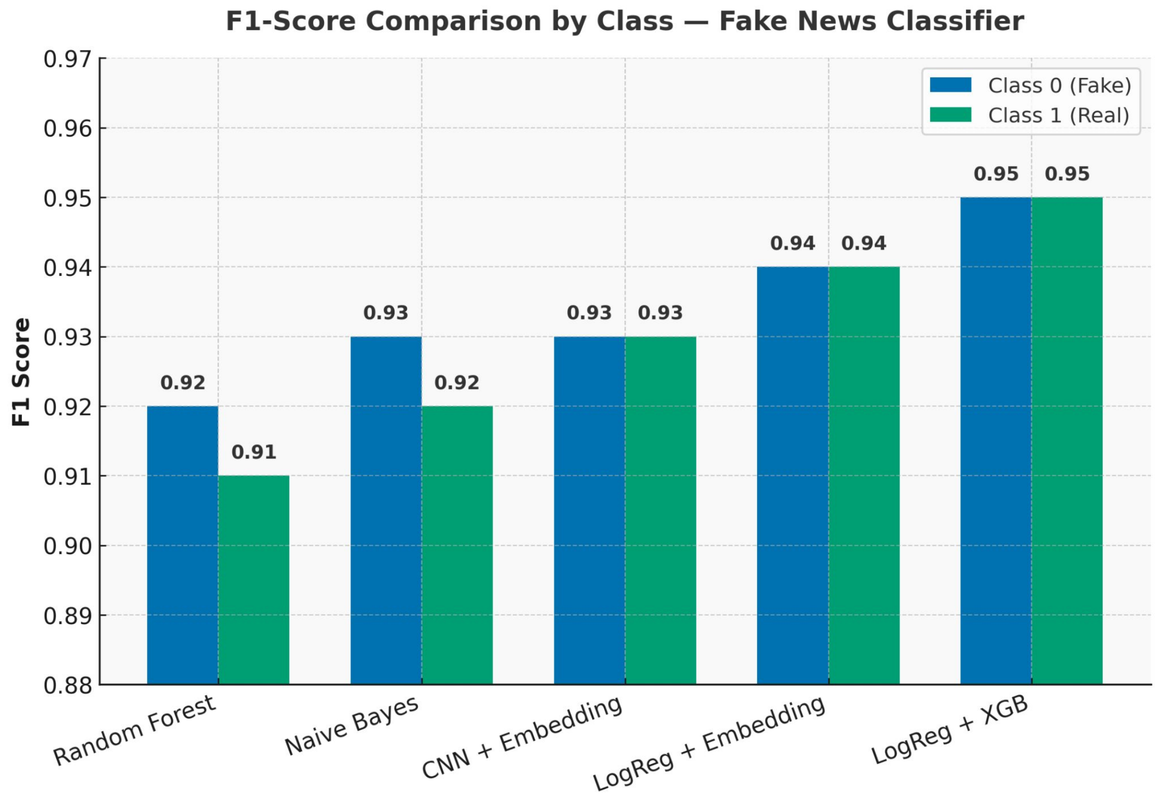
3. Structural Features
Count ! and ?, measure capitalization ratio, text length, and lexical diversity.

4. Vectorization
Convert text into numerical form using Bag-of-Words and TF-IDF (1–3-grams, 10k features each).

5. Feature Combination
Merge BoW, TF-IDF, and all extra numeric features into a single feature matrix for model training.

Methods (models)





Optimizing CNN Parameters for Higher Accuracy

```
# ---------------------------------------------------------
# Deep Neural Network (Sequential Model)
# ---------------------------------------------------------
from tensorflow.keras.optimizers import Adam

model = Sequential([
    Dense(512, activation='relu', input_shape=(X_train_final.shape[1],),
          kernel_regularizer=regularizers.l2(0.002)),
    BatchNormalization(),
    Dropout(0.4),

    Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    BatchNormalization(),
    Dropout(0.3),

    Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
    Dropout(0.25),

    Dense(1, activation='sigmoid')
])

optimizer = Adam(learning_rate=1e-3)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

Key Adjustments Made:

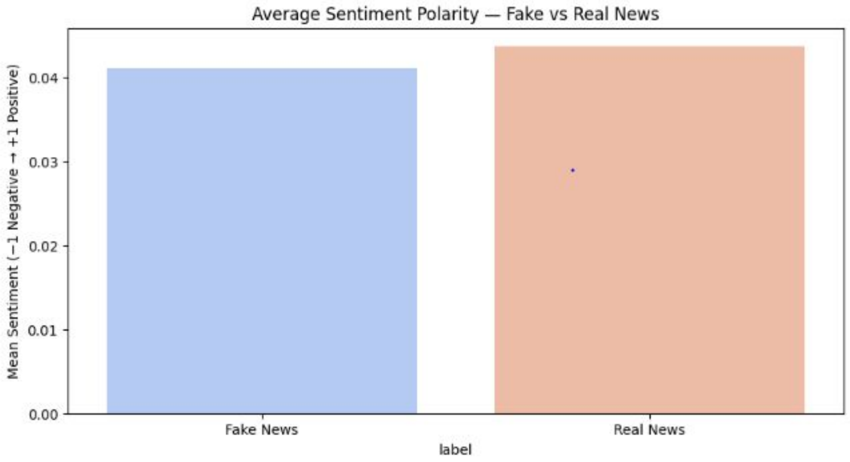
1. **Adding Layers** – to increase model depth and capture more complex patterns.
2. **Changing the Number of Neurons** – to optimize the network's capacity and avoid under/overfitting.
3. **Adjusting Dropout Rate** – to reduce overfitting and improve generalization.

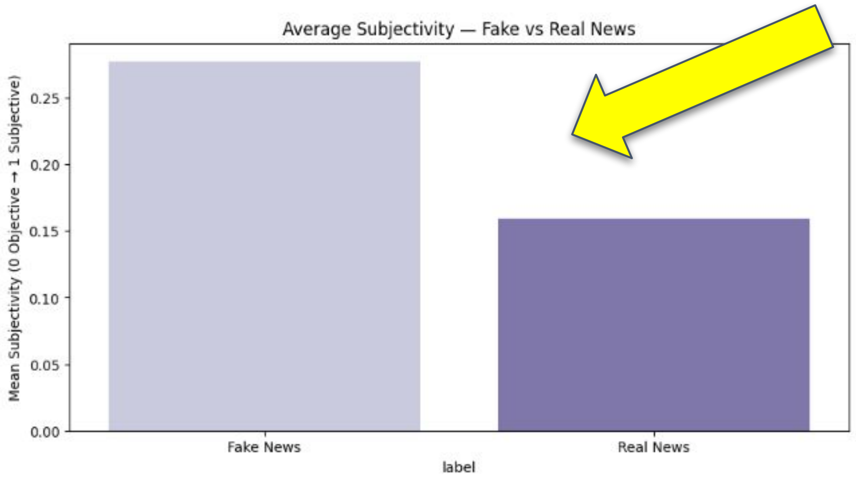
Training vs Validation accuracy:

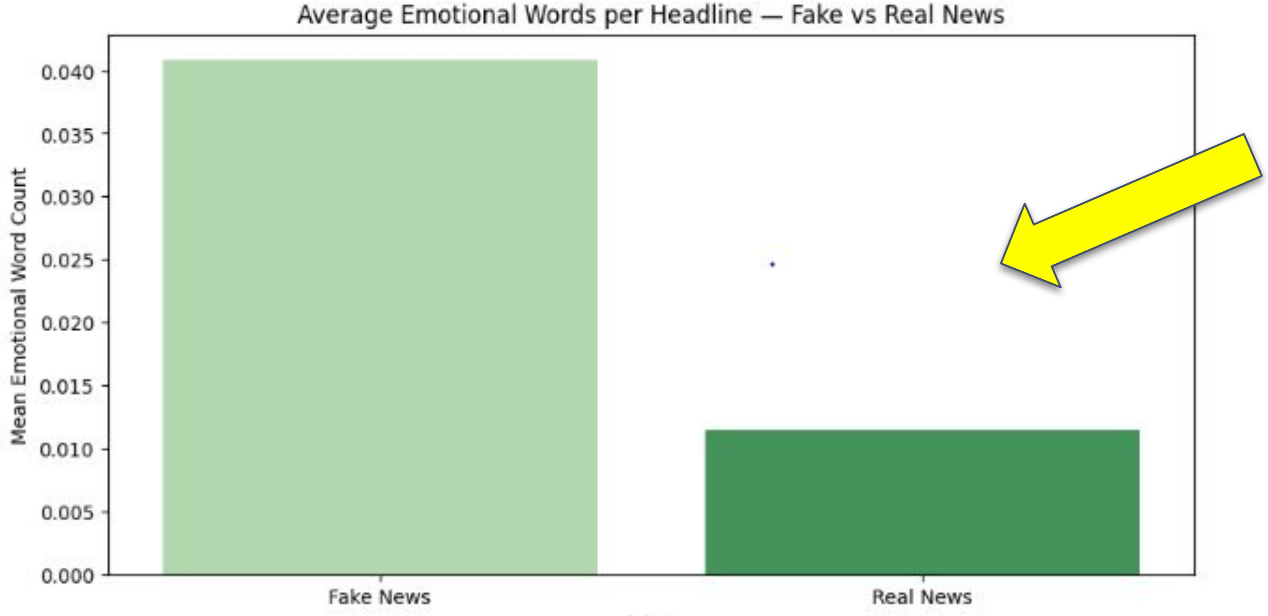
FIXING OVERFITTING

```
Epoch 19/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9652 - loss: 0.2255 - val_accuracy: 0.9278 - val_loss: 0.3413 - learning_rate: 0.00
Epoch 20/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9627 - loss: 0.2312 - val_accuracy: 0.9293 - val_loss: 0.3394 - learning_rate: 0.00
Epoch 21/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9684 - loss: 0.2122 - val_accuracy: 0.9322 - val_loss: 0.3196 - learning_rate: 5.00
Epoch 22/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9776 - loss: 0.1676 - val_accuracy: 0.9308 - val_loss: 0.3197 - learning_rate: 5.00
Epoch 23/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9778 - loss: 0.1562 - val_accuracy: 0.9287 - val_loss: 0.3297 - learning_rate: 5.00
Epoch 24/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9822 - loss: 0.1391 - val_accuracy: 0.9312 - val_loss: 0.3259 - learning_rate: 2.50
Epoch 25/25
427/427 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9843 - loss: 0.1206 - val_accuracy: 0.9287 - val_loss: 0.3317 - learning_rate: 2.50

✅ Final Validation Accuracy: 0.9322
214/214 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step

📊 Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      3515
           1       0.93      0.93      0.93      3316

    accuracy                           0.93      6831
   macro avg       0.93      0.93      0.93      6831
weighted avg       0.93      0.93      0.93      6831
```

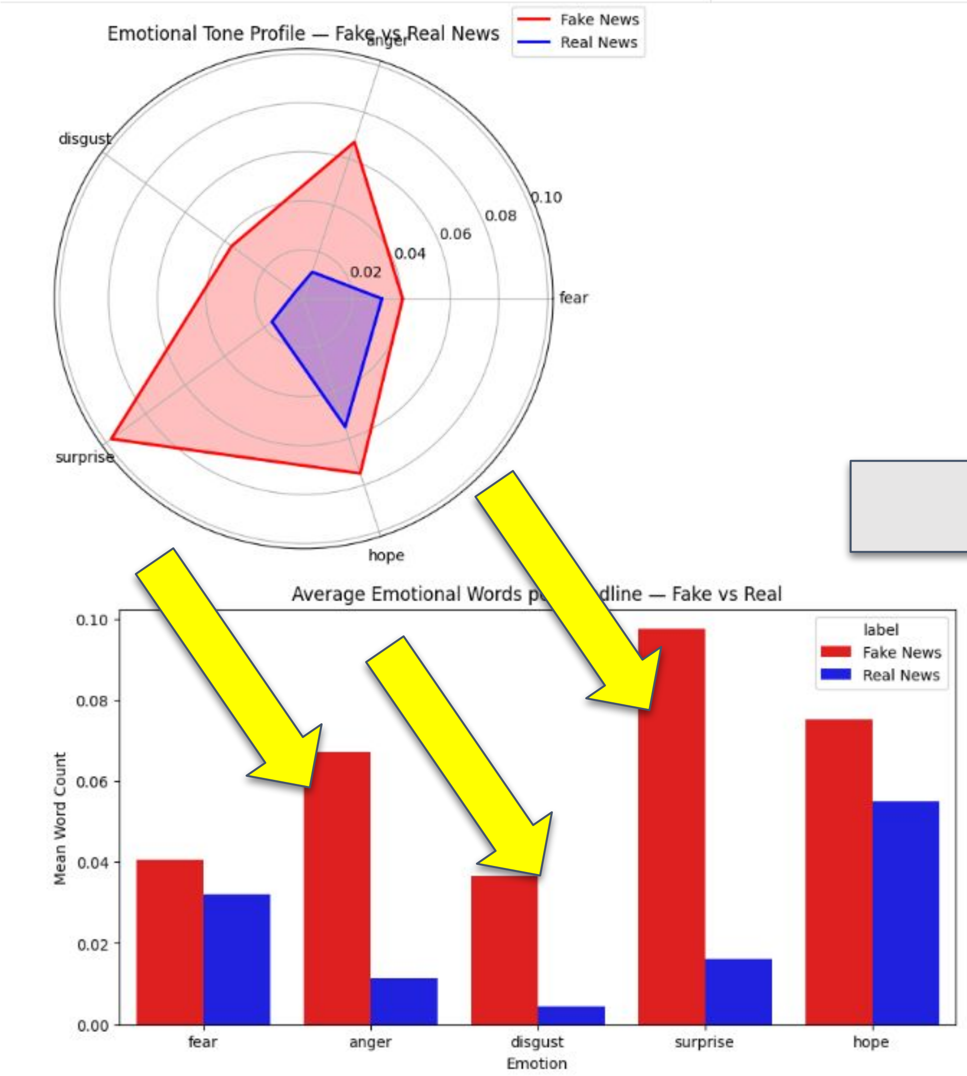
STEP BACK TO PREPROCESS DATA AGAIN







Emotional analysis FAKE NEWS VS REAL NEWS




```
# -------------------------------------------------------------
# Emotion lexicons – anger, disgust, surprise
# -------------------------------------------------------------
nltk.download("wordnet")
nltk.download("omw-1.4")
lemmatizer = WordNetLemmatizer()

emotion_seeds = {
    "anger": ["anger","rage","furious","outrage","hate","mad","irritated","annoyed"],
    "disgust": ["disgust","gross","nasty","horrible","filthy","vile","repulsive","sick
    "surprise": ["surprise","shock","amazing","unexpected","astonishing","wow","sudden
}

def get_synonyms(word):
    syns = set()
    for syn in wordnet.synsets(word):
        for lemma in syn.lemmas():
            name = lemma.name().replace("_"," ").lower()
            if len(name) > 2:
                syns.add(name)
    return syns
```

Takeaways

- **Text representation matters most.**
  Upgrading from basic TF-IDF to semantic embeddings significantly improved generalization.

- **Simple ≠ weak.**
  Logistic Regression, when enhanced with proper features and ensemble boosting, outperformed deeper neural networks.

- **Hybrid approaches win.**
  Combining linear models for structure + tree models for interaction patterns delivered the best accuracy and stability.

- **Balanced evaluation is crucial.**
  Equal F1 for both classes shows model fairness — essential in fake news detection.

- **Feature engineering is still king.**
  Sentiment, subjectivity, and clickbait cues added valuable context beyond raw text.