2025 Data Mining HW3 Report

Name: 鄭睿宏
Student ID: 314554025

Q1: Describe how you solve this problem. Details include preprocessing, embeddings, model selection, and hyperparameters.

Ans:

To address the News Recommendation Prediction task, I implemented the **Neural News Recommendation with Multi-Head Self-Attention (NRMS)** architecture. This approach utilizes self-attention mechanisms to learn deep representations of both news content and user interests. To handle the large-scale dataset efficiently, I optimized the training process using **Distributed Data Parallel (DDP)** on 8 GPUs.

My implementation process is detailed below:

1. Data Preprocessing

- **Text Cleaning & Tokenization:**
  - For news titles, I applied regular expressions to remove punctuation and converted all text to lowercase to ensure consistency.
  - I constructed a vocabulary dictionary using the top 40,000 most frequent words, mapping the rest to an `<UNK>` token.
  - The maximum title length was set to **30 tokens** (`MAX_TITLE_LEN = 30`). Titles shorter than this were padded, and longer ones were truncated.
- **Category Encoding:**
  - News categories were encoded into integer IDs to be used for Category Embeddings, providing additional semantic context beyond just the text.
- **History Processing:**
  - User behavior history was fixed at a length of **50** (`MAX_HISTORY_LEN = 50`). If a user had fewer interactions, the sequence was padded with zeros; if more, only the most recent 50

interactions were kept to capture the latest user interests.

- **Negative Sampling:**
  - During training, I employed a **1:4 negative sampling ratio** (`NUM_NEGATIVES = 4`). For every positive click (impression), 4 non-clicked news items were randomly sampled from the same session to form the training batch.

2. Embeddings

- **Word Embeddings:** I utilized the pre-trained **GloVe 100-dimensional** vectors (`glove.6B.100d.txt`) to initialize the embedding layer. This layer was set to be trainable (fine-tuning) during the process to adapt to the specific domain of the news dataset.
- **Category Embeddings:** A separate embedding layer with a dimension of 50 (`CAT_EMB_DIM = 50`) was initialized to represent news categories.

3. Model Architecture (NRMS)

The model consists of two core components: the **News Encoder** and the **User Encoder**.

- **News Encoder:**
  i. **Input:** Takes the news title sequence and category ID.
  ii. **Representation:** Applies word embeddings followed by a Dropout layer.
  iii. **Multi-Head Self-Attention:** Utilizes **20 attention heads** (`NEWS_HEADS = 20`) to capture the contextual interaction between words in the title.
  iv. **Additive Attention:** Compresses the sequence of word vectors into a single title vector.
  v. **Concatenation:** The title vector (250-dim) is concatenated with the category vector (50-dim) and passed through a fully connected layer to produce the final **300-dimensional news vector**.
- **User Encoder:**
  i. **Input:** A sequence of news vectors representing the user's browsing history (Batch x 50 x 300).
  ii. **Multi-Head Self-Attention:** Uses **20 attention heads** (`USER_HEADS = 20`) to capture the relationships between different news items in the history (e.g., identifying consistent interest in specific topics).
  iii. **Additive Attention:** Aggregates the history sequence into a single **User Interest Vector** (300-dim).
- **Prediction:**
  - The click probability is calculated using the **Dot Product** between the User Vector and the Candidate News Vector.

4. Optimization & Hardware Acceleration

- **Distributed Data Parallel (DDP):** The code was engineered to run on **8x V100 GPUs**. I used `torch.distributed` to parallelize the training, with `DistributedSampler` ensuring that each GPU processes a distinct slice of the dataset.
- **Automatic Mixed Precision (AMP):** I implemented `torch.cuda.amp.GradScaler` to enable mixed-precision training (FP16). This significantly reduced VRAM usage and accelerated computation on the V100 Tensor Cores.
- **SyncBatchNorm:** Converted standard Batch Normalization to `SyncBatchNorm` to ensure accurate statistics synchronization across multiple GPUs.

5. Hyperparameters

Parameter	Value	Description
`BATCH_SIZE`	512	Per-GPU batch size. Global batch size = 4096 (512 x 8).
`NUM_EPOCHS`	20	Number of training epochs.
`LEARNING_RATE`	0.001	Used with the Adam optimizer.
`WORD_EMB_DIM`	100	GloVe dimension.
`NEWS_DIM`	250	Dimension after News Attention.
`TOTAL_NEWS_DIM`	300	`NEWS_DIM + CAT_EMB_DIM` (50).
`NUM_NEGATIVES`	4	Negative sampling ratio 1:4.

Q2: Choose a variable (e.g., different model, different approach) excluding hyperparameters and compare their performance. Explain what causes the difference.

Ans:

Variable Compared: Negative Sampling Ratio

- **Approach A:** Ratio 1:1 (1 negative sample per positive sample).

- **Approach B (My Implementation):** Ratio 1:4 (4 negative samples per positive sample).

Performance Comparison:
In my experiments, **Approach B (1:4)** yielded a significantly higher AUC score compared to Approach A.

Explanation of the Difference:

1. **Harder Decision Boundary:**
   News recommendation is essentially a retrieval task with a massive class imbalance (a user ignores most news). With a 1:1 ratio, the model can easily minimize loss by distinguishing the positive news from a single random irrelevance. Increasing the ratio to 1:4 forces the model to identify the correct target among a larger amount of "noise." This forces the encoders to learn more **discriminative features** to separate the user embedding from multiple negative news vectors in the latent space.
2. **Higher Information Gain per Step:**
   With 4 negative samples, the model is exposed to a wider variety of "what the user does not like" in every training step. This provides richer gradient signals during backpropagation. The User Encoder learns not just to align with the positive history, but to actively distance itself from a broader range of negative content categories.
3. **Better Approximation of Real Distribution:**
   In the test set and real-world scenarios, the ratio of clicked to non-clicked items is extremely low. Training with a higher negative sampling ratio (1:4) creates a data distribution that is closer to the inference phase, thereby improving the model's generalization capability.

Q3: Do some error analysis or case study. Is there anything worth mentioning while checking the mispredicted data?

Ans:

After analyzing the cases where the model failed to predict accurately (Mispredicted Data), I observed several patterns worth mentioning:

- **1. The Cold Start Problem & Sparse History:**
  - **Observation:** Some users in the test set had empty or very short history sequences (e.g., < 3 items). In my code, these are padded with zeros.

    - **Analysis:** For these users, the Self-Attention mechanism in the User Encoder cannot extract meaningful patterns. The resulting User Embedding tends to be a generic average or random vector. Consequently, the model fails to personalize the ranking, likely relying on random guesses or inherent biases in the News Encoder, leading to low AUC.
    - **Future Improvement:** Incorporating non-behavioral features (e.g., user metadata) or using a popularity-based fallback for new users could mitigate this.
- **2. Information Loss due to Title Truncation:**
    - **Observation:** I set `MAX_TITLE_LEN = 30`. However, some news titles place the most critical entity (e.g., a specific person or location) at the very end of a long sentence.
    - **Analysis:** Truncation causes the News Encoder to miss the most distinct keywords. If the first 30 words are generic introductory text, the model cannot link the specific entity in the candidate news to the user's history, resulting in a false negative prediction.
- **3. Interest Drift:**
    - **Observation:** The model struggles with sudden changes in user interest. For example, a user with a history dominated by "Technology" might click on a trending "Sports" article (e.g., World Cup). The model predicts a low probability for this click.
    - **Analysis:** The Dot Product mechanism relies on similarity. A "Tech" user vector is far from a "Sports" news vector in the embedding space. The current architecture assumes consistent interest and lacks a mechanism to model "exploratory" behavior or global trending events.
- **4. Entity Overlap vs. Sentiment:**
    - **Observation:** A user might click on negative news about a politician but avoid positive propaganda about the same person. The model often predicts a high probability for the latter but gets it wrong.
    - **Analysis:** The News Encoder heavily relies on Word Embeddings and Attention to match keywords (Entities). It sees the same politician's name and assumes relevance. However, the model likely lacks a deep understanding of **sentiment** or **stance**, failing to distinguish between "interested in the person" and "interested in a specific viewpoint."