LiBai: A Comprehensive Chinese Poem Analyzing System

EBA5004: Practical Language Processing

Group 8

Full Name	Student ID
Yan Jiahuan	A0261968M
Li Qinyuan	A0261801N
Liang Zijian	A2062016R

Project Supervisor: Dr. Wang Aobo

Contents

1 | Introduction

Chinese poetry has always been rich and complex, often requiring a deep understanding of the language and culture so that one can fully appreciate it. These literary masterpieces typically possess unique linguistic structures and multiple layers of meaning, making their analysis quite challenging. This project is aimed at leveraging modern natural language processing techniques to break down these complex structures into more accessible plain text and annotations. This will enable poetry lovers to better understand and appreciate Chinese poems in a more comfortable and enjoyable manner.

1.1 | Background

In an era of rapid globalization and cultural fusion, the importance of preserving and understanding traditional cultures cannot be overstated. One such cultural heritage that carries profound historical significance is classical Chinese poetry. However, the intricate language systems employed in these pieces of art pose substantial challenges to contemporary readers, especially those accustomed to modern colloquial Chinese. The differences in grammar, vocabulary, and expressive nuances between the two language systems often create barriers to comprehension, thus hindering the appreciation of the rich tapestry of emotions, allusions, and imagery inherent in these classical poems.

Here is a simple demonstration of the hardness for Chinese beginners or non-native speakers to understand one Chinese poem thoroughly. ”会当凌绝顶，一览众山小”, the first level of confusion is not even understanding the semantics of this poem, which means their cognition remains at the stage of word translation; Medium players may understand it is the description of the grandiose of mountain Tai; Advanced player will bring their cognition to the peak of the understanding towards this utterance: To praise the human tenacity to overcome adversity, and finally get the joy of success.

It is hard for non-professional Chinese speakers (including most Chinese people) to understand a poem/poetry when it is unfamiliar to them. As the difficulty may arise from the ancient words, special grammatical structure and hidden background information which may not be visualized textural-based. Hence, we decided to use certain AI tools to solve the problems and build a comprehensive, beautiful and useful system to help all Chinese poem lovers.

1.2 | Overall Design

Our project proposes an innovative solution to this problem, focusing on the development of a comprehensive poetry analysis application specifically designed for classical Chinese poetry. This application aims to alleviate the issues of incompatibility between the classical and contemporary Chinese language systems, ensuring the tradition remains alive and accessible to modern readers.

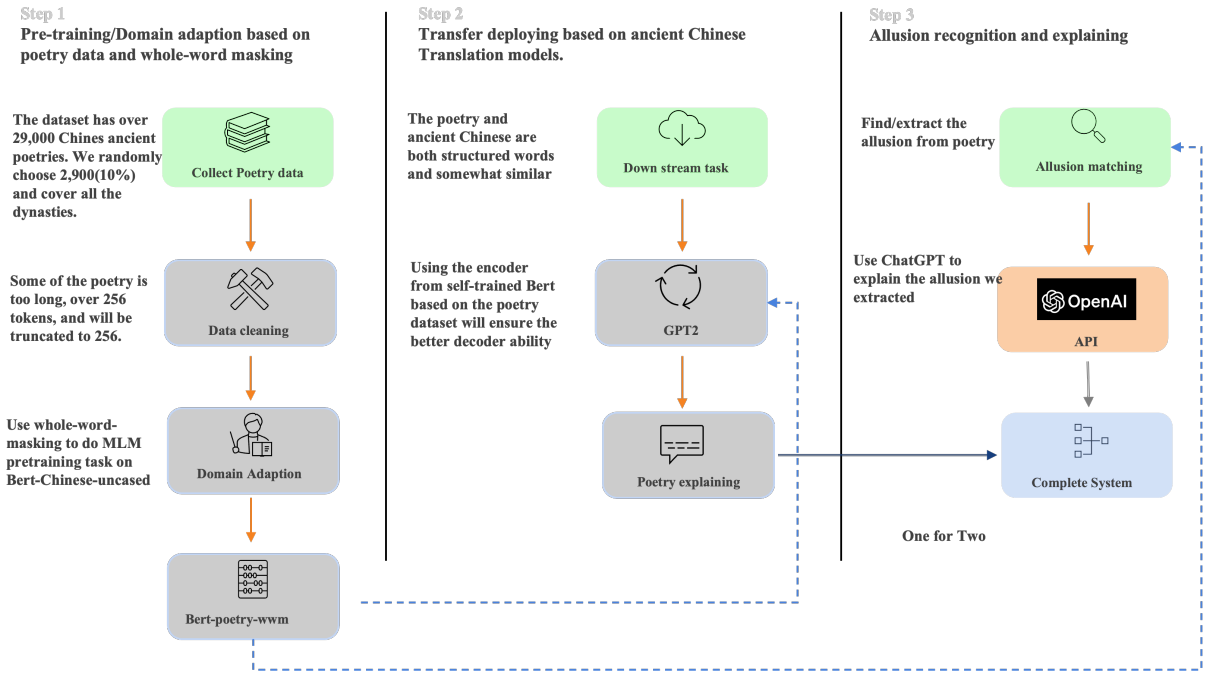

Figure 1.1: The overall system design.

The approach to this project is three-fold, centering on the translation of classical poetry into contemporary colloquial language, visualization of the imagery embedded in the poems, and comprehensive allusion analysis. By presenting the content of the poetry in a more accessible manner, we seek to enhance the reader's understanding and appreciation of the ancient works.

1.3 | Business Value

The potential applications of this project are vast, spanning cultural tourism, museum guide services, and educational training fields. Our solution can be instrumental in enhancing the presentation of classical poems, reducing comprehension difficulties, and ultimately, making the dissemination of traditional Chinese culture more engaging and less daunting. Furthermore, the project stands to make a substantial impact on primary and secondary school Chinese language classrooms, providing students with a vivid, interactive means of understanding and appreciating classic cultures. By doing so, we aim to not only enhance their humanistic literacy and language proficiency but also lay a robust foundation for their future development.

In the report that follows, we delve into the details of our project, elucidating the challenges we encountered, the methodologies we adopted, and the solutions we developed in our quest to bring classical Chinese poetry closer to the contemporary audience. We firmly believe that by bridging this gap, we can open up a whole new world of understanding, appreciation, and love for this significant part of Chinese culture.

2 | Datasets and Training Environment

2.1 | Datasets Introduction

- **Domain adaption task:**
  The Chinese-poetry database[1] is a comprehensive database of Chinese poetry, comprising a total of 55,000 Tang poems, 260,000 Song poems, and 21,000 Song lyrics and other forms of poetry. The dataset is too large for our hardware, thus we collected 29,000 poetries from all dynasties (Tang and Song as the majority). This practical setup can help us allievate the training consumption and also cover the poetry domain at certain level.

- **Translation task:**
  The Classical-Chinese database[2] is a parallel corpus consisting of classical and modern texts. It provides comprehensive coverage of traditional ancient writings and was derived from online sources. The corpus includes a total of 960,000 sentence pairs, divided by Chinese periods, semicolons, exclamation marks, and question marks, as shown in Figure 2.1. The translation of modern texts in the Classical-Chinese database is achieved by manual translation.

曰：否。	那年轻人回答说：没有。
绐问童子，亦如之。	就像先前一样，编造谎言问那些孩子。
帝固邀与宿，凡其留者，悉饮食之。	太祖坚持邀请照烈和自己同宿，凡是留下来的，全都让他们又吃又喝。
吾儿宜识之。	我儿应当记住啊。
君能益吾右翼，吾将夺其弧矢也。	您要是能够增援我的右翼，我就能够取代他的地位。
帝悦，曰：以此众战，何忧不胜。	太祖很高兴地说：凭着这样的部下去作战，还担心不能取胜么？
今吾观其气势，殆非往时矣。	现在我看他们的气势，恐怕再不是从前的他们了。
明日，余众悉降。	第二天，剩下的残部全都投降了。
帝瘳。	皇上病愈。

Figure 2.1: Classical and Modern Texts Pairs on Translation Task Dataset

- **Allusion task:**
  The original database used for model training is the same as the domain adaption task. The actual allusion dataset used for model training and preprocessing is generated by data preprocessing on the original database.

2.1.1 | Data Pre-processing

For poetry domain adaption, the main processing is set words boundaries, where we tried several ways, and the best result is from LTP[1], which will be discussed in section 3.

For the translation part, the data is already been divided into classical text and human translation text sentences pairs. We use 40,000 sentence pairs from different generations. Besides, Punctuation provides information about the structure of the sentences. Allowing the model to learn both punctuated and unpunctuated data helps to enhance its understanding of language and its perception of context. To adapt the different input methods and improve the robustness of our model, we randomly remove 50% of the punctuation in the training data.

[1] LTP: a platform also a useful tool in Chinese NLP

For the allusion part, since the Chinese-poetry database cannot be directly used for fine-tuning the model of the allusion task, we retrieved the verses containing allusion annotations in the database, and included a target according to each piece of data The format of poetry and a candidate allusion is stored. As shown in Figure 2.2, the label is 1 if there is a reference relationship between the allusion and the poem, and 0 if not.


```
"data": [
    {
        "target_sentence": "藕丝秋色浅，人胜参差剪。",
        "candidate_allusion": "人胜",
        "label": 1.0
    },
    {
        "target_sentence": "小园芳草绿，家住越溪曲。",
        "candidate_allusion": "东篱",
        "label": 0.0
    },
    {
        "target_sentence": "两蛾愁黛浅，故国吴宫远。",
        "candidate_allusion": "吴宫",
        "label": 1.0
    },
```


Figure 2.2: Allusion Dataset

2.2 | Training Environment

2.2.1 | Linux Server

The main training procedure is conducted in Linux clusters, with two nodes (4*V100 GPUs and 2*A6000 GPUs). The detailed hardware information is listed in Table 2.1.

Node 1	Operating System	Ubuntu 20.04.4 LTS
	GPU	2 * NVIDIA GeForce RTX A6000
	CPU	AMD EPYC 7643 48-Core Processor
	RAM	48GB DDR4
Node 2	Operating System	Ubuntu 18.04.6 LTS
	GPU	4 * Quadro GV100
	CPU	Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz
	RAM	32GB DRR4

Table 2.1: Hardware Information of the Two Nodes

2.2.2 | Front-end Environment

The front-end uses the Vue architecture and calls Flask library for GET and POST requests to connect with the back-end API to achieve the parameter transfer. Our front-end uses third-party libraries, Vuetify and Axios, making the framework easier to build.

The environments of front-end are as below:

- Node.js 14.20.1

- npm 8.19.2
- Vue 2.9.6
- node-sass 4.14.1
- sass-loader 7.3.1

2.2.3 | Runnable System Installation

To run and test our project through website, you could follow the steps below to achieve.

Shell commands

```
$ cd runnableProgram group8
$ pip install tensorflow
$ pip install flask
$ pip install flask-cors
$ pip install numpy
$ pip install pandas
$ cd my-project
$ python backend/
```

3 | Language Modelling in Poetry Domain

The using of bidirectional encoding representation from Transformers (Bert) [3, 4] has been proved efficient as a paradigm of language embedding. This architecture has also been widely used as an inputting embedding strategy in marvelous downstream tasks. The researchers have pre-trained different Bert based on specific languages (e.g., Chinese, German, French) by simply using a pre-train methodology named mask language modelling (**MLM**) inspired by Cloze task[5]. The overall concept of this method looks alike a wide known game called "Cloze", which is to make a sentence or short phrases with blanks complete by filling in the right words. This "Cloze" game has been used as a standard to train and test the language model.

Bert-base-chinese is one of the version of Bert trained on specific Chinese dataset collected from Chinese WiKipedia. It is also used the initial check point on downsteam tasks related to Chinese. However, there are **two** limitations in this model/embedding strategy:

1. The representation of Chinese is trained from single WordPiece tokens instead of short phrases (e.g., ["古", "诗"] intead of ["古诗"]) which may cause semantic fragmentation issue.

2. It is trained on Chinese corpus instead of Poetry corpus, which may cause deviation.

This project has proved this two limitations solid by demonstrating the performance improvement after using our own model instead of the original Bert, the evidence has been listed as Table or Graph in the following section of this report.

This section will illustrate how the domain adaption (overcome deviation issue) and whole-word-masking (overcome semantic segmentation issue) are applied in our system.

3.1 | Whole-word-masking for Chinese

Whole-word-masking (**WWM**) [6] is one of the methods to overcome the issue cased by using WordPiece tokenization and embedding. The concept is simple: if one word is masked, its subwords (e.g., for word "WordPiece", the "Word" is considered as main word, "##Piece" is defined as subwords by Google to overcome the rare words issue) should also be masked. This strategy was proposed to alleviate the influence brought by rare words (words not in vocab list), as normally they can divide the rare words into some known words. And it has been proved useful in English corpus.

Chinese characters have different representation preference with English or English-like languages, as Chinese characters are not formed by alphabet-like symbols but individual lexical symbols, thus only the words-level seperation is suitable for Chinese instead of word-level.

Inspired by Dr. Cui's research [7], we proposed the similar solution to deal with Chinese character embeddings. We use LTP tool to automatically set the boundary of words, use space placeholder as the marker of two words, if the masking procedure covers one of the word in two space placeholders, each word there will be masked simultaneously. The structure of this part has been illustrated in following pseudocode.

Algorithm 1 wwm for Chinese by using LTP

Here N represents all data in corpus. W means word. M means the stats of one specific token (masked or not)

for $i = 1$ to N **do**
  boundary = ltp.pipeline(i, tasks = ['cws'])
  if $w.state$ in boundary $= M$ **then**
    $w[:].state = M$
  else
    pass
  end if
end for

3.2 | Domain Adaption

The concept of domain adaption is simple: Bert-base-chinese is not trained in Poetry dataset, which makes it self-evident that the model may not be fit in downstream tasks like poem translation or classification, as the poem/poetry has even more structured linguistic representation than Chinese articles. The scope of this project is done under the Chinese poetry domain, but we will not cover the very rare words in poems. It is possible that some words like "葳蕤" exists in a very limited number of poems but they are not covered in the original 21128 vocabularies of bert-base-chinese. In that case, our solution is to replace it by '[UNK]', which is a special character in Bert to represent unknown words.

In our project, we use the data from a public repository in github, and we organzie the data into a structured dataset, which contains 29,000 poems in main dynasties like Tang, Song. The longest poem has over 80 sentecnes and contains more than 600 tokens, which means it has exceeds the max length of 512. If any poem is exceeding 512 in token size, we will truncated it as an easy solution to avoid the error.

The initial checkpoint we use is bert-base-chinese from an open source web community named huggingface. This checkpoint has already been trained by **MLM** on a large Chinese corpus, we use it as the initial weights and continuously train our own model.The architecture of this procedure is shown in Figure 3.1. We follow the instruction of original paper and still choose 15% as the mask ratio, which means the we will randomly select 15% of tokens in each sample to conduct a masking procedure. The selected words with its corresponding words (set by aforementioned boundaries) will be replaced by a special token '[MASK]', the processing structure of this part has been visulaized in Figure 3.2.

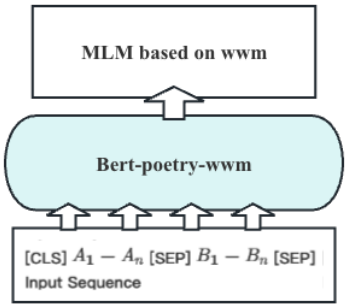

Figure 3.1: Neural architecture of Bert-poetry-wwm.

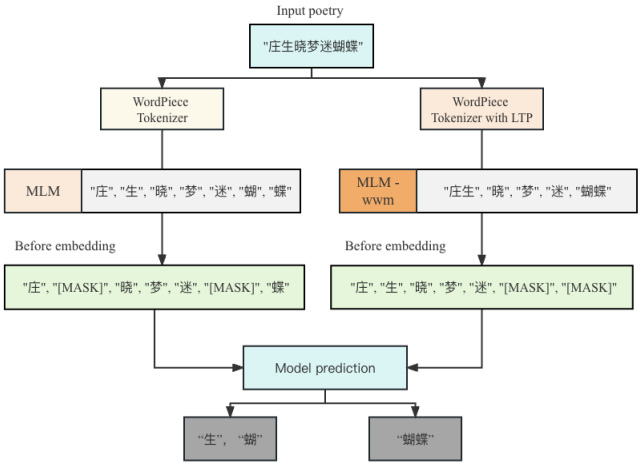

Figure 3.2: The processing pipeline with MLM and WWM.

After domain adaption, we will pop the last several classification layers to make the alignment for down stream task model initiation, which is shown in Figure 3.3.

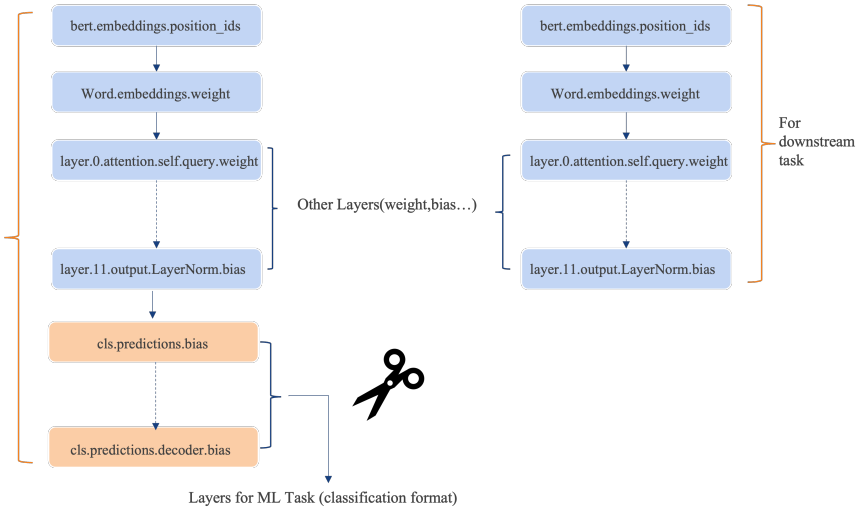

Figure 3.3: Pop classifiers so suite down stream tasks.

The model pre-trained by MLM will automatically add classifiers near the output layer, and we need to manually pop this layers when apply it as a language model instead of classification model.

4 | Translation Model

The process of representing poems and lyrics in modern text can be equated with a translation task, and we implement it by applying a translation model.

4.1 | Model Design

In this part, we implement a translation model from poems and lyrics to modern text based on the sequence-to-sequence model.

The sequence-to-sequence model is constructed by an encoder part and a decoder part. The encoder is responsible for converting the input sequence into a fixed-length context vector. The decoder receives the context vector from the encoder and generates a target sequence corresponding to the length of the input sequence.

In our model, the encoder part converts the poetic phrase to a set of hidden states, while the decoder part uses these hidden states to generate the plain text.

We used our Self-Trained-Bert-Poetry which is explained in Chapter 3. as the encoder and used GPT-2-Chinese-Poem[8] as the decoder, and implemented an attention-based sequence-to-sequence model, as shown in Figure 4.1.

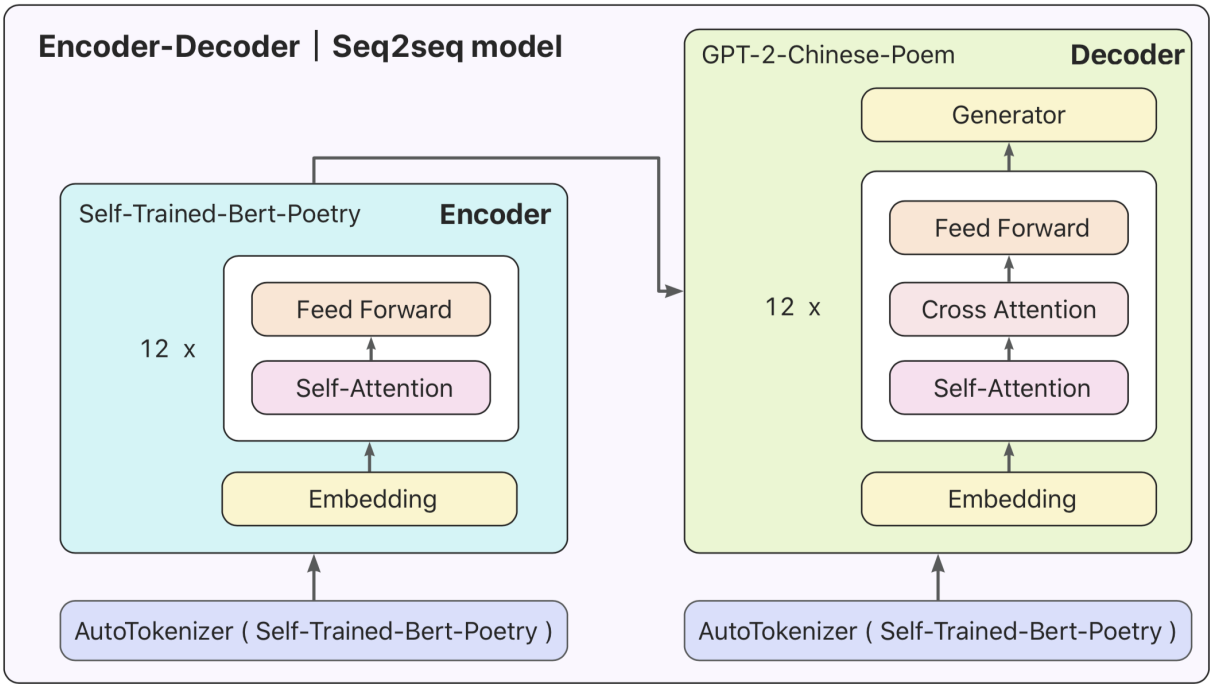

Figure 4.1: Translation Model Architecture

For the encoder part, Self-Trained-Bert-Poetry contains 12 Transformer Encoder layers. Each layer consists of a self-attentive mechanism and a feed-forward neural network. The self-attentive layer is used to capture the contextual relationships within the input sequence, and the feedforward neural network layer is used to perform a nonlinear transformation on the output of the self-attentive sublayer.

The encoder part also includes a pooling layer (BertPooler) for extracting the semantic representation of the whole sentence from the last layer of hidden states of the encoder. For

the decoder part, GPT-2-Chinese-Poem is a pre-trained model specifically for generating Chinese poems. It used a dataset containing 400,000 ancient Chinese poems to train and learn how to generate fluent language through unsupervised learning.

GPT-2-Chinese-Poem contains 12 Transformer Decoder layers, each layer consists of a self-attention mechanism, a cross-attention mechanism, and a feed-forward neural network. In particular, the self-attention layer is used to capture the contextual relationships within the decoder input sequence, the cross-attention layer is used to introduce the interaction information between the encoder and the decoder, and the feed-forward neural network layer is used to perform nonlinear transformations on the output of the attention layer.

The decoder also includes a linear transformation layer (lm_head) for mapping the hidden states of the last layer of the decoder to the probability distribution of the modern text vocabulary.

4.2 | Model Implementation

We used transformers to implement the construction of the seq2seq model. The model is built by introducing the *EncoderDecoderModel* from transformers.

Besides, we use AutoTokenizer to automatically load the Tokenize based on the name or path of the pre-trained model. Since our decoder part also uses Transformer and uses a similar input method as the encoder part, we use the tokenizer of our Self-Trained-Bert-Poetry in both the encoder and decoder parts, making it easier and more efficient to use Chinese language processing in our model.

In order to better control and adjust the weight update rate of each layer and control the gradient update magnitude when training the neural network model, we set different learning rates for different layers of the model, as shown in Table 4.1.

Encoder / Decoder	Layer	Learning Rate
encoder part	embeddings	1e-5
	feed_forward and self-attention	1e-5
	pooler	1e-3
decoder part	cross_attention	1e-3
	feed_forward and self-attention	1e-5
	linear_transformation	1e-4
encoder-decoder	/	2e-3

Table 4.1: Learning Rate in Different Layers

We chose Adam as the optimizer of the model. Also, in order to streamline the model and reduce the training time, we used Pytorch-lightning for the training.

5 | Allusion

In this part, we complete the task of allusion recognition and explaining. Allusions refer to words with extended meanings quoted in poems, and also generally refer to characters or events that are educational and familiar to the public. Compared with general poem entities, allusions cannot be directly translated because their extended meanings are more complex. Therefore, recognizing and explaining allusions is also an important part of lowering the threshold for understanding ancient poetry.

5.1 | Model Design

At present, there are two common directions for allusion recognition. The first one is rule-based method, which does not use machine learning. The disadvantage is that this method is unable to recognize indirect allusions. The second direction is an entity recognition based on natural language processing. Its disadvantage is that it cannot distinguish allusions from general entities.

Inspired by these two methods, we use the method of combining rule-based and pre-trained models. First, we use rules to screen the allusions containing keywords in the candidate allusion data set. Then, based on the fine-tuning of the allusion data set, the Bert-allusion model is used to calculate the sentence similarity between the target poem and the candidate allusion, and give less than ten allusions that may be cited, realizing the recognition of direct and indirect allusions.

Next, we call GPT-3.5-turbo API to explain the meaning of the allusion. With the definition of the allusion as the background information, as long as we enter the allusion with the highest sentence similarity, we can get the historical event and detailed extended meaning explanation of the allusion.

5.2 | Model Implementation

Model fine-tuning mainly involves the following parts, data preprocessing, text encoding, model packaging, training and testing.

In the data preprocessing part, we first load the training and testing datasets with a pre-divided data volume ratio of 1:7 from the file. Each dataset contains three parts of information: target sentence, candidate allusion and label. We extract these information separately as training and test cases. Then in the text encoding part, we used the BertTokenizer of the Bert-Chinese pre-trained model to encode the text, and also performed truncation and padding operations so that all inputs have the same length. Then we build datasets and data loaders for loading data in batches. And the fine-tuning model Bert-Allusion is defined in the model encapsulation section, which adds a linear layer on the basis of Bert-Chinese for binary classification tasks. And the parameters of the BERT model are frozen in advance to prevent them from being updated during the fine-tuning process. In the training and testing settings, we used the AdamW optimizer and set the learning rate to 5e-5. The loss function uses BCEWithLogitsLoss, which is commonly used in binary classification tasks.

6 | System Evaluation

6.1 | Domain Adaption Part

During the training, the loss function is based on traditional cross-entropy, and it is also seen as a simple evaluation metric to see whether the model is converge or not. However, this metric can only tell if the model predict the masked tokens correctly, it is not a comprehensive metric to tell if this model is suitable for being a language model or not. In short, there are better ways to evaluate a language model.

One of the widely recognized method is to run the model on different benchmarks in same setting. Considering the time and hardware capacity we have, we can't ensure all the benchmarks related to language (or Chinese to be specific).

	DRCD [9] EM/F1	CJRC EM/F1	XNLI [10] Accuracy	ChnSentiCorp Accuracy
Bert-base-Chinese	82.2/89.2	55.1/75.2	77.8	95.0
Bert-Chinese-wwm	**82.8/89.7**	**55.1/75.4**	**78.2**	**95.4**

Table 6.1: Ablation study based on several Chinese NLP benchmarks. DRCD: Delta Reading Comprehension Dataset; CJRC: China AI and Law Challenge; XNLI: The Cross-Lingual NLI Corpus; ChnSentiCorp: A Chinese corpus for sentiment analysis;

In Table 6.1 is the demonstration of the performance improvement by using **WWM** on Bert. The results are run on the same random seed and standardized by deviation in 10 times. However, we do not find any well organized benchmark for Chinese poem/poetry, the demonstration of domain adaption is then displayed in another format.

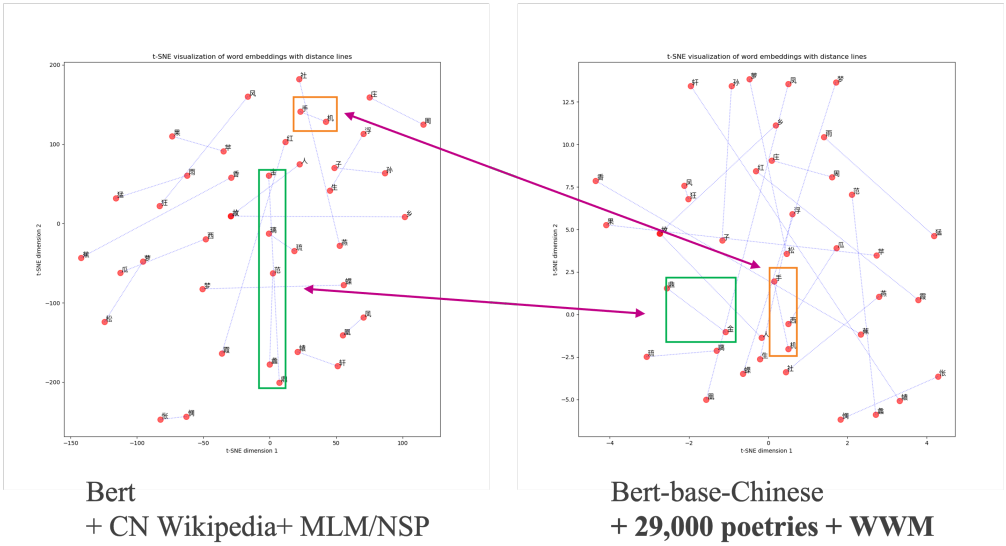

Figure 6.1: Visualization of domain adaption

In Figure 6.1, we try to visualize the result of domain adaption by using T-SNE [11], which is a method proposed by Geoffrey Hinton. It is reasonable to have the phenomenon that the distance in usual pairs of words in poetry get shorter after training, while the unusual pairs will have longer distance. For example: "手机" is a very usual word in

Chinese corpus like WiKipedia, while it may never be seen in any poetry or poem, thus the distance should be shorter so that "手" and "机" can have closer relationship with other characters that can be a potential candidates in poetry vocab space.

6.2 | Translation Part

6.2.1 | Parameters in Test Function

After the training is completed, we tested the optimal model using the *model.generate*() function, generating a sequence of vernacular text. In the test part, there are some hyper-parameters that can be modified, as shown below:

- *attention_mask*: the attention mask of the input sequence. It indicates which positions are valid and which positions are filled. The attention mechanism will decide whether to pay attention to the information in these positions based on the attention mask.
- *num_beams*: The width of the beam search during generation. Beam search is used to preserve multiple alternative output sequences during the generation process, and the first K possible results are preserved at each decoding step.
- *max_length*: The maximum length limit of the generated sequences.
- *bos_token_id*: token ID of the start token.
- *eos_token_id*: token ID of the end token. When the value of the token ID is set incorrectly, the complete sentence may not be translated.
- *pad_token_id*: token ID of the pad token. Pad tokens are usually used to pad the input to keep the length of the sequence consistent.
- *early_stopping*: Whether to enable early stopping policy.
- *no_repeat_ngram_size*: The size of the non-repeating n-gram. When generating sequences, this parameter specifies that consecutive n-gram tokens are not allowed to be repeated to avoid generating duplicate fragments.

Finally, we set $num\_beams = 3, max_length = 256, bos_token_id = 101, early\_stopping = True$.

6.2.2 | BLEU

For the evaluation part of the translation model, we will use BLEU (Bilingual Evaluation Understudy), which calculates the accuracy of translation results by comparing the output of our model with the reference translation.

More specifically, we compare the n-grams of the model translation output and the translation done by humans and compute the overlap between them, then calculate a score based on the degree of the overlap. the higher the BLEU score, the better the quality of the model output. We implement this through the *sentence_bleu* function in the NLTK library.

For n-grams (from 1 to the maximum n-gram), first calculate the number of exact matches for each n-gram in the candidate sentence c with the count ratio $p_n = \frac{M}{C}$. where M is the

number of exact matches between the candidate sentence c and the reference sentence r_i, and C is the count of each n-gram in the candidate sentence c.

Then the exact match count and count ratio of each n-gram are weighted and summed using the standard BLEU weights: $w = \left(\frac{1}{n}\right)_{n=1}^{N}$

$$BLEU = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log(p_n)\right)$$

Finally, the BLEU score is calculated by the formula above, where BP is Brevity Penalty.

The n-gram parameter in the BLEU evaluation indicates the length of the n-gram considered in the calculation of the BLEU score. Different n-values can have different effects on the performance performance of the model. Smaller n-values focus more on capturing phrase-level matches, while larger n-values focus more on capturing longer contextual relationships. Since poetry translations are mostly short sentences, we set $n = 2$.

Finally, we got the BLEU score of both training datasets and testing datasets, as shown in Table 6.2. If the sentence pair is same, the score should be 1.

	Score of Training Datasets	Score of Testing Datasets
BLEU Score	0.613	0.493

Table 6.2: BLEU Score

6.2.3 | Word2Vec Similarity

Since BLEU lacks Sentence Structure, to evaluate the results of single sentence translation output, we calculate the similarity between the two sentences, the model output and the human translation result. We obtain sentence similarity by measuring the word overlap and semantic similarity of sentences.

First, for the word overlap part, we use TF-IDF to evaluate the importance of words in the set and calculate the exact match sentence pair score. The key to measuring the consistency of two candidate sentences is to calculate how many words are identical between them. TF denotes the word frequency and is used to get the words that have many repetitions in the sentences. IDF denotes the inverse document frequency, which is used to filter out common words in the sentence and thus retain the critical words. Frequency-based weights are introduced for each word to emphasize the importance of low-frequency words, and the rarer words are given higher weights to obtain the words that best characterize the current sentence. Finally, all the words that overlap in a sentence pair are found and their weighted sums are obtained, which is the final word overlap score.

For the semantic similarity part, we generate sentence vectors by summing the word vectors and representing the similarity of the sentence pair by the angle of the sentence vectors, the smaller the angle, the more similar the sentence pair is. The Word2Vec model is applied to generate the word vectors. The vectors of the sentences are obtained by simply summing of the obtained word vectors. Finally, the similarity of the sentence pair vectors is represented by the angle of the vectors, and the semantic similarity score $S = 1 - \frac{u \cdot v}{\|u\|\|v\|}$ is obtained.

After assigning weights to these two values, the similarity of the sentences is obtained by adding these two values together. Since we mainly compare sentences but not long articles or documents, we assign a weight of 0.2 to the TF-IDF part and 0.8 to the Word2Vec part, and get the final score S, $S = \lambda \cdot S_{\text{tf-idf}} + (1 - \lambda) \cdot S_{\text{word2vec}}, \lambda = 0.2$.

Finally, we got the Word2Vec similarity of both training datasets and testing datasets, as shown in Table 6.3. When the Similarity is 1, means the sentence pair is same.

	Similarity of Training Dataset	Similarity of Testing Dataset
Word2Vec Similarity	0.862	0.698

Table 6.3: Word2Vec Similarity

Since our model is trained in the form of sentence pairs, that may be the reason that it has a weak understanding of the indicative pronouns, as shown in Figure 6.2.

```
Org:  不允。                          Org:  朝右皆惮之。
Translate:  没有答应他们的要求。      Translate:  朝廷的大官都很怕他。
Predict:  皇上没有批准。              Predict:  朝廷的人都很怕他。
0.680604213476181                     0.9355618596076966
```

Figure 6.2: Translation Result

6.3 | Allusion Part

To achieve the evaluation of the model results, we perform the same fine-tuning, encapsulation and training on the Bert-chinese model and our Bert-poetry model. We refer to the quantification methods of other papers in the same direction, and use the correct rate of correct allusions that appear in the first, top three and top five in the sentence similarity calculation results as a measure of model accuracy. Table 6.4 shows our test results. The Other model is the Bert-chinese fine-tuned allusion model, and the other is the result of the Bert-allusion model based on our upstream task.

	Accuracy in Top1	Accuracy in Top3	Accuracy in Top5
Other model	67	83	90
Bert-Allusion	**72**	**88**	**96**

Table 6.4: Allusion detection accuracy comparison

The results obtained by calling the GPT-3.5-turbo API are shown in Figure 6.3. We can see that by inputting the name of the allusion, we get the historical events of the allusion and the detailed interpretation of the extended meaning.

```
1. Allusions: 织锦
2. Background and meaning: 织锦 (zhī jǐn) literally means "weaving brocade". It is an
 allusion to an ancient story about a woman named Zhuo Wenjun, who was skilled in
 weaving brocade. In the story, she was forced to marry a man she did not love, but she
 expressed her emotions through her poetry and brocade weaving. The phrase 织锦 is often
 used in poetry to describe the intricate and beautiful weaving of emotions and thoughts
  into the fabric of the verse.
```

Figure 6.3: Background and Meaning of Allusions

7 | Front-end

We build a front-end system to use our model based on Vue, and use Flask to interact the front-end and the back-end.
On the main page, when input the poem into the text box, and press the bottom button, then users could wait for the translation results, as shown in Figure 7.1.

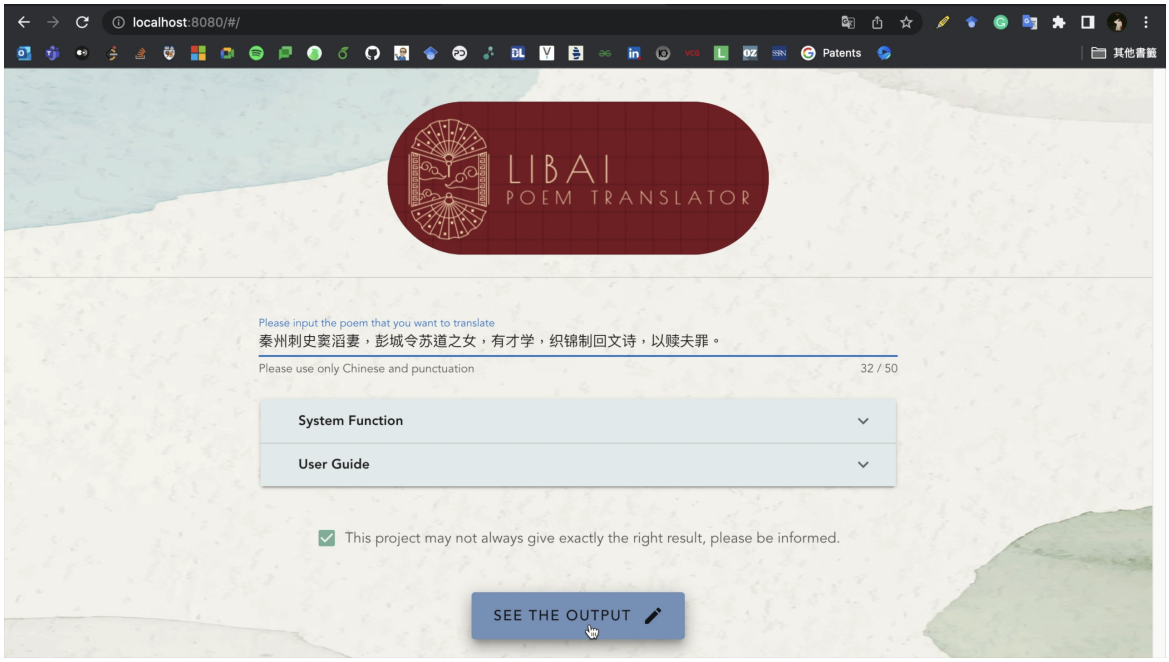

Figure 7.1: Main Page

On the results page, we will show the translation results and the allusion explanation of the poem, as shown in Figure 7.2 and Figure 7.3.

Figure 7.2: Translation Results in Results Page

Figure 7.3: Allusion Explanation in Results Page

Then, we will give an allusion ranking for the user to let them know more about similar allusions, as shown in Figure 7.4.

Figure 7.4: Allusion Ranking in Results Page

If users are interested in how we implement our system, they could also see our GitHub page by clicking the link, as shown in Figure 7.5.

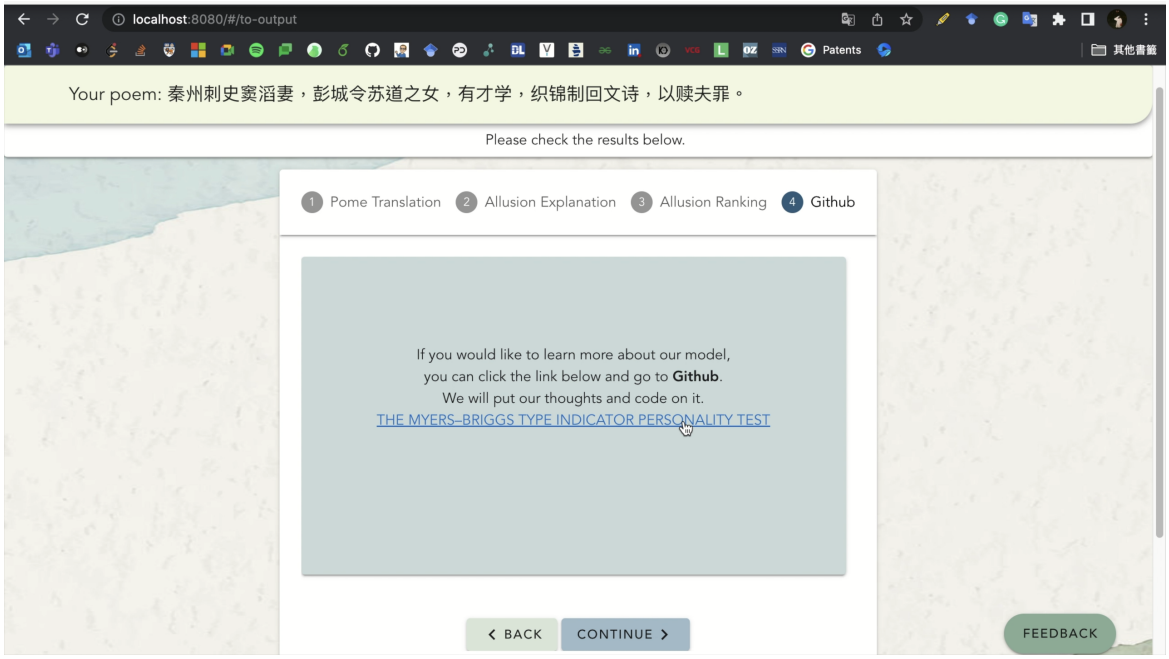

Figure 7.5: GutHub Link in Results Page

Users could also give us a feedback if they want, as shown in Figure 7.6.

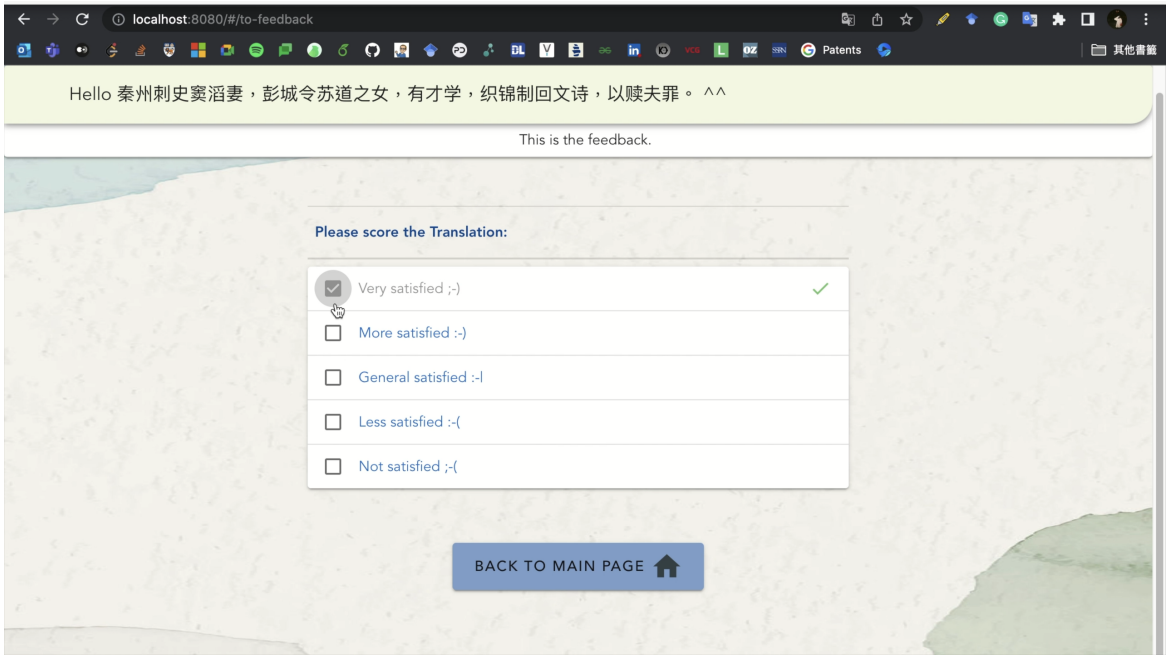

Figure 7.6: GutHub Link in Results Page

And they could conveniently go back to the start page to translate other poems. We hope the users could gain some inspiration from our system, learn more about poetry and enjoy the magnificent traditional Chinese culture.

8 | Conclusion

In this project, we proposed a complete system design for the demands of poem analysis. The entire system is composed of three parts: 1. Poetry language modelling. 2. Translation. 3. Allusion explaination. Each part has its own model architecture and performance evaluation, and they have been covered in detail in previous chapters.

From our research, the association of word representation does have positive effect to down stream tasks, and it has been analyzed in our ablation study in previous section.

We also achieved success in training and deploying our system, which is composed by our translation model and allusion model. The system can run smoothly under the right installation instruction, which can be found in our repo. [2]

8.1 | Future Wrok

This system has the following parts that can be improved, which will be implemented in future arrangements:

1. The data we applied in each section is not a union form, which can lead to domain bias issue to some extent [12]. There will be overlapping parts, but we haven't conducted any further research into it.

2. The mask-language-modelling strategy now for pre-training might not be sufficient. We will consider using masking-as-correction [7] instead in the future.

3. We will deploy our system in cloud to make it easier accessible for people who don't have hardware ensurance.

8.2 | Acknowledgement

This endeavor forms an integral part of our coursework in Practical Language Processing (EBA5004, NUS-ISS). We have been fortunate to embark on this journey under the astute guidance and mentorship of Dr. Aobo Wang and the dedicated teaching faculty at the Institute of Systems Science, National University of Singapore (NUS-ISS). Their invaluable insights and unwavering support have been instrumental in the realization of this project. We wish to express our profound gratitude for their contribution to our academic journey.

[2] Link to our peoject: https://github.com/YanJiaHuan/Poem_Analyst

9 | References

[1] Chinese-poetry database. https://github.com/chinese-poetry/chinese-poetry.

[2] Classical-chinese database. https://github.com/BangBOOM/Classical-Chinese.

[3] Charles W Bert and Moinuddin Malik. Differential quadrature method in computational mechanics: a review. 1996.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[5] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. Journalism quarterly, 30(4):415–433, 1953.

[6] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[7] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. Pre-training with whole word masking for chinese bert. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29:3504–3514, 2021.

[8] Gpt-2-chinese-poem. https://huggingface.co/uer/gpt2-chinese-poem.

[9] Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. Drcd: a chinese machine reading comprehension dataset, 2019.

[10] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations, 2018.

[11] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008.

[12] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, pages 67–73, 2018.