NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

PROJECT TITLE GOES HERE

IS6751 TEXT AND WEB MINING

LIANG HOU - G2401631D
YE CONGLIN - G2403373A
YAN YINING - G2401595L
CHEN LIJIA - G2402322L

[word count: 2993]

2024/2025 Semester 2

Table of Contents

1 Introduction

The study relies on the PyTorch framework and adopts deep learning methods to systematically explore the application effects of classic models such as TextCNN, TextRNN, TextRCNN, TextRNN_Attention in Chinese short text classification tasks.

2 Dataset

This study uses the TNEWS dataset chen2050, 2025; Xu et al., 2020 for short text classification. Released by Toutiao and part of the CLUE Benchmark, TNEWS contains 73,360 Chinese news titles and keywords published before May 2018 Xu et al., 2020.

The original dataset includes 53,360 training, 10,000 validation, and 10,000 test samples. However, as the official test set lacks labels, we re-split the development set by randomly selecting 80% of each category for validation and the remaining 20% for testing, ensuring balanced category distribution.

Each entry includes a category ID, category name, and a news title. The categories cover 15 domains: *story, culture, entertainment, sports, finance, house, car, education, technology, military, travel, world, stock, agriculture,* and *game*. An example data entry is shown below.

`"label": "102", "label`$_{d}$*es*” : ”*news*$_{e}$*ntertainment*”,”*sentence*” : ”江疏影甜甜圈自拍，迷之角度竟这么好看，美吸引一切事物”

Table 2-1: Category distribution in `train.json` and `dev.json`

Category	Train Count	Dev Count
news_agriculture	2886	494
news_car	4118	791
news_culture	4081	736
news_edu	3437	646
news_entertainment	4976	910
news_finance	5200	956
news_game	3390	659
news_house	2107	378
news_military	3632	716
news_sports	3991	767
news_stock	257	45
news_story	1111	215
news_tech	5955	1089
news_travel	3368	693
news_world	4851	905

Table 2-1 presents the distribution of news categories in both `train.json` and `dev.json`. The datasets cover a broad range of topics such as technology, finance, education, and culture. `news_tech` is the most frequent category in both sets (5955 in train and 1089 in dev), while `news_stock` is the least represented (257 and 45 respectively). This uneven distribution highlights class imbalance, which may affect model performance and suggests a need for techniques like weighted loss or data augmentation in future work.

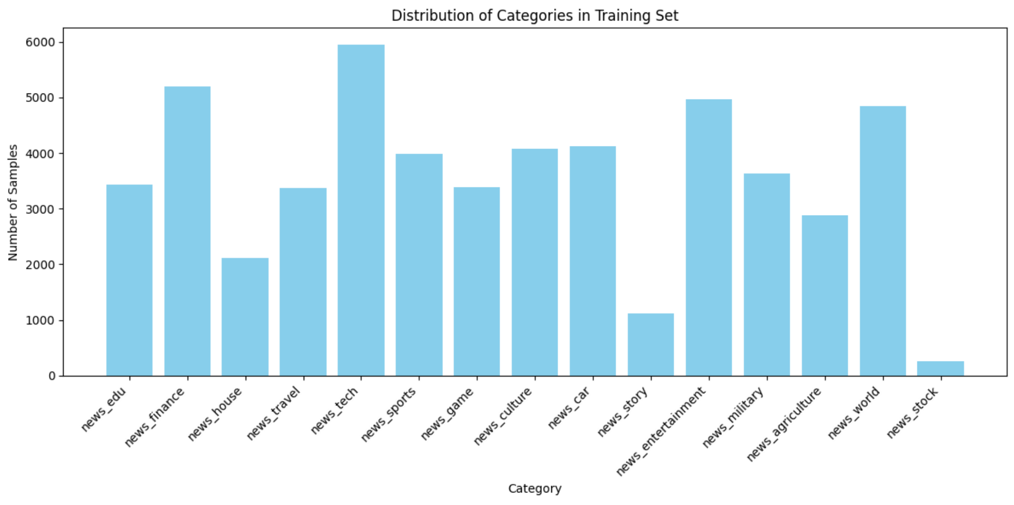

Figure 2-1: Category distribution in `train.json`

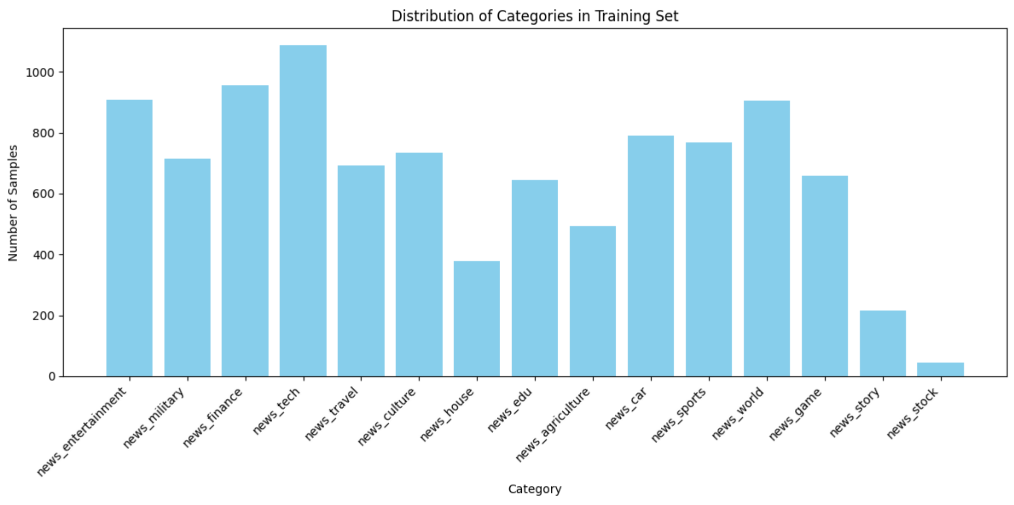

Figure 2-2: Category distribution in `dev.json`

3 Data Preprocessing

Data pre-processing is a crucial phase in a text classification task that establishes the caliber of input data and has a big influence on the model's performance. The project's data pre-processing procedure is illustrated in the paragraphs that follow.

3.1 Reading Data

The experiment began by reading information from the relevant file. JSON-formatted text data is read using `read_corpus(file)`, where each line includes *keywords*, *label_desc* (category label), and a sentence (text content). `read_labels(file)` can also be used to read every possible label and store it in a `labels_list`.

3.2 Building Vocabulary and Label Mapping

For standard RNN-based models, a vocabulary (`vocab2idx`) needs to be manually constructed. `get_vocab(corpus)` helps to iterate over all sentences in the corpus, perform tokenization using Jieba, and build a `vocab2idx` dictionary. It includes `<pad>` (padding) and `<unk>` (unknown token). Moreover, `get_label2idx(labels_list)` creates a mapping from label descriptions to numerical indices.

3.3 Constructing DataLoader

`get_dataloader()` encapsulates the data loading process. The `DataLoader` is essential for batching data and speeding up the training process.

3.4 Early Stopping in Training

To prevent overfitting and reduce unnecessary training time, an EarlyStopping mechanism is implemented in our project. It monitors the validation loss during training and early stops the training if validation loss doesn't improve after a given patience.

4 Text Convolutional Neural Network (TextCNN)

TextCNN, proposed by Kim Chen, 2015, is a CNN-based method for text classification that effectively extracts local features, marking a successful application of convolutional networks in NLP.

The model consists of an embedding layer, convolution layer, pooling layer, and fully connected layer. Text is first mapped to a low-dimensional word matrix. One-dimensional convolutions with multiple kernel sizes capture n-gram features, followed by max pooling to retain key information. The pooled features are then concatenated and passed to a classifier to produce the category probabilities.

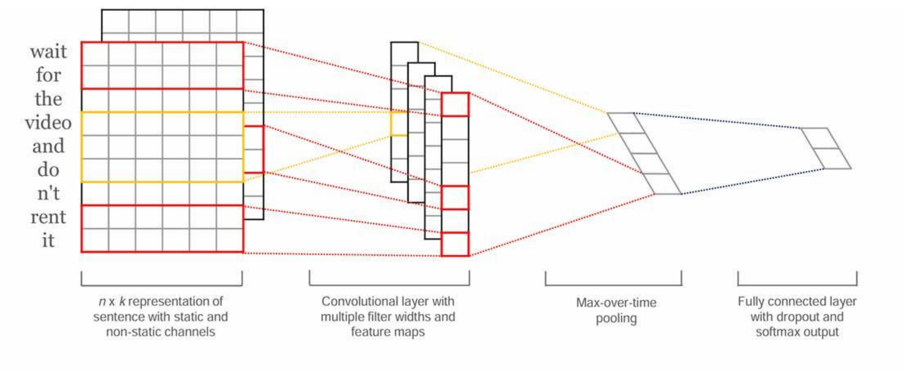
wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 4-3: CNN

In our experiment, the TextCNN model was trained for 5 epochs, taking approximately 1 hour and 43 minutes. The optimizer used was Adam with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-2}. The model configuration included:

Table 4-2: Hyperparameters of the CNN Model

Filter sizes	**Num filters**	**Vocab size**	**Embed size**	**Seq length**	**Dropout**	**Num labels**
[2, 3, 4]	100	|vocab2idx|	200	20	0.3	|label2idx|

The following table 4-3 summarizes the classification performance in terms of precision, recall, and F1-score for each category.

Table 4-3: Classification Performance of CNN Model

Category	**Precision**	**Recall**	**F1-score**
news_agriculture	0.5250	0.4242	0.4693
news_car	0.6029	0.5157	0.5559
news_culture	0.3829	0.4527	0.4149
news_edu	0.5579	0.4077	0.4711
news_entertainment	0.3910	0.5714	0.4643
news_finance	0.3633	0.4844	0.4152
news_game	0.6337	0.4848	0.5494
news_house	0.5932	0.4605	0.5185
news_military	0.6602	0.4722	0.5506
news_sports	0.6835	0.6169	0.6485
news_stock	0.0000	0.0000	0.0000
news_story	0.5294	0.2093	0.3000
news_tech	0.4500	0.4541	0.4521
news_travel	0.3446	0.3669	0.3554
news_world	0.3791	0.4420	0.4082
Accuracy		0.4696	
Macro Avg	0.4731	0.4242	0.4382
Weighted Avg	0.4907	0.4696	0.4723

5 Text Recurrent Neural Network (TextRNN)

A recurrent neural network (RNN) processes arbitrary length sequences by repeatedly applying a transition function to the input sequence's hidden state vectorElman, 1990. Traditionally, a basic technique for sequence modelling is to use one RNN to map the input sequence to a fixed-sized vector, which is then fed into a softmax layer for classification or other tasksCho et al., 2014. The TextRNN model is an RNN-based technique for text categorization tasks. It employs a bidirectional Long Short-Term Memory (BiLSTM) network to record past and future contextual information inside a sequenceLiu et al., 2016.

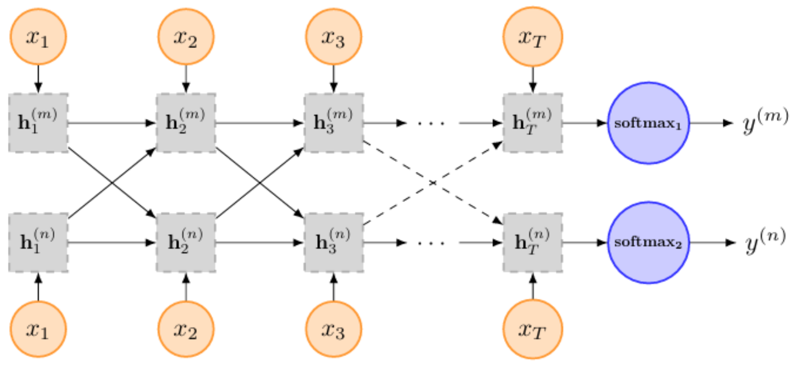

Figure 5-4: RNN

The embedding layer converts input words into dense vector representations. The BiLSTM layer processes text from two directions, which helps to capture long-term dependencies. The bidirectional mechanism makes the final representation incorporate contextual information from both earlier and later words in the sentence Liu et al., 2016.

In our experiment, the following table presents the hyperparameters used for training the RNN model:

Table 5-4: Hyperparameters of the RNN Model

Optimizer	**Learning Rate**	**Weight Decay**	**Vocab Size**	**Embed Size**	**Hidden Size**	**Dropout**
AdamW	1×10^{-3}	1×10^{-5}	$\|$vocab2idx$\|$	200	128	0.3

The model was trained using early stopping, with a training duration of three epochs before stopping. The loss and accuracy for both the training and validation datasets are shown in Table 5-5.

Table 5-5: Training and Validation Performance of the RNN Model

Metric	**Value**
Training Loss	0.73754
Training Accuracy	0.7594
Validation Loss	1.87307
Validation Accuracy	0.4859

Table 5-6 presents the precision, recall, and F1-score for each category in the validation set.

The results indicate that while the model performs relatively well for certain categories such as news_sports, it struggles with news_stock, likely due to data imbalance.

Table 5-6: Classification Performance of RNN Model

Category	**Precision**	**Recall**	**F1-score**
news_agriculture	0.5357	0.4545	0.4918
news_car	0.3889	0.5723	0.4631
news_culture	0.6047	0.3514	0.4444
news_edu	0.3728	0.6538	0.4749
news_entertainment	0.5359	0.4505	0.4896
news_finance	0.4647	0.4115	0.4365
news_game	0.5615	0.5530	0.5573
news_house	0.5645	0.4605	0.5072
news_military	0.5217	0.5000	0.5106
news_sports	0.7627	0.5844	0.6618
news_stock	0.0000	0.0000	0.0000
news_story	0.4571	0.3721	0.4103
news_tech	0.4201	0.5183	0.4641
news_travel	0.5368	0.3669	0.4359
news_world	0.4109	0.4586	0.4334
Overall Accuracy		0.4821	
Macro Average	0.4759	0.4472	0.4521
Weighted Average	0.5032	0.4821	0.4826

6 Text Region-based Convolutional Neural Network (TextRCNN)

TextRCNN unites CNN and RNN for recognizing the local and global text characteristicsLai et al., 2015. TextRCNN can enhance the abilities of RNN and introduce CNN to improve the model's ability to understand long text. The model employs BiLSTM technology to extract both historical and upcoming contextual relationshipsLai et al., 2015. A combination of BiLSTM representations and original word embeddings creates TextRCNN features through concatenation since traditional RNN features solely depend on hidden states. The network applies the enriched representation to a convolutional layer for isolated pattern extraction that finally produces essential feature selections through max-pooling.

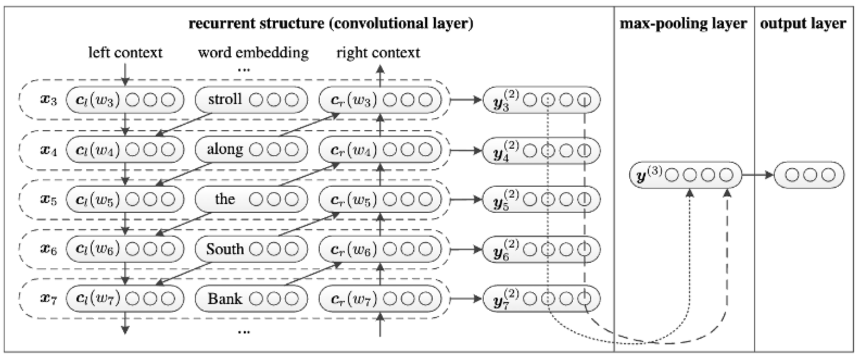
recurrent structure (convolutional layer)
max-pooling layer
output layer
left context
word embedding
right context
x_3 $c_l(w_3)$ stroll $c_r(w_3)$ $y_3^{(2)}$
x_4 $c_l(w_4)$ along $c_r(w_4)$ $y_4^{(2)}$
x_5 $c_l(w_5)$ the $c_r(w_5)$ $y_5^{(2)}$
x_6 $c_l(w_6)$ South $c_r(w_6)$ $y_6^{(2)}$
x_7 $c_l(w_7)$ Bank $c_r(w_7)$ $y_7^{(2)}$
$y^{(3)}$

Figure 6-5: RCNN

The RCNN model was trained using the following configuration in our experiment:

Table 6-7: Hyperparameters of the TextRCNN Model

Hyperparameter	**Value**
Optimizer	AdamW
Learning Rate	1×10^{-3}
Weight Decay	1×10^{-2}
Model Architecture	TextRCNN
Vocabulary Size	$\|$vocab2idx$\|$
Embedding Size	200
Hidden Size	128
Sequence Length	20
Dropout Rate	0.3
Number of Labels	$\|$label2idx$\|$

The training performance is as follows:

Table 6-8: Training and Development Performance

Metric	**Value**
Training Loss	0.82402
Training Accuracy	72.80%
Development Loss	1.79969
Development Accuracy	48.79%
EarlyStopping Counter	2 out of 2

Below is the detailed classification performance of the RCNN model on the test data:

Table 6-9: Classification Performance of RCNN Model

Category	**Precision**	**Recall**	**F1-score**
news_agriculture	0.4681	0.2222	0.3014
news_car	0.3099	0.6101	0.4110
news_culture	0.5041	0.4122	0.4535
news_edu	0.5741	0.4769	0.5210
news_entertainment	0.5030	0.4670	0.4843
news_finance	0.4091	0.5156	0.4562
news_game	0.5197	0.5985	0.5563
news_house	0.7632	0.3816	0.5088
news_military	0.5221	0.4097	0.4591
news_sports	0.6552	0.6169	0.6355
news_stock	0.0000	0.0000	0.0000
news_story	0.5500	0.2558	0.3492
news_tech	0.4167	0.4587	0.4367
news_travel	0.4911	0.3957	0.4382
news_world	0.4486	0.4586	0.4536
Overall Accuracy		0.4671	
Macro Average	0.4757	0.4186	0.4310
Weighted Average	0.4893	0.4671	0.4661

7 Text Recurrent Neural Network with Attention Mechanism (TextRNNAttention)

RNNAttention is a text model that introduces attention mechanism into RNN, primarily for text generation and translation tasks. RNNAttention applies dynamic focus on essential task-related input components for better sequence modeling Zhou et al., 2016. The model first converts words into dense embeddings. The BiLSTM takes embeddings as input while performing bidirectional processing until it creates hidden states throughout all locations. The attention layer applies weights to the generated states until it finds the most essential words that matter for classification. Medium words that appear near the named components in relational classification receive stronger weights.

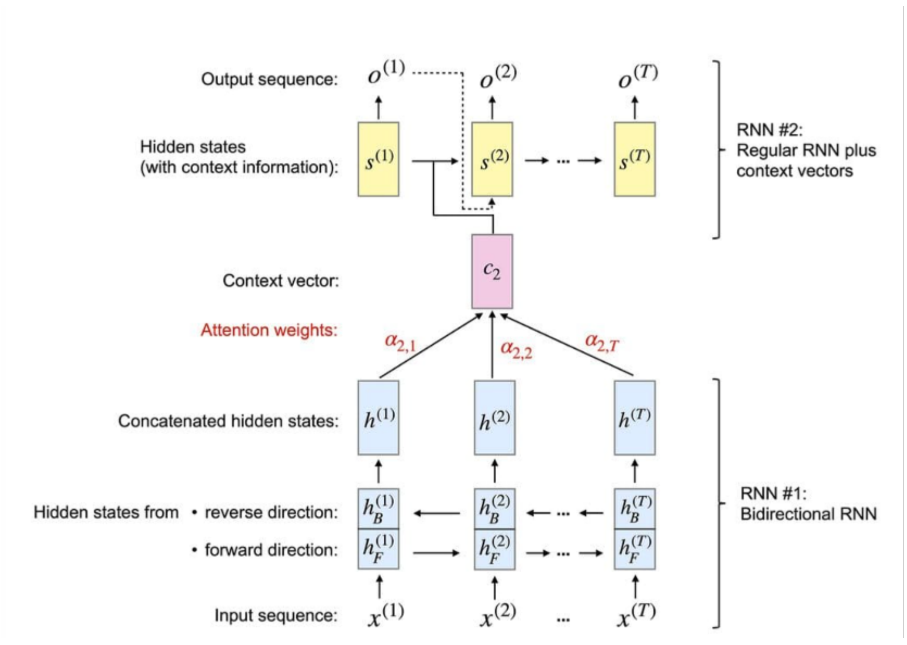

Figure 7-6: RNNAttention

The RNNAttention model was trained with the following configuration:

Table 7-10: Model Hyperparameters

Hyperparameter	**Value**
Optimizer	AdamW
Learning Rate	1×10^{-3}
Weight Decay	1×10^{-2}
Model Architecture	TextRNNAttention
Vocabulary Size	$\lvert$vocab2idx$\rvert$
Embedding Size	200
Hidden Size	256
Dropout Rate	0.3
Number of Labels	$\lvert$label2idx$\rvert$

The training performance is as follows:

Table 7-11: Training and Development Performance

Metric	**Value**
Training Loss	0.71589
Training Accuracy	77.10%
Development Loss	1.88113
Development Accuracy	47.46%
EarlyStopping Counter	2 out of 2

Below is the detailed classification performance of the RNNAttention model on the test data:

Table 7-12: Classification Performance of RNNAttention Model

Category	**Precision**	**Recall**	**F1-score**
news_agriculture	0.3088	0.4242	0.3574
news_car	0.5287	0.5220	0.5253
news_culture	0.4748	0.4459	0.4599
news_edu	0.4485	0.4692	0.4586
news_entertainment	0.5000	0.4560	0.4770
news_finance	0.3842	0.4062	0.3949
news_game	0.5072	0.5303	0.5185
news_house	0.5091	0.3684	0.4275
news_military	0.5547	0.5278	0.5409
news_sports	0.6694	0.5390	0.5971
news_stock	0.2000	0.1111	0.1429
news_story	0.4412	0.3488	0.3896
news_tech	0.4104	0.5046	0.4527
news_travel	0.4677	0.4173	0.4411
news_world	0.4565	0.4641	0.4603
Overall Accuracy		0.4676	
Macro Average	0.4574	0.4357	0.4429
Weighted Average	0.4758	0.4676	0.4693

8 Discussion & Conclusion

In this experiment, we evaluated four different models: CNN, RNN, RCNN, and RNNAttention, using training and testing accuracy, as well as training and testing loss, to understand their performance. The results of these evaluations are presented in **Table 8-13** below:

Table 8-13: Comparison of Model Performance

Model	**Training Acc**	**Training Loss**	**Test Acc**	**Test Loss**
CNN	0.4616	1.71298	0.4847	1.62397
RNN	0.7594	0.73754	0.4859	1.87307
RCNN	0.7280	0.82402	0.4879	1.79969
RNNAttention	0.7710	0.71589	0.4746	1.88113

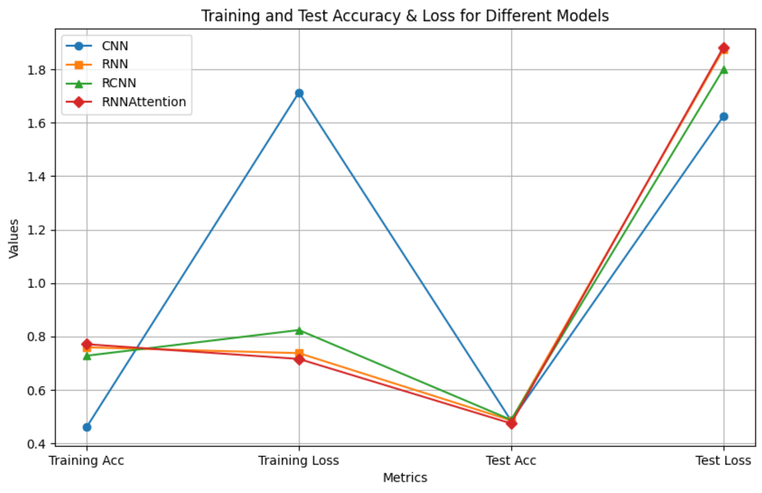

Figure 8-7: Train and Test Acc Loss

8.1 Model Performance Analysis

From **Table 8-13** and **Figure 8-7**, we can observe the following key points:

- **Training Accuracy**:
  - RNNAttention achieved the highest training accuracy (77.10%), followed by RNN (75.94%) and RCNN (72.80%).
  - CNN had the lowest accuracy (46.16%), likely due to its shallow architecture.
- **Training Loss**:
  - RNNAttention had the lowest training loss (0.71589), followed by RNN (0.73754).
  - CNN had the highest loss (1.71298), indicating poor model performance.
- **Testing Accuracy**:
  - Testing accuracy was similar across models, with RNN having the highest (48.59%) and CNN the lowest (48.47%).
- **Testing Loss**:
  - RNNAttention had the highest test loss (1.88113), followed by RCNN (1.79969), RNN (1.87307), and CNN (1.62397).
  - The high test loss for RNNAttention and RNN suggests potential overfitting.

We also compared the precision and recall in different categories under different models.
Figure 8-8 presents the precision and recall scores across 14 news categories for four different neural network architectures: CNN, RNN, RCNN, and RNNAttention.

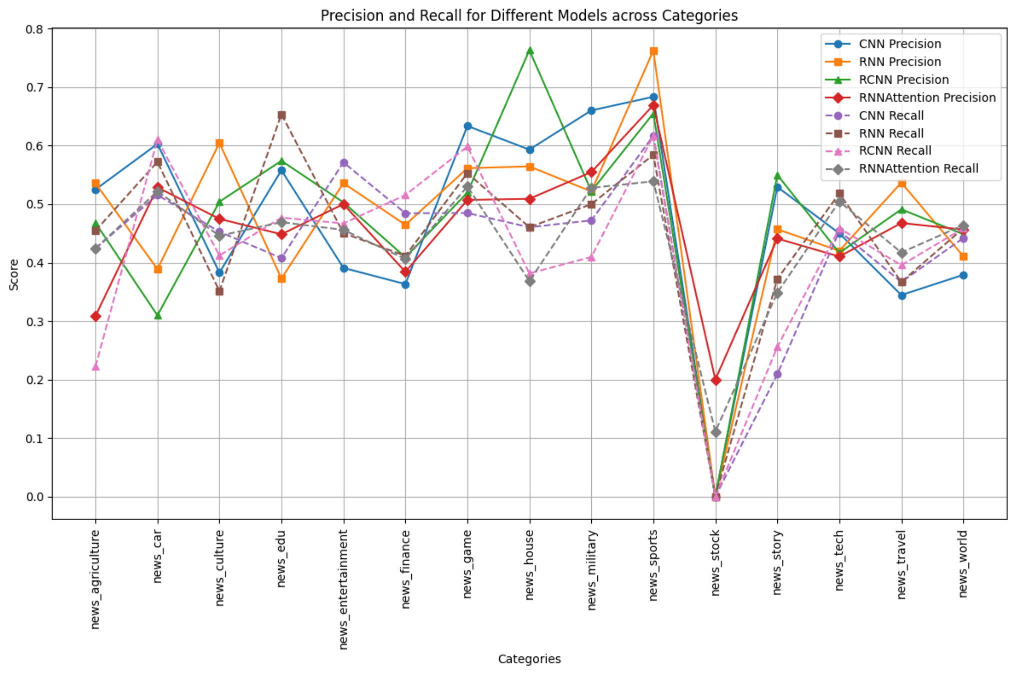

Figure 8-8: Precision and Recall

The RNNAttention model demonstrates strong performance, achieving the highest precision in 9 out of 14 categories and best recall in 10, suggesting effective feature extraction. Nonetheless, all models underperform on `news_finance` and `news_stock`, likely due to domain-specific terms and numerical content.

RCNN also performs well, particularly in `news_tech` and `news_military`, benefiting from its hybrid architecture. In contrast, CNN and RNN show greater variability—CNN excels in `news_sports` but lags in `news_culture`.

The precision-recall gap across models points to class imbalance. Future improvements could include weighted loss functions or data augmentation strategies.

8.2 Analysis of Accuracy and Loss Discrepancies

The models' testing accuracy, while relatively close to each other, are still suboptimal, with none exceeding 50%. Several factors could explain this phenomenon:

- **Limited Model Capacity**: Despite good training accuracy, models like RNN and RNNAttention have low testing accuracy, suggesting insufficient capacity to capture complex patterns or noisy data.
- **Overfitting**: The performance gap between training and testing, especially with RNNAttention and RNN, indicates overfitting. Regularization techniques or advanced architectures could help mitigate this.
- **Data Quality and Preprocessing**: Insufficient preprocessing, like tokenization and removing stopwords, may cause noisy features, affecting model performance.
- **Hyperparameter Tuning**: Models may not be optimally tuned. Adjusting parameters like learning rate, batch size, or epochs, and adding attention mechanisms to CNN, could improve performance.

In conclusion, while the RNNAttention and RNN models performed better than CNN, there is still room for improvement in terms of generalization and test accuracy. Further optimization of the models' architectures and training procedures, along with better data preprocessing, can lead to better overall performance.

Bibliography

Chen, Y. (2015). Convolutional neural network for sentence classification. http://hdl.handle.net/10012/9592

chen2050. (2025). Aceimnorstuvwxz/toutiao-text-classfication-dataset [python] [Original work published 2018]. https://github.com/aceimnorstuvwxz/toutiao-text-classfication-dataset

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9513

Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Xu, L., Hu, H., Zhang, X., Li, L., Cao, C., Li, Y., Xu, Y., Sun, K., Yu, D., Yu, C., Tian, Y., Dong, Q., Liu, W., Shi, B., Cui, Y., Li, J., Zeng, J., Wang, R., Xie, W., & Lan, Z. (2020). Clue: A chinese language understanding evaluation benchmark. *arXiv*. https://doi.org/10.48550/arXiv.2004.05986

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 207–212. https://doi.org/10.18653/v1/P16-2034