Beyond Clickbait

COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS FOR HEADLINE SIMILARITY

Kenya Williams

With special thanks to supervisor:
Jizheng Wan

Project Motivation

Problem: News sources often cover the same events from varying perspectives, making it difficult to identify trustworthy information. This issue is exacerbated by intentionally misleading headlines, a.k.a. ‘clickbait’.

Context: Automating headline similarity detection can help determine whether two news sources are discussing the same event, thereby tackling news misinformation.

Motivation: Explore how data science can contribute to media verification.

Research aim:

Systematically assess machine learning models to classify headline similarity.

Project Design (each involving Cosine Similarity)

EMBEDDING TECHNIQUES

1. TF-IDF Vectorisation
2. BERT Embedding
3. Word2Vec Embedding

Total combinations, including hyperparameter tuning: **82+ (w/ 410 training cycles)**

MACHINE LEARNING MODELS

A. Random Forest (RF)

B. Support Vector Machine (SVM)

C. Gradient Boosting Machine (XGB)

D. State-of-the-Art option: Large Language Model (ChatGPT)

Comparing Traditional Methods with LLM Results

THE PROMPT

Given the dataset below ('test.csv'), each row contains two headlines – 'headline_a' and 'headline_b'. Please classify each headline pair as either discussing the same event or not. These headlines are from real news articles covering global events. Label pairs that discuss the same event as '1' and pairs that do not as '0'. Provide the classifications in a list where each entry corresponds to a row in the dataset.

Note that the headlines belong to different subject categories (as seen in column 'timeline_id') so there will be headlines that talk about the same broader topic (e.g. whistleblowing in the USA) but not necessarily the same exact event (e.g. Edward Snowden and President Obama negotiations)

Instructions:
1. Review each headline pair for syntactic, semantic, thematic and contextual similarities.
2. Consider the use of specific keywords, temporal details, and locations mentioned in the headlines.
3. Return a list of binary labels where each label corresponds to the similarity classification of the headline pair.
4. Compare your prediction with the actual classification that is provided in the column 'label'. Return the following metrics: accuracy, precision, recall, f1-score, confusion matrix, classification report.
Here is the dataset:

THE RESULTS

The classification results using a basic TF-IDF cosine similarity approach are as follows:

- Accuracy: 0.79
- Precision: 1.0
- Recall: 0.12
- F1-Score: 0.22

Confusion Matrix:

```lua
[[1905,    0],
 [ 517,   73]]
```

Classification Report:

```yaml
              precision    recall  f1-score   support

           0       0.79      1.00      0.88      1905
           1       1.00      0.12      0.22       590

    accuracy                           0.79      2495
   macro avg       0.89      0.56      0.55      2495
weighted avg       0.84      0.79      0.72      2495
```

↓

These results indicate that while the model achieves high precision, it suffers from low recall,

benchmarks. We propose to repurpose a Headline Generator for the task of headline grouping, based on prompting it for the likelihood of a headline swap, and achieve within 3 F-1 of the best supervised model, paving the way for other unsupervised methods to repurpose generators for NLU. Analy-

Related Literature

News Headline Grouping as a Challenging NLU Task (Laban et al.)

- Gives framework to compare LLM's results with the best supervised model results
- Misleading headlines in mainstream media are more dangerous than outright fake news. (Colomé, J.P. (2024))
  - Motivates the need for automated verification & factchecking

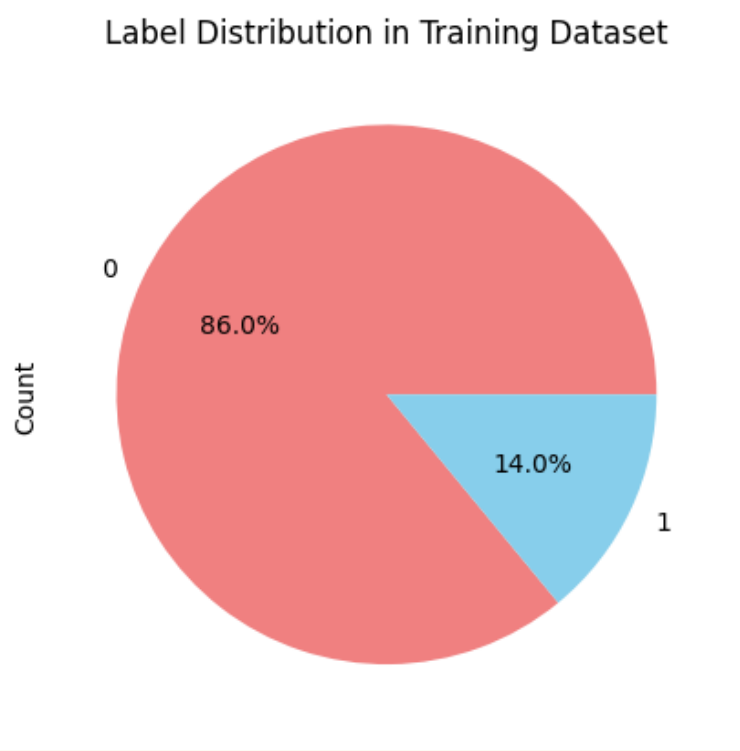
Special mention: Journal of Computational Social Science

Code Demonstration

My Repository

```
# Round 2
rf_param_grid_r2 = {
    'n_estimators': [150],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2],
    'min_samples_leaf': [2],
    'bootstrap': [True],
    'max_features': ['sqrt'],
}

best_random_forest_bert_model, X_rf_bert_resampled, y_rf_bert_resampled = bert_grid_search(
    RandomForestClassifier(random_state=42), rf_param_grid_r2, X_bert_resampled, y_bert_resampled)
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
Best Parameters: {'bootstrap': True, 'max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 2,
Best Score: 0.9835876806999865
```



Applying GridSearch to TF-IDF Vectorised Training Data

Initialise GridSearch hyperparameters for different classifiers

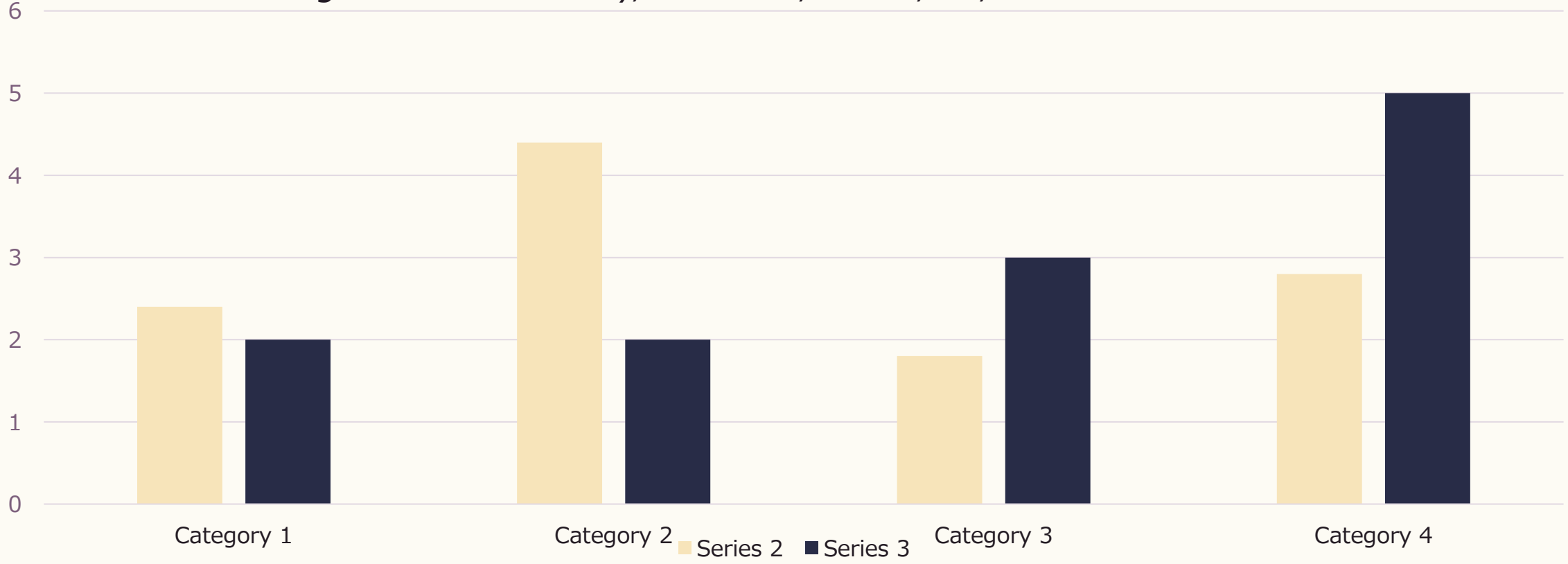
```
# Define hyperparameter grid for RandomForest

rf_param_grid = {
    'n_estimators': [150],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [2],
    'bootstrap': [True],
    'max_features': ['sqrt'],
}


# Define hyperparameter grid for SVM

svm_param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear'],
```

Evaluation

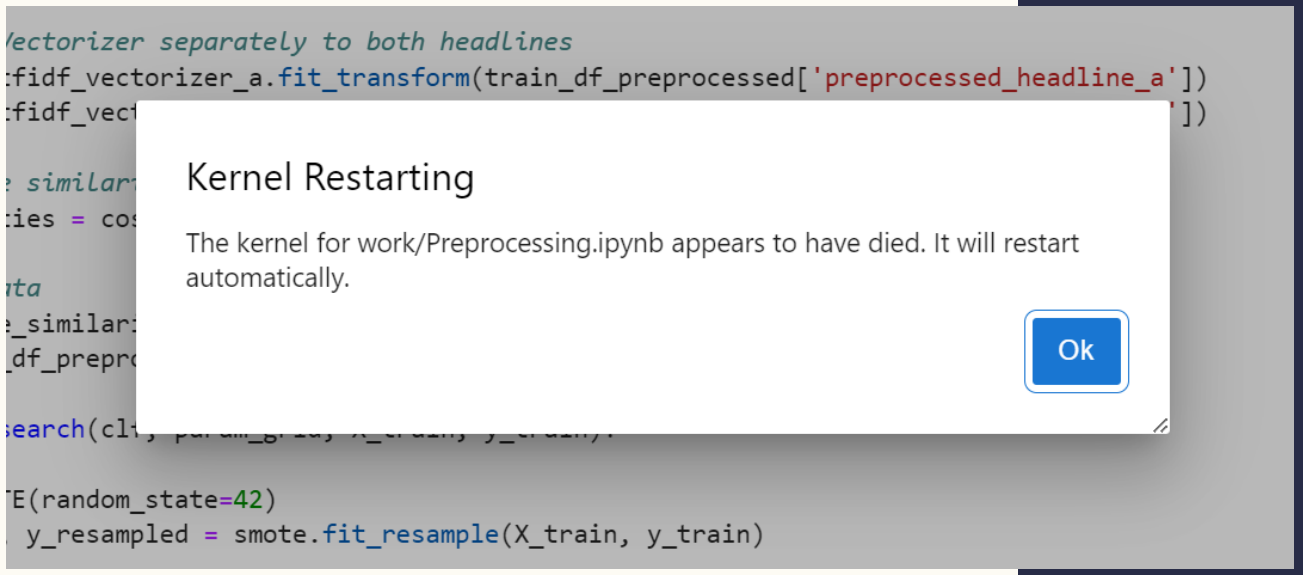
Expected best combination (embedding+model): **BERT + SVM**

Best *performing combination on TF-IDF vectorisation* = _____ TBD

Based on the following metrics: *Accuracy, Precision, Recall, F1, Confusion Matrix*



Executing (2h 19m 37s) <cell line: 33>



Main Adjustments

- Time
  - No scraping of COVID-19 headlines for testing
  - Focus on headlines over content
- Computational Resources
  - Reduce number of K-Folds for Cross-validation to 5
  - Do separate GridSearch for hyperparameter tuning of the same model
- Questions?