UMLT Assignment 3: Language Detection

1.Problem Understanding and Challenges

1.1. Introduction and Motivation.

With the rapid development of large language models (LLMs), we need reliable detection methods to determine whether text was generated by AI.

1.2. Problem Definition (Machine Learning Perspective).

The task is formulated as a binary classification problem that aims to distinguish AI-generated (label = 1) from human-written (label = 0) texts at the paragraph level. The dataset provides sentence-level embeddings, each represented by a 100 × 768 tensor. The final evaluation metric is the Area Under the ROC Curve (AUC) at the paragraph level, which measures the model's ability to separate the two classes across all thresholds. Since the evaluation is conducted on a hidden private leaderboard, the generalisation ability is more important rather than overfitting to the public split.

1.3. Key challenges

Data Scale and Memory Constraints.
The training set, comprising over 16,000 high-dimensional embeddings, resulting in significant memory overhead. Efficient out-of-core data handling pipelines are required to process the data without exceeding hardware limits.

Sentence-to-Paragraph Gap.
The primary challenge is bridging the gap between sentence-level input data and paragraph-level evaluation. This requires developing an effective aggregation strategy to synthesize multiple sentence predictions into a single, accurate paragraph score.

Handling Padding in Embeddings.
Each paragraph is padded to 100 sentences, but exploratory analysis shows that only about 25–27 sentences on average are non-zero. The presence of a large proportion of zero vectors introduces noise, making it necessary to design pooling strategies that can separate meaningful sentences from padding.

Model Selection Trade-offs.
There is an inherent trade-off between lightweight models such as Logistic Regression, which rely on pooling to compress information but are computationally efficient, and sequence-aware models like CNNs and LSTMs, which can capture richer contextual patterns but require greater training resources.

Generalisation to Hidden Test Data.
Since Kaggle evaluation is conducted on a hidden private leaderboard rather than the public split, models that perform well on validation may still fail to generalise. To mitigate this risk, careful cross-validation, appropriate scaling, and relatively simple yet robust architectures are needed to avoid overfitting.

2. Data Exploration

2.1 Dataset Overview

We have a training dataset of 16,322 sentences, an independent validation dataset of 20 paragraphs, and a test dataset of 180 paragraphs which has no labels. This class-balanced training set includes 8,161 AI-generated and 8,161 human-written sentences. Each sentence is represented as a 100×768 embedding vector produced by a RoBERTa-based detector.

Effective sentence length. The RoBERTa generates the embedding with zero paddings if the number of tokens is less than 100 in a sentence, so we count the number of non-padding tokens in training set for each class, as shown in Figure 2.1. Both distributions are right-skewed, confirming that most sentences are shorter than the maximum length of 100. The distributions reveal a significant difference between the two classes. The distribution for human-written text is peaked around 20 tokens, while the AI-generated peak is shifted to the right (around 25 tokens). This difference in length distribution is a strong discriminative signal that can be leveraged by the machine learning models for classification.

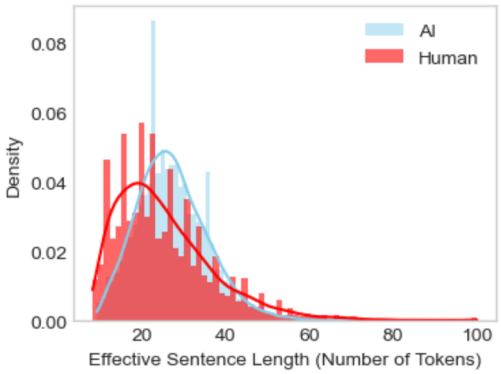

Figure 2.1. The Distributions of Effective Sentence Length in Training Data.

Distribution of Sentences per Paragraph. The number of sentences in a validation or test paragraph distribution is shown in Figure 2.2. The peaks for both the validation and test sets are concentrated around 10 sentences, indicating that the most common paragraph length in the test set is well-represented in the validation set. However, the most critical difference is the outliers in the validation set, suggesting it contains at least two extremely long paragraphs. This is typical for the small sample size (20 paragraphs). While the validation set shares some core similarities with the test set, it has significant limitations due to its small sample size, making it an unreliable proxy for final performance evaluation.

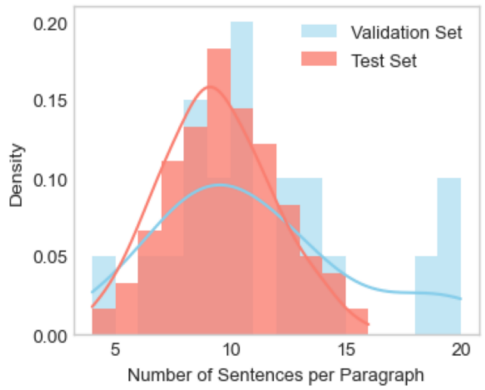

Figure 2.2. The number of sentences per paragraph distributions.

2.2 Data Cleaning

Since the dataset was already preprocessed into fixed-size embeddings, there isn't much traditional cleaning to do. Our main issue comes from zero padding. To deal with this, we use different pooling strategies that either drop or keep zero paddings, and this data

cleaning effect is discussed in Section 2.3.1. This way, the representations stay robust and less affected by artificial noise.

2.3 2D Data Preprocessing

2.3.1 Feature Extraction: Pooling Strategy Evaluation

For non-sequential models like logistic regression, we convert each 100×768 sequence to a fixed-size vector using Mean, Max, and Top-K Mean pooling, each in two variants—Remove-Zero (drop paddings) and Keep-Zero (retain them)—plus a [CLS] token proxy. The Figure 2.3 suggests the Remove-Zero strategies perform better than Keep-Zero ones on validation AUC under a stratified 80/20 split with a logistic-regression baseline. This confirms that the data cleaning of the Remove-Zero method can effectively filter noises. We retain Mean (Remove-Zero) as the default for non-sequential models due to its best performance and

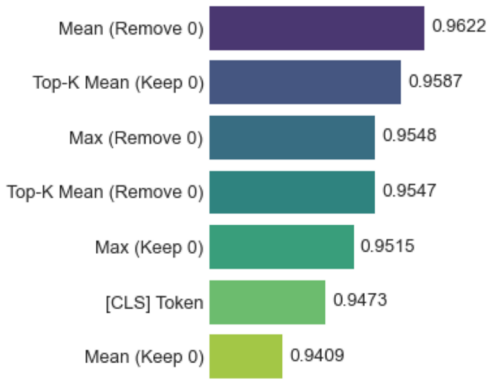

Figure 2.3. Validation AUC across pooling strategies.

2.3.2 Feature Scaling Strategy Evaluation

The logistic regression is sensitive to the input values, so we compare the impacts of different scaling methods after using mean pooling with data cleaning to preprocess training data. The methods include Standardization, Min–Max scaling, Robust scaling, and a no-scaler control within the same logistic-regression baseline, using scikit-learn implementations [2]. The results (Figure 2.4) are comparable across options, with Robust scaling yielding the highest validation AUC and most stable optimization. We therefore adopt Robust scaling for logistic regression models.

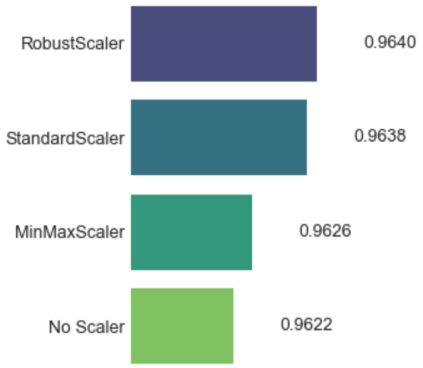

Figure 2.4. Validation AUC across feature-scaling strategies.

2.4. Data Visualization

We randomly take a subsample of 2000 sentences (1000 AI and 1000 Human) in training data to visualize the data. The data has been preprocessed by mean pooling with data cleaning and a robust scaler.

2.4.1. PCA projection

We project the embeddings onto the first two principal components using Principal Component Analysis (PCA) [1] in Figure 2.5. The axes annotate the fraction of variance explained by each component as computed by PCA. The projection reveals a curved manifold with substantial overlap between classes. We therefore treat PCA as a qualitative diagnostic rather than a decision tool, and train models on the full-dimensional features.

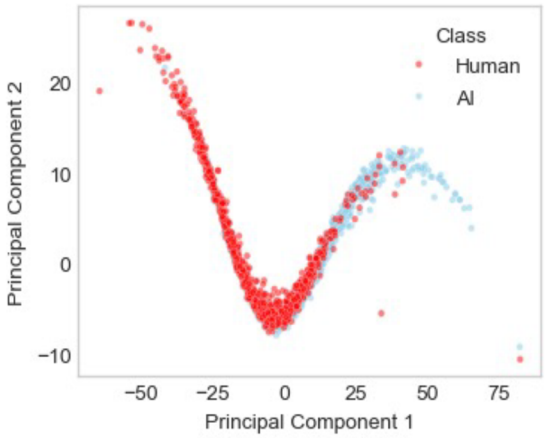

Figure 2.5. 2D PCA of mean-pooled robust-scaled embeddings.

2.4.2 Embedding magnitude (ℓ2 norm)

For each embedding we compute the ℓ2 norm and visualize class-wise distributions shown in Figure 2.6. The two distributions show modest but consistent shifts, indicating that global magnitude carries a weak yet usable discriminative signal that downstream models can leverage.

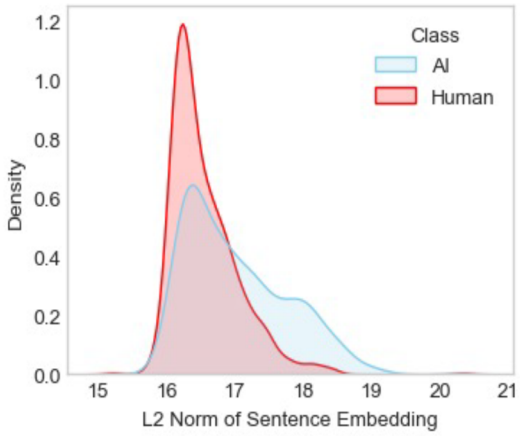

Figure 2.6. The distributions of l2 norms for mean-pooled robust-scaled embeddings.

2.5. Construction of an Efficient 3D Data Pipeline

To control memory footprint while streaming large arrays (the total training data size exceeding 5GB), we load NPY files with NumPy memory mapping and feed batches through tf.data.Dataset.from_generator [3]. This avoids loading the full dataset into RAM and mirrors the training/inference input format.

2.6. Training Data Splitting

For robust model tuning and evaluation, the full training corpus of 16,322 sentences was partitioned using an 80/20 Stratified Sampling strategy. This created an internal training set (13,058 samples) and an internal validation set (3,264 samples). Stratified sampling ensures that the class distribution (AI vs. Human) is preserved across both splits, preventing sampling bias.

3. Modelling and Hyperparameter Optimisation

3.1 Model Selection

The model selection follows a guidance of establishing from simple baselines to more complex and powerful architectures. We select five models to explore the problem, including logistic regression (LR), XGBoost, MLP, CNN and LSTM, which spans linear, decision-tree, and deep network architectures. This diversity is a key prerequisite for ensembling, as models with different inductive biases are likely to make uncorrelated errors.

In summary, the modelling strategy proceeds in three specific phases:

(1) Establish non-sequential (LR, MLP, XGBoost) on pooled data.
(2) Explore complex, sequence-aware architectures (CNN, LSTM) to directly leverage contextual information.
(3) Combine the best models by an ensemble approach to achieve a higher level of robustness and generalization performance (discussed in section 4.4).

In all experiments, we set the random seed as 42 for scikit-learn, TensorFlow and NumPy. For each model, the basic workflow is shown in Figure 3.1.

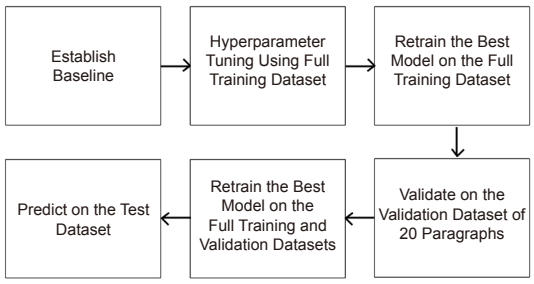

Figure 3.1. Modelling Workflow.

3.2 Paragraph Aggregation Strategy

One of the key challenges in this task is to make the paragraph-level prediction based on the sentence-level predictions. This is a critical step that directly impacts the final evaluation. To explore this, we propose and evaluate five different aggregation strategies for our models:

Mean: This strategy calculates the average of all sentence probabilities. The hypothesis is that AI text exhibits a consistent, distributed signal. This method is a strong, stable baseline that considers all probabilities equally.

Max: This strategy takes the highest sentence probability as the score for the entire paragraph. It hypothesizes that AI is likely to produce a sentence with a distinctly non-human, high-confidence signal.

Median: This strategy takes the median value of the sentence probabilities. It is to find the most typical sentence score, making it highly robust to outliers.

Standard Deviation: This strategy takes the variance of the sentence probabilities. The hypothesis is that a high variance might indicate an inconsistent writing style, which could be a potential signal of AI.

75th Percentile: This strategy finds the value below which 75% of the sentence probabilities fall. This is robust since it hypothesizes that a paragraph is AI if a significant portion of its sentences are classified as AI.

3.3 Non-sequential Models

3.3.1 Logistic Regression: The Baseline

We begin with LR as our fundamental baseline. As a linear model, LR provides a necessary performance benchmark and helps ascertain the linear separability in the feature space after pooling.

Baseline. We establish a LR pipeline including a RobustScaler. The hyper-parameter max iteration is set to 1000, while others are the default values. This baseline LR achieves the AUC of 0.9640 on the 20% validation sentences (Internal Val AUC).

Hyperparameter Tuning (Grid Search Cross Validation). We use GridSearchCV in scikit-learn to optimize the model. The best LR (Best LR) is the model with the C (inverse of regularization strength) of 0.1 and the 'liblinear' solver, and more details about hyperparameter space can be found in Appendix B. The Best LR achieves the Internal Val AUC of 0.9667.

External Validation. After retraining the LR Best on the full training dataset, we use the separate validation dataset to evaluate it. It achieves the sentence-level AUC of 0.7688. The paragraph-level AUC results using different aggregation strategies are shown in Table 3.1. Based on the results, we select the Median as the aggregation strategy for the LR Best model.

Strategy	Mean	Max	Median	Standard Deviation	75% Percentile
AUC	0.93	0.73	0.98	0.67	0.93

Table 3.1. Best LR AUC results using different aggregation strategy

Retrain the Best LR and Predict the Test Data. Lastly, we concatenate full training data and all sentences in the validation dataset to create a "big train" dataset, and retrain the LR Best on it. We then predict the 180 paragraphs of test data using the Median strategy, and the Kaggle public score is 0.94084, which serves as the baseline for our experiments.

3.3.2 XGBoost

We employ XGBoost, a state-of-the-art gradient boosting framework renowned for its performance in Kaggle competitions [4]. Its ensemble of decision trees is particularly effective at learning complex decision boundaries in high-dimensional feature spaces like our 768-dimensional embedding vectors.

Baseline. We set the objective as binary classification, evaluation metric as logarithmic loss function and early stopping rounds as 10 to prevent overfitting. This baseline XGBoost has the Internal Val AUC of 0.9409.

Hyperparameter Tuning (Random Search Cross Validation). We use the RandomizedSearchCV in scikit-learn to optimize the hyperparameters as it is more efficient in exploring large hyperparameter spaces (Appendix B). The Best XGB has the estimator number of 500, max depth of 6, learning rate of 0.2, training instance subsample ratio of 1.0, of column subsample ratio of 0.8 and gamma of 0, and it achieves the Internal Val AUC of 0.9585.

External Validation. The sentence-level AUC of the full-trained Best XGB is 0.7697. The paragraph-level AUC are shown in Table 3.2. We select the Median aggregation strategy for the Best XGB.

Strategy	Mean	Max	Median	Standard Deviation	75% Percentile
AUC	0.96	0.68	0.97	0.73	0.96

Table 3.2. Best XGB AUC results using different aggregation strategy

Retrain the Best XGB and Predict the Test Data. Best XGB predicts the test data using the Median strategy, and the Kaggle public score is 0.91852.

3.3.3 MLP

Based on the Universal Approximation Theorem [5] , a sufficiently large MLP can approximate any continuous function, making it a powerful tool for learning patterns from the pooled embeddings.

Baseline. We use a simple MLP of two hidden fully connected (FC) layers, which has 128 and 64 units respectively (Appendix A.) This baseline achieves the highest Internal Val AUC of 0.9622.

Hyperparameter Tuning (Random Search Single Holdout Validation). Considering the training time, we use the single holdout validation combined with KerasTuner to random search (Appendix B). The structure of the Best MLP is shown in Figure 3.2, which achieves the highest Internal Val AUC of 0.9638 at epoch 49.

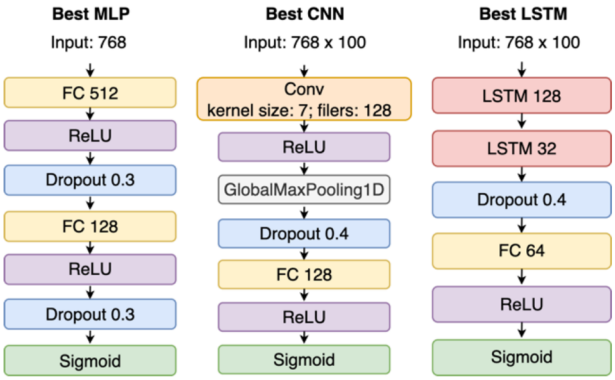

Figure 3.2. Best MLP, Best CNN and Best LSTM Architectures.

Validate on the Validation Dataset. The sentence-level AUC of the Best MLP is 0.7870. The paragraph-level AUC are shown in Table 3.3. We select the Mean aggregation strategy for the Best MLP, because it considers all probabilities equally and is robust and stable.

Strategy	Mean	Max	Median	Standard Deviation	75% Percentile
AUC	1.00	0.83	1.00	0.76	0.99

Table 3.3. Best MLP AUC results using different aggregation strategy

Retrain the LR Best and Predict on the Test Dataset. Best MLP predicts the test data using the Mean strategy, and the Kaggle public score is 0.94499.

3.4 Sequential Models

While pooled features are effective, they discard crucial sequential information such as word order and local context. Therefore, we implement sequence-aware models.

3.4.1 1D Convolutional Neural Network (CNN)

Inspired by the work on using CNNs for text classification [6], we design a 1D-CNN.

Baseline. The model employs one CNN layer of 128 filters with a kernel size of 5, which acts as detectors for 5-gram patterns (Appendix A). A GlobalMaxPooling1D layer identifies the most important features. It achieves the highest Internal Val AUC of 0.9646.

Hyperparameter Tuning (Random Search Single Holdout Validation). The random search details are in Appendix B. The structure of the Best CNN is shown in Figure 3.2, with the optimized learning rate of 0.0001, which achieves the highest Internal Val AUC of 0.9698 at epoch 42.

External Validation. The sentence-level AUC of the Best CNN is 0.7646. The paragraph-level AUC results are shown in Table 3.4. The Mean aggregation performs best.

Strategy	Mean	Max	Median	Standard Deviation	75% Percentile
AUC	0.97	0.88	0.95	0.88	0.96

Table 3.4. Best CNN AUC results using different aggregation strategy

Retrain the LR Best and Predict on the Test Dataset. Best CNN predicts the test data using the Mean strategy, and the Kaggle public score is 0.95433.

3.4.2 Long Short-Term Memory Network (LSTM)

To capture long-range dependencies, we implement LSTM networks. LSTM can learn contextual relationships across long sequences due to the gating mechanism (forget, input, and output gates).

Baseline. The baseline consists of one 128-unit LSTM layer (Appendix A), which can process the sequence and output a final hidden state vector representing the entire sentence's semantic content. It achieves the highest Internal Val AUC of 0.9697.

Hyperparameter Tuning (Random Search Single Holdout Validation). Random search details can be found in Appendix B. The structure of the Best LSTM is shown in Figure 3.2, which achieves the Internal Val AUC of 0.9683 at epoch 42 with the learning rate of 0.001.

External Validation. The sentence-level AUC of the Best CNN is 0.7658. The paragraph-level AUC results are shown in Table 3.5. Based on the results, we choose the Mean aggregation for LSTM.

Strategy	Mean	Max	Median	Standard Deviation	75% Percentile
AUC	0.95	0.88	0.92	0.88	0.93

Table 3.5. Best LSTM AUC results using different aggregation strategy

Retrain the LR Best and Predict on the Test Dataset. Best LSTM predicts the test data with the Mean strategy, resulting in the Kaggle public score of 0.96730.

4. Results

4.1 Evaluation set-up

We report the same signals used in the notebook so comparisons are consistent and reproducible.

Train AUC for optimisation feedback.

Internal Val AUC at the sentence level using an 80/20 split of the training corpus.

External Val AUC (sentence) on the separate validation set.

External Val AUC (paragraph) on the same set after mean aggregation of sentence probabilities by id, or the best aggregation rule found for that model.

Test AUC (public) on the competition public test set, the score is computed on 49% of the test set.

For non-sequential models, sentence features are 768-d vectors produced by Mean pooling with data cleaning. For paragraph decisions we evaluated several sentence-to-paragraph rules during development, then locked a single rule per model before generating test predictions.

4.2 Single-model performance and efficiency

Each model's performance and training time are shown in Table 4.1. Internal Val AUC is computed on a sentence split and runs higher than the external signals, which is expected for this protocol. The ordering of models is stable across metrics. Among pooled baselines, MLP improves over LR and XGBoost on both internal and external validation, confirming that moderate nonlinearity helps on 768-d embeddings. Sequence-aware models that ingest the full 100×768 tensors transfer better to the public test. LSTM is the strongest single model on Test AUC and remains competitive on the external set.

Model	Internal Val AUC (sent.)	External Val AUC (sent.)	External Val AUC (para.)	Aggregation Strategy	Test AUC (public)	Training time*
Logistic Regression	0.9667	0.7688	0.9800	Median	0.94084	4.6s
XGBoost	0.9585	0.7697	0.9700	Median	0.91852	18.6s
MLP	0.9638	0.7870	1.0000	Mean	0.94499	1m20s for 49 epochs
1D-CNN	0.9698	0.7646	0.9700	Mean	0.95433	20m07s for 42 epochs
LSTM	0.9683	0.7658	0.9500	Mean	0.96730	46m23s for 42 epochs

Table 4.1. Single-Model Performance Summary: Validation/Test AUCs, Aggregation Rule, and Training Time (*Times are from the notebook for model development runs on Environment: MacBook Air (Apple M1, 8 GB unified memory), using the stated number of epochs and random seed of 42.)

From the Figure 4.1, inference and training cost increase from LR and XGBoost to CNN and LSTM. This growth is directly due to the dramatic increase in model complexity, the growing number of parameters that need to be learned.

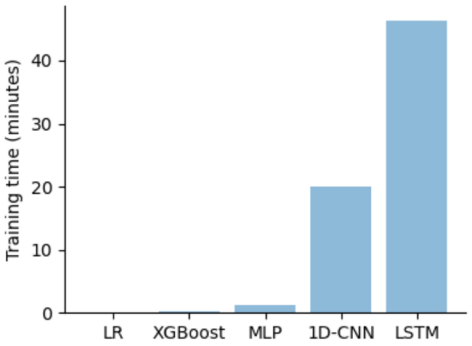

Figure 4.1. Training time for single models.

4.3 Paragraph aggregation study

We compared five sentence-to-paragraph rules on the external validation paragraphs. The best rule per model is shown in Table 4.2.

Aggregation	LR	XGB	MLP	CNN	LSTM
Mean	0.93	0.96	**1.00**	**0.97**	**0.95**
Max	0.73	0.68	0.83	0.88	0.88
Median	**0.98**	**0.97**	1.00	0.95	0.92
Std	0.67	0.73	0.76	0.88	0.88
75th	0.93	0.96	0.99	0.96	0.93

Table 4.2. External Val AUC (paragraph) by aggregation (locked choices bolded).

Why the aggregation choice is locked. Table 4.2 and Figure 4.2 report the full comparison of sentence-to-paragraph rules across models on the external validation set. Median is the most reliable rule for LR and XGB, while Mean consistently dominates for MLP, CNN, and LSTM. The locked choices reflect these outcomes and are highlighted in the table. Given the small evaluation set (20 paragraphs), 95% confidence intervals are shown in Figure 4.2 to communicate uncertainty.

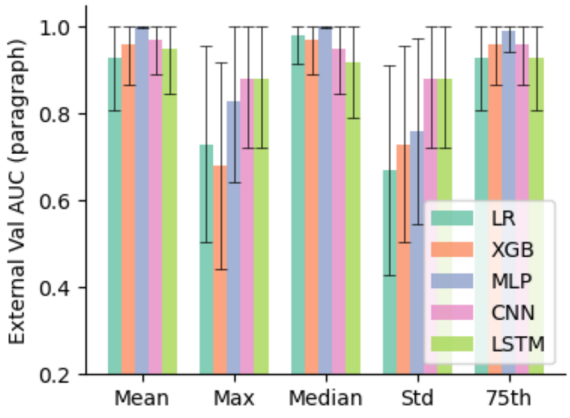

Figure 4.2. Aggregation strategy comparison on External validation paragraphs (95% Confidence Interval via Hanley-McNeil, 10 human / 10 AI)

Small-sample caveat. The external paragraph set contains 20 items (10 human, 10 AI). The MLP's AUC of 1.00 falls within a wide interval for this sample size and should be interpreted accordingly.

4.4 Ensembles

Each base model first produces sentence probabilities. One paragraph score per model is computed using the locked aggregation rule. Model paragraph scores are then combined with equal weights. The ensemble configurations and Test AUC are shown in Table 4.3.

Ensemble configuration	Combination rule across models	Test AUC (public)
LR + MLP + CNN + LSTM	Median of model paragraph scores	0.95796
MLP + CNN + LSTM	Mean of model paragraph scores	0.95692
CNN + LSTM	Mean of model paragraph scores	**0.95848**

Table 4.3. Ensemble configuration details and Test AUC.

Intuitively (Figure 4.3), including weaker members reduces the benefit of ensembling. The four-model median is robust to outliers yet trails sequence-only ensembles because LR contributes higher error. Dropping LR and averaging MLP + CNN + LSTM improves over the four-model median, indicating that ensembling helps when all members are competitive. The CNN + LSTM mean is the strongest ensemble at 0.95848, CNN captures local patterns and LSTM captures longer context, providing complementary inductive biases. The top ensemble remains below the single LSTM public score of 0.96730, indicating residual error correlation among sequence models and the limits of equal weighting when one model dominates.

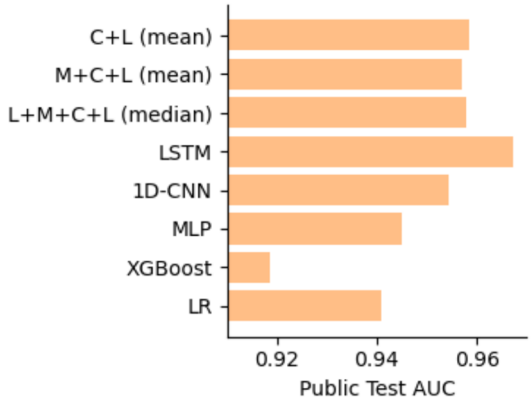

Figure 4.3. Public Test AUC for single models and ensembles

4.5 Cross-metric view and model selection under a 49% public split

Because the public leaderboard covers 49% of the test set, selection is made to balance stability and peak score.

Primary system: two-model sequence ensemble (CNN + LSTM, mean across model paragraph scores) - best ensemble on the public split, complementary biases, and improved stability across paragraph length and style.

Secondary system: single LSTM - highest public AUC among all models, retained when the absolute best headline score is required.

4.6 Trial-and-error that changed outcomes

Feature handling for pooled models. Token Mean Pooling (drop zero paddings) with robust scaling produced stable LR and MLP baselines and removed sensitivity to outliers.

Leveraging sequence context. Moving from pooled features to CNN and LSTM improved external and public behaviour, confirming that order and local context are important.

Fixing the paragraph rule. Locking Mean Aggregation of Sentence Probabilities for sequence models eliminated volatility from max and median on shorter paragraphs and simplified test-time logic.

Choosing ensemble members. Equal weights help only when members are both diverse and individually strong. Removing LR improved ensemble performance. Pairing CNN with LSTM produced the best configuration among those tested.

4.7 Final training and submission path

For the final submissions each selected model was retrained on the full training data plus the separate validation data to maximise effective sample size.
Primary submission: 'submission_ensemble_2models_mean.csv' (CNN + LSTM mean on paragraph scores).
Secondary submission: 'lstm_best.csv' ('LSTM only).

5. Reflection and Future Work

Risk	Mitigation (Action)	Private Leaderboard Effect
Split integrity	Deduplicate in training. Group similar sentences with cosine similarity. Use grouped K-fold cross-validation with class balance. Run fixed-seed cross-validation and report mean with confidence interval. Use External Val AUC as the tiebreaker	Lowers leakage optimism and improves stability on the hidden 51%
Small external set	Report confidence intervals. Use bootstrap when needed. Avoid decisions based on small deltas	Reduces over-confidence from a tiny sample
Aggregation sensitivity	Stress-test across length/ sentences; consider compact learned pooling/attention only if sensitivity confirmed	Mitigates length-profile shift
Ensemble correlation and calibration	Stacking on out-of-fold predictions; Platt/isotonic calibration; retune threshold	Reduces ranking flips from shared errors or calibration drift
Sequence model stability - LSTM can overfit without controls	Use grouped early stopping and light regularisation. Keep seeds fixed and logs reproducible	Better private-split transfer

6. Team Collaboration

Member	Contributions
Ziqi Hao	Non-sequential model code, code integration and fixes, Modelling section
Hanzhi Zhang	Problem Understanding and Data Exploration section, final typesetting
Siying Ren	Data Exploration and Collaboration section, Cross-section QA/synthesis
Deepin Kuchroo	Sequential model code, Results and Reflection section, progress manager
Christopher Nugent	Initial data exploration and logistic regression code, Results section

7. References

[1] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A, 374(2065):20150202, 2016.

[2] F. Pedregosa et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[3] M. Abadi et al. TensorFlow: A System for Large-Scale Machine Learning. In OSDI, 2016.

[4] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD), pages 785–794, 2016.

[5] Wikipedia contributors. Universal approximation theorem. Wikipedia. Accessed August 2025.

[6] Y. Kim. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, 2014.

Appendix A: Baseline Models

All the following baseline models are trained with the max epochs of 50 and batch size of 32. The loss function is the binary cross-entropy for this binary classification task. During the training, we use the Internal Val AUC to monitor overfitting with the patience of 10, and the optimizer of Adam with the learning rate of 0.001. We use the ReLU activation after hidden layers, and Sigmoid layer as output.

1 MLP
The dropout layers after each FC layer have the rate of 0.3.

2 CNN
The dropout layer after global max pooling is 0.5 followed by a FC layer of 64 units.

3 LSTM
The dropout layer after the LSTM layer is 0.5 followed by a FC layer of 64 units.

Appendix B: Hyperparameter Spaces and Search Strategies

The MLP, CNN and LSTM use single holdout validation during random search. They are trained with the max epochs of 50 and batch size of 32. Besides, each parameter setting only runs once. The loss function is the binary cross-entropy. We use the Internal Val AUC to monitor overfitting with the patience of 10, and the optimizer of Adam with learning rate of 0.001. We use the ReLU activation after hidden layers, and Sigmoid layer as output.

1 LR
The C is from 0.1, 1, 10, 100 and solver from 'liblinear', 'saga' and 'lbfgs'. We use the GridSearchCV in scikit-learn.

2 XGBoost
The number of estimators: 100, 200, 300, 500. The max depth: 3∼7. The learning rate: 0.01, 0.05, 0.1, 0.2. The training instance subsample ratio: 0.7, 0.8, 0.9, 1.0. The column subsample ratio: 0.7, 0.8, 0.9, 1.0. The gamma: 0, 0.1, 0.2. We use the RandomizedSearchCV in scikit-learn.

3 MLP
The number of hidden layers ranges from 1 to 4, and the number of units chosen in each layer are [256, 512], [128, 256], [64, 128] and [32, 64]. The dropout rate is selected from 0.1, 0.3, 0.5.

4 CNN
The number of units in the 1D-Conv layer: 64, 128, 256. The kernel size: 3, 5, 7. The dropout rate: 0.2, 0.3, 0.4, 0.5. The number of units in the FC layer: 32, 64, 128. The learning rate: 0.01, 0.001, 0.0001.

5 LSTM
The number of LSTM layers: 1, 2. Bidirectional: True, False. The number of units in the 1st LSTM layer: 64, 128, 256, and in the 2nd LSTM layer: 32, 64, 128. The dropout rate: 0.2, 0.3, 0.4, 0.5. The number of units in the FC layer: 32, 64, 128. The learning rate: 0.001, 0.0001.