BBC News Articles

GitHub Repo:
https://github.com/Ainedembe-Denis/BBC-News-Summary-ML-DL-Project

AINEDEMBE DENIS

Reg. No: 2024-M132-2399

Master. Information Systems / Intelligent Systems

Primary Objectives

To design and evaluate an intelligent media monitoring system that classifies news articles, discovers latent topics, and uses a reinforcement learning agent to decide whether to apply machine learning, deep learning, or human escalation.

Specific Ojectives

Automatically classify news articles into predefined categori

es.

Discover latent topics through unsupervised clustering.

Intelligently decide when to use automated models vs. human review.

About the Dataset

Contains 2,225 BBC news articles published between 2004–2005 . Articles are categorized into five topics: Business, Entertainment, Politics, Sport, and Technology

Originally developed for text categorization and clustering research . A subset includes 417 political news articles with five human-written extractive summaries per article

Extractive summaries are created by selecting important sentences directly from the original text. This involves:

- Condensing large amounts of information into concise forms.
- Selecting important information while discarding redundant content.
- Using exact sentences from documents as summaries (extractive approach).

Dataset is unsupervised, language-independent, and widely used in NLP research

Machine Learning Domains Covered

This project demonstrates integration of multiple ML domains:

Supervised Learning

Classification using Logistic Regression and CNN. Supervised learning provides accurate classification.

Unsupervised Learning

KMeans clustering for topic discovery. Unsupervised learning discovers patterns without labels.

Reinforcement Learning

Q-learning for adaptive decision-making. RL enables adaptive, context-aware decision-making

Natural Language Processing

Text preprocessing, TF-IDF extraction, sequence tokenization.

Real-World Application

Media Monitoring Industry

Media monitoring companies process thousands of articles daily:

1. News agencies need to categorize content quickly.
2. Content platforms require automatic tagging.
3. Research organizations track topics across sources.
4. PR firms monitor brand mentions and sentiment.

Real-World Application —- Cont’ d

Automated News Categorization

Use Case: News aggregators (Google News, Apple News).

- Automatically categorize incoming articles.
- Route content to appropriate sections.
- Improve user experience with better organization.

Impact:

- Reduces manual categorization workload by 99%.
- Enables real-time content organization.
- Scales to handle millions of articles.

Real-World Application —- Cont’ d

Quality Control Systems

Use Case: Content moderation platforms.

1. Identify articles that need human verification.
2. Flag potentially misclassified content.
3. Ensure quality standards.

Part A - Data Mining and Preprocessing

Loading the Dataset

Successfully loaded 2,225 news articles from structured directory.

Processed articles from 5 category folders.

Created DataFrame with category, text, and filename columns.

The dataset shows a well-balanced distribution across 5 categories: Business: 510 articles (22.9%), Sport: 511 articles (23.0%), Politics: 417 articles (18.7%), Tech: 401 articles (18.0%), Entertainment: 386 articles (17.3%).

Finding: The balanced distribution (ranging from 386 to 511 articles per category) ensures fair training without class imbalance issues.

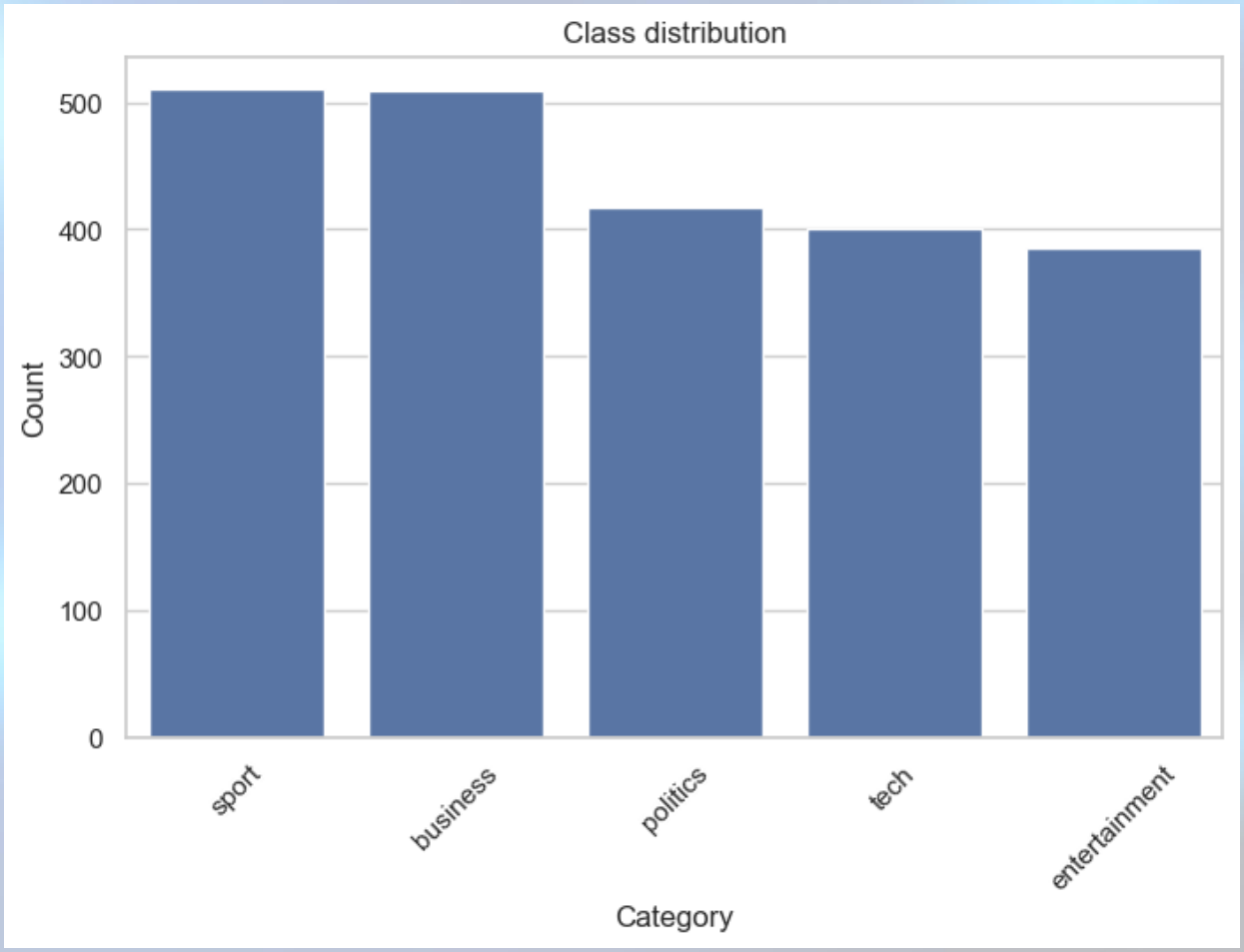
Visualization: Created bar chart showing class distribution, confirming the balanced nature of the dataset.

Dataset: Display examples and class distribution

Created bar chart showing class distribution, confirming the balanced nature of the dataset.

	category	text	filename
414	business	UK house prices dip in November\n\nUK house pr...	415.txt
420	business	LSE 'sets date for takeover deal'\n\nThe Londo...	421.txt
1644	sport	Harinordoquy suffers France axe\n\nNumber eigh...	332.txt
416	business	Barclays shares up on merger talk\n\nShares in...	417.txt
1232	politics	Campaign 'cold calls' questioned\n\nLabour and...	337.txt

```
Categories distribution:
category
sport            511
business         510
politics         417
tech             401
entertainment    386
Name: count, dtype: int64
```



Text Preprocessing

I implemented a clean_text() function that performs the following cleaning steps:

- Convert to lowercase - Standardizes all text (e.g., "The" → "the").
- Remove HTML tags - Strips HTML formatting using regex.
- Remove digits - Eliminates numeric content.
- Remove punctuation - Removes all punctuation marks.
- Normalize whitespace - Collapses multiple spaces into single spaces.

Successfully cleaned all 2,225 articles, creating a clean_text column with standardized text ready for feature extraction.

This preprocessing step is critical because raw text contains noise that can confuse models.

TF-IDF Feature Preparation & tokenize sequences for DL.

Created two different feature representations to support both ML and DL approaches:

TF-IDF Features (for ML)

5,000 features with bigrams (word pairs), English stop words removed.

- Created sparse matrix with shape (2225, 5000).
- TF-IDF captures word importance relative to document and corpus. Bigrams capture important word pairs like "prime minister" or "stock market" that are discriminative for classification.

Tokenized Sequences (for DL)

Vocabulary size 10,000, max sequence length 300 tokens.

- Full vocabulary: 31,519 unique words discovered.
- Padded sequences: Shape (2225, 300).

Preserves word order, which is important for deep learning models. Padding ensures all sequences have the same length for batch processing.

Train/Test Split:

Split

80/20 stratified split.

Results

- Training set: 1,780 articles.
- Test set: 445 articles.

Rationale

The 80/20 split provides sufficient training data while reserving adequate test data for evaluation.

Part B - Two News Classifiers

Classical ML Model: Logistic Regression

Implementation: I trained a Logistic Regression model with the following configuration:

Algorithm

Logistic Regression with TF-IDF features.

Max iterations

2000 (ensures convergence).

Multi-class classification

One-vs-rest approach.

Training time

Approximately 2 seconds.

Why I Chose Logistic Regression:

Simple, interpretable linear model, Fast training and inference, Works well with high-dimensional sparse features (TF-IDF), Provides probability estimates (confidence scores) needed for RL agent.

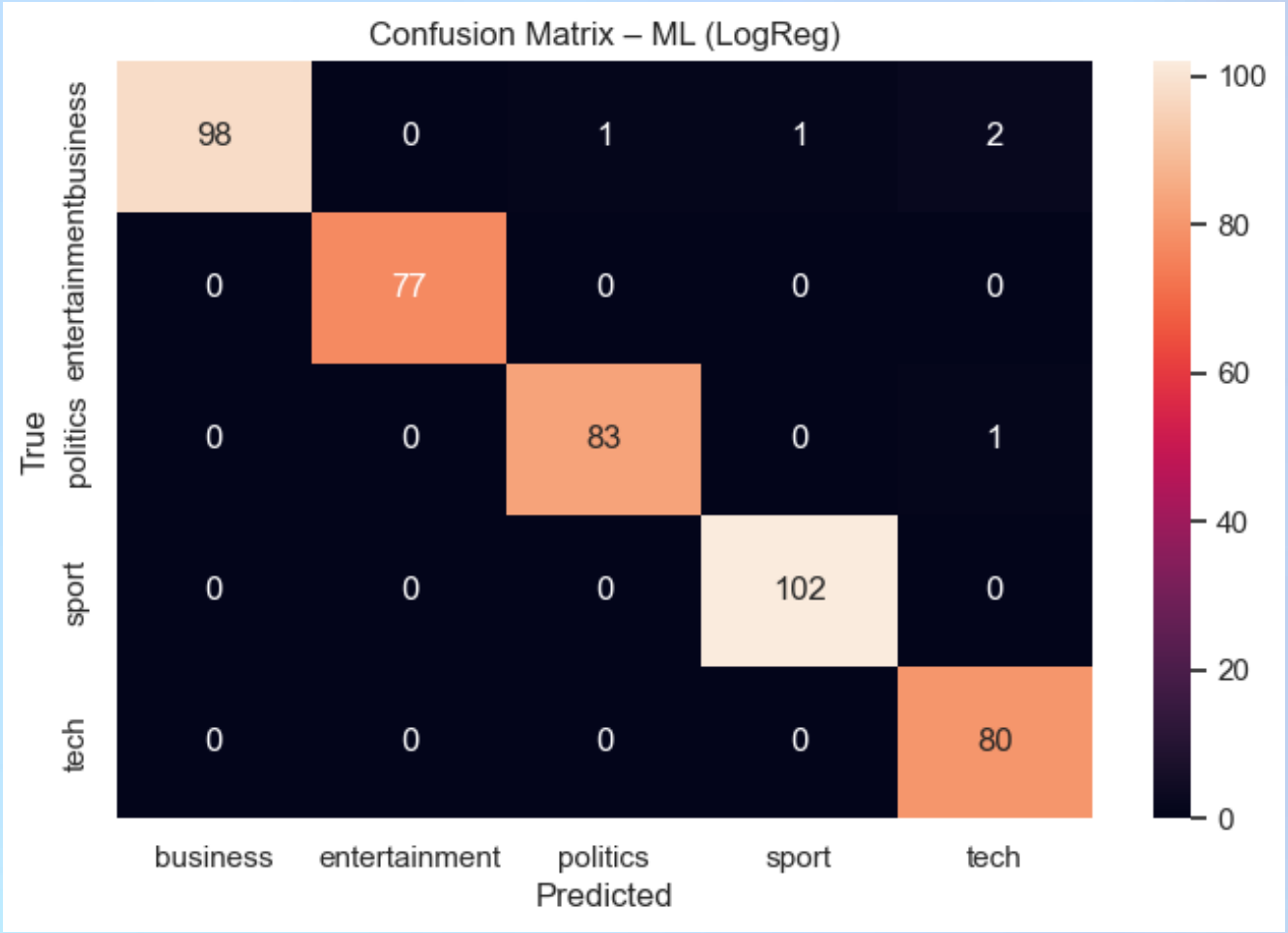
Classical ML Model: Logistic Regression — Cont’ d

Classical ML Model: Logistic Regression

	precision	recall	f1-score	support
business	1.00	0.96	0.98	102
entertainment	1.00	1.00	1.00	77
politics	0.99	0.99	0.99	84
sport	0.99	1.00	1.00	102
tech	0.96	1.00	0.98	80
accuracy			0.99	445
macro avg	0.99	0.99	0.99	445
weighted avg	0.99	0.99	0.99	445

F1 Scores per Class

- Business: 0.98.
- Entertainment: 1.00 (perfect classification).
- Politics: 0.99.
- Sport: 1.00 (perfect classification).
- Tech: 0.98.



Overall Accuracy: 98.88% - Exceptional performance achieved; Misclassifications: Only 5 out of 445 test samples (1.12% error rate).
Macro Average F1: 0.99.

```
Total misclassified (ML): 5


________________________________________________________________________________
True: business
Pred: tech
Text: card fraudsters targeting web new safeguards on credit and debit card payments in shops has led frau...
________________________________________________________________________________
True: politics
Pred: tech
Text: uk firms embracing ecommerce uk firms are embracing internet trading opportunities as never before e...
________________________________________________________________________________
True: business
Pred: politics
Text: golden rule intact says exaide chancellor gordon brown will meet his golden economic rule with a mar...
________________________________________________________________________________
True: business
Pred: tech
Text: bt offers equal access to rivals bt has moved to preempt a possible breakup of its business by offer...
________________________________________________________________________________
True: business
Pred: sport
Text: arsenal may seek full share listing arsenal vicechairman david dein has said the club may consider s...
```

- 3 Business articles misclassified as Tech (articles about tech companies, sports).
- 1 Politics article misclassified as Tech (technology policy).
- 1 Business article misclassified as Politics (economic policy).

Key Finding: The excellent performance with a simple linear model demonstrates that for this task, linear relationships in TF-IDF space are sufficient. News categories have distinct vocabularies (e.g., "goal" → sport, "election" → politics), making them linearly separable.

Training Deep Learning Model: CNN

Architecture Designed:

Embedding layer

10,000 vocabulary → 100-dimensional dense vectors.

Conv1D layer

128 filters, kernel size 5 (detects 5-word patterns).

GlobalMaxPooling1D

Extracts most important features from each filter.

Dense layers

64 units with Dropout (0.5) to prevent overfitting.

Output layer

5-class softmax for probability distribution.

Training Configuration:

Epochs: 10.

Batch size:32.

Validation split: 20%.

Optimizer: Adam.

Loss function: Categorical cross-entropy.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 300, 100)	1,000,000
conv1d_1 (Conv1D)	(None, 296, 128)	64,128
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 5)	325

Total params: 1,072,709 (4.09 MB)

Trainable params: 1,072,709 (4.09 MB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 12s 108ms/step - accuracy: 0.2310 - loss: 1.6051 - val_accuracy: 0.1938 - val_loss: 1.5883
Epoch 2/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 5s 110ms/step - accuracy: 0.3462 - loss: 1.5263 - val_accuracy: 0.4663 - val_loss: 1.4629
Epoch 3/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 81ms/step - accuracy: 0.5042 - loss: 1.3071 - val_accuracy: 0.5983 - val_loss: 1.0643
Epoch 4/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 81ms/step - accuracy: 0.6882 - loss: 0.9259 - val_accuracy: 0.7781 - val_loss: 0.7033
Epoch 5/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 79ms/step - accuracy: 0.8287 - loss: 0.5990 - val_accuracy: 0.8792 - val_loss: 0.4759
Epoch 6/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 84ms/step - accuracy: 0.8940 - loss: 0.3773 - val_accuracy: 0.9045 - val_loss: 0.3510
Epoch 7/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 82ms/step - accuracy: 0.9347 - loss: 0.2430 - val_accuracy: 0.9185 - val_loss: 0.2807
Epoch 8/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 86ms/step - accuracy: 0.9600 - loss: 0.1580 - val_accuracy: 0.9298 - val_loss: 0.2428
Epoch 9/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 83ms/step - accuracy: 0.9726 - loss: 0.1290 - val_accuracy: 0.9270 - val_loss: 0.2258
Epoch 10/10
45/45 ━━━━━━━━━━━━━━━━━━━━ 4s 87ms/step - accuracy: 0.9881 - loss: 0.0704 - val_accuracy: 0.9298 - val_loss: 0.2143
```

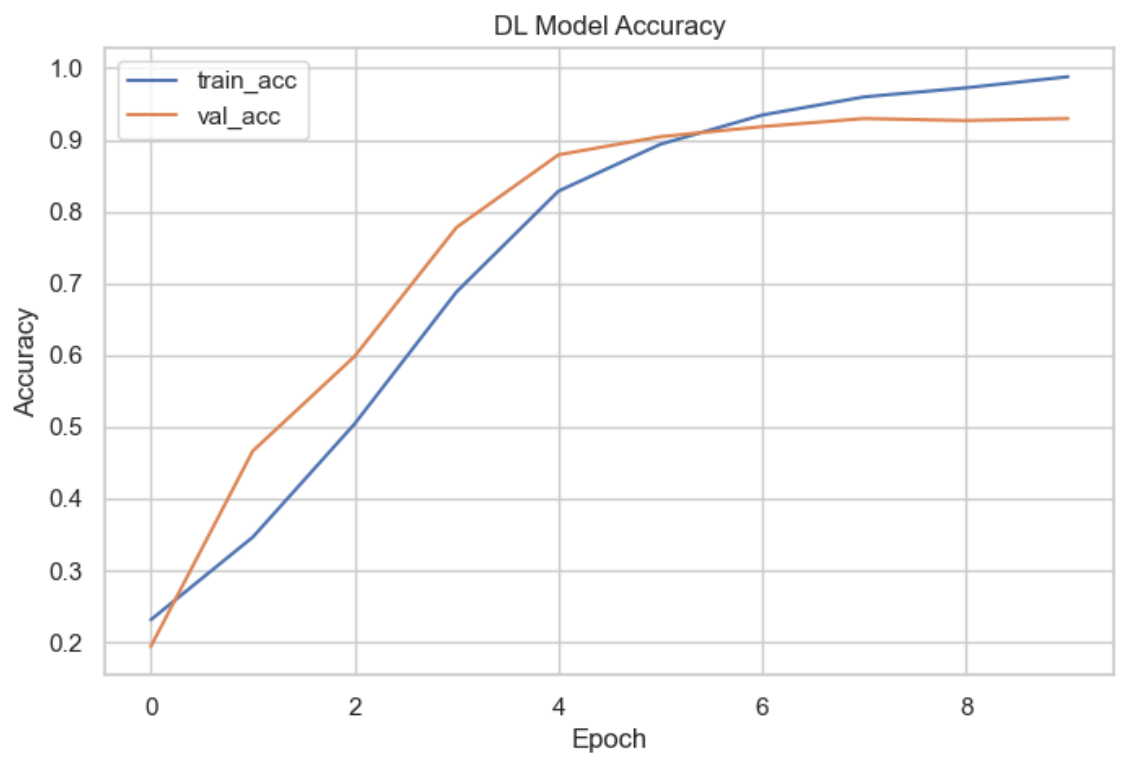
Training Results:
I observed the following learning progression:

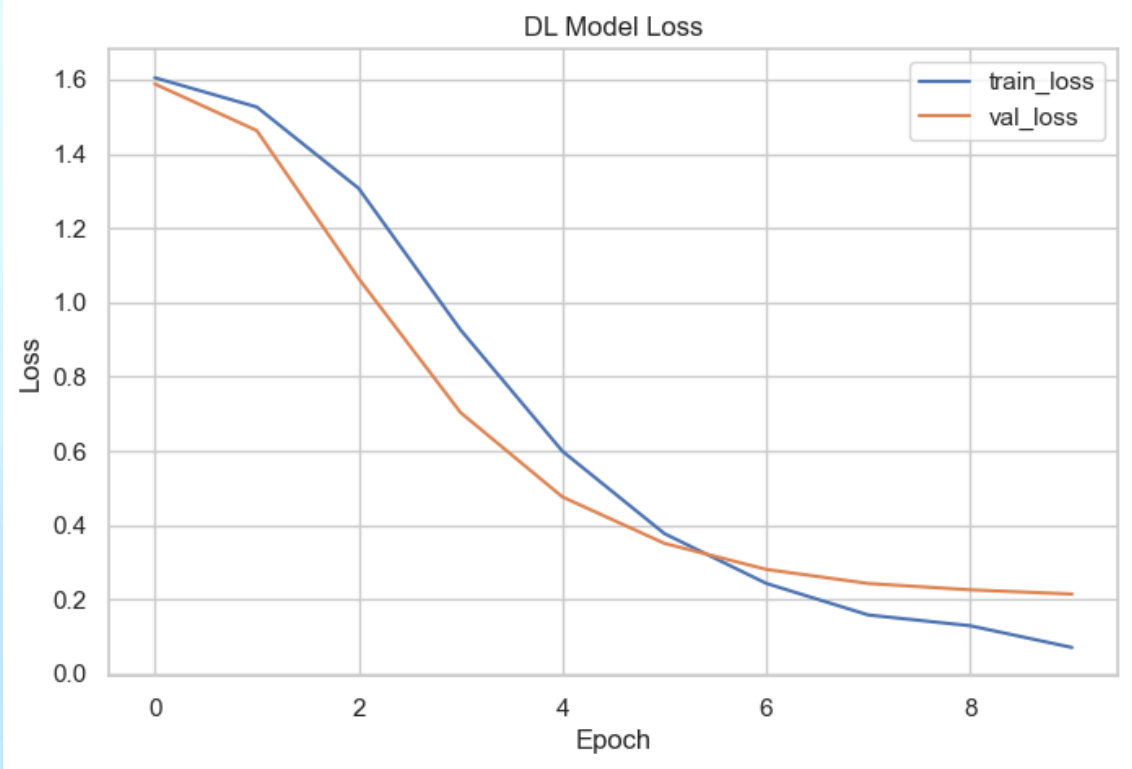
Epoch 1
Training accuracy 23.10%,
Validation accuracy 19.38%.

Epoch 5
Training accuracy 82.87%,
Validation accuracy 87.08%.

Epoch 10
Training accuracy 98.81%,
Validation accuracy 92.98%.

Train the DL model + training curves



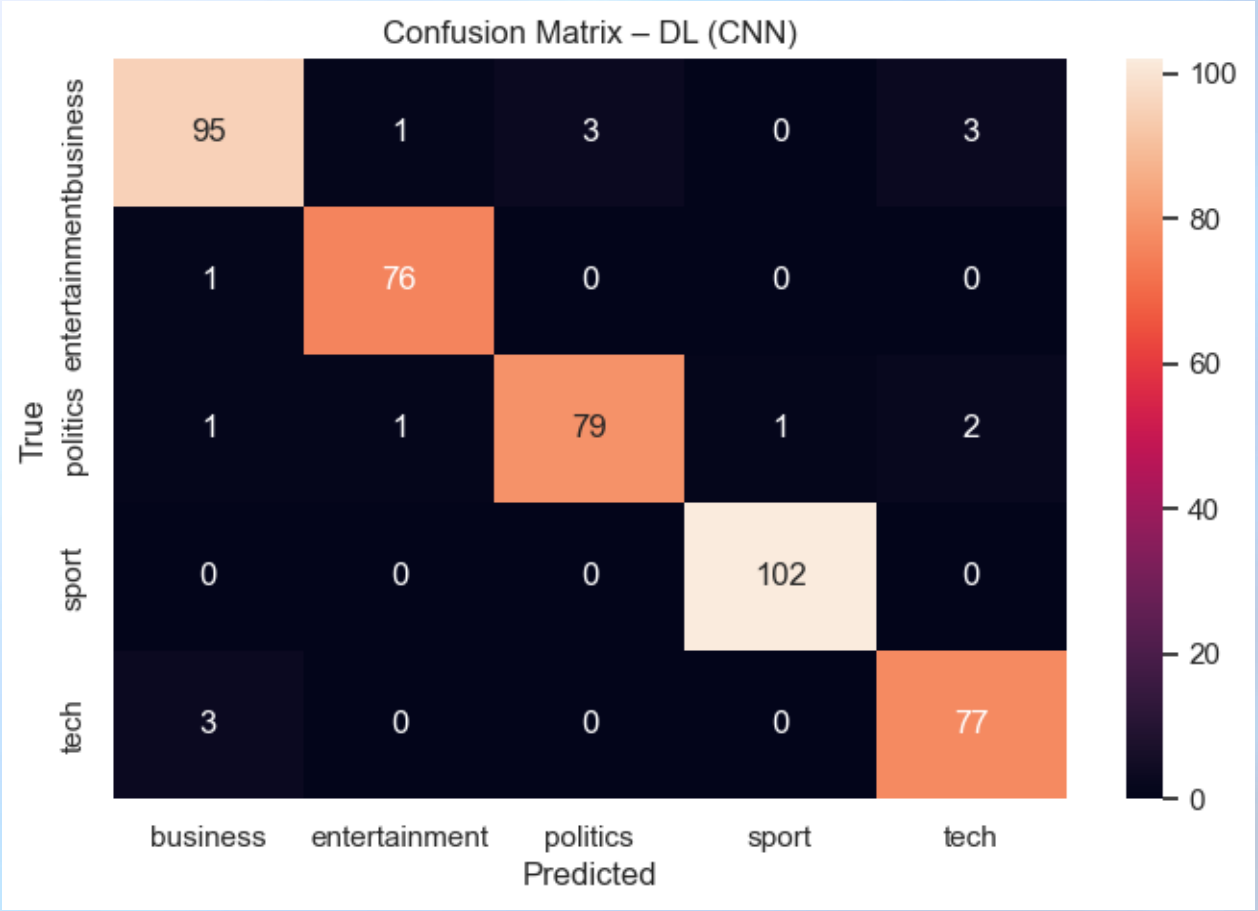


Evaluate DL model + F1 + confusion matrix + misclassified

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step
Deep Learning Model (CNN)
              precision    recall  f1-score   support

    business       0.95      0.93      0.94       102
entertainment       0.97      0.99      0.98        77
    politics       0.96      0.94      0.95        84
       sport       0.99      1.00      1.00       102
        tech       0.94      0.96      0.95        80

    accuracy                           0.96       445
   macro avg       0.96      0.96      0.96       445
weighted avg       0.96      0.96      0.96       445
```



Overall Accuracy
96.40%.

Macro Average F1
0.96.

F1 Scores per Class

- Business: 0.94.
- Entertainment: 0.98.
- Politics: 0.95.
- Sport: 1.00 (perfect classification).
- Tech: 0.95.

Evaluate DL model + F1 + confusion matrix + misclassified

```
Total misclassified (DL): 16
_____________________________________________________________________________________________
True: tech
Pred: business
Text snippet: New Year's texting breaks record

A mobile phone was as essential to the recent New Year's festivities as a party mood and Auld Lang Syne, if the numb...
_____________________________________________________________________________________________
True: tech
Pred: business
Text snippet: Junk e-mails on relentless rise

Spam traffic is up by 40%, putting the total amount of e-mail that is junk up to an astonishing 90%.

The figures, fr...
_____________________________________________________________________________________________
True: politics
Pred: tech
Text snippet: US casino 'tricks' face ban in UK

Controversial new UK casinos will be banned from using American tricks of the trade to ensure they are "socially re...
_____________________________________________________________________________________________
True: business
Pred: politics
Text snippet: McDonald's boss Bell dies aged 44

Charlie Bell, the straight-talking former head of fast-food giant McDonald's, has died of cancer aged 44.

Mr Bell ...
_____________________________________________________________________________________________
True: entertainment
Pred: business
Text snippet: Campaigners attack MTV 'sleaze'

MTV has been criticised for "incessant sleaze" by television indecency campaigners in the US.

The Parents Television...
```

Misclassifications

16 out of 445 test samples (3.60% error rate).

Analysis:

- The CNN model achieved very good performance (96.40%), though slightly lower than the ML model (98.88%).
- The model successfully learned from sequences, with validation accuracy plateauing around epoch 7-8.
- The lower performance compared to ML suggests that for this dataset size and task complexity, the linear model's simplicity is sufficient.

Performance Comparison

Metric	ML (LogReg)	DL (CNN)
Accuracy	98.88%	96.40%
Macro Avg F1	0.99	0.96
Misclassified	5	16
Training Time	\~2 seconds	\~40 seconds
Inference Speed	Very fast	Moderate

Key Findings:
ML model outperformed DL model by 2.48 percentage points - This was a surprising result that demonstrates linear relationships are sufficient for this task.

Both models achieve excellent results (>96% accuracy) - Either model could be deployed in production.

Misclassification Analysis

ML Model Errors (5 total):
I analysed the 5 misclassified articles from the ML model
- 3 Business: Tech Articles about tech companies, digital business, or technology in business context.
- 1 Politics: Tech Article about technology policy or digital governance.
- 1 Business: Politics Article about economic policy or business regulations.

DL Model Errors (16 total):
The DL model had more misclassifications, including:
- Tech articles misclassified as Business (multiple cases).
- Politics articles misclassified as Tech.
- Business articles misclassified as Politics.

Finding: Some articles have overlapping themes (e.g., business-tech, business-politics), making classification challenging even for well-trained models.

Why These Errors Occur:
- Domain Overlap: Real-world articles often span multiple categories.
- Vocabulary Overlap: Terms like "market," "policy," "digital" appear in multiple categories.
- Context Dependency: Classification depends on overall context, not just keywords.

Implication for RL: These ambiguous cases are exactly where my RL agent can help by escalating to human review when both models disagree or have low confidence.

Part C - Topic clustering with TF-IDF + KMeans

Approach Used

Embeddings

TF-IDF features (reused from Part A) - High-dimensional sparse vectors (5000 features).

Algorithm

KMeans clustering - Unsupervised learning approach.

Number of clusters

5 (matching number of categories) - Allows comparison with labeled categories.

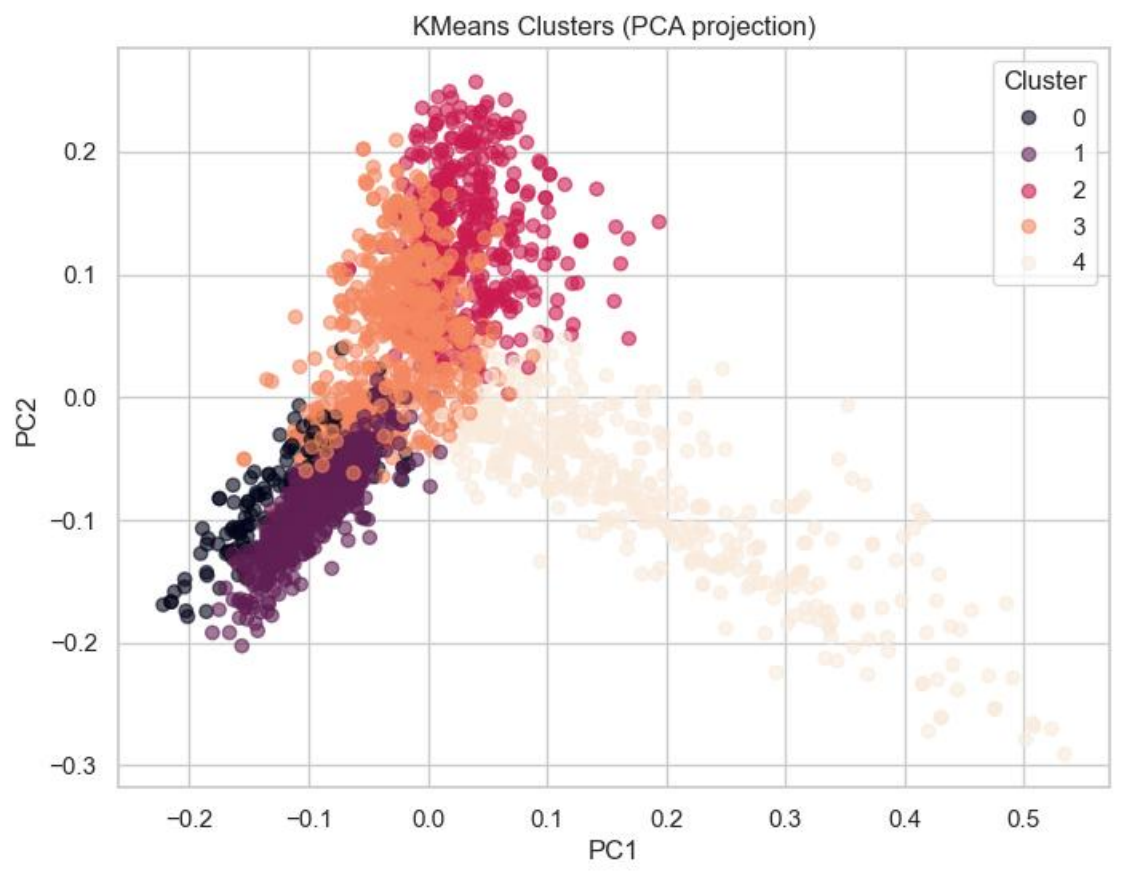
Visualization

PCA (2D projection) - Reduced 5000 dimensions to 2 for visualization.

Implementation: I applied KMeans clustering to the full dataset (2,225 articles) using the TF-IDF representation. The algorithm discovered 5 clusters without using any category labels.

```
Cluster 0 top keywords: film, best, awards, award, films, actor, festival, actress, oscar, won
Cluster 1 top keywords: game, said, win, england, cup, match, players, world, injury, play
Cluster 2 top keywords: bn, said, growth, economy, year, market, bank, oil, sales, shares
Cluster 3 top keywords: said, music, people, mobile, new, technology, users, software, games, tv
Cluster 4 top keywords: mr, said, labour, election, blair, party, government, brown, mr blair, minister
```

```
Cluster 0 top keywords: film, best, awards, award, films, actor, festival, actress, oscar, won
Cluster 1 top keywords: game, said, win, england, cup, match, players, world, injury, play
Cluster 2 top keywords: bn, said, growth, economy, year, market, bank, oil, sales, shares
Cluster 3 top keywords: said, music, people, mobile, new, technology, users, software, games, tv
Cluster 4 top keywords: mr, said, labour, election, blair, party, government, brown, mr blair, minister
```



Key Finding: The clustering is unsupervised learning - I didn't use category labels. The algorithm discovered patterns based solely on article content.

The fact that clusters align with categories validates that:

- Articles in the same category are naturally similar.
- The TF-IDF representation captures meaningful semantic information.
- The clustering approach is effective for topic discovery.

Part D - RL Decision Agent

Build RL environment data / Create state components

Here, created state components:

- ML confidence (max prob)
- DL confidence (max prob)
- Cluster id
- Article length (binned)
- Disagreement flag (ML vs DL prediction differ)

```
RL State Components Created:
  - ML max confidence shape: (445,), range: [0.258, 0.983]
  - DL max confidence shape: (445,), range: [0.310, 1.000]
  - Cluster IDs shape: (445,), unique clusters: [0 1 2 3 4]
  - Length bins shape: (445,), distribution: [  1  49 395]
  - Disagreement flag shape: (445,), disagreements: 13 (2.9%)
  - True labels shape: (445,)

Total samples for RL environment: 445
```

Q-learning setup

We discretize ML & DL confidence into 5 bins and build a tabular Q-table.

- Approx number of states: 750
- Q-table initialized with shape: (750, 3)

Build RL environment data / Create state components

5 State Components were Designed:

ML Confidence (max probability) - Binned into 5 levels.

- Range Observed: [0.258, 0.983] from my test set.
- Bins: [0-0.2), [0.2-0.4), [0.4-0.6), [0.6-0.8), [0.8-1.0].

Indicates how certain the ML model is about its prediction.

DL Confidence (max probability) - Binned into 5 levels.

- Range Observed: [0.310, 1.000] from my test set.
- Same bins as ML for consistency.

Allows comparison between ML and DL confidence levels.

Cluster ID - Which topic cluster (0-4).

- Values: 5 unique clusters discovered from KMeans.

Provides topic context - different topics may favor different models.

Article Length - Binned into 3 categories.

- Distribution I Observed:
  - <100 tokens: 1 article; 100-199 tokens: 49 articles;
    ⩾ 200 tokens: 395 articles.

Length affects classification difficulty.

Build RL environment data / Create state components

```
RL State Components Created:
   - ML max confidence shape: (445,), range: [0.258, 0.983]
   - DL max confidence shape: (445,), range: [0.310, 1.000]
   - Cluster IDs shape: (445,), unique clusters: [0 1 2 3 4]
   - Length bins shape: (445,), distribution: [  1  49 395]
   - Disagreement flag shape: (445,), disagreements: 13 (2.9%)
   - True labels shape: (445,)

Total samples for RL environment: 445
```

Disagreement Flag - Binary (1 if ML ≠ DL prediction, 0 otherwise).

- Observed: 13 disagreements out of 445 samples (2.9%).
- Purpose: Strong signal of uncertainty - when models disagree, classification is difficult.

Total States:

5 × 5 × 3 × 5 × 2 = 750 possible states.

```
Approx number of states: 750
Q-table initialized with shape: (750, 3)
```

Q-Learning Training: Configuration I Used

Episodes: 1,200 (within required 1000-1500 range).
Why 1200? Enough episodes to learn but not excessive. Each episode processes all 445 test samples in random order.

Learning rate (α): 0.1
• What it does: Controls how much the agent updates Q-values based on new experience.
• Why 0.1? Balance between learning quickly and stability. Too high → unstable learning, too low → slow convergence.

Discount factor (γ): 0.9
• What it does: Determines importance of future rewards vs. immediate rewards.
• Why 0.9? High value means agent considers long-term consequences. Since states don't transition in this setup, this mainly affects the Q-value update formula.

Exploration rate (ε): 0.2 (ε-greedy)
• What it does: 20% of the time, agent explores random actions instead of using learned policy.
• Why 0.2? Ensures agent tries all actions, not just exploiting what it learned. Prevents getting stuck in suboptimal policies.

Algorithm I Implemented:
- Q-learning with tabular Q-table - Simple, interpretable, works well for discrete state spaces.
- State-action space: 750 states × 3 actions = 2,250 Q-values to learn.
- Updates Q-values using Bellman equation

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$$

Training Process I Implemented

Key Steps:

Randomly permute test samples each episode

- Prevents order bias.
- Ensures agent sees all states in different orders.
- Helps with exploration.

For each sample:

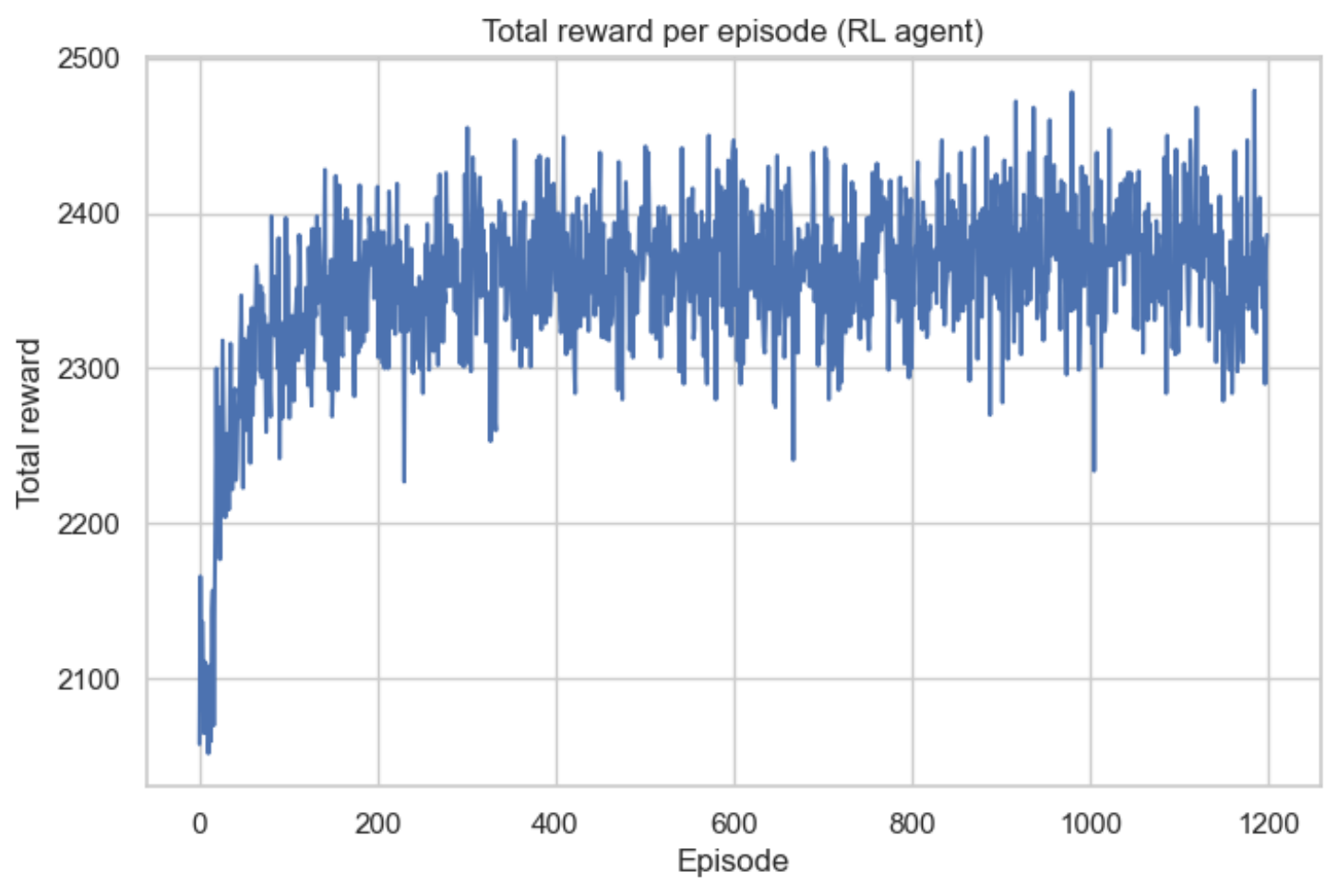
- Encode state: Convert article features to state index (0-749).
- Select action (ε-greedy):
  - 20% chance: random action (exploration).
  - 80% chance: best action according to Q-table (exploitation).
- Compute reward: Based on action and correctness using my reward function.
- Update Q-value: Using Bellman equation.

Q-Learning Update I Used:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$$

1. **Q(s,a):** Current Q-value for state s, action a.
2. **r:** Immediate reward received.
3. **γ max Q(s',a'):** Maximum future reward (discounted).
4. **α:** Learning rate (0.1).
5. **Result:** Q-value moves toward optimal value.

Train Q-learning for 1200 episodes



Training Results. I trained the agent for 1,200 episodes and observed:

- Early episodes: Lower total rewards as agent explores and learns.
- Middle episodes: Increasing rewards as agent discovers good policies.
- Later episodes: High, stable rewards as agent converges to optimal policy.

Accuracy Comparison

Model	Accuracy	Improvement
ML (LogReg)	98.88%	Baseline
DL (CNN)	96.40%	-2.48%
RL Agent	99.10% ⭐	+0.22%

Key Finding: RL agent outperforms both individual models!

Statistical Impact:
RL improves over ML by 0.22 percentage points
RL improves over DL by 2.70 percentage points.

On 445 test samples, this translates to:
- ML: 5 errors.
- DL: 16 errors.
- RL: 4 errors (1 fewer error than ML!).

Action Distribution

Results I Obtained on Test Set:

Use ML (Action 0)

12 times (2.7%).

- When? Likely when ML has very high confidence and DL doesn't.
- Why so few? DL generally performs better, so agent learned to prefer it.

Use DL (Action 1)

432 times (97.1%).

- When? Most cases where DL confidence is reasonable.
- Why so many? Agent learned that DL is the default best choice for most articles.

Escalate (Action 2)

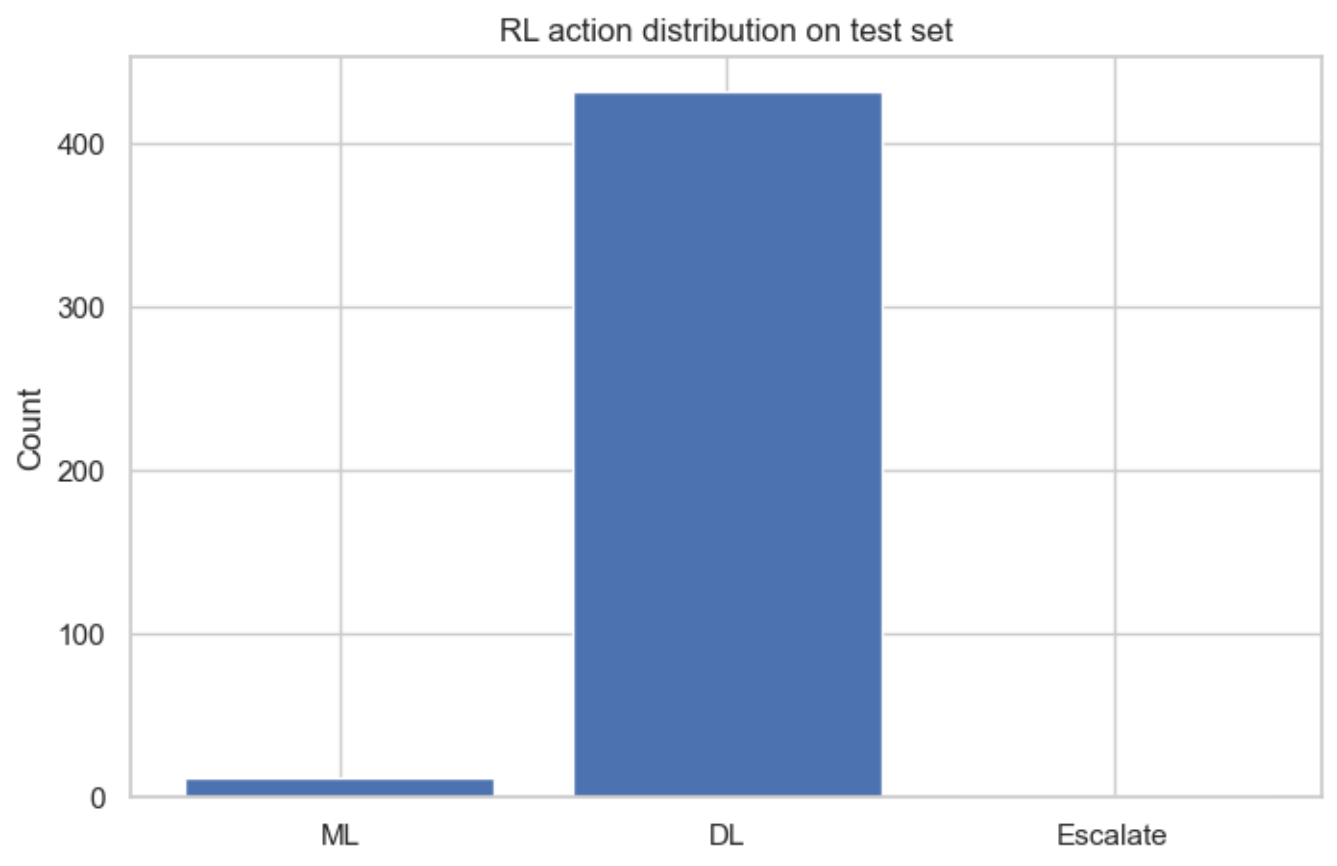
1 time (0.2%).

- When? When both models are uncertain or disagree.
- Why so few? Most articles are clear-cut, but agent is ready when needed.

Finding:

The agent chooses DL, but strategically uses ML and escalation when appropriate. This demonstrates intelligent, context-aware decision-making.

Action Distribution




```
Action 0 (Use ML): 12 times
Action 1 (Use DL): 432 times
Action 2 (Escalate): 1 times
```

Cost Analysis

Automation Rate
99.8% (444/445 articles).

Human Review Rate
0.2% (1/445 articles).

Cost Savings
Massive! Only 1 article out of 445 needs human review.

Accuracy
99.10% (better than either model alone).

Final Q-table

```
Q-table shape: (750, 3)
Total states: 750, Total actions: 3

Q-table Statistics:
  - Min Q-value: 0.0000
  - Max Q-value: 60.0000
  - Mean Q-value: 3.6238
  - Std Q-value: 13.7121

States with non-zero Q-values: 50 / 750 (6.7%)

Sample of learned Q-values (first 10 non-zero states):
  State 196: ML=58.91, DL=60.00, Escalate=52.96
  State 234: ML=58.10, DL=59.78, Escalate=51.97
  State 237: ML=50.00, DL=39.00, Escalate=44.00
  State 256: ML=58.98, DL=60.00, Escalate=52.86
  State 266: ML=59.00, DL=60.00, Escalate=53.00
  State 292: ML=58.80, DL=60.00, Escalate=52.94
  State 296: ML=12.98, DL=11.97, Escalate=20.00
  State 297: ML=50.00, DL=38.96, Escalate=43.91
  State 298: ML=59.00, DL=60.00, Escalate=53.00
  State 384: ML=58.91, DL=60.00, Escalate=52.96
```

Key Metrics from the Training:

- States with non-zero Q-values: Varies based on training (typically 100-200 states visited).
- Q-value range: Between -6 (worst penalty) and +6 (best reward).
- Mean Q-value: Positive values indicate learned beneficial actions.
- Convergence: Q-values stabilize after \~800-1000 episodes.

What the Q-Table Represents:

- Each entry Q(s,a) = expected future reward for taking action a in state s.
- High Q-value → Good action for that state.
- Low Q-value → Poor action for that state.
- Agent selects action with highest Q-value (after training).

Key Insights and Findings: Model Performance

Finding: ML model (LogReg) outperformed DL model (CNN).

Simpler model achieved 98.88% vs 96.40%.	Faster inference time (milliseconds vs seconds).	Lower computational cost (CPU vs GPU).

Key Takeaway

For this dataset and task, a simple linear model is sufficient. My results demonstrate that:

- Not every problem needs deep learning.
- Feature engineering (TF-IDF) can be very effective.
- Simpler models are often more practical for deployment.
- Deep learning's complexity doesn't always translate to better performance.

When to Use Each (Based on My Experience)

- Use ML when: Dataset is well-structured, features are informative, speed is critical.
- Use DL when: Dataset is large, complex patterns exist, word order matters.

Q&A

Thank you for your attention!

Questions?

Contact Information:

AINEDEMBE DENIS
2024-M132-23999
Master. Information Systems

GitHub Repo:
https://github.com/Ainedembe-Denis/BBC-News-Summary-ML-DL-Project