Introduction to Artificial Intelligence Final Project Report

Part 1: Vector Space Model Construction

I deployed a Streamlit app for part 1 here.

1. *Overview*

Each verse of Truyện Kiều is treated as a document and encoded into a vector space using TF-IDF. The model supports both search queries and authorship classification.

2. *Data Preparation*

Each line from Truyện Kiều is preprocessed individually to form a standalone document. First, I use the Unicode normalization in NFC form to ensure consistent character representation. I also apply lowercasing to eliminate case-based discrepancies, like what was already learnt in class. The punctuation and extra white space are also removed. I apply the tokenization using a whitespace-based split, with punctuation stripped from each token. This initial phase will ensure all verses are uniformly cleaned and segmented, making them suitable for vectorization using TF-IDF and other downstream tasks.

Image 1: Testing result on the first phrase

```
→  /usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:517: UserWarning:
   warnings.warn(
        a  a hoàn   ai  ai ai  ai biết  ai bì  ai có  ai cũng  ai dám  ai dễ  \
   0  0.0     0.0  0.0     0.0      0.0    0.0    0.0      0.0     0.0     0.0
   1  0.0     0.0  0.0     0.0      0.0    0.0    0.0      0.0     0.0     0.0
   2  0.0     0.0  0.0     0.0      0.0    0.0    0.0      0.0     0.0     0.0
   3  0.0     0.0  0.0     0.0      0.0    0.0    0.0      0.0     0.0     0.0
   4  0.0     0.0  0.0     0.0      0.0    0.0    0.0      0.0     0.0     0.0

      ...  ở bắc  ở chung  ở tay  ở trong  ở trên  ở đâu  ở đây  ở đời    ủ  \
   0  ...    0.0      0.0    0.0      0.0     0.0    0.0    0.0    0.0  0.0
   1  ...    0.0      0.0    0.0      0.0     0.0    0.0    0.0    0.0  0.0
   2  ...    0.0      0.0    0.0      0.0     0.0    0.0    0.0    0.0  0.0
   3  ...    0.0      0.0    0.0      0.0     0.0    0.0    0.0    0.0  0.0
   4  ...    0.0      0.0    0.0      0.0     0.0    0.0    0.0    0.0  0.0

                                        verse
   0              trăm năm trong cõi người ta,
   1      chữ tài chữ mệnh khéo là ghét nhau.
   2                  trải qua một cuộc bể dâu,
   3   những điều trông thấy mà đau đớn lòng.
   4                     lạ gì bỉ sắc tư phong,

   [5 rows x 4118 columns]
```

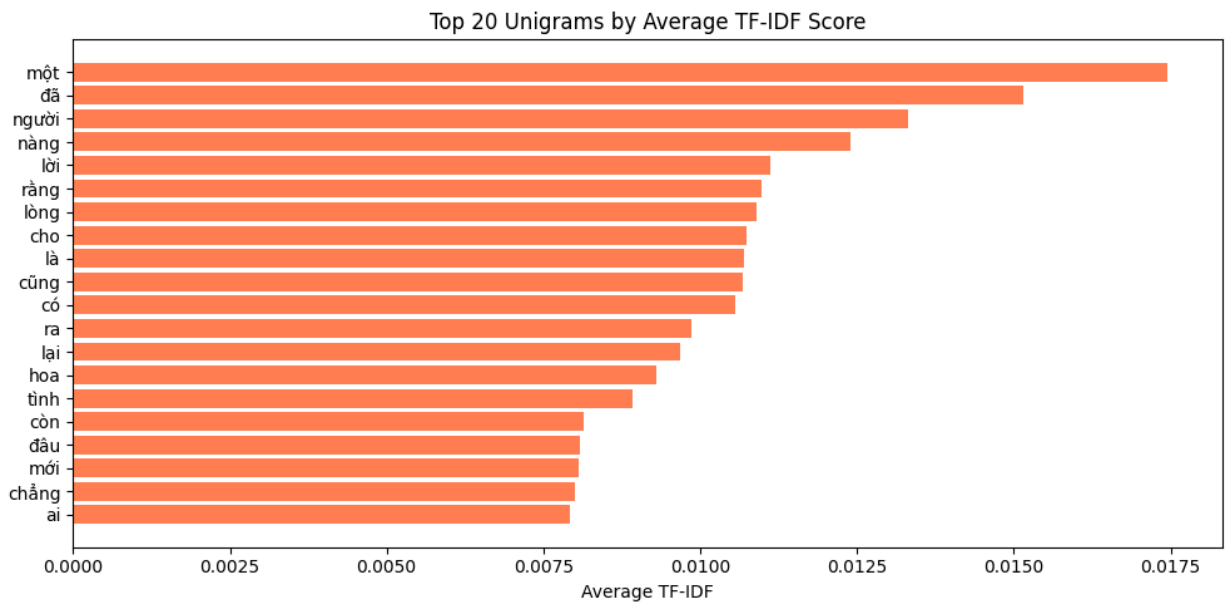Image 2: Preview TF-IDF Matrix

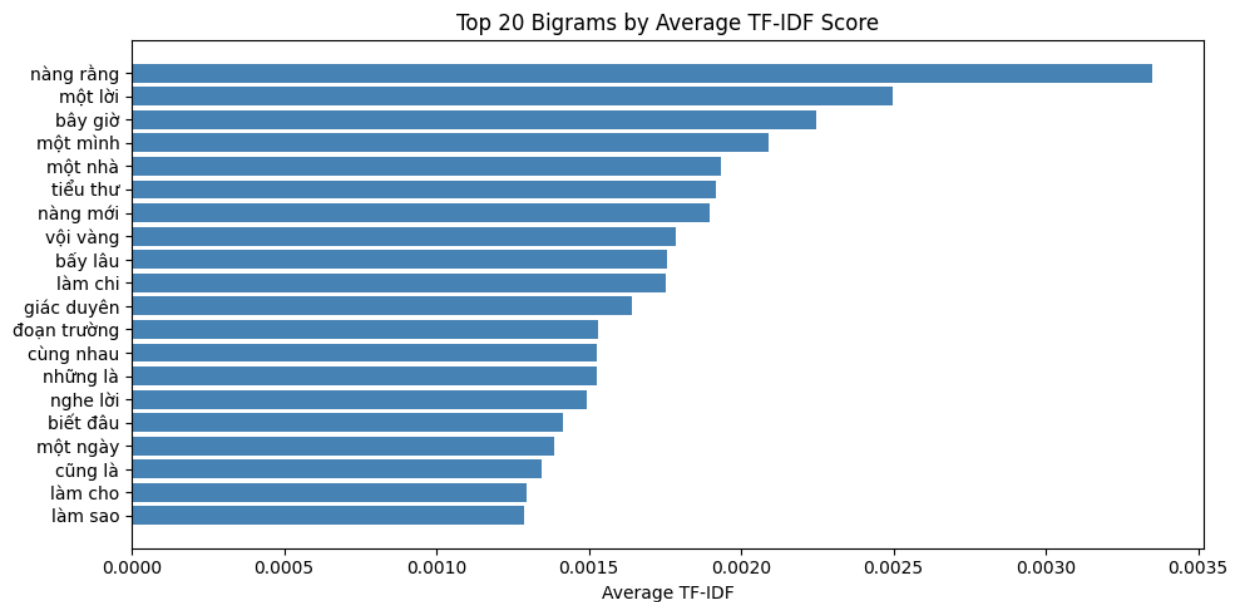Image 3: Top 20 Unigrams by Average TF-IDF Score

Image 4: Top 20 Bigrams by Average TF-IDF Score

3. Search Engine Development

Using the TF-IDF, I develop a lightweight search engine that supports two retrieval modes: word overlap and cosine similarity. Given a keyword-based query such as "ngày xuân", the exact-token overlap search counts shared words between query and each verse. The verse that have highest overlap score will be returned.

Image 5: Searching by Overlap

Alternatively, the system computes the similarity score between the query and all verses in the document matrix when I enter the keyword "ngày xuân". It then ranks the most semantically related verses using cosine distance.

Truyen Kieu Search & Authorship Prediction

Classify Verse   **Search by Cosine**   Search by Overlap   Inverted Index

Search by cosine similarity

ngày xuân

Top matching verses:

- hoa xuân đương nhụy, ngày xuân còn dài. (score: 0.61)

- ngày xuân lắm lúc đi về với xuân. (score: 0.59)

- ngày xuân con én đưa thoi, (score: 0.55)

- ngày xuân em hãy còn dài, (score: 0.54)

- ngày xuân càng gió càng mưa càng nồng. (score: 0.46)

Image 6: Searching by Cosine

Next, to enhance retrieval efficiency and enable keyword-based indexing, an inverted index is implemented. It aims to map each token to the set of verses it appears in, allowing instant lookups for users interested in where specific words occur.

Image 7: Searching Inverted Index

4. Authorship Attribution

To predict whether Nguyễn Du wrote a given verse, I trained an SVM model by using vector representations of known verses by Nguyễn Du and a contrasting corpus of non-kiều poetic lines. The TF- IDF features extracted from these verses served as input to the model. Upon the testing, the model achieved a 91% accuracy in distinguishing verses written by Nguyễn Du.

The precision, recall, and F1-scores for each class are well-balanced, demonstrating that the model is not biased toward either class.

```
=== Classification Report ===
               precision    recall  f1-score   support

Other Author        0.93      0.85      0.89       466
   Nguyễn Du        0.90      0.96      0.93       652

    accuracy                            0.91      1118
   macro avg        0.92      0.90      0.91      1118
weighted avg        0.91      0.91      0.91      1118
```

Image 8: Classification Result

Precision of 0.90 and Recall of 0.96 for Nguyễn Du, meaning it correctly identified most of his verses and made relatively few false positives.

Precision of 0.93 and Recall of 0.85 for Other Author, showing it was slightly more cautious but still accurate in rejecting non-Nguyễn Du verses.

The confusion matrix later confirms this, with 623 correct predictions out of 652 Nguyễn Du verse, and 398 correctly identified non-Kiều lines.The model only misclassified 68 "Other Author" lines as Nguyễn Du and 29 Nguyễn Du lines as other, highlighting its robustness.
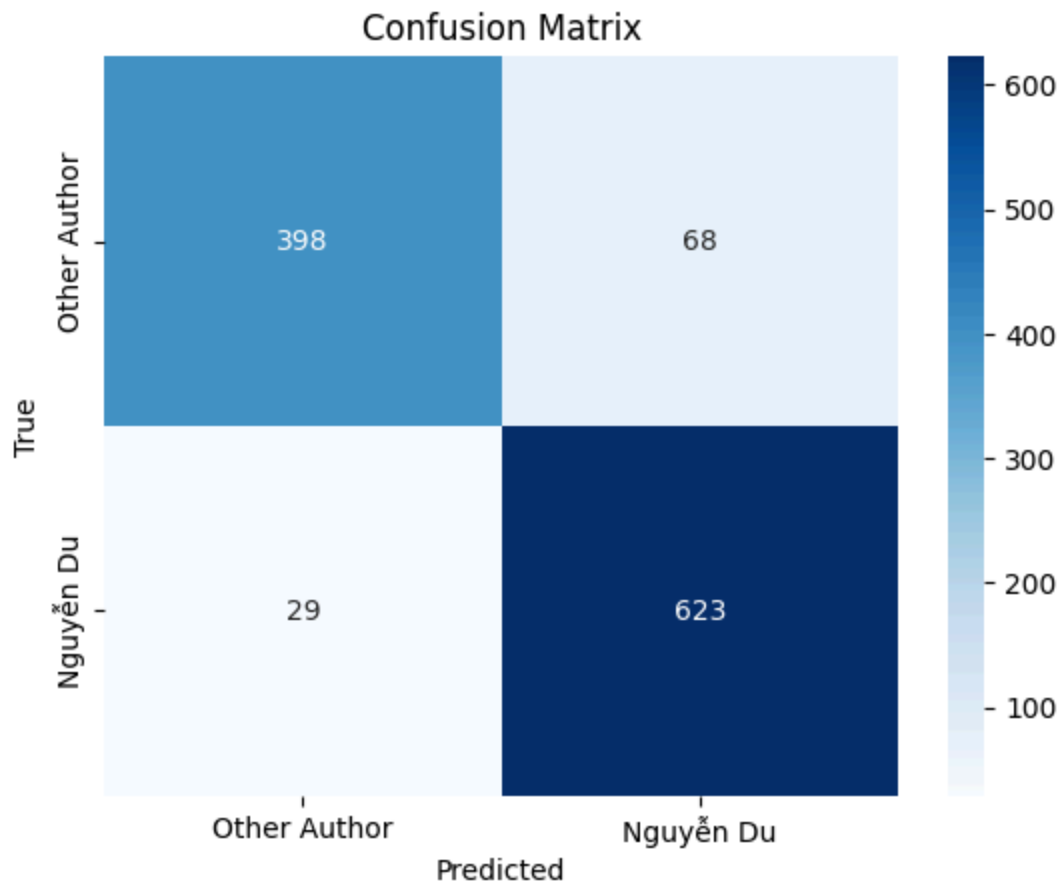
Image 9: Confusion Matrix

Image 10: Testing on Nguyễn Du verse

Truyen Kieu Search & Authorship Prediction

Classify Verse   Search by Cosine   Search by Overlap   Inverted Index

Input a verse to predict the author

bay mất vào bóng tối mênh mông

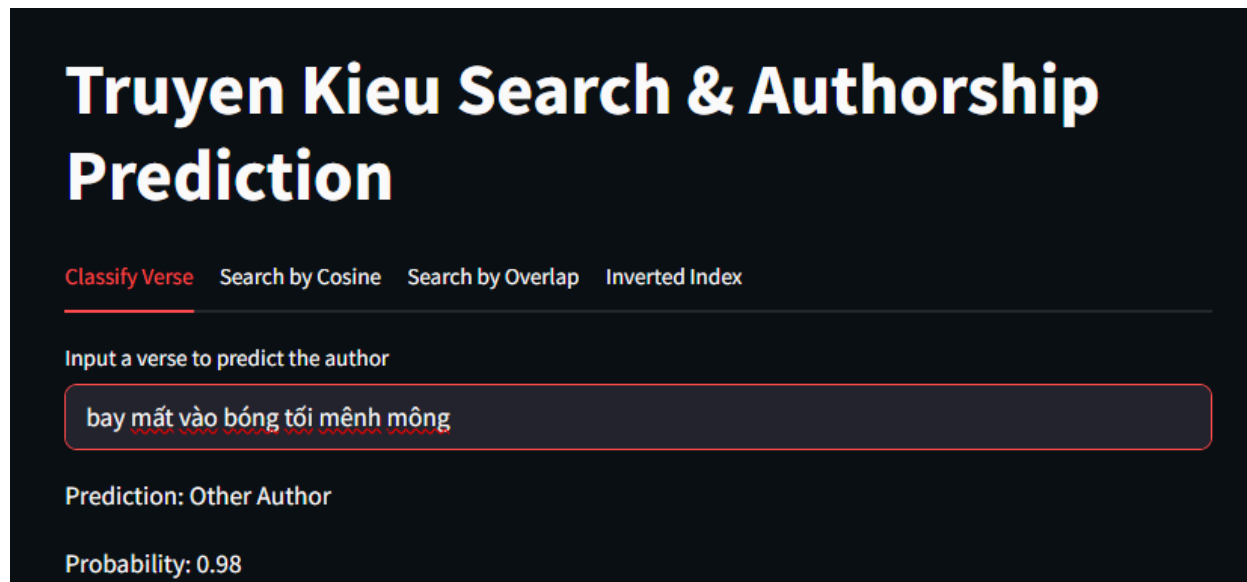Prediction: Other Author

Probability: 0.98

Image 11: Testing on a Haiku verse

These results suggest that stylistic features embedded in the vector representation of each verse, such as frequent word pairings and phrase structures, can reliably distinguish Nguyễn Du's poetic voice, supporting the effectiveness of vector space modeling for authorship attribution tasks in Vietnamese classical literature.

Limitations

While TF-IDF effectively captures word frequency patterns and stylistic tendencies, it lacks semantic and historical context awareness. Since Nguyễn Du and Nguyễn Trãi both wrote in classical Vietnamese with elevated, formal vocabulary, a TF-IDF-based model may confuse stylistic overlap as authorship similarity. Since the model was trained only on Truyện Kiều and one contrasting corpus (Haku), it is not familiar with other authors from the same era. This reinforces that a more nuanced model should be considered for robust, real-world authorship attribution across historical texts.

Image 12: Missclassification

Part 2: Language Modeling

1. *Building a language model based on the text of Truyện Kiều and extra sources*

In this part, I explore two approaches to building a language model using the Vietnamese classical text Truyện Kiều by Nguyễn Du. I aimed to create a model that generates new verses based on an initial input phrase. The two strategies implemented include: (1) building a language model entirely from scratch using PyTorch, and (2) fine-tuning a pre-trained Vietnamese GPT-2 model on the *Truyện Kiều* corpus. This dual approach allowed for foundational understanding and practical performance comparison in generative language modeling.

The first approach involves building a language model from scratch. The process began with data preprocessing, where each line of Truyện Kiều was treated as a separate verse. These lines were read from a text file and cleaned by stripping whitespace and removing special characters. Vietnamese word segmentation was applied using the underthesea library to tokenize the text into meaningful word units, which is particularly important for Vietnamese due to the language's multi-syllabic word structure.

```
Cleaned line: Chữ tài chữ mệnh khéo là ghét nhau.
Token IDs: [556, 130, 170, 412, 1221, 12, 1458, 75, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Tokens: ['Chữ', 'tài', 'chữ', 'mệnh', 'khéo', 'là', 'ghét', 'nhau', '.', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]',
Attention Mask: [1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Decoded text: Chữ tài chữ mệnh khéo là ghét nhau .
```

Image 13: Tokenization

The datasets are then wrapped in a PoemDataset class that returns input IDs, target IDs (for next-token prediction), and attention masks. Each training sample is a pair of input and shifted target sequences. For example, the first training item produces:

```
Number of valid stanzas: 814
Number of stanzas processed: 814
Number of samples: 814
Input IDs: tensor([ 363,   80,   45, 575,   11,   63,    5, 556, 130, 170, 412, 1221,
          12, 1458,   75,    6, 1813, 199,    8, 965, 203, 724,    5,    1,
           1])
Target IDs: tensor([ 168,  107,   93,   60,   37, 258, 1084,  16,    6,    1,    1,    1,
            1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
            1])
Attention mask: tensor([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
          1., 1., 1., 1., 1., 0., 0.])
```

Image 14:

A single training example from the custom dataset showing input IDs, target IDs, and the corresponding attention mask. Padding ensures equal sequence lengths while the attention mask filters it out during training.

Sample Construction and Transformer Model

To enhance the model's ability to capture long-range dependencies in Truyện Kiều, training samples were constructed using a prefix-based strategy. Each four-line stanza was encoded with [SOS], [EOL], and [EOS] tokens, then split into incremental input-target pairs. Padding and attention masks ensured fixed-length sequences and proper masking during training.

A Transformer-based model was implemented, consisting of an embedding layer, sinusoidal positional encoding, multiple TransformerEncoder layers with multi-head self-attention, and a linear decoder. The architecture supports padded input via key padding masks, which were initialized using Xavier uniform initialization.

Beam Search Decoding

Beam search decoding was applied to the trained Transformer model to evaluate generation quality. Beam search explores multiple candidate sequences at each decoding step, selecting the most probable complete sequence based on accumulated log-probabilities. This deterministic method yields more coherent outputs than greedy decoding, thus avoiding early commitment to suboptimal tokens.

However, in practice, the model produced repetitive and degenerate outputs with the prompt = "[SOS] Trăng "
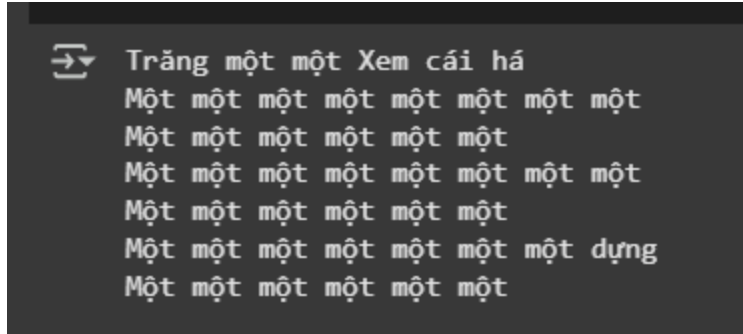
Image 15: Output from Beam Search Decoding

The error then indicates over-reliance on high-frequency tokens and insufficient regularization.

Temperature-Controlled Sampling

Next, I implement temperature-controlled sampling to improve the diversity of verse generation. The method modifies the models output distribution by scaling logits with a temperature parameter before applying softmax. A temperature of 1.2 was used to increase randomness and avoid repetition while preserving syntax. Given the prompt = "[SOS] quang"
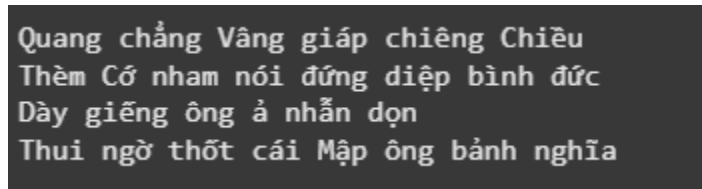

Image 16: Output after control

Although semantically imperfect, the output displays rhythmic and lexical variation, making it stylistically closer to Vietnamese poetry than earlier outputs. A formatting function was used to enforce the traditional lục bát structure by alternating lines of six and eight syllables and applying tone-based validation. Compared to beam search, this method produced more natural and engaging text, confirming the effectiveness of stochastic decoding in poetic generation.

Approach 2: Fine-Tuning GPT-2 on Truyện Kiều

In addition to the from-scratch Transformer model, a pre-trained Vietnamese GPT-2 model (danghuy1999/gpt2-viwiki) was fine-tuned on Truyện Kiều to compare performance. GPT-2, a generative transformer trained on large-scale Vietnamese text, provides strong prior linguistic knowledge, making it a suitable candidate for domain adaptation to classical poetry. The text was tokenized using GPT-2's tokenizer and formatted into a Hugging Face Dataset for efficient training. Special tokens such as [SOS], [EOS], and [EOL] were preserved to maintain verse structure. In this phase, I also apply the lowercase to avoid the uppercase word that will appear in the middle of the sentence. Fine-tuning was conducted

using the Hugging Face Trainer API with parameters including a batch size of 4. Due to Colab's resource constraints, I could only train for 5 epochs.

```
<ipython-input-178-9aeafd9022fc>:20: FutureWarning: `tokenizer` is deprec
  trainer = Trainer(
                                                    [1835/1835 06:07, Epoch 5/5]
```

Epoch	Training Loss	Validation Loss
1	No log	6.460541
2	5.948500	6.272570
3	5.273300	6.207601
4	5.273300	6.203349
5	4.970300	6.212679

```
TrainOutput(global_step=1835, training_loss=5.284172037316928, metrics={'
'train_samples_per_second': 39.845, 'train_steps_per_second': 4.989, 'tot
```

Image 17: Training Log

```
GPT2LMHeadModel(
  (transformer): GPT2Model(
    (wte): Embedding(50257, 768)
    (wpe): Embedding(1024, 768)
    (drop): Dropout(p=0.1, inplace=False)
    (h): ModuleList(
      (0-11): 12 x GPT2Block(
        (ln_1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (attn): GPT2Attention(
          (c_attn): Conv1D(nf=2304, nx=768)
          (c_proj): Conv1D(nf=768, nx=768)
          (attn_dropout): Dropout(p=0.1, inplace=False)
          (resid_dropout): Dropout(p=0.1, inplace=False)
        )
        (ln_2): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (mlp): GPT2MLP(
          (c_fc): Conv1D(nf=3072, nx=768)
          (c_proj): Conv1D(nf=768, nx=3072)
          (act): NewGELUActivation()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (ln_f): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  )
  (lm_head): Linear(in_features=768, out_features=50257, bias=False)
)
```

Image 18: Model Evaluation

The first generating attemp with prompt = "thương sao cho trọn thì thương" seems so weird

```
Generated poem:
 thương sao cho trọn thì thương lòng, chẳng thương cả hai bên kia, nên ai cũng thương lòng này là người này hay không
```

Image 19: First generation

I developed a tone rule checker that enforces Vietnamese poetic conventions and then combined it with the GPT-2 model. A post-processing function validated generated verses against expected tone patterns, specifically requiring alternating six- and eight-syllable lines and correct placement of "bằng" and "trắc" tones at fixed positions in eight-syllable lines. Generation was performed iteratively using top-k and nucleus sampling (top_k=40, top_p=0.95) with temperature set to 0.9. If a sample failed to meet the structural criteria, a new attempt was generated until either a valid poem was produced or a maximum number of retries was reached.

Image 20: Second generation after filtering and formatting

The generated poem seems better than the previous one but still seems senseless. I decided to implement a tone-checking mechanism. This system validated each 8-syllable line by enforcing correct tonal positions: bằng tones on syllables 2, 4, 6, and 8, with rejection of any line violating this constraint. If a generated output failed the rule, it was discarded, and generation was retried.
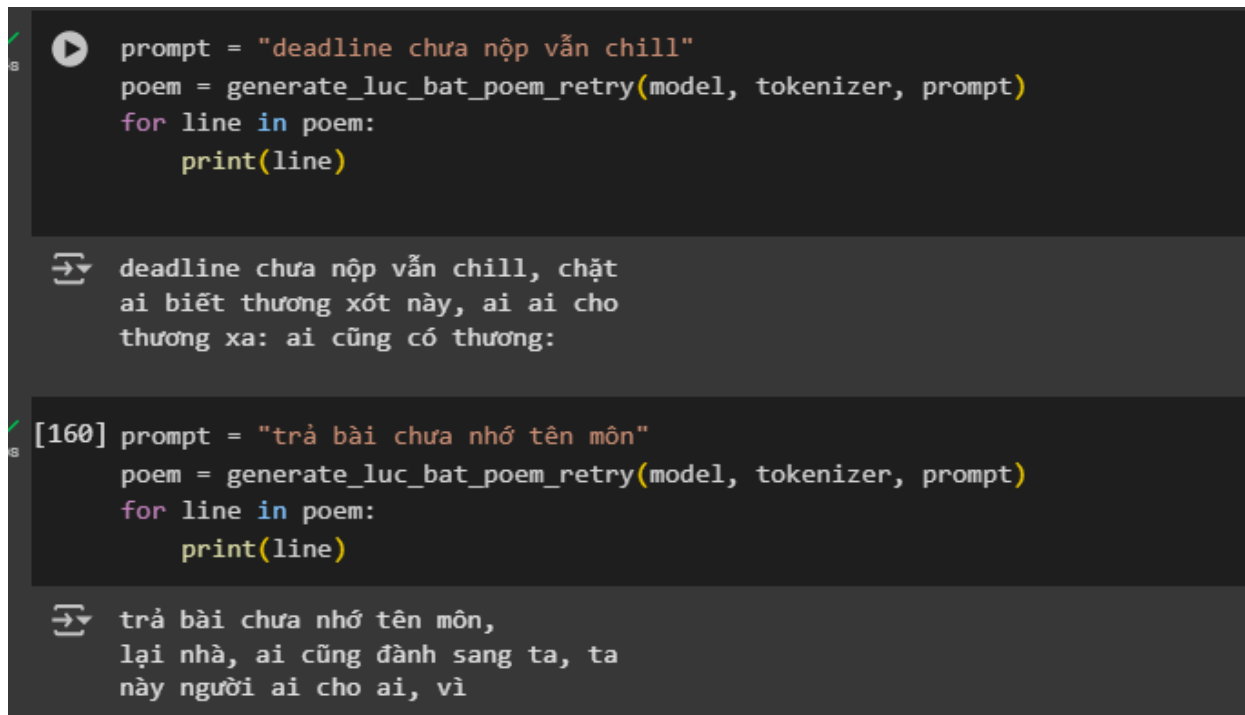
This rejection-based filtering method proved effective but computationally expensive. In one example, only 1 out of 10 attempts produced a fully valid stanza. The process iterated through dozens of rejected outputs with incorrect tone patterns (e.g., misplaced "trắc" syllables), before successfully producing

Image 20: Tone Rule Filtering in Lục Bát Generation with Retry Log

To evaluate the model's adaptability beyond traditional poetic themes, I tested it using contemporary verses:

```
prompt = "deadline chưa nộp vẫn chill"
poem = generate_luc_bat_poem_retry(model, tokenizer, prompt)
for line in poem:
    print(line)
```

```
deadline chưa nộp vẫn chill, chặt
ai biết thương xót này, ai ai cho
thương xa: ai cũng có thương:
```

```
[160] prompt = "trả bài chưa nhớ tên môn"
poem = generate_luc_bat_poem_retry(model, tokenizer, prompt)
for line in poem:
    print(line)
```

```
trả bài chưa nhớ tên môn,
lại nhà, ai cũng đành sang ta, ta
này người ai cho ai, vì
```

Image 21: Testing with modern language

These results showcase how the model retained the prompt's casual tone and rhythm while maintaining structural integrity. The output aligns well with the lục bát form despite the use of modern slang like "chill." Although slightly fragmented in meaning, the model still generated correct syllabic structure and poetic phrasing, demonstrating its ability to blend classical structure with informal.

Finally, the fine-tuned GPT-2 can flexibly respond to stylistic shifts while respecting the formal constraints of *lục bát* poetry based on the result. This opens potential for creative AI-assisted poetry generation in classical and contemporary Vietnamese literary styles.

2. *Generate an image from a given input verse.*

One feasible approach involves leveraging pretrained text-to-image models such as DALL·E 2, Stable Diffusion, or Gemini. These models accept natural language prompts and generate corresponding visual scenes. However, since the verses in Truyện Kiều are metaphorical and culturally rich, they often require preprocessing in the form of prompt expansion, translating poetic lines into vivid, descriptive English that the model can interpret effectively.I input the prompt in gemini and receive the following image

Image 22: Image generating prompt

Image 23: Image inspired by the verse: "cỏ non xanh tận chân trời, cành lê trắng điểm một vài bông hoa"

3.   *Find a Verse in Truyện Kiều that is Most Relevant for a Given Image*

To explore the potential of cross-modal retrieval between images and poetic text, we prompted Gemini to identify the most semantically relevant verse from Truyện Kiều for a given input image—a "dark, gloomy landscape with a country road in snow."

This suggests that multimodal language models can effectively match abstract visual mood with poetic themes, even when exact lexical overlap is absent. Such capabilities demonstrate the feasibility of using AI for cultural and literary interpretation across modalities, though fine-tuning would be necessary for higher precision.
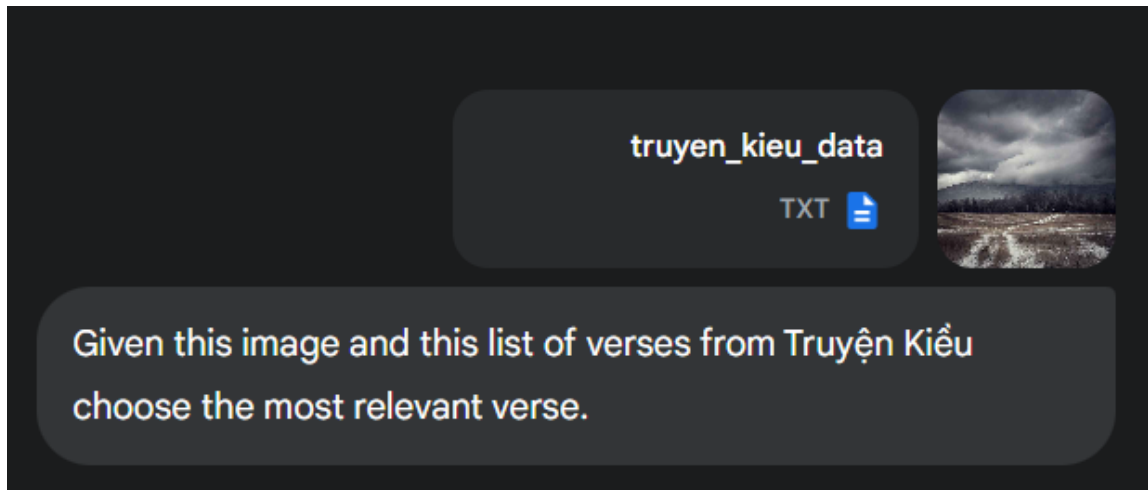

Image 24: Generating verse from a given Image

Image 25: Given image

Although no verse directly described this scene, the model successfully retrieved several thematically appropriate options that conveyed a sense of melancholy and solitude. For instance, "Bóng tà như giục cơn buồn, Khách đã lên ngựa, người còn ngẩn ngơ." (The setting sun urges sadness, the guest has mounted his horse, the person still follows.)

Based on the image and the search results, it's difficult to find a direct verse that perfectly matches the "Dark Gloomy Landscape with Country Road in Snow". However, some verses evoke a sense of melancholy and bleakness, which might be relevant to the image's atmosphere. Here are a few options:

- **"Bóng tà như giục cơn buồn, Khách đà lên ngựa, người còn nghé theo."** (The setting sun urges sadness, the guest has mounted his horse, the person still follows.) This verse depicts a melancholic scene at sunset, which can be interpreted as gloomy.

- **"Mảnh trăng chênh chếch dòm song, Vàng gieo ngấn nước, cây lồng bóng sân."** (The slanting moon peers through the window, gold scatters on the water, trees cast shadows on the yard.) This verse creates a sense of loneliness and darkness, although it doesn't explicitly mention snow.

- **"Ba thu dồn lại một ngày dài ghê. Mây Tần khóa kín song the, Bụi hồng lẽo đẽo đi về chiêm bao."** (Three autumns compressed into a terribly long day. Qin clouds lock the thin window, pink dust wearily goes back and forth in dreams.) This verse conveys a feeling of confinement and gloominess.

It's important to note that these are interpretations, and Truyện Kiều is rich in imagery, allowing for multiple valid interpretations. Would you like me to explore other verses or aspects of the

Image 26: The final result

This suggests that multimodal language models can effectively match abstract visual mood with poetic themes, even when exact lexical overlap is absent. Such capabilities demonstrate the feasibility of using AI for cultural and literary interpretation across modalities, though fine-tuning would be necessary for higher precision.

Conclusion

The project explored how AI can be applied to Vietnamese classical literature, using Truyện Kiều as the core dataset. The vector space model enabled effective verse search and authorship classification, showing that even simple TF-IDF methods can capture poetic style.

I built and fine-tuned language models in the second part to generate new verses. While the from-scratch model faced limitations, the fine-tuned GPT-2 produced "structurally" lục bát poetry, especially when combined with tone-checking and filtering techniques.

Finally, by testing Gemini's ability to link poetry with images, we showed that AI can match poetic verses to visual scenes and vice versa. These experiments highlight the creative potential of multimodal AI in understanding, generating, and interacting with Vietnamese poetry. Though far from perfect, the results suggest promising directions for combining AI with literary and cultural work.

References:

Duyet. (n.d.). Truyen Kieu word2vec.ipynb [Jupyter notebook]. GitHub.
https://github.com/duyet/truyenkieu-word2vec/blob/master/word2vec.ipynb