VERSE INTO VISION “化诗入画” - AI TOOL TO VISUALIZE ANCIENT CHINESE POEM: THE IMPLEMENTATION AND THE DISCUSSION OF AI ART

by

Yantao Mei

Signature Work Product, in partial fulfilment of the Duke Kunshan University Undergraduate Degree Program

March 10th, 2024

Signature Work Program

Duke Kunshan University

APPROVALS

Mentor: Benjamin Bacon, Art and Humanity

Marcia B. France, Dean of Undergraduate Studies

TABLE OF CONTENTS

ABSTRACT

This Signature Work project delves into artificial intelligence's (AI) ability to interpret Chinese poetry, highlighted through the development of the "Verse Into Vision" tool and the accompanying collection, "In Poetry, Paintings; In Paintings, Poetry." Integrating concepts from machine learning, semiotics, and information theory, the research scrutinizes AI's effectiveness in grasping the complex essence of poetry. Utilizing statistical modeling, semiotic analysis, and Systemic Functional Linguistics, it addresses the challenges of poetic interpretation and translation. The project's outcomes reveal both the potential and current limitations of AI in fully appreciating the nuances of Chinese poetry, advocating for advancements in AI's comprehension of historical context and cultural nuances to achieve more profound interpretations. The methodological journey also encompasses data science practices such as data collection, processing, and model tuning, alongside website development. Future directions include deeper philosophical explorations and the development of specialized AI models for translating and interpreting poetry, underlining the project's contribution to the dialogue between technology and traditional literature.

摘要

这个标志性作品项目探讨了人工智能解读中国诗歌的能力，通过开发“化诗入画”工具及其附带的作品集“诗中有画，画中有诗”来突显。该研究融合了机器学习、符号学和信息论的概念，审视 AI 把握诗歌复杂本质的有效性。利用统计建模、符号分析和系统功能语言学等理论，讨论了 AI 对诗歌解释和翻译的挑战。项目成果揭示了 AI 在欣赏中国诗歌细微方面的潜力与当前的局限，倡导在 AI 对历史背景和文化细节的理解方面进行进一步的提升，以实现更深入多元的解读。项目实践过程还包括了许多数据科学专业的技术应用，如数据收集、处理和模型调参，以及网站开发。未来，可能的研究方向包括更深入的哲学探索和专门用于翻译及解释诗歌的 AI 模型的开发，以展现该项目对技术与传统文学对话的贡献。

ACKNOWLEDGEMENTS

I am deeply grateful for the guidance, encouragement, and invaluable support I received throughout this project. First and foremost, I wish to express my sincerest gratitude to my mentor, Benjamin Bacon, whose expertise in Media and Art significantly shaped this work. His creative insights and problem-solving skills were instrumental in overcoming numerous challenges.

Special thanks go to my peers, Zezhen Wang and Yizhou Bi, for their essential technical support with APIs and website deployment. Their assistance in building and setting up the local server was crucial for the project's success.

I also extend my gratitude to everyone who participated in the test run of the website. Your constructive feedback was indispensable and has greatly contributed to the improvement of this work.

Lastly, I want to acknowledge my friends, who have been a constant source of encouragement and support. Your belief in me has been a pillar of strength, enabling me to complete this significant project.

Without the combined efforts and contributions of all mentioned, this project would not have been possible. Thank you for making this journey memorable and enriching.

TABLE OF FIGURES

INTRODUCTION

Images stand as essential mediums through which human engage with and interpret their surroundings, largely perceiving the world through visual stimulation. From the sharp snapshots taken by cameras to the creative visions expressed in paintings and digital art, visual media shapes our perceptions. Historically preserved on paper, canvas, and light-sensitive materials like film, the evolution of imaging technology has ushered in a digital era where images are increasingly stored, shared, and manipulated in digital formats. This progression is underpinned by significant advances in digital acquisition technology and signal processing theories, reshaping our interaction with images, and expanding the horizons of artistic and scientific expression. This background forms a crucial context for exploring the dynamic interplay between AI and image, highlighting a transformative shift in how we create, disseminate, and perceive images in contemporary society.

The development of AI-generated art, particularly in drawings, marks a revolutionary chapter in the history of artistic expression and media theory. This innovation traces its basis to the exploration of automata, a field where technology and artistry intersect, challenging traditional definitions of authorship and artistic genius. AI drawings originate from the desire to extend the capabilities of human creativity with the precision versatility and computation ability of machine intelligence. Early experiments in this domain sought to understand whether machines could replicate or even augment the creative process, leading to the development of algorithms capable of producing visual art. Today, AI's capability in art generation has transcended mere replication, embodying the ability to synthesize novel visual expressions from vast datasets of existing artworks. These systems, powered by advanced machine learning techniques such as Generative Adversarial Networks (GANs) and deep learning models, can now generate images that resonate with human aesthetics, yet are entirely new creations. This breakthrough not only frees artistic production, making it accessible to those without formal training, but also stimulates a profound discourse on the essence of creativity, the role of the artist, and the evolving relationship between humans and machines in the creation of art. AI-generated drawings emerge not just as technological singularity, but as a medium that challenges and expands the boundaries of traditional media theory, inviting a reevaluation of art's place in the digital age. It also prompts a profound reevaluation of media theories and fosters a deeper comprehension of the intricate relationship between technology and artistic creativity.

Jean-Paul Sartre made a clear distinction between perception and imagination in his work, The Imaginary: A Phenomenological Psychology of the Imagination. He explained that perception involves observing the world through our senses, but it's inherently limited because we can only see things from one angle at a time (3-14). Imagination, on the other hand, is more free-form and subjective. It allows us to envision something from all angles at once, combining our knowledge and desires to form a complete picture. Sartre described imagination as a kind of 'quasi-observation' (7-8). This means our imaginative ideas are very unique and personal, shaped by our past experiences and what we want to see. When we apply Sartre's ideas to AI-generated art, some fascinating comparisons emerge. AI art pushes the boundaries of traditional media by using what could be seen as a machine's form of imagination. It pulls together vast amounts of data to create something new, which might be said to reflect the machine's 'intentions,' as designed and programmed by its creators. Thus, AI-generated artworks serve as a technological form of 'quasi-observation,' where each piece reflects a mix of historical data and the algorithm's current processes. This invites us to look not only at the artwork itself but also at the intentions and the embedded knowledge that shaped it. It's a fashion way to engage with art, starting discussions about creativity and the role of technology in art.

Classical Chinese poetry, like Gu Shi, unfolds vivid scenes—from majestic natural landscapes to intimate emotions of joy, sorrow, separation, and reunion. This is done with minimal words but powerful impact, adhering to the strict rules of Yun Lv (rhythm) and Dui Zhang (antithesis). These poems use Yi Xiang (imagery)—symbols that spark the imagination of the reader. By connecting Jean-Paul Sartre's theory of imagination and semiotics to these ancient texts, we can see how these poems aren't just words on a page; they are gateways to deep, imaginative engagement. The carefully structured verses of these poems showcase how language can evoke whole worlds within the reader's mind, adhering to Sartre's view of imagination as a space where everything about an object—or in this case, a scene or emotion—is completely presented through the synthesis of the reader's knowledge and desires. The concise and rich language of these poems serves as a semiotic tool, each character loaded with historical and cultural significance, inviting readers to look beyond the text and uncover layers of meaning. In this way, Classical Chinese poetry exemplifies the profound skill of ancient poets to create immersive, imagistic experiences that illustrate the interaction between the signifier (the word) and the signified (the meaning), and the imaginative leap needed to fully appreciate the imagery.

The project 'Verse into Vision' initiative introduces a groundbreaking tool that uses AI technology to transform ancient Chinese poetry into visual art. This approach allows for a novel exploration of poetic imagery within a digital context. This project not only demonstrates the fusion of textual and visual aesthetics but also enhances public engagement by making these artworks accessible through an easy-to-use website. Aiming to explore the potential of AI in understanding and interpreting Chinese poetry, this paper discussed it through the lenses of information theory, semiotics, and other relevant fields. Although successful, the project faces challenges such as time constraints, financial limitations, and intellectual property concerns, which may affect the depth of AI model exploration and the discussion of media theory and semiotics.

MEDIA AND LITERATURE REVIEW

In developing the "Verse Into Vision' tool, I extensively explored theories from machine learning, semiotics, and information theory. This investigation raises a compelling question: Can AI truly "understand" Chinese poetry? Understanding is a complex concept with definitions that vary across disciplines. Generally, to "understand" means to interpret or grasp the intended meaning of something. From a semantic perspective, it involves comprehending the messages communicated through words, sentences, and symbols within a specific context. Information theory defines understanding as the complete reception of an entity's information, while semiotics views it as the decoding of symbols to uncover underlying messages. Although these definitions provide a foundational backdrop, this paper focuses on exploring the nuanced meaning of "understand" within the unique context of Chinese poetry, guided by these diverse theoretical insights.

Chinese poetry, with its roots in China's cultural environment, presents unique challenges for comprehension. Researchers exploring different interpretive methods have highlighted two main approaches. One innovative method involves using statistical techniques such as the Weighted Personalized PageRank (WPPR) to analyze Chinese poetry. This approach measures the emotional tone of words by assessing their similarity to both positive and negative reference words within a comprehensive poetry corpus, as discussed by Hou and Frank (15-24). They note that this graph-based method views the lexical network holistically, achieving what might be considered an optimal solution for identifying the emotional orientations of words in classical Chinese poetry (Hou and Frank 22). Such advancements suggest that AI may indeed have the capacity to 'understand' the sentiment expressed in these poems. However, like other data-heavy statistical methods, WPPR struggles with predicting metaphors and providing accurate interpretations without historical context. Despite these shortcomings, further refinements in data quality and preprocessing, along with more sophisticated models like BERT-BiLSTM-CRFs—a hybrid model combining BERT (Bidirectional Encoder Representations from Transformers)'s deep learning contextual capabilities, BiLSTM (Bidirectional Long Short-Term Memory)'s sequential data handling, and CRF (Conditional Random Field)'s sequence labeling accuracy—are expected to enhance our ability to interpret specific metaphors accurately, as indicated by Zhang et al (115-120).

On the other hand, the fields of semantics and semiotics also play an important role in the interpretation of Chinese poetry. For instance, the Systemic Functional Linguistics (SFL) model is a practical tool for translating Chinese poetry into English (Huang 1). Huang's analysis reveals the difficulty to translate Chinese poetry, where every word choice and syntactic decision can dramatically alter the imagery and emotional resonance of the final piece (2-3). Moreover, His findings highlight the invaluable role of functional linguistics in translation studies, demonstrating how essential semantics is when trying to get to the heart of Chinese poems (Huang 2-9). Meanwhile, as machine learning continues to evolve, we're seeing more models that can pinpoint the syntax and key elements within sentences. Additionally, when we apply the theories of semiotics pioneers like Saussure and Barthes to ancient poetry, we can dissect the language into three main components, as noted by Cui (56-57). First, there's the signifier—essentially the physical form of the language, such as sounds or written characters. Next, the primary signified, which refers to the concrete things these symbols represent, for example, "夕阳西下" (Sunset). Finally, the secondary signified, involves the ideas and mental images the poetry evokes, for example, the sense of sad and lonely that the "夕阳西下" (Sunset) brings. Exploring these concepts allows us to appreciate the poems on both a surface and a deeper level, providing a richer and colorful understanding of their meaning. This method not only helps us appreciate the material aspects of the signs but also the emotional and conceptual reactions they provoke, blending the tangible with the abstract. These semiotic theories, when linked with previous statistical methods, reveal a profound and sometimes missed connection between the ongoing development of AI technology and the traditional realms of semiotics and information theory.

Before we further explore the technological and theoretical approaches to AI's interpretation of Chinese poetry, it is vital to address a common misconception about the term "understand" in this context. Many may assume that "understand" a poem's meaning involves finding a singular, "optimal" interpretation. Yet, Chinese poetry, with its layers of complexity and historical depth, resist such straightforward decoding. The poets' original intentions often remain unknow in mystery, and the poems themselves do not simply convey clear-cut messages. Instead, they invite readers to dive into a personal exploration and interpretation of their themes and emotions. This unique characteristic of Chinese poetry serves as a rich canvas that engages readers on a deeper personal level. Understanding this aspect is crucial, and through the lenses of information theory, imagination, and semiotics, we can appreciate how the interpretative process transcends conventional analysis, allowing for a multitude of individual perspectives and insights.

With the information theory, our focus move from what Chinese poetry does say to what Chinese poetry could say. The theory developed by Shannon consider communication as a sequence of symbols transmitted from sender to receiver without necessitating an understanding of the symbols' meanings (623-656). It quantifies information as the logarithmic value of potential symbol choices and utilizes the Markov chain model, suggesting that each symbol’s appearance and the conveyed information are contingent upon preceding selections. This theory implies that messages offering a greater number of decoding options inherently carry more information, especially in scenarios where multiple interpretations are equally probable. Thus, the inherent ambiguity and multiplicity of meanings in Chinese poetry elevate its information entropy, offering expansive room for imagination rather than delivering precise messages. This unique characteristic underscores the Chinese poetry's capacity to evoke a wide array of interpretations, enriching its cultural and artistic value. Interpreting Chinese poems through information theory involves reducing the poems' information entropy. This reduction process isn't a goal in itself but demonstrates that each interpretation is part of understanding the poem's broader message. This approach doesn't invalidate interpretations but rather validates them as contributions to the comprehensive understanding of the poem's full possible meaning.

The appreciation of Chinese poetry, as explored in the works of Zhao Guangfa and Pan Jianping, reveals a nuanced understanding of poetic images and the expansive realm of interpretation they offer. Zhao's analysis explores the translation of metaphorical images in ancient Chinese poetry, emphasizing the cognitive psychological approach to metaphor theory (44-45). This investigation reveals various translation methods tailored for different metaphorical images, thereby enhancing cross-cultural communication and deepening the appreciation of Chinese poetic beauty. On the other hand, Pan's discussion on the aesthetic theory of mood in Chinese ancient poetry underscores the intrinsic connection between the poetic images and the evocation of mood (83). The theory elucidates how Chinese poetry, steeped in profound cultural heritage, provides a vast space for imagination through its images. It categorizes images into three types, yet emphasizes that their interpretation relies on a blend of common sense and personal understanding, thus making each encounter with a poem a unique experience (Pan 83). These discussions collectively assert that Chinese poetry, with its deep-seated cultural roots and intricate use of images, invites readers into a space where imagination and personal interpretation play crucial roles. The act of decoding these images isn't just about uncovering a singular, intended meaning but rather about engaging with the poem in a way that is deeply personal and reflective of one's own experiences and worldview.

Chinese poetry, rich in imagery and open-ended meanings, serves as an ideal medium for exploring Sartre's theory. According to Sartre, poetry provides “analogon”— that resonate with the reader's personal experiences, knowledge, and emotions to assist their imagination in understanding the scenario (55-57). Which is also corresponds to "secondary signified" in Cui's theory (56-57). While reading a Chinese poem, readers do not simply observe the imagery; instead, they imaginatively reconstruct it, infusing it with personal experience and emotion. This interpretive process emphasizes the diversity of understanding poetry. Moreover, "quasi-observation", where our personal interpretations are deeply influenced by our own life experiences, can be interpreted that the imagination offers us a unique kind of freedom (Sartre 6-8). When we read Chinese poetry with this mindset, it's not just about appreciating its beauty; it becomes a liberating journey. This process allows readers to break free from their restricted realities and immerse themselves in their emotions, dreams, imaginations, and histories. Therefore, reading Chinese poetry is a deeply personal exploration that reveals the limitless scope of human creativity and our unique ways of finding meaning in art.

The discussion has circled back to the insight that to "understand" Chinese poetry is, in essence, to interpret it. This interpretation involves decoding messages and reducing information entropy based on individual perspectives. Having delved into the subjective nature of interpretation through semiotics, information theory, and the imaginative framework, we stand at the threshold of a new inquiry: the capability of machines or AI to interpret Chinese poetry. This question invites us to consider the extent to which can AI navigate the complexities of language, culture, and subjective experience that define the act of interpreting poetry.

Building on the previous exploration of AI's ability to understand complex forms of language, Monti's research delves into the intricate relationships between cybernetics, semiotics, and AI, specifically focusing on the potential and limitations of AI in creative interpretation (89-103). His thorough analysis details the historical evolution of cybernetics, emphasizing its significant impact on how AI systems currently conceptualize meaning (Monti 93-96). By drawing on Umberto Eco's crucial distinctions between information and meaning, Monti emphasizes the formidable challenges AI faces in comprehending the symbolic and often ambiguous elements (91-96). Thus, AI naturally struggles to interpret Chinese poetry, especially when it delves into symbolic and abstract imagery. This is particularly relevant to the multifaceted nature of Chinese poetry, which is deeply related with metaphors, historical allusions, and philosophical insights—elements that require a higher level of interpretation far beyond simple information processing. Monti's investigation thus raises profound questions about AI's capacity to fully grasp and interpret the nuanced expressions and cultural depths embedded within poetic works, aligning closely with ongoing debates about the role of AI in the arts and humanities.

Similarly, Nadin delves into how AI, considered as semiotic machines, processes and constructs meaning, offering a nuanced perspective on AI's interpretive capabilities (85-92). Nadin's focus on the dynamic and interactive aspects of semiotic processes presents a framework for analyzing how AI might approach the interpretation of poetry. This exploration is crucial for assessing AI's potential to decode the rich tapestry of meanings in Chinese poetry, deeply entrenched in a complex semiotic tradition. Nadin argues that integrating semiotic principles more deeply into AI design could potentially enhance its proficiency in dealing with the cultural and linguistic subtleties of word (18-19). His works call for a deeper examination of whether AI can truly engage with and interpret poetry in a manner that respect its cultural depth and linguistic nuances, aligning with humanistic understanding and sensitivity.

However, AI may brings new meaning and experience through the interpration of Chinese poetry. Poschinger and Coon introduces a pioneering model that reconceptualizes the role of AI in the meaning-creation process (393-396). The concept of the three-dimensional semiotic pyramid does more than just place AI in the role of an active contributor to meaning-making. It unveils the complex interconnections between AI's language outputs and the everyday digital world, influenced by how we use language online and through predictive texting (Poschinger and Coon 396-407). The potential of this model to transform our understanding of Chinese poetry is immense. It suggests that

AI could significantly shape how we interpret the subtle nuances and rich textures of poetic language. While we are discussing whether AI can understand Chinese poetry the pyramid model reminds us to reconsider the role of AI in the arts, promising a deeper integration of technology with creative expression.

In conclusion, it’s clear that while AI is favored for interpreting Chinese poetry, there are still significant hurdles to overcome. AI struggles with the symbolic and ambiguous elements of poetry, especially when it comes to Chinese poetry's deep metaphors, historical references, and philosophical insights. However, there are still some adjustments we can apply to improve this. AI can certainly help in the process of interpreting Chinese poems, but fully appreciating the subtleties of Chinese poetry requires more than what current AI technology can offer. This journey invites us to reconsider AI’s role in the arts instead of a dead machine.

PROCESS, MATERIALS, AND METHODS

The inspiration of this project was driven by my personal transformation from reluctance towards learning Chinese ancient poetry, due to its traditional recitation methods, to a profound appreciation of its inherent beauty. This shift inspired the creation of a tool designed to bring these poetic works to life visually, enabling users to interact with and understand the poems on a deeper level. This tool, engineered to take a poem as input and generate an English prompt for an AI drawing model, culminates in the visual representation of the poem. The ambition was to freely interact with the rich tapestry of Chinese poetry through technology, allowing for personalization and easy image download. To showcase this tool's capabilities, the project also includes a collection of visualized Chinese poems, presented on a website where viewers can engage by rating and viewing images. Utilizing my data science background, the project encompassed coding in diverse programming languages, data management, database and website development, algorithm tuning, and meticulous design of human-computer interactions. This journey not only utilized my technical skills but also encouraged a contemplation on how algorithms interpret symbols and icons within Chinese poetry, engaging deeply with semiotic and informational theories.

VERSE INTO VISION: THE VISUALIZATION TOOL

The design of the application to visualize Chinese poetry into images involves two models: a translating model to convert Chinese poetry into English sentences, and a drawing model to transform these sentences into visual representations. Recognizing that different drawing models require specific types of instructions—for instance, Dalle (the AI model developed by OpenAI company that generate images based on text) prefers narrative sentences, while Stable Diffusion (the AI model developed by Stability AI) favors a list of descriptive words punctuated and structured distinctly—a unifying layer for input standardization is crucial. This system integration, coupled with a user-friendly interface, enables seamless conversion from Chinese poetry to image output. The development process entailed programming the translation, processing, and drawing functionalities, followed by constructing a website to serve as the interface, encapsulating all functions within an accessible tool.

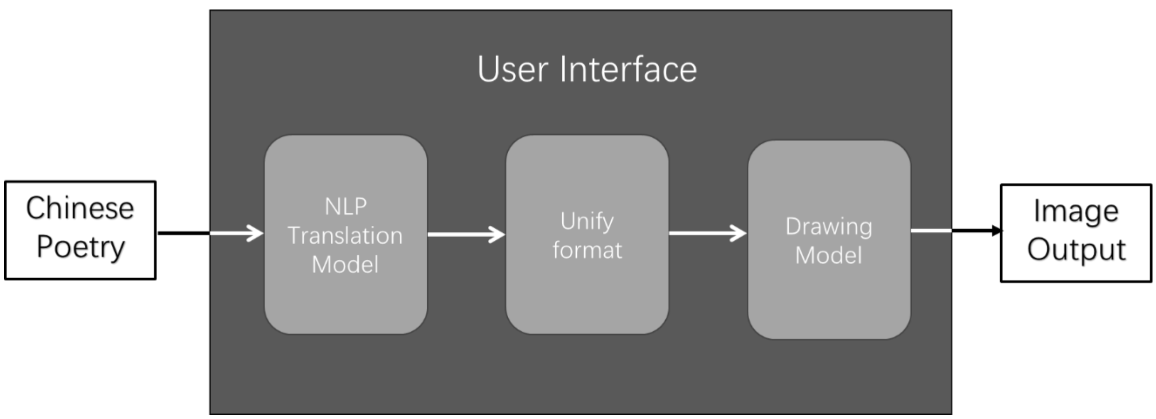


Figure 1: The Framework of Verse Into Vision

Figure 2: The Screenshot of Verse Into VisionFigure 3: The Framework of Verse Into Vision

The process of programming the translation of Chinese poetry into images was an intricate journey of exploration and refinement. Starting with the AI drawing model, the ideal scenario would involve developing a custom-trained model specifically for visualizing Chinese poetry. However, accomplishing this in an individual project is nearly impossible due to constraints in time, funding, and equipment. Consequently, I explored different well-known drawing models—DALL-E, Midjourney, and Stable Diffusion—to identify the best method for visual translation. Evaluating these models based on their acquisition difficulty, cost, compatibility with Chinese poetry, and capacity for large-scale image production led to the early exclusion of Midjourney due to accessibility issues. The comparison ultimately focused on Stable Diffusion and DALL·E-2. Stable Diffusion, deployable locally, offers significant advantages in customizing image styles through various style packages. It also accepts prompt words in both Chinese and English, facilitating the input preparation. However, my attempts revealed that word segmentation models could not accurately break down sentences from ancient Chinese poems for this purpose (see Appendix A). Furthermore, Stable Diffusion struggled with ancient Chinese nouns and adjectives, let alone creating images based on them, and existing translation tools were inadequate for accurate English translations. In contrast, DALL·E-2, with its official OpenAI API, offered easier access and greater accuracy in translating entire poems, considering contextual meaning. Adjusting DALL·E-2's style was straightforward, following guidelines from *The DALL·E 2 Prompt Book* (OpenAI, p. 16). Despite DALL·E-2's cost of approximately $0.020 per image, its efficiency and comprehensive guidance on parameter adjustments made it the chosen model for implementing the tool, setting aside Stable Diffusion until after the prototype was completed.

To support the analysis of information theory and semiotics in relation to the tool, it's vital to go through the methodology behind the DALL-E 2 model. This exploration not only contextualizes the tool's capabilities within the broader theoretical landscape but also offers insights into the discussion between technological innovation and symbolic communication. DALL-E 2, developed by OpenAI, uses a sophisticated diffusion model conditioned on CLIP image embeddings, a technique that significantly advances the fidelity and contextual relevance of generated images from textual descriptions (Radford et al.). This model, utilizing 3.5 billion parameters, demonstrates an exceptional capability to interpret and visualize complex textual inputs into detailed and semantically accurate images (Ramesh et al.). Its operational framework is rooted in a transformative approach where text descriptions are translated into visual representations through a process that progressively refines random noise into coherent imagery, echoing the intricacies of human creativity. The training process of DALL-E 2, integrating vast datasets of text-image pairs, equips the model with a nuanced understanding of visual and textual correlations. This endows DALL-E 2 with the unique ability to produce images across a wide array of styles and compositions, from photorealistic renditions to stylized interpretations, effectively capturing the essence of the described scenes or objects (Johnson). The application of DALL-E 2 to translate Chinese poetry into visual art presented an opportunity to explore the convergence of language, culture, and AI-driven creativity. By feeding the model with carefully crafted prompts derived from the poetic texts, the project achieved results that resonate with the symbolic depth and aesthetic appeal of the original poems. Moreover, the integration of CLIP into DALL-E 2's workflow enhances the model's ability to evaluate and select the most appropriate visual outputs based on textual prompts, ensuring that the generated images not only align with the descriptive content but also uphold artistic and cultural fidelity. This nuanced translation from text to image underscores the project's innovative use of AI to bridge traditional artistic domains with cutting-edge technology, offering a fresh lens through which to appreciate the timeless beauty of Chinese poetry.

Central to the methodology is the translation model that parsed the essence of the poetry, focusing on capturing the objects, emotions, and scenes described. I developed a dynamic script to transform Chinese verses into English prompts, optimized for an AI drawing model's interpretation. However, the quest for an ideal language model for this task proved challenging. Given the selection of DALL-E 2 as the drawing model, the language model needed to accurately convey not just the poem's basic content but also its mood, composition, and color tones. During the exploration in Stats 302 (Appendix A), it was found that existing NLP models like Latent Dirichlet Allocation (LDA) could identify themes and emotions through the analysis of word vectors. Yet, the precision of these models fell short of the requirements for this application. To enhance accuracy, manual labeling became indispensable, rendering these smaller language models insufficient for translating Chinese poetry with precise emotional and object recognition. Additionally, constructing a composition reflective of the poem's narrative proved difficult for the current models. Consequently, Large Language Models (LLM) such as GPT, LaMDA, and LLaMA emerged as the leading candidates for translating ancient Chinese poetry and generating the descriptive text prompts for DALL-E 2. Considering both convenience and API consistency, I opted for the GPT model as the language model. Through various tests, both GPT-3.5 and GPT-4.0 demonstrated the capability and processing speed to accomplish the task with comparable outcomes. Given the cost-efficiency, GPT-3.5 was selected for extensive usage to optimize budget allocation. This decision was influenced by the balance between performance and cost, ensuring the project remained within financial constraints while achieving the desired quality in translating and visualizing Chinese poetry.

The GPT model operates on the transformer architecture, which employs attention mechanisms to weigh the importance of different words in a sentence, enabling it to predict the next word in a sequence. This model is trained on vast datasets of text, learning patterns, and structures of language. During training, it adjusts its internal parameters (weights) to minimize the difference between its predictions and the actual outcomes. This process allows the GPT model to generate coherent and contextually relevant text based on the input it receives. (Radford et al. pp 3-4) To elucidate the process by which GPT's Large Language Model (LLM) transforms Chinese poetry into English prompts for DALL-E 2, an understanding of its complex algorithms and extensive dataset training is imperative. The GPT employs a sophisticated understanding of linguistic structures, cultural nuances, and semantic depths inherent in the poetry, facilitated by its training on a diverse corpus of texts. This enables it to accurately capture and translate the essence, tone, and imagery of Chinese poems into coherent English prompts. Such prompts are then utilized by DALL-E 2 to generate visual representations that are not only contextually relevant but also deeply resonant with the original poetic vision. For example, in the translation process for "远芳侵古道，晴翠接荒城" (The endless fragrant grass covered the ancient road, and the green extended to the deserted city.) GPT's LLM first analyzes the text, identifying key elements such as "distant fragrance," "ancient path," "bright greenery," and "desolate city." It then synthesizes these elements into a coherent English prompt that reflects the poetic imagery and mood. An illustrative result might be, "A distant fragrance invades the ancient path, where bright greenery meets the remnants of a desolate city," showcasing the model's capacity to maintain the poetic essence, Once the translation and drawing models are chosen, the implementation code is crafted using Python and its libraries: 'openai' for model interactions and 'wget' for image downloads. The program begins by setting up the OpenAI API key, initiating the process to convert poetry into visual art. It prompts the GPT-3.5 model to analyze a given Chinese poem, extracting and interpreting objects, emotions, and the overall setting described within the text. This prompt is carefully crafted to ensure the output is succinct yet descriptive, adhering to a limit of 1000 characters. The program then leverages this refined text to instruct the DALL-E model, resulting in a visual representation that captures the essence of the poem, including specific colors, positions, and ancient Chinese stylistic elements, without including text or watermarks. The outcome is an image and a URL linking to the generated image, demonstrating the production achieved by the combination of two AI models.

Through meticulous tuning and experimentation, including adjustments to prompts for clarity and thematic accuracy, I enhanced the script's ability to generate prompts that align with the poem's imagery. The initial prompt sent to the GPT-3.5 model was broad and vague: ‘Take this Chinese poem and turn it into a detailed text prompt within 1000 letters (including space and marks) for DALLE to draw a painting without watermark or text on it:’. To refine the content GPT should generate, including theme color and emotion, I revised the prompt hundreds of times and tested the results. Despite the improved second version appearing and testing well, it faltered during small batch production tests due to its length causing the GPT model to lose focus on key tasks, such as the length limitation and translation. Additionally, since GPT model billing is based on tokens—parts of or entire words (e.g., “GPT” counts as one token)—a longer prompt significantly increases costs. Consequently, the third prompt version was streamlined: ‘Take a Chinese poem to analyze the objects in it and the emotions or feelings that the poem describe. Turn it into a text prompt less than 1000 letters for DALLE. The prompt must be shorter than 1000 letters (including space). Describe objects in the picture with their color and their position in the picture. Then, give a brief description about what the background should look like (color and scene). Also, don't forget this is a Chinese poem, so the view and buildings and other things should look like ancient Chinese style. There should not be any text and watermark in this generated image. Here is the Chinese poem:’. Trimming just a few words significantly altered the output. The script successfully integrates the analysis of poetic elements with the generation of detailed, contextually rich visual prompts, ensuring that the resulting images are imbued with the poem's atmospheric and stylistic nuances. As an example, here is a poetic prompt that the code generated corresponding to “远芳侵古道晴翠接荒城”:

The distant fragrance invades the ancient pathway,

As clear and verdant scenery connects with the abandoned city.

The vibrant blooms and their sweet scent,

Permeate the air, a reminder of days long gone.

The winding pathway, worn with time,

Symbolizes the passage of years, of memories left behind.

Surrounded by lush, green vegetation,

A serene and peaceful atmosphere fills the air.

In the background, a deserted city stands,

With crumbling walls and dilapidated buildings,

Nestled amidst rolling hills and mountains,

Painted in shades of emerald green.

The sky above is clear and bright,

The sun shining down, casting a warm golden glow.

A moment frozen in time, evoking nostalgia,

And a sense of tranquility that lingers in the air.

In this scene from ancient China,

The beauty of nature and the remnants of the past,

Merge together in a harmonious embrace,

Creating a picture of timeless beauty and serenity.

This delicate balance of technology and creative insight resulted in a collection of images that not only represent the poem's visual narrative but also embrace the aesthetic and cultural depth of ancient Chinese settings, devoid of any modern elements or anachronisms.

In order to implement the whole blueprint of the project as quickly as possible instead of wasting time struggling on the part of the content, I implemented the user interface through the website instead of developing software for different platforms. The web user interface is developed using Flask, a lightweight WSGI web application framework in Python, to offer a user-friendly tool. Flask enables the creation of web applications with minimal setup, integrating Python functions with HTML through templating. This setup allows users to interact with the application via a webpage, where they can input text to generate images. The Flask application handles requests, processes input using the 'generated_image' function, and dynamically updates the webpage with the generated image and prompt, enhancing accessibility and engagement for users.

"IN POETRY, PAINTINGS; IN PAINTINGS, POETRY" — WORK COLLECTIONS

To showcase the capabilities and quality of Verse Into Vision, the tool was applied to create visual interpretations of Wang Wei's poetry from Quan Tang Shi. This endeavor resulted in a collection of 1053 images for 351 poems, with each poem inspiring three distinct images. This extensive compilation demonstrates the tool's potential for wider public utilization and affirms the high quality of its outputs. The collection's title, "In Poetry, Paintings; In Paintings, Poetry" is inspired by the accolades for Wang Wei, symbolizing the interplay between his poetry and the Verse Into Vision tool.

This collection of works uses Wang Wei’s poems for the following reasons. First, Su Shi, the great poet of the Song Dynasty, once said in 《东坡题跋・书摩诘〈蓝田烟雨图〉》("Dongpo Inscription·Shu Mojie's 'Lantian Misty Rain Picture'"): "味摩诘之诗，诗中有画；观摩诘之画，画中有诗。" It means carefully reviewing the poems of Wang Wei (also known as Mojie), His poems seem to contain pictures. When viewing Wang Wei's paintings, you can feel the elegant artistic conception like poetry. Wang Wei is both a poet and a painter. His achievement is not only being good at poetry and painting but also melting and then organically combining poetry and painting in art through his other works. His poems represent a major school of ancient Chinese poetry - the landscape pastoral school (山水田园派). In terms of discovering the beauty of nature, they can not only describe majestic and majestic scenery in general, but also depict the dynamics of natural things in detail; they have special insights into the observation of natural scenery and can skillfully capture the scenery that is suitable for expressing their life taste. Thus, Wang Wei’s poem is naturally suitable for this project as the landscape pastoral school poems contain lots of descriptions of the scenes. Second, Wang Wei's works are almost included in Quan Tang Shi —"Complete Tang Poems", which allows them to be well preserved and read. "Complete Tang Poems" was compiled in the 44th year of the reign of Emperor Kangxi of the Qing Dynasty (1705), based on Hu Zhenheng's 《唐音统签》("Tang Yin Tong Qian") of the Ming Dynasty and《唐诗》("Tang Poems") of Ji Zhenyi of the Qing Dynasty (Sturgeon). Down to the smallest details. The book consists of more than 900 volumes and contains more than 48,900 poems by more than 2,200 poets. Luckily, Quan Tang Shi was found with completely electronic version and included in a GitHub repository ‘chinese-poetry’, which reduced the difficulty in searching and process data. At the same time, Wang Wei and his large number of works have also proved to us that the tool Verse Into Vision has the ability to handle large-volume workloads.

To create the database for the collection "In Poetry, Paintings; In Paintings, Poetry" the electronic version of Quan Tang Shi from the "chinese-poetry" GitHub repository was utilized. This repository contains poems divided into 900 JSON files corresponding to the anthology's volumes. Each JSON file structures the data into categories like author, title, and content, among others. Using Python and Pandas, these files were merged, filtering for essential data such as author, title, and poem content. Traditional characters were converted to simplified ones using the zhconv library, and the results were stored in a CSV file. Specifically targeting Wang Wei's poems, a new CSV file was generated to serve as the foundation for creating the collection. This methodical approach underscores the project's technical and artistic endeavor to bridge ancient Chinese poetry with contemporary digital art forms, ensuring both accuracy and cultural integrity in the process.

To integrate the processes of selecting the most poetic sentences from Wang Wei's poems and visualizing them, I developed a comprehensive approach. Initially, I encountered challenges when inputting entire poems for visualization due to their complex content and length. This led to the decision to limited the focus on individual sentences to ensure clarity and effectiveness in the visual representation. By implementing a loop in Python, I utilized the GPT 3.5 model to select the most evocative sentences within each poem. Subsequently, these selected sentences were processed through another loop, employing the Verse Into Vision tool to generate three distinct images per sentence. This choice was informed by observing the high variability in DALL-E 2's output, where producing multiple images per sentence offered a broader range of visual interpretations and enhanced the overall quality of the collection. Each generated image was meticulously cataloged with a descriptive prompt used for the drawing model and named according to a structured format: the poem's title, the chosen sentence, and an index number. This methodical approach facilitated efficient storage, easy searchability, and practical use of the images, thereby optimizing the tool's capability to visually interpret and celebrate the essence of Wang Wei's poetry.

With over a thousand images to showcase, the challenge of presenting them in an engaging manner arose. Various strategies were considered: initially, a selection of images was published as a photo album, and subsequently, a digital gallery was created to enable specific image searches. Yet, these methods lacked interactive engagement with audiences. Aiming for a presentation format that enhances viewer enjoyment and interaction, the idea emerged to incorporate user ratings to gauge the visualization's accuracy and compare machine versus human interpretation. Inspired by dating app interfaces, the collection "In Poetry, Paintings; In Paintings, Poetry" was thus presented. Utilizing a simple logic, individual images are displayed on the website alongside pertinent details (author, title, corresponding poem sentences) and options to 'like' or 'dislike'. This approach not only facilitates interactive engagement but also allows for image ranking based on user preferences, offering a unique and interactive presentation method while collecting valuable feedback.

WEBSITES DEPLOYMENT

The deployment process involved leveraging Flask to construct the websites, making deployment straightforward. The initial plan was to deploy on a server within Duke Kunshan University's intranet for ease of access and enhanced security. However, encountering issues with DKU's network proxy blocking access to OpenAI's API interfaces, the deployment shifted to Duke University's virtual machine. This adjustment provided dual benefits: seamless integration within the Duke intranet, including DKU, and unrestricted access to necessary APIs. The deployment was tested from November 11th to 17th, inviting peer feedback on both the Verse Into Vision tool and the digital works collection "In Poetry, Paintings; In Paintings, Poetry".

FURTHER EXPLORATION

Throughout the creative process, I experimented with the impact of different types of poems on image outcomes. Additionally, encouraged by suggestions from peers, I explored using modern Chinese poetry for image generation to observe potential differences. This process led to numerous intriguing experiments by both my peers and me, fostering exactly the kind of diverse interactions between ancient poetry and artificial intelligence that I had envisioned. Unfortunately, due to time constraints and other factors, many of these remarkable attempts could not be included in this paper.

FINAL OUTCOME

The outcomes of this project primarily include a web tool "Verse Into Vision," a collection of AI-generated artworks based on the poems of Wang Wei titled "In Poetry, Paintings; In Paintings, Poetry," and a website designed to display the collection and gather user feedback. Due to the complexity of this project and the extensive documentation involved, all programs, databases, and the generated collection have been uploaded to a GitHub repository. This section will only partially showcase my achievements; for more detailed information, please visit my GitHub repository (Appendix B).

VERSE INTO VISION: THE VISUALIZATION TOOL

Here is the screenshot of the tool.

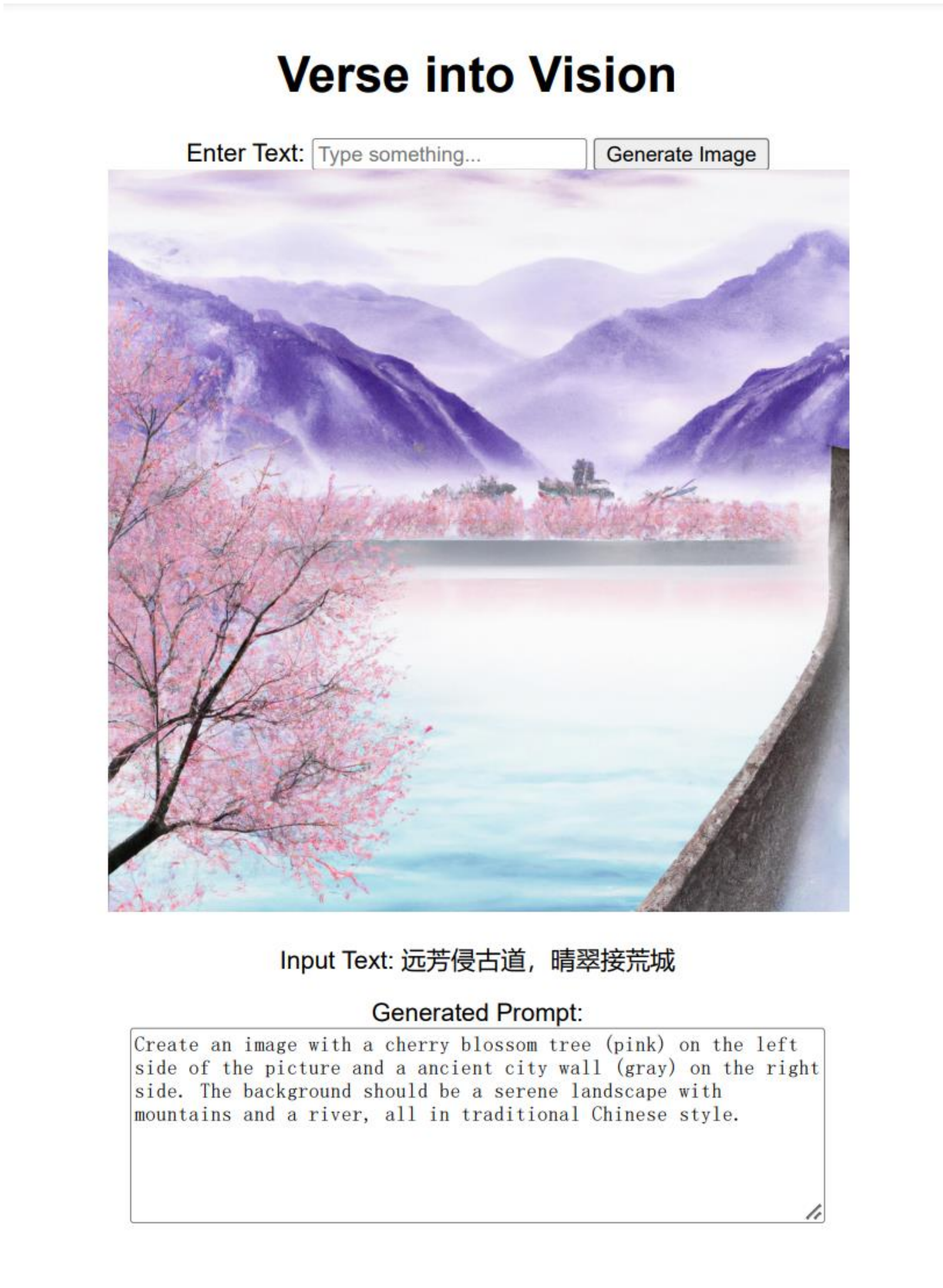


Figure 4: The Screenshot of Verse Into Vision

As depicted in Figure 1, the interface includes a text box designated for entering Chinese poetry. Upon clicking the "Generate Image" button, the input poem and the resulting image is displayed at the center of the webpage, accompanied by the specific text prompt utilized for its creation.

"IN POETRY, PAINTINGS; IN PAINTINGS, POETRY" — WORK COLLECTIONS

In the collection, I've gathered six intriguing images. While they might not always perfectly represent the essence of the poems, they convey meaningful messages or offer a touch of humor.

Six images picked form the " In Poetry, Paintings; In Paintings, Poetry.":

1. Image generated based on《酬比部杨员外暮宿琴台朝跻书阁率尔见赠之作》:"桃源迷汉姓，松树有秦官". This paradise, reminiscent of the Peach Blossom Spring, makes the Han people lose their direction, while the posture of the pine trees is as dignified and solemn as that of a minister from the Qin Dynasty. (Figure 3)
2. Image generated based on《冬夜书怀》:"草白霭繁霜，木衰澄清月". Grass stretches vast under thick frost, while sparse leaves shimmer in the cold, bright moon. (Figure 4)
3. Image generated based on《奉和圣制暮春送朝集使归郡应制》:"杨花飞上路，槐色荫通沟". Willow catkins drift along the road, while the dense shade of the locust trees envelops the ditch. (Figure 5)
4. Image generated based on《洛阳女儿行》:"洛阳女儿对门居，才可容颜十五余". There was a girl living across from my house in Luoyang, in the prime of her youth at fifteen or sixteen, and she is extraordinarily beautiful. (Figure 6)
5. Image generated based on《慕容承携素馔见过》:"纱帽乌皮几，闲居懒赋诗". Wearing a black silk hat and leaning against a small table wrapped in black lambskin, I leisurely enjoy my home life, too lazy to write poetry. (Figure 7)

6. Image generated based on《寓言二首》:“骊驹从白马，出入铜龙门”. One after another, white horses enter and exit through the palace gate adorned with bronze dragons. (Figure 8)

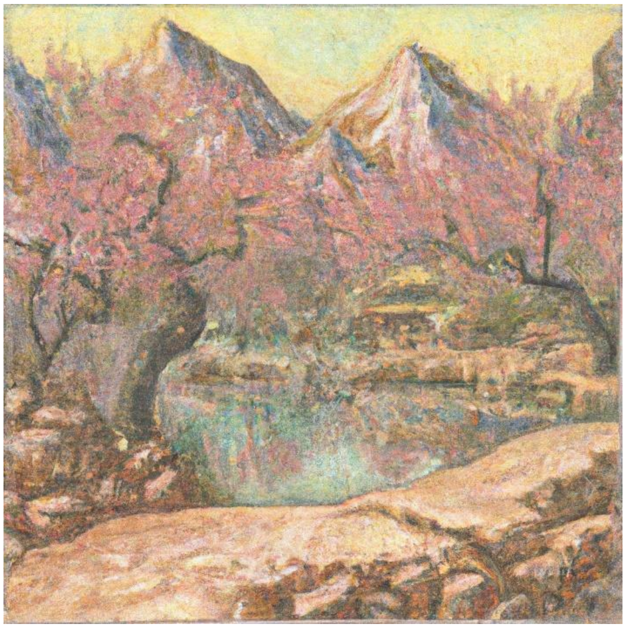


Figure 3: 桃源迷汉姓，松树有秦官

Figure 4: 草白霭繁霜，木衰澄清月

Figure 5: 杨花飞上路，槐色荫通沟

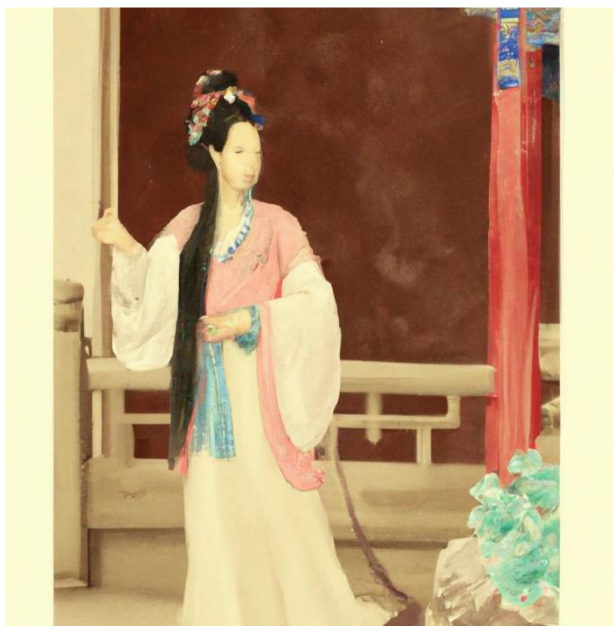


Figure 6: 洛阳女儿对门居，才可容颜十五余

Figure 7: 纱帽乌皮几，闲居懒赋诗

Figure 8: 骊驹从白马，出入铜龙门

The first three images depict sceneries from different seasons, showcasing the poems' content with precision and beauty. However, the subsequent three images encounter various types of errors, primarily due to issues with the visualization model. The DALL-E model faced challenges in detailing the facial features of a fifteen-year-old girl in Figure 6. In Figure 7, a Chinese official's cap is mistakenly rendered as a Western hat, likely because the DALL-E model's training set lacks data on Chinese official caps. Observation of the prompt database suggests that in Figure 8, DALL-E struggled to distinguish between the imagery of a white horse and Longmen, resulting in a white dragon's portrayal. This issue is also partly due to the translation model; a more accurate translation of the "Longmen"(Dragon Gate) imagery might have prevented this error.

VOTING WEBSITE FOR WORK COLLECTIONS

Here is a screenshot for the voting website to present work collection and collect user feedbacks:

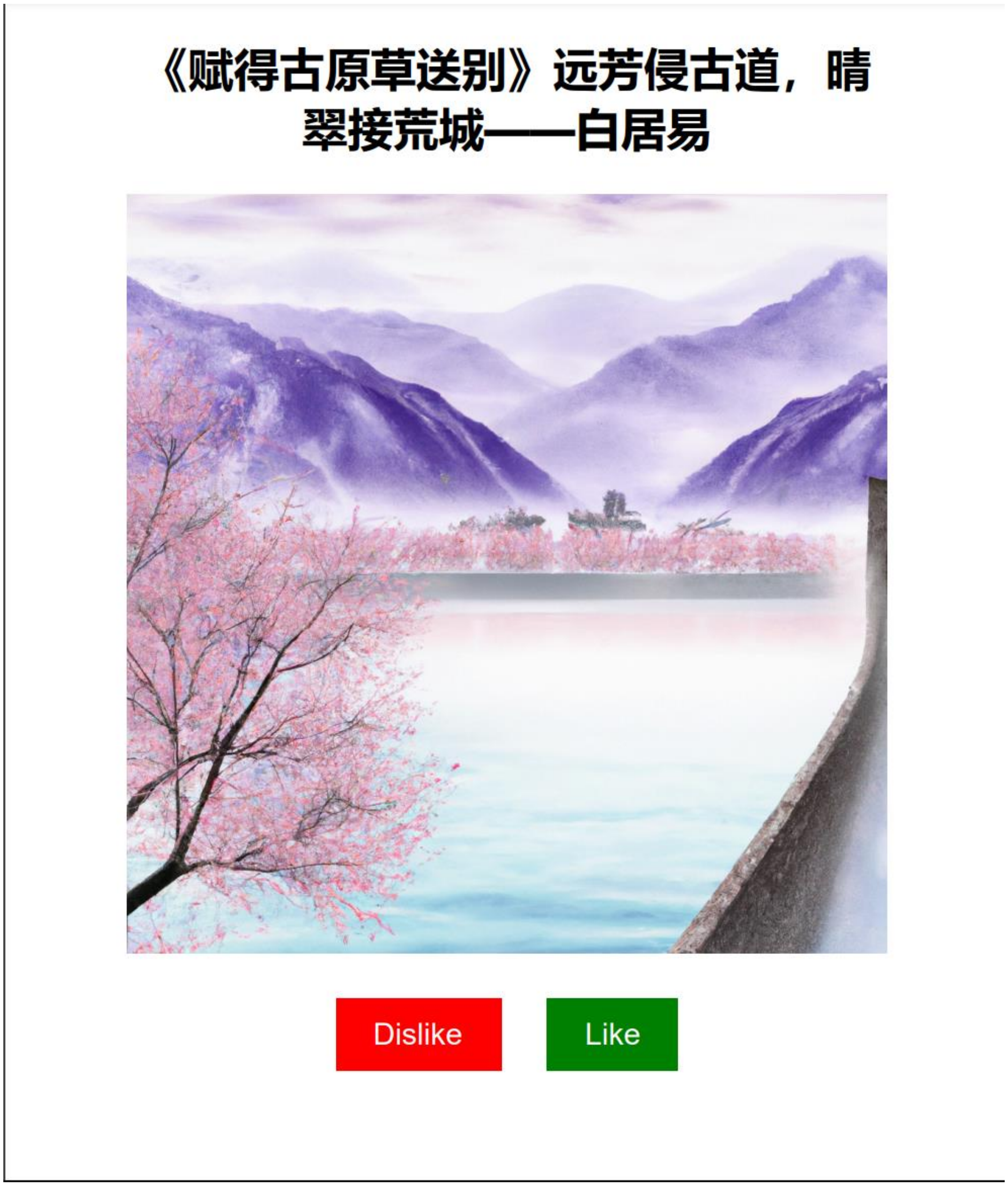


Figure 7: The Screenshot for Voting Website

The interface features two buttons that allow users to express their appreciation or lack thereof for each image, alongside the pertinent poem details. Upon selecting either option, the user's choice is logged on the server, and a new image, chosen at random, is displayed for their consideration.

REFLECTIONS

In the capstone of this project, the innovative tool developed for visualizing Chinese poetry as imagery, alongside the meticulously curated collection "In Poetry, Paintings; In Paintings, Poetry," marks a significant stride in bridging the realms of artificial intelligence, art, and literature. This endeavor illuminates the potential of AI to delve into the nuanced domain of Chinese poetry, offering interpretations that resonate with the complexity and richness of this literary form. Through the discussions of machine learning, semiotics, and information theory, the research navigates the multifaceted concept of "understanding" in the context of poetry, dissecting the layers of meaning that define the poetic experience.

Drawing upon diverse methodological approaches, including statistical modeling and semiotic analysis, this work explores the capacity of AI to engage with the interpretive challenges presented by Chinese poetry. The employment of techniques of the models shows the evolving landscape of computational linguistics and its application to the arts. Moreover, the incorporation of Systemic Functional Linguistics and semiotic frameworks emphasizes the importance of semantics and symbolism in the translation and interpretation of poetic texts.

The discussion extends into the theoretical debates surrounding the interpretation of poetry, challenging the traditional constraints of understanding and opening up a space for a plurality of meanings. Inspired by Shannon's information theory, the project posits that interpretation is an act of navigating information entropy, where each reading contributes to a broader comprehension of the poem's message. This perspective is further enriched by the exploration of poetic imagery and mood, revealing the intricate relationship between language, imagination, and cultural context.

The project's success in generating a vast collection of images from Wang Wei's poetry not only demonstrates the practical application of AI in the arts but also engages with critical theoretical discussions on the nature of meaning, interpretation, and the human-AI interface in creative processes. This synthesis of theoretical insight and technological innovation invites a reevaluation of AI's role in understanding and interpreting the layered nuances of Chinese poetry, marking a significant contribution to the fields of digital humanities and AI research.

Reflecting on the research question of AI's capabilities with Chinese poetry, it becomes clear that neither AI nor humans can fully grasp the entire context of Chinese poetry. However, AI's ability to interpret Chinese poetry is discussed with theories and proved with the implementation of Verse into Vision. For AI, advancing in understanding the historical context, poets' backgrounds, and the intricacies of ancient Chinese language represents significant areas for development. These are critical for achieving nuanced interpretations that resonate more profoundly with human perspectives and cultural insights. For humans, engaging with AI-generated interpretations can offer new angles and understandings, enriching the dialogue between technology and traditional literature. Thus, the journey into interpreting Chinese poetry with AI highlights not only the limitations but also the potential for cross-disciplinary growth and the continuous quest for deeper understanding.

Looking ahead, the intersection of AI, Chinese poetry, and philosophy beckons for more profound exploration to enrich our understanding of this multifaceted topic. My ambition to not only employ but also innovate within language models for translating Chinese poetry signals a groundbreaking direction for future research. Imagining and perhaps even developing a bespoke model tailored specifically for the nuances of Chinese poetic expression could unveil new layers of interpretation and artistic rendition, particularly through specialized AI-driven drawing models that capture the essence of poetic imagery. This journey has also been a mirror, reflecting personal limitations and gaps in knowledge and skills. Yet, it's these realizations that carve the path for growth and learning. Inspired by the challenges encountered during this signature work, the commitment to deepen my knowledge base and enhance my technical skills becomes a cornerstone for future endeavors. This venture into the uncharted territories of AI and Chinese poetry not only promises to extend the boundaries of current understanding but also to personally transform, guided by the lessons learned and the infinite possibilities that lie ahead in bridging technology with the timeless beauty of Chinese poetry.

REFERENCES

Chinese-Poetry. “Chinese-Poetry/Chinese-Poetry: The Most Comprehensive Database of Chinese Poetry 最全中华古诗词数据库, 唐宋两朝近一万四千古诗人, 接近 5.5 万首

唐诗加 26 万宋诗. 两宋时期 1564 位词人，21050 首词。.” *GitHub*, github.com/chinese-poetry/chinese-poetry. Accessed 9 Mar. 2024.

Cui, Yiwen. “从索绪尔的语言符号学思想解读中国古诗词.” 文学界: 理论版 , vol. 2, 2012, pp. 56–57.

Hou, Yufang, and Anette Frank. “Analyzing Sentiment in Classical Chinese Poetry.” *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage*, Social Sciences, and Humanities (LaTeCH), 2015, pp.15-24, doi:10.18653/v1/w15-3703.

Huang, Guowen. “功能语言学分析对翻译研究的启示.” *外语与外语教学*, vol. 5, 2002, pp. 1-11.

Johnson, Khari. “OpenAI Debuts Dall-e for Generating Images from Text.” *VentureBeat*, 5 Jan. 2021, venturebeat.com/business/openai-debuts-dall-e-for-generating-images-from-text/.

Monti, Niccolò. “The Unmeaning Machine. Cybernetics from Semiotics to AI.” *Philosophy Kitchen*. Rivista Di Filosofia Contemporanea, vol. 18, 2023, pp. 89–103, doi:10.13135/2385-1945/7834.

Nadin, Mihai. “Semiotic Machine.” *Public Journal of Semiotics*, vol. 1, no. 1, 1 Jan. 2007, pp. 85–114, doi:10.37693/pjos.2007.1.8815.

OpenAI. *The DALL·E 2 Prompt Book*. Version 1.02, 2022, https://dallery.gallery/wp-content/uploads/2022/07/The-DALL%C2%B7E-2-prompt-book-v1.02.pdf.

Radford, Alec, et al. “Learning Transferable Visual Models from Natural Language Supervision.” *arXiv.Org*, 26 Feb. 2021, arxiv.org/abs/2103.00020.

Radford, Alec, et al. "Improving language understanding by generative pre-training." *OpenAI*, 2018, https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Pan, Jianping. “中国古代意境理论的生成及古诗意境赏析” *Journal of Shanxi Radio & TV University*, May. 2007, pp. 83-83.

Poschinger, Felix, and Robert Coon. “Toward a Semiotic Pyramid: Language Studies, Ai, and Knowledge Exchange Economy.” *Language and Semiotic Studies*, vol. 9, no. 3, 25 Aug. 2023, pp. 393–407, doi:10.1515/lass-2023-0016.

Ramesh, Aditya, et al. “Hierarchical Text-Conditional Image Generation with Clip Latents.” *arXiv.Org*, 13 Apr. 2022, arxiv.org/abs/2204.06125.

Sartre, Jean-Paul. The Imaginary: A Phenomenological Psychology of the Imagination. *Routledge*, 2004, pp. 3-14, 55-85

Shannon, C. E. “A Mathematical Theory of Communication.” *Bell System Technical Journal*, vol. 27, no. 4, Oct. 1948, pp. 623–656, doi:10.1002/j.1538-7305.1948.tb00917.x.

Zhang, Wei, et al. “Sentiment Term Extraction and Application of Chinese Ancient Poetry Text for Digital Humanities.” *Journal of Library Science in China*, vol. 47, July 2021, pp. 113–131, doi:10. 13530 / j. cnki. jlis. 2021033.

Zhao, Guangfa. “浅析中国古诗中的隐喻意象翻译” *Journal of Liaoning Economic Management Cadre Institute*, 2013, pp. 44-45.

"全唐詩 – 中國哲學書電子化計劃." Edited by Donald Sturgeon, *Chinese Text Project*, ctext.org/quantangshi/zh. Accessed 10 Mar. 2024.

APPENDICES

APPENDIX A: STAT 303 Final Project Document (The Jupyter Notebook File)

Tang Shi (Tang Dynasty Poem) Analysis

Yantao Mei

ym174

Introduction

The whole notebook file will seperated into following parts:

- Build a CSV database of Quan Tang Shi
- Visualization of the database
- NLP processing of poems
- Visualization of the result
- Discussion and summary

Setup

In [70]:
```
import numpy as np
import pandas as pd

import os
import json
from zhconv import convert
import copy
from pprint import pprint

import matplotlib
import matplotlib.pyplot as plt
from wordcloud import WordCloud

import jieba
import stanza
import thulac

import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel
from gensim import corpora, models, similarities
```

Build the CSV database based on JSON files

Introduction of the data and database

The data source I use is from the GitHub project: https://github.com/chinese-poetry/chinese-poetry. This project uses the web scraping method to get different kinds of Chinese poems including Tang poems and make them into separate JSON files. Thus, this project gives me the perfect database I need so that I do not need to find additional APIs or do more web scraping to get the data. Moreover, this data source is permitted for private uses and commercial use, which allows me to make full use of it. However, I still need to sort the database to filter out useful information for my project and read JSON files into CSV files.

According to the description (see README files in folder) of the database, the Quan Tang Shi is stored and sorted in volumes of original sequences of books. This means that we have a total of 900 volumes that contain about 48,900 poems.

In the JSON files, each poem is stored by: title, author, biography, paragraphs, notes, volume, and number. Here is an example in volume 1:


```
{
    "title": " 飲馬長城窟行 ",
    "author": " 李世民",
    "biography":  "",
    "paragraphs":  [
        "塞外悲風切， 交河冰已結。瀚海百重波， 陰山千里雪。",
        "迥戍危烽火， 層巒引高節。悠悠卷旆旌， 飲馬出長城。",
        "寒沙連騎跡， 朔吹斷邊聲。胡塵清玉塞， 羌笛韻金鉦。",
        "絕漠干戈戢， 車徒振原隰。都尉反龍堆， 將軍旋馬邑。",
        "揚麾氛霧靜， 紀石功名立。荒裔一戎衣， 靈台凱歌入。"
    ],
    "notes":  [
        ""
    ],
    "volume": " 卷一",
    "no#":  2
}
```


Combined with the project needs, we need to screen out the title, author, and paragraphs.

References of the original data sourses:

[中國哲學書電子化計劃 《御定全唐詩》影印古籍]

[中國哲學書電子化計劃 《全唐詩》]

[國學原文 全唐詩 卷795]

[維基文庫 《御定全唐詩》 (四庫全書本)/卷795]/%E5%8D%B7795#)

Get the dirctory and find all data files' name

In [71]:
```
cur_dir = os.getcwd()
directory = cur_dir + '\quan_tang_shi\json'
```

In [72]:
```
filenames = []

for filename in os.listdir(directory):
    if not os.path.isdir(os.path.join(directory,filename)) and filename.endswith('json'):
        filenames.append(os.path.join(directory,filename))

print(len(filenames))
```

```
900
```

Here we can examine that we have 900 files.

With the name list, now we can import the data from json files into pd Dataframe

Before that, we need to make the paragraphs (a list of String) into one String, otherwise the pd Dataframe will store every sentences seperately instead of store

them poem by poem. This might influence the analyze result, which require us to fix this problem while forming the database. Here is an example:

In [73]:
```
test_dic = directory + '\\001.json'
filter = ['title','author','paragraphs']

with open(test_dic, 'r',encoding='utf-8') as file:
        poems = json.load(file)

poems[1]['paragraphs']
```

Out[73]:
```
['塞外悲風切，交河冰已結。瀚海百重波，陰山千里雪。',
 '迥戍危烽火，層巒引高節。悠悠卷旆旌，飲馬出長城。',
 '寒沙連騎跡，朔吹斷邊聲。胡塵清玉塞，羌笛韻金鉦。',
 '絕漠干戈戢，車徒振原隰。都尉反龍堆，將軍旋馬邑。',
 '揚麾氛霧靜，紀石功名立。荒裔一戎衣，靈台凱歌入。']
```

The case we do not unite the paragraphs:

In [74]:
```
dic = dict([key,poems[1][key]] for key in filter)
temp_df = pd.DataFrame(data = dic)
temp_df
```

Out[74]:

	title	author	paragraphs
0	飲馬長城窟行	李世民	塞外悲風切，交河冰已結。瀚海百重波，陰山千里雪。
1	飲馬長城窟行	李世民	迥戍危烽火，層巒引高節。悠悠卷旆旌，飲馬出長城。
2	飲馬長城窟行	李世民	寒沙連騎跡，朔吹斷邊聲。胡塵清玉塞，羌笛韻金鉦。
3	飲馬長城窟行	李世民	絕漠干戈戢，車徒振原隰。都尉反龍堆，將軍旋馬邑。
4	飲馬長城窟行	李世民	揚麾氛霧靜，紀石功名立。荒裔一戎衣，靈台凱歌入。

The case that we unite them:

In [75]:
```
dic = dict([key,poems[1][key]] for key in ['title','author'])
dic['paragraphs'] = ''.join(poems[1]['paragraphs'])
temp_df = pd.DataFrame(data = dic, index = [0])
temp_df
```

Out[75]:

	title	author	paragraphs
0	飲馬長城窟行	李世民	塞外悲風切，交河冰已結。瀚海百重波，陰山千里雪。迥戍危烽火，層巒引高節。悠悠卷旆旌，飲馬出長...

Build dataframe

Although some libraries and models of NLP like jieba and stanza works for both traditional and simplified Chinese, we will take simplified Chinese as the language type since it is easier to be read and understood by us. Thus we need to use the zhconv library to convert traditional Chinese into simplified Chinese.

In [76]:
```
poem_df = pd.DataFrame()

filter = ['title','author','paragraphs']

n = 0                                                                   ## index

for filename in filenames:

    ## Read json files one by one

    with open(filename, 'r',encoding='utf-8') as file:
```

```
        poems = json.load(file)

    ## Read poems one by one

    for poem in poems:

        dic = {}

        dic['title'] = convert(poem['title'], 'zh-hans')

        dic['author'] = convert(poem['author'], 'zh-hans')

        dic['paragraphs'] = convert(''.join(poem['paragraphs']), 'zh-hans')

        temp_df = pd.DataFrame(data = dic, index = [n])

        n += 1

        poem_df = pd.concat([poem_df,temp_df])

print(n)

poem_df.to_csv('quan_tang_shi.csv', mode = 'w', encoding="utf-8_sig")
```

```
43103
```

Now we can examine the database and amount of poems

In [77]:
```
poem_df.shape[0]
```

Out[77]:
```
43103
```

In [78]:
```
poem_df.sample(10)
```

Out[78]:

	title	author	paragraphs
10573	送陆三出尉	钱起	春草晚来色，东门愁送君。盛才仍下位，明代负奇文。且乐神仙道，终随鸳鹭群。梅生寄黄绶，不日在青云。
14183	送魏州李相公	王建	百代功勋一日成，三年五度换双旌。闲来不对人论战，当朝面受新恩去，算料妖星不敢生。
1607	杂曲歌辞：古曲五首	施肩吾	可怜江北女，惯唱江南曲。摇荡木兰舟，双凫不成浴。郎为匕上香，妾为笼上灰。归时虽暖热，去罢生尘...
18394	茅舍	元稹	楚俗不理居，居人尽茅舍。茅苫竹梁栋，茅疏竹仍罅。边缘堤岸斜，诘屈檐楹亚。篱落不蔽肩，街衢不容...
22814	鱼上冰	王季则	北陆收寒尽，东风解冻初。冰消通浅溜，气变跃潜鱼。应节似知化，扬鬐任所如。浮沉非乐藻，沿溯异传...
32890	赠念经僧	周朴	庵前古折碑，夜静念经时。月皎海霞散，露浓山草垂。鬼闻抛故冢，禽听离寒枝。想得天花坠，馨香拂白眉。
8031	府舍月游	韦应物	官舍耿深夜，佳月喜同游。横河俱半落，泛露忽惊秋。散彩疏群树，分规澄素流。心期与浩景，苍苍殊未收。
30965	送友人归江南	聂夷中	泉州五更鼓，月落西南维。此时有行客，别我孤舟归。上国身无主，下第诚可悲。
7704	自巩洛舟行入黄河即事，寄府县僚友	韦应物	夹水苍山路向东，东南山豁大河通。寒树依微远天外，为报洛桥游宦侣，扁舟不系与心同。
13674	唐昌观玉蕊花	杨凝	瑶华琼蕊种何年，萧史秦嬴向紫烟。时控彩鸾过旧邸，摘花持献玉皇前。

Load the database

In [79]:
```python
poems = pd.read_csv('quan_tang_shi.csv', index_col=[0])
poems.head()
```

Out[79]:

	title	author	paragraphs
0	帝京篇十首	李世民	秦川雄帝宅，函谷壮皇居。绮殿千寻起，离宫百雉余。连甍遥接汉，飞观迥凌虚。云日隐层阙，风烟出绮...
1	饮马长城窟行	李世民	塞外悲风切，交河冰已结。瀚海百重波，阴山千里雪。迥戍危烽火，层峦引高节。悠悠卷旆旌，饮马出长...
2	执契静三边	李世民	执契静三边，持衡临万姓。玉彩辉关烛，金华流日镜。无为宇宙清，有美璇玑正。皎佩星连景，飘衣云结...
3	正日临朝	李世民	条风开献节，灰律动初阳。百蛮奉遐赆，万国朝未央。虽无舜禹迹，幸欣天地康。车轨同八表，书文混四...
4	幸武功庆善宫	李世民	寿丘惟旧迹，酆邑乃前基。粤予承累圣，悬弧亦在兹。弱龄逢运改，提剑郁匡时。指麾八荒定，怀柔万国...

Play with data

Before we implement NLP models and other methods to analyze these poems, we decided to discover the database first.

Most "Hard Working" Poet

We use groupby function and sort function to see who produce most poems in Quan Tang Shi

In [80]:
```python
poet_produce_rank = poems.groupby('author').count().sort_values(by = 'title', ascending=False
poet_produce_rank.head(10)
```

Out[80]:

author	title	paragraphs
白居易	2642	2642
杜甫	1158	1158
李白	896	896
不详	842	842
齐己	783	783
刘禹锡	703	703
元稹	593	593
李商隐	555	555
贯休	553	553
韦应物	551	551

In [81]:
```python
2642/poems.shape[0]
```

Out[81]:
```
0.06129503746838967
```

According to the result, we can see that Juyi Bai produce 2642 poems that makes him produce 6.13% of poems in Quan Tang Shi.

In [82]:
```python
font_path="MSYH.TTC"
```

```python
WC = WordCloud (width = 3840, height = 2160, background_color = "royalblue", max_words = 200,
                colormap = "Set2", relative_scaling = 0.75 ,font_path=font_path)

WC.generate_from_frequencies(poet_produce_rank['title'])

plt.figure(figsize = (12, 7))
plt.axis("off")
plt.imshow(WC)
# plt.savefig("Poet_produce_rank.png")
plt.show()
```

Title distribution

Now we use groupby function to see what is the most popular title that poets liked to use

In [83]:
```python
title_rank = poems.groupby('title').count().sort_values(by = 'paragraphs', ascending=False)
title_rank.head(10)
```

Out[83]:

title	author	paragraphs
句	567	567
古意	39	39
长门怨	33	33
七夕	30	30
塞下曲	27	27
牡丹	27	27
柳	25	25
送别	25	25
从军行	23	23
行路难	22	22

In [84]:
```python
font_path="MSYH.TTC"

WC = WordCloud (width = 3840, height = 2160, background_color = "royalblue", max_words = 200,
```

```
                    colormap = "Set2", relative_scaling = 0.75 ,font_path=font_path)

WC.generate_from_frequencies(poet_produce_rank['paragraphs'])

plt.figure(figsize = (12, 7))
plt.axis("off")
plt.imshow(WC)
# plt.savefig("Title_rank.png")
plt.show()
```

NLP Pre Processing

References: 23_NLP_preprocessing

1. Text segementation models

Based on the understanding of the database we have, now we are going to do the NLP analysis of the data. Like what we learned in the class, NLP of Chinese poem also follow the step of standarlize data, split the words and then do the manipulation and anlysis. However, different from English sentences, Chinese sentence do not have space between words, which require us to split the sentences with the help of text segementation function like jieba, stanza (Stanford) and THULAC (Tsinghua)

Now we need to compare these three models and decide which one we are going to use, here we use 将进酒(君不见黄河之水天上来) as an example to compare them

In [85]:
```
poem = poems.loc[788]
sentences = poem.iloc[2].split("。")
print(poems.loc[788])
```

```
title                                                    乐府杂曲：鼓吹曲辞：将进酒
author                                                                    李白
paragraphs     君不见黄河之水天上来，奔流到海不复回。君不见高堂明镜悲白发，朝如青丝暮成雪。人生得意须
尽欢，...
Name: 788, dtype: object
```

jieba

URL: https://github.com/fxsjy/jieba

In [86]:
```
for sentence in sentences:
    ls = jieba.lcut(sentence)
    print('/'.join(ls))
    print()
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\YANTAO~1\AppData\Local\Temp\jieba.cache
Loading model cost 0.631 seconds.
Prefix dict has been built successfully.
```

君不见/黄河/之水/天上/来/，/奔流/到/海不复/回

君不见/高堂/明镜/悲/白发/，/朝如/青丝/暮成/雪

人生/得意/须尽欢/，/莫使/金尊空/对/月

天生我材必有用/，/岑/夫子/，/丹丘/生/，/将进酒/，/杯莫停

与/君歌/一曲/，/古来/圣贤/皆/寂寞/，/惟有/饮者/留其名

陈王/昔时/宴/平乐/，/五花马/，/千金/裘/，/呼儿/将/出换/美酒/，/与尔同销/万古愁

Advantages:

- Can split words like "天上" (sky) and "须尽欢" (need to indulge)
- The seperation of words keeps the original meaning from the sentence like "天生我才必有用" (no one is useless), is actually an classic saying in China. It helps to analysis the words' meaning and function to the poem.

Disadvantages:

- Some words are seperated with error like "莫使/金尊空/对/月" should be "莫使/金尊/空/对月" (Don't waste the wine in the golden cup and staring at the moon in a daze).

stanza

URL: https://stanfordnlp.github.io/stanza/

In [ ]:
```
### Download the model (http://nlp.stanford.edu/software/stanza/1.0.0/zh-hans/default.zip)
# stanza.download('zh')
```

In [87]:
```
nlp = stanza.Pipeline('zh')
```

```
2023-03-06 16:48:50 INFO: Checking for updates to resources.json in case models have been updated.  Note: this behavior can be turned off with download_method=None or download_method=DownloadMethod.REUSE_RESOURCES
Downloading  https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.4.1.json:   0%|   …
```

```
2023-03-06 16:48:51 INFO: "zh" is an alias for "zh-hans"
2023-03-06 16:48:53 INFO: Loading these models for language: zh-hans (Simplified_Chinese):
============================
| Processor    | Package   |
----------------------------
| tokenize     | gsdsimp   |
| pos          | gsdsimp   |
| lemma        | gsdsimp   |
| depparse     | gsdsimp   |
| sentiment    | ren       |
| constituency | ctb       |
| ner          | ontonotes |
============================

2023-03-06 16:48:53 INFO: Use device: cpu
2023-03-06 16:48:53 INFO: Loading: tokenize
2023-03-06 16:48:53 INFO: Loading: pos
2023-03-06 16:48:54 INFO: Loading: lemma
2023-03-06 16:48:54 INFO: Loading: depparse
2023-03-06 16:48:54 INFO: Loading: sentiment
2023-03-06 16:48:55 INFO: Loading: constituency
2023-03-06 16:48:55 INFO: Loading: ner
2023-03-06 16:48:56 INFO: Done loading processors!
```

In [88]:
```
doc = nlp(poem.iloc[2])
```

In [89]:
```
for sent in doc.sentences:
    # print("Sentence: " + sent.text) # original
    print("Tokenize: " + '/'.join(token.text for token in sent.tokens)) # seperation
    print()
```

```
Tokenize: 君/不/见/黄河/之/水/天/上/来/，/奔/流/到/海/不/复回/。

Tokenize: 君/不/见/高/堂/明/镜/悲/白/发/，/朝/如/青/丝/暮/成雪/。

Tokenize: 人生/得意/须/尽欢/，/莫/使/金/尊/空对/月/。

Tokenize: 天生/我/材必/有用/，/岑/夫子/，/丹丘/生/，/将/进酒/，/杯/莫停/。

Tokenize: 与/君歌/一/曲/，/古/来/圣/贤/皆/寂/寞/，/惟/有/饮者/留/其/名/。

Tokenize: 陈/王/昔/时/宴/平乐/，/五/花/马/，/千金/裘/，/呼儿/将/出换/美/酒/，/与/尔同销万/古愁/。
```

Advantages:

- It seperate every sentences into accurate smallest unit of words.

Disadvantages:

- Some segementation is wrong.
- The seperation is too detailed. To help you understand, you can see that almost every one character or two in Chinese poem works like an English words that stands for a meaning. For example, "高/堂/明/镜/" (large / room / bright/ mirror). However, if you group them together like "高堂/明镜" (large room / bright mirror), with this seperation, the word "高堂" can be understand as (parent). Seperation of sentences, especially in Chinese poem, plays a vital role while analyzing and undersanding the poems.

THULAC

URL: https://github.com/thunlp/THULAC-Python

In [90]:
```
thu = thulac.thulac(seg_only=True)
```

```
Model loaded succeed
```

In [91]:
```python
for sentence in sentences:
    text = thu.cut(sentence, text=True)
    print(text)
```

```
君 不 见 黄河 之 水 天上 来 ， 奔流 到 海 不 复 回君
不 见 高堂明镜 悲白发 ， 朝 如 青丝 暮 成 雪
人生 得意 须 尽欢 ， 莫 使 金尊空 对 月
天生我材必有用 ， 岑夫子 ， 丹丘生 ， 将 进酒 ， 杯 莫 停与
君歌 一 曲 ， 古 来 圣贤 皆 寂寞 ， 惟有饮者 留 其 名
陈王昔时 宴平乐 ， 五花马 ， 千金裘 ， 呼儿 将 出 换 美酒 ， 与尔 同 销万 古 愁
```

Advantages:

- This model actually is similar to jieba, which keeps more important words instead of split them into a smaller part, like "高堂明镜".
- Have an auto filter

Disadvantages:

- The mistakes is too much compare to jieba.

Conclusion (for text segementation)

With the observation and analysis after trying to seperate several poems with these models, I choose **jieba** as the model to do text segmentation as it can split out most of important words and has less error compare to other models.

2. Text segementation

In [92]:
```python
with open('Segemented_Poem.txt','w') as file:

    for index, poem in poems.iterrows():

        sentences = poem['paragraphs'].split("。")

        for sentence in sentences:

            ls = jieba.lcut(sentence)

            if '，' in ls:

                ls.remove('，')


            s = ' '.join(ls)

            if not s == '':

                file.write(s)

                file.write('\n')
```

Read segmentation file

In [93]:
```python
file = open('Segemented_Poem.txt')
sentences = [line.strip('\n') for line in file.readlines()]
```

3. Dictionary of tokens

References: 23_NLP_Preprocessing

In [94]:
```
TokenDict = dict ()

def build_token_dict_from_text(text, Dict):

    StandardText = text
    Set = StandardText.split(" ")

    for Element in Set:
        if (len(Element) == 0):
            continue
        if (Element in Dict):
            Dict[Element] = Dict[Element] + 1
        else:
            Dict[Element] = 1

    return

for sentence in sentences:

    build_token_dict_from_text(text=sentence, Dict = TokenDict )

print("Length of Dictionary: ", len ( TokenDict ) )
# print(sorted(TokenDict.items(), key=lambda x: x[1], reverse= True))
```

```
Length of Dictionary:  284922
```

In [95]:
```
print(TokenDict['兮'])
print(TokenDict['花'])
```

```
927
878
```

4. Lemmatisation

Since Chinese wrods don't have word form, we will skip this step. For example "你/吃/饭/了吗" (Did you eat lunch?), "我/吃/了" (I ate lunch already). Two "吃" here both means eat but do not different form.

5. Remove stop words

Resources:

- stop words databse from github: https://github.com/goto456/stopwords
- 23 Notebook

In [96]:
```
file2 = open('stop_words.txt')
stopwords = [line.strip('\n') for line in file2.readlines()]
print(stopwords)
```

```
['$', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '?', '_', '“', '”', '、', '。', '，',
'，', '．', '《', '》', '了', '在', '与', '后', '又', '无', '此', '前', '却', '一', '一些', '一
何', '一切', '一则', '一方面', '一旦', '一来', '一样', '一般', '一转眼', '万一', '上', '上下',
'下', '不', '不仅', '不但', '不光', '不单', '不只', '不外乎', '不如', '不妨', '不尽', '不尽然',
'不得', '不怕', '不惟', '不成', '不拘', '不料', '不是', '不比', '不然', '不特', '不独', '不管',
'不至于', '不若', '不论', '不过', '不问', '与', '与其', '与其说', '与否', '与此同时', '且', '且不
说', '且说', '两者', '个', '个别', '临', '为', '为了', '为什么', '为何', '为止', '为此', '为着',
'乃', '乃至', '乃至于', '么', '之', '之一', '之所以', '之类', '乌乎', '乎', '乘', '也', '也好',
'也罢', '了', '二来', '于', '于是', '于是乎', '云云', '云尔', '些', '亦', '人', '人们', '人家',
'什么', '什么样', '今', '介于', '仍', '仍旧', '从', '从此', '从而', '他', '他人', '他们', '以',
'以上', '以为', '以便', '以免', '以及', '以故', '以期', '以来', '以至', '以至于', '以致', '们',
'任', '任何', '任凭', '似的', '但', '但凡', '但是', '何', '何以', '何况', '何处', '何时', '余外',
'作为', '你', '你们', '使', '使得', '例如', '依', '依据', '依照', '便于', '俺', '俺们', '倘', '倘
使', '倘或', '倘然', '倘若', '借', '假使', '假如', '假若', '傥然', '像', '儿', '先不先', '光是',
'全体', '全部', '兮', '关于', '其', '其一', '其中', '其二', '其他', '其余', '其它', '其次', '具体
地说', '具体说来', '兼之', '内', '再', '再其次', '再则', '再有', '再者', '再者说', '再说', '冒',
'冲', '况且', '几', '几时', '凡', '凡是', '凭', '凭借', '出于', '出来', '分别', '则', '则甚',
'别', '别人', '别处', '别是', '别的', '别管', '别说', '到', '前后', '前此', '前者', '加之', '加
以', '即', '即令', '即使', '即便', '即如', '即或', '即若', '却', '去', '又', '又及', '及', '及
其', '及至', '反之', '反而', '反过来', '反过来说', '受到', '另', '另一方面', '另外', '另悉',
'只', '只当', '只怕', '只是', '只有', '只消', '只要', '只限', '叫', '叮咚', '可', '可以', '可是',
'可见', '各', '各个', '各位', '各种', '各自', '同', '同时', '后', '后者', '向', '向使', '向着',
'吓', '吗', '否则', '吧', '吧哒', '吱', '呀', '呃', '呕', '呗', '呜', '呜呼', '呢', '呵', '呵
呵', '呸', '呼哧', '咋', '和', '咚', '咦', '咧', '咱', '咱们', '咳', '哇', '哈', '哈哈', '哉',
'哎', '哎呀', '哎哟', '哗', '哟', '哦', '哩', '哪', '哪个', '哪些', '哪儿', '哪天', '哪年', '哪
怕', '哪样', '哪边', '哪里', '哼', '哼唷', '唉', '唯有', '啊', '啐', '啥', '啦', '啪达', '啷当',
'喂', '喏', '喔唷', '喽', '嗡', '嗡嗡', '嗬', '嗯', '嗳', '嘎', '嘎登', '嘘', '嘛', '嘻', '嘿',
'嘿嘿', '因', '因为', '因了', '因此', '因着', '因而', '固然', '在', '在下', '在于', '地', '基于',
'处在', '多', '多么', '多少', '大', '大家', '她', '她们', '好', '如', '如上', '如上所述', '如下',
'如何', '如其', '如同', '如是', '如果', '如此', '如若', '始而', '孰料', '孰知', '宁', '宁可', '宁
愿', '宁肯', '它', '它们', '对', '对于', '对待', '对方', '对比', '将', '小', '尔', '尔后', '尔
尔', '尚且', '就', '就是', '就是了', '就是说', '就算', '就要', '尽', '尽管', '尽管如此', '岂但',
'己', '已', '已矣', '巴', '巴巴', '并', '并且', '并非', '庶乎', '庶几', '开外', '开始', '归', '归
齐', '当', '当地', '当然', '当着', '彼', '彼时', '彼此', '往', '待', '很', '得', '得了', '怎',
'怎么', '怎么办', '怎么样', '怎奈', '怎样', '总之', '总的来看', '总的来说', '总的说来', '总而言
之', '恰恰相反', '您', '惟其', '慢说', '我', '我们', '或', '或则', '或是', '或曰', '或者', '截
至', '所', '所以', '所在', '所幸', '所有', '才', '才能', '打', '打从', '把', '抑或', '拿', '按',
'按照', '换句话说', '换言之', '据', '据此', '接着', '故', '故此', '故而', '旁人', '无', '无宁',
'无论', '既', '既往', '既是', '既然', '时候', '是', '是以', '是的', '曾', '替', '替代', '最',
'有', '有些', '有关', '有及', '有时', '有的', '望', '朝', '朝着', '本', '本人', '本地', '本着',
'本身', '来', '来着', '来自', '来说', '极了', '果然', '果真', '某', '某个', '某些', '某某', '根
据', '欤', '正值', '正如', '正巧', '正是', '此', '此地', '此处', '此外', '此时', '此次', '此间',
'毋宁', '每', '每当', '比', '比及', '比如', '比方', '没奈何', '沿', '沿着', '漫说', '焉', '然则',
'然后', '然而', '照', '照着', '犹且', '犹自', '甚且', '甚么', '甚或', '甚而', '甚至', '甚至于',
'用', '用来', '由', '由于', '由是', '由此', '由此可见', '的', '的确', '的话', '直到', '相对而言',
'省得', '看', '眨眼', '着', '着呢', '矣', '矣乎', '矣哉', '离', '竟而', '第', '等', '等到', '等
等', '简言之', '管', '类如', '紧接着', '纵', '纵令', '纵使', '纵然', '经', '经过', '结果', '给',
'继之', '继后', '继而', '综上所述', '罢了', '者', '而', '而且', '而况', '而后', '而外', '而已',
'而是', '而言', '能', '能否', '腾', '自', '自个儿', '自从', '自各儿', '自后', '自家', '自己', '自
打', '自身', '至', '至于', '至今', '至若', '致', '般的', '若', '若夫', '若是', '若果 ', '若非',
'莫不然', '莫如', '莫若', '虽', '虽则', '虽然', '虽说', '被', '要', '要不', '要不是', '要不然',
'要么', '要是', '譬喻', '譬如', '让', '许多', '论', '设使', '设或', '设若', '诚如', '诚然', '该',
'说来', '诸', '诸位', '诸如', '谁', '谁人', '谁料', '谁知', '贼死', '赖以', '赶', '起', '起见',
'趁', '趁着', '越是', '距', '跟', '较', '较之', '边', '过', '还', '还是', '还有', '还要', '这',
'这一来', '这个', '这么', '这么些', '这么样', '这么点儿', '这些', '这会儿', '这儿', '这就是说',
'这时', '这样', '这次', '这般', '这边', '这里', '进而', '连', '连同', '逐步', '通过', '遵循', '遵
照', '那', '那个', '那么', '那么些', '那么样', '那些', '那会儿', '那儿', '那时', '那样', '那般',
'那边', '那里', '都', '鄙人', '鉴于', '针对', '阿', '除', '除了', '除外', '除开', '除此之外', '除
非', '随', '随后', '随时', '随着', '难道说', '非但', '非徒', '非特', '非独', '靠', '顺', '顺着',
'首先', '！', '，', '；', '：', '？']
```

In [97]:
```python
def remove_stopwords_from_dict( Dict, StopwordList ):
    DictEx = copy.deepcopy(Dict)
    for Key in list ( DictEx.keys () ):
        if (Key in StopwordList ):
            del DictEx[Key]

    return (DictEx)
```

```
TokenCleanDict = remove_stopwords_from_dict(TokenDict, stopwords)
```

In [98]:
```
print("Length of Dictionary: ", len ( TokenCleanDict ) )
```

```
Length of Dictionary:  284483
```

6. visualize the token frequency using a word cloud

In [99]:
```
font_path="MSYH.TTC"

WC = WordCloud (width = 3840, height = 2160, background_color = "lightpink", max_words = 200,
                colormap = "Set2", relative_scaling = 0.75 ,font_path=font_path)

WC.generate_from_frequencies(TokenCleanDict)

plt.figure(figsize = (12, 7))
plt.axis("off")
plt.imshow(WC)
# plt.savefig("Frequency.png")
plt.show()
```

LDA processing bags of words

Glove is hard to install, I wish to do word vectors -_-

Creat word bag

In [119…
```
file = open('Segemented_Poem.txt')
sentences = [line.strip('\n') for line in file.readlines()]
```

In [120…
```
fenci = []

for sentence in sentences:
    keys = sentence.split(' ')
    for key in keys:
        if key in stopwords:
            keys.remove(key)
    fenci.append(keys)
```

In [121…

```
', ' in fenci
```

Out[121]:

```
False
```

In [122…

```
id2word = corpora.Dictionary(fenci)

corpus = [id2word.doc2bow(word) for word in fenci]

print(corpus[:1])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)]]
```

Try to train LDA model with 30 topics

In [123…

```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                id2word=id2word,
                num_topics=30,
                # random_state=100,
                # update_every=1,
                # chunksize=100,
                # passes=10,
                # alpha='auto',
                # per_word_topics=True
                )
```

In [126…

```
pprint(lda_model.print_topics(num_words=10))
```

```
[(11,
  '0.055*"空" + 0.042*"香" + 0.034*"山" + 0.034*"相逢" + 0.030*"何人" + 0.026*"残" + '
  '0.024*"无限" + 0.024*"苔" + 0.023*"日月" + 0.017*"帏"'),
 (8,
  '0.132*"欲" + 0.054*"清" + 0.038*"诗" + 0.036*"久" + 0.035*"轻" + 0.030*"惊" + '
  '0.028*"明" + 0.020*"天地" + 0.017*"居" + 0.014*"何如"'),
 (19,
  '0.102*"道" + 0.057*"皆" + 0.025*"传" + 0.023*"会" + 0.019*"卷" + 0.018*"乍" + '
  '0.018*"鹤" + 0.018*"绕" + 0.012*"罗" + 0.012*"迴"'),
 (2,
  '0.106*"时" + 0.062*"恨" + 0.039*"棹" + 0.036*"明月" + 0.029*"晚" + 0.025*"身" + '
  '0.021*"天涯" + 0.017*"郡" + 0.017*"穿" + 0.014*"自有"'),
 (25,
  '0.059*"路" + 0.033*"霜" + 0.032*"当时" + 0.027*"树" + 0.022*"名" + 0.021*"乾坤" + '
  '0.019*"偏" + 0.019*"千年" + 0.016*"画" + 0.015*"千载"'),
 (16,
  '0.042*"云" + 0.040*"隔" + 0.034*"初" + 0.029*"莺" + 0.026*"分明" + 0.022*"独" + '
  '0.020*"风吹" + 0.018*"古" + 0.015*"不觉" + 0.014*"何曾"'),
 (0,
  '0.094*"年" + 0.056*"事" + 0.045*"人间" + 0.041*"闲" + 0.032*"绿" + 0.031*"在" + '
  '0.020*"澹" + 0.018*"流水" + 0.018*"歇" + 0.017*"笙歌"'),
 (23,
  '0.126*"更" + 0.047*"客" + 0.045*"水" + 0.039*"有" + 0.029*"外" + 0.026*"闻" + '
  '0.022*"岁" + 0.020*"爱" + 0.017*"散" + 0.013*"虚"'),
 (28,
  '0.061*"新" + 0.059*"深" + 0.045*"吾" + 0.029*"正" + 0.028*"冷" + 0.027*"回首" + '
  '0.025*"意" + 0.025*"夜" + 0.020*"兼" + 0.017*"不能"'),
 (21,
  '0.051*"春风" + 0.048*"寒" + 0.033*"东风" + 0.028*"尘" + 0.023*"作" + 0.020*"阙" + '
  '0.017*"沉沉" + 0.016*"折" + 0.016*"西" + 0.014*"绮"'),
 (1,
  '0.052*"斜" + 0.048*"似" + 0.043*"落" + 0.040*"应" + 0.037*"情" + 0.020*"拂" + '
  '0.017*"无心" + 0.015*"遥" + 0.011*"变" + 0.010*"谒"'),
 (26,
  '0.075*"不知" + 0.049*"倚" + 0.041*"寄" + 0.035*"吹" + 0.029*"乱" + 0.023*"可怜" + '
  '0.022*"言" + 0.019*"卧" + 0.019*"断" + 0.014*"足"'),
 (7,
  '0.046*"花" + 0.045*"碧" + 0.045*"便" + 0.044*"风" + 0.037*"为" + 0.030*"不" + '
  '0.028*"相" + 0.027*"玉" + 0.024*"掩" + 0.021*"风流"'),
 (24,
  '0.045*"酒" + 0.040*"烟" + 0.037*"金" + 0.034*"秋风" + 0.033*"声" + 0.020*"家" + '
  '0.019*"今朝" + 0.016*"楼" + 0.013*"翡翠" + 0.011*"元"'),
 (10,
  '0.067*"□" + 0.058*"惆怅" + 0.050*"千里" + 0.041*"谁" + 0.037*"欹" + 0.032*"暖" + '
  '0.023*"忆" + 0.020*"阑" + 0.017*"离别" + 0.011*"从来"'),
 (17,
  '0.102*"日" + 0.034*"僧" + 0.032*"犹" + 0.025*"近" + 0.018*"遂" + 0.017*"行人" + '
  '0.014*"凤凰" + 0.014*"浮云" + 0.014*"佳人" + 0.013*"有余"'),
 (18,
  '0.053*"老" + 0.052*"今日" + 0.052*"坐" + 0.030*"砌" + 0.024*"鬓" + 0.023*"长安" + '
  '0.022*"非" + 0.020*"色" + 0.020*"寂寥" + 0.016*"悲"'),
 (15,
  '0.181*"月" + 0.046*"出" + 0.037*"笑" + 0.034*"难" + 0.031*"知" + 0.030*"不可" + '
  '0.018*"成" + 0.018*"低" + 0.017*"莺" + 0.017*"忘"'),
 (27,
  '0.038*"寻" + 0.035*"罢" + 0.027*"迟" + 0.026*"蝉" + 0.023*"转" + 0.023*"早" + '
  '0.022*"横" + 0.021*"春色" + 0.018*"珠帘" + 0.017*"紫"'),
 (6,
  '0.063*"春" + 0.049*"处" + 0.048*"长" + 0.047*"心" + 0.047*"是" + 0.024*"三" + '
  '0.021*"书" + 0.020*"分" + 0.020*"江南" + 0.017*"不堪"')]
```

Find the Perplexity and Coherence Score

- Perplexity: is a statistical measure of how well a probability model predicts a sample. This is calculated by splitting the dataset into two, train and test documents.

        - *Intuition:* higher likelihood implies a better model
- Coherence Score: is used for assessing the quality of the learned topics.
    - *Intuition:* topic is good if the word constituting the topic co-occur together. The score is used for deciding the required number of topics in the model.
- References: https://pahulpreet86.github.io/standard-metrics-for-lda-model-comparison/

In [127…

```
print('Perplexity: ', lda_model.log_perplexity(corpus))
coherence_model_lda = CoherenceModel(model=lda_model, texts=fenci, dictionary=id2word, cohere
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda)
```

```
Perplexity:  -36.67794154148063
Coherence Score:  0.5628651210588521
```

Find the optimal theme amount

In [138…

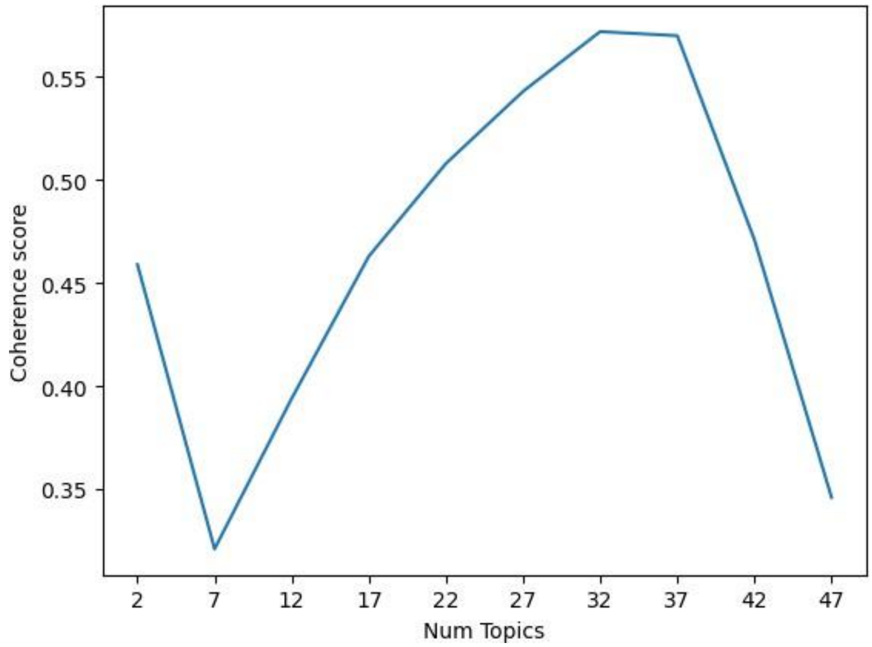
```
coherence_values = []
model_list = []
for num_topics in range(2,52,5):

    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                        id2word=id2word,
                        num_topics=num_topics,
                        # random_state=100,
                        # update_every=1,
                        # chunksize=100,
                        # passes=10,
                        # alpha='auto',
                        # per_word_topics=True
    )

    model_list.append(lda_model)
    coherencemodel = CoherenceModel(model=lda_model, texts=fenci, dictionary=id2word, coheren
    coherence_values.append(round(coherencemodel.get_coherence(),3))
```

In [139…

```
x = range(2,52,5)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.xticks(x)
plt.savefig('Best_model_rough.png')
plt.show()
```



We run it again to find the accurate value

In [140…

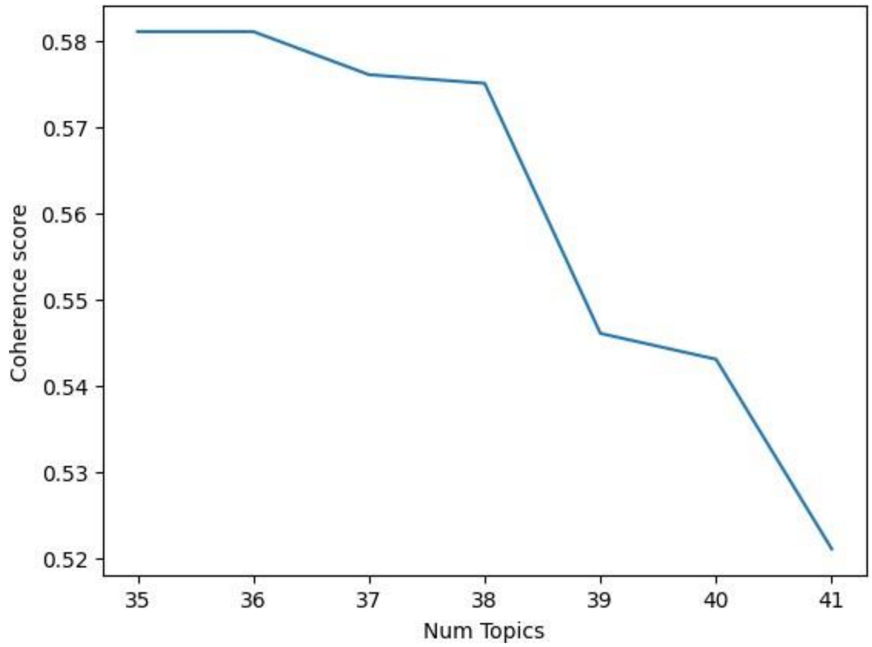
```
coherence_values_d = []
model_list_d = []
for num_topics in range(35,42):

    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                    id2word=id2word,
                    num_topics=num_topics,
                    # random_state=100,
                    # update_every=1,
                    # chunksize=100,
                    # passes=10,
                    # alpha='auto',
                    # per_word_topics=True
    )

    model_list_d.append(lda_model)
    coherencemodel_d = CoherenceModel(model=lda_model, texts=fenci, dictionary=id2word, coher
    coherence_values_d.append(round(coherencemodel_d.get_coherence(),3))
```

In [141…

```
x = range(35,42)
plt.plot(x, coherence_values_d)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.xticks(x)
plt.savefig('Best_model_detailed.png')
plt.show()
```



Coherence values to see the detailed information

In [149…

```
x = range(35,42)
for m, cv in zip(x, coherence_values_c):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 35  has Coherence Value of 0.581
Num Topics = 36  has Coherence Value of 0.581
Num Topics = 37  has Coherence Value of 0.576
Num Topics = 38  has Coherence Value of 0.575
Num Topics = 39  has Coherence Value of 0.546
Num Topics = 40  has Coherence Value of 0.543
Num Topics = 41  has Coherence Value of 0.521
```

Now we can see the optimal value for topics amount is 36

Best Theme

In [150…

```
optimal_model = model_list_c[1]
model_topics = optimal_model.show_topics(formatted=True)
pprint(optimal_model.print_topics(num_words=12))
```

```
[(35,
  '0.068*"出" + 0.039*"久" + 0.039*"住" + 0.036*"外" + 0.029*"无限" + 0.028*"迟" + '
  '0.025*"作" + 0.021*"西风" + 0.018*"西" + 0.017*"又" + 0.016*"青" + 0.016*"带"'),
 (31,
  '0.081*"心" + 0.074*"客" + 0.041*"闻" + 0.037*"向" + 0.026*"散" + 0.023*"曲" + '
  '0.022*"紫" + 0.017*"马" + 0.016*"收" + 0.013*"旌" + 0.011*"披" + 0.011*"图"'),
 (3,
  '0.120*"春" + 0.039*"分明" + 0.036*"兼" + 0.030*"不能" + 0.025*"移" + 0.016*"一点" + '
  '0.015*"麝" + 0.012*"神" + 0.011*"一半" + 0.008*"岫" + 0.008*"树影" + 0.006*"牛"'),
 (19,
  '0.048*"莺" + 0.045*"书" + 0.042*"风流" + 0.018*"何须" + 0.018*"窥" + 0.015*"林下" + '
  '0.013*"断续" + 0.012*"礼" + 0.012*"黍" + 0.011*"缕" + 0.008*"卮" + 0.008*"屡"'),
 (16,
  '0.092*"不知" + 0.051*"酒" + 0.037*"晚" + 0.033*"残" + 0.030*"传" + 0.029*"可怜" + '
  '0.026*"独" + 0.023*"弦" + 0.017*"俱" + 0.014*"叶" + 0.013*"月华" + 0.013*"识"'),
 (8,
  '0.248*"月" + 0.050*"高" + 0.039*"落" + 0.039*"诗" + 0.023*"思" + 0.021*"送" + '
  '0.015*"立" + 0.013*"竟" + 0.011*"灯" + 0.010*"镜" + 0.010*"君子" + 0.010*"变"'),
 (7,
  '0.080*"深" + 0.072*"梦" + 0.070*"空" + 0.043*"岂" + 0.036*"旧" + 0.030*"江南" + '
  '0.023*"乾坤" + 0.022*"花落" + 0.018*"弄" + 0.018*"三千" + 0.016*"帐" + 0.013*"属"'),
 (30,
  '0.079*"清" + 0.075*"人间" + 0.052*"在" + 0.036*"长安" + 0.019*"寒食" + 0.017*"春来" + '
  '0.015*"遍" + 0.014*"勿" + 0.012*"清明" + 0.011*"阳春" + 0.011*"呈" + 0.010*"樽"'),
 (27,
  '0.173*"，" + 0.044*"犹" + 0.042*"重" + 0.028*"逢" + 0.024*"偏" + 0.020*"何必" + '
  '0.016*"不须" + 0.014*"光" + 0.013*"依稀" + 0.012*"咫尺" + 0.011*"几多" + 0.009*"凄凄"'),
 (20,
  '0.084*"事" + 0.054*"入" + 0.051*"问" + 0.025*"枝" + 0.024*"郡" + 0.023*"今朝" + '
  '0.016*"罗" + 0.015*"千万" + 0.015*"殿" + 0.013*"风光" + 0.012*"便是" + 0.012*"求"'),
 (11,
  '0.072*"愁" + 0.060*"听" + 0.058*"醉" + 0.047*"烟" + 0.042*"莫" + 0.037*"归去" + '
  '0.032*"柳" + 0.029*"会" + 0.017*"足" + 0.015*"迴" + 0.013*"山川" + 0.010*"泣"'),
 (33,
  '0.076*"红" + 0.058*"便" + 0.050*"暮" + 0.043*"白" + 0.042*"天" + 0.039*"寻" + '
  '0.039*"不" + 0.033*"当时" + 0.023*"流水" + 0.014*"唱" + 0.010*"迎" + 0.009*"学"'),
 (32,
  '0.117*"见" + 0.106*"道" + 0.034*"东风" + 0.033*"相" + 0.033*"吟" + 0.026*"日月" + '
  '0.025*"将" + 0.023*"言" + 0.020*"卧" + 0.019*"帏" + 0.016*"春光" + 0.015*"绮"'),
 (34,
  '0.065*"倚" + 0.057*"回" + 0.049*"绿" + 0.024*"乍" + 0.024*"行人" + 0.019*"浮云" + '
  '0.016*"文章" + 0.013*"一朝" + 0.013*"谢" + 0.011*"夸" + 0.011*"荷" + 0.011*"题诗"'),
 (12,
  '0.080*"皆" + 0.041*"三" + 0.029*"寂寥" + 0.020*"城" + 0.020*"合" + 0.020*"想" + '
  '0.018*"尽日" + 0.017*"秋色" + 0.017*"昨日" + 0.015*"到" + 0.015*"九" + 0.015*"剑"'),
 (10,
  '0.060*"香" + 0.057*"万里" + 0.041*"暖" + 0.026*"疏" + 0.024*"忽" + 0.019*"何如" + '
  '0.017*"珠" + 0.016*"池塘" + 0.013*"李" + 0.012*"玄" + 0.012*"扫" + 0.012*"借问"'),
 (13,
  '0.083*"雨" + 0.078*"垂" + 0.033*"忆" + 0.028*"天下" + 0.021*"陪" + 0.021*"蓬莱" + '
  '0.017*"平" + 0.016*"捧" + 0.015*"群" + 0.010*"无为" + 0.007*"嫩" + 0.006*"珰"'),
 (6,
  '0.058*"闲" + 0.054*"似" + 0.050*"生" + 0.047*"共" + 0.038*"帘" + 0.032*"近" + '
  '0.023*"拂" + 0.019*"频" + 0.017*"栊" + 0.016*"余" + 0.015*"婵娟" + 0.015*"通"'),
 (23,
  '0.063*"难" + 0.033*"青山" + 0.030*"翠" + 0.026*"阴" + 0.023*"石" + 0.018*"走" + '
  '0.018*"白发" + 0.015*"可惜" + 0.015*"南" + 0.014*"元" + 0.013*"袅袅" + 0.012*"孤舟"'),
 (26,
  '0.130*"中" + 0.064*"远" + 0.060*"长" + 0.055*"路" + 0.042*"云" + 0.033*"砌" + '
  '0.029*"意" + 0.023*"忘" + 0.021*"鸟" + 0.020*"卷" + 0.019*"动" + 0.016*"没"')]
```

Conclusion for LDA part

As we can see here, the best theme is grouped by many topics. Inside these topics, we can see that each topics is build by key words. Each keyword contributes a certain weight to the topic, and the weight reflects the contribution of the keyword to the subject. For example, in topic 11, we can see that the word "路"(road) and "酒"(wine) have a heavy weight, which we can infer that this topic might related to someone who is on the road or poet are farewell friends. It is acutally an good model that some topics is understandable and siginiticant.

With LDA, we again compare three kinds of text segemntation

Jieba

The model with 30 topics has the perplexity and coherence Score as:

In [135…

```
print('Perplexity: ', optimal_model.log_perplexity(corpus))
coherence_model_lda = CoherenceModel(model=optimal_model, texts=fenci, dictionary=id2word, co
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda)
```

```
Perplexity:  -45.777008744749686
Coherence Score:  0.5901818539336339
```

Stanza

In [109…

```
nlp = stanza.Pipeline('zh')
```

```
2023-03-06 21:15:55 INFO: Checking for updates to resources.json in case models have been upd
ated.  Note: this behavior can be turned off with download_method=None or download_method=Dow
nloadMethod.REUSE_RESOURCES
Downloading  https://raw.githubusercontent.com/stanfordnlp/stanza-resources/main/resources_1.
4.1.json:   0%|    …
2023-03-06 21:15:56 INFO: "zh" is an alias for "zh-hans"
2023-03-06 21:15:58 INFO: Loading these models for language: zh-hans (Simplified_Chinese):
============================
| Processor    | Package   |
----------------------------
| tokenize     | gsdsimp   |
| pos          | gsdsimp   |
| lemma        | gsdsimp   |
| depparse     | gsdsimp   |
| sentiment    | ren       |
| constituency | ctb       |
| ner          | ontonotes |
============================

2023-03-06 21:15:58 INFO: Use device: cpu
2023-03-06 21:15:58 INFO: Loading: tokenize
2023-03-06 21:15:58 INFO: Loading: pos
2023-03-06 21:15:59 INFO: Loading: lemma
2023-03-06 21:15:59 INFO: Loading: depparse
2023-03-06 21:15:59 INFO: Loading: sentiment
2023-03-06 21:16:00 INFO: Loading: constituency
2023-03-06 21:16:00 INFO: Loading: ner
2023-03-06 21:16:01 INFO: Done loading processors!
```

In [116…

```
with open('Segemented_Poem_stanza.txt','w') as file:

    for index, poem in poems.iterrows():

        doc = nlp(poem['paragraphs'])

        for sentence in doc.sentences:

            s = ' '.join(token.text for token in sentence.tokens if token.text not in [', ','
```

```
            if not s == '':
                file.write(s)
                file.write('\n')
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
d:\File\2023_Spring\STAT_302\Final_Project\final_project.ipynb 单元格 106 in <cell line: 1>()
      <a href='vscode-notebook-cell:/d%3A/File/2023_Spring/STAT_302/Final_Project/final_project.ipynb#Y224sZmlsZQ%3D%3D?line=0'>1</a> with open('Segemented_Poem_stanza.txt','w') as file:
      <a href='vscode-notebook-cell:/d%3A/File/2023_Spring/STAT_302/Final_Project/final_project.ipynb#Y224sZmlsZQ%3D%3D?line=2'>3</a>     for index, poem in poems.iterrows():
----> <a href='vscode-notebook-cell:/d%3A/File/2023_Spring/STAT_302/Final_Project/final_project.ipynb#Y224sZmlsZQ%3D%3D?line=4'>5</a>         doc = nlp(poem['paragraphs'])
      <a href='vscode-notebook-cell:/d%3A/File/2023_Spring/STAT_302/Final_Project/final_project.ipynb#Y224sZmlsZQ%3D%3D?line=6'>7</a>         for sentence in doc.sentences:
      <a href='vscode-notebook-cell:/d%3A/File/2023_Spring/STAT_302/Final_Project/final_project.ipynb#Y224sZmlsZQ%3D%3D?line=8'>9</a>             s = ' '.join(token.text for token in sentence.tokens if token.text not in [', ','。 ','? ','; ',': ','! '])

File d:\Anaconda\lib\site-packages\stanza\pipeline\core.py:408, in Pipeline.__call__(self, doc, processors)
    407 def __call__(self, doc, processors=None):
--> 408     return self.process(doc, processors)

File d:\Anaconda\lib\site-packages\stanza\pipeline\core.py:397, in Pipeline.process(self, doc, processors)
    395     if self.processors.get(processor_name):
    396         process = self.processors[processor_name].bulk_processif bulk else self.processors[processor_name].process
--> 397         doc = process(doc)
    398 return doc

File d:\Anaconda\lib\site-packages\stanza\pipeline\constituency_processor.py:66, in ConstituencyProcessor.process(self, document)
     63 if self._tqdm:
     64     words = tqdm(words)
---> 66 trees = trainer.parse_tagged_words(self._model.model,words, self._batch_size)
     67 document.set(CONSTITUENCY, trees, to_sentence=True)
     68 return document

File d:\Anaconda\lib\site-packages\stanza\models\constituency\trainer.py:900, in parse_tagged_words(model, words, batch_size)
    897 model.eval()
    899 sentence_iterator = iter(words)
--> 900 treebank = parse_sentences(sentence_iterator, build_batch_from_tagged_words, batch_size, model)
    902 results = [t.predictions[0].tree for t in treebank]
    903 return results

File d:\Anaconda\lib\site-packages\torch\autograd\grad_mode.py:27, in _DecoratorContextManager.__call__.<locals>.decorate_context(*args, **kwargs)
     24 @functools.wraps(func)
     25 def decorate_context(*args, **kwargs):
     26     with self.clone():
---> 27         return func(*args, **kwargs)

File d:\Anaconda\lib\site-packages\stanza\models\constituency\trainer.py:856, in parse_sentences(data_iterator, build_batch_fn, batch_size, model, best)
    854 while len(tree_batch) > 0:
    855     _, transitions = predict(tree_batch)
--> 856     tree_batch = parse_transitions.bulk_apply(model, tree_batch, transitions)
    858     remove = set()
    859     for idx, tree in enumerate(tree_batch):

File d:\Anaconda\lib\site-packages\stanza\models\constituency\parse_transitions.py:614, in bulk_apply(model, tree_batch, transitions, fail)
    611     return tree_batch
    613 new_transitions = model.push_transitions([tree.transitionsfor tree in tree_batch], transitions)
--> 614 new_constituents = model.push_constituents(constituents, new_constituents)
    616 tree_batch = [state._replace(num_opens=state.num_opens+ transition.delta_opens(),
    617                              word_position=word_position,
```

W

```
    618                               transitions=transition_stack,
    619                               constituents=constituents)
    620               for (state, transition, word_position, transition_stack, constituents)
    621               in zip(tree_batch, transitions, word_positions, new_transitions, new_co
nstituents)]
    623 return tree_batch

File d:\Anaconda\lib\site-packages\stanza\models\constituency\lstm_model.py:744, in LSTMMode
l.push_constituents(self, constituent_stacks, constituents)
    742 hx = torch.cat([current_node.lstm_hx for current_node in current_nodes], axis=1)
    743 cx = torch.cat([current_node.lstm_cx for current_node in current_nodes], axis=1)
--> 744 output, (hx, cx) = self.constituent_lstm(constituent_input,(hx, cx))
    745 # Another possibility here would be to use output[0, i, :]
    746 # from the constituency lstm for the value of the new node.
    747 # This might theoretically make the new constituent include
   (...)
    752 # 150 epochs: 0.8971 to 0.8953
    753 # 200 epochs: 0.8985 to 0.8964
    754 new_stacks = [stack.push(ConstituentNode(constituent.value,constituents[i].tree_hx,
hx[:, i:i+1, :], cx[:, i:i+1, :]))
    755               for i, (stack, constituent) in enumerate(zip(constituent_stacks,consti
tuents))]

File d:\Anaconda\lib\site-packages\torch\nn\modules\module.py:1190, in Module._call_impl(sel
f, *input, **kwargs)
   1186 # If we don't have any hooks, we want to skip the rest of the logic in
   1187 # this function, and just call forward.
   1188 if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _gl
obal_backward_hooks
   1189         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1190     return forward_call(*input, **kwargs)
   1191 # Do not call functions when jit is used
   1192 full_backward_hooks, non_full_backward_hooks = [], []

File d:\Anaconda\lib\site-packages\torch\nn\modules\rnn.py:774, in LSTM.forward(self, input,
hx)
    772 self.check_forward_args(input, hx, batch_sizes)
    773 if batch_sizes is None:
--> 774     result = _VF.lstm(input, hx, self._flat_weights, self.bias, self.num_layers,
    775                       self.dropout, self.training, self.bidirectional, self.batch_fir
st)
    776 else:
    777     result = _VF.lstm(input, batch_sizes, hx, self._flat_weights, self.bias,
    778                       self.num_layers, self.dropout, self.training, self.bidirectiona
l)

KeyboardInterrupt:
```

Stanza is **tooooo slow**: 20 mins for only 4000 sentences (we have total 203037 sentences). Thus we deside to only compare THULAC and jieba

THULAC

In [108…

```python
thu = thulac.thulac(seg_only=True,filt=True)
```

```
Model loaded succeed
```

In [107…

```python
with open('Segemented_Poem_THULAC.txt','w') as file:

    for index, poem in poems.iterrows():

        sentences = poem['paragraphs'].split("。")

        for sentence in sentences:

            texts = thu.cut(sentence, text=False)
```

```
            s = ' '.join([text[0] for text in texts if not text[0] == ', '])

            if not s == '':

                file.write(s)

                file.write('\n')
```

We do all things again in LDA

In [130…

```
file2 = open('Segemented_Poem_THULAC.txt')
sentences2 = [line.strip('\n') for line in file2.readlines()]
```

stop words

In [131…

```
fenci2 = []

for sentence in sentences2:
    keys = sentence.split(' ')
    for key in keys:
        if key in stopwords:
            keys.remove(key)
    fenci2.append(keys)
```

In [132…

```
id2word2 = corpora.Dictionary(fenci2)

corpus2 = [id2word2.doc2bow(word) for word in fenci2]

print(corpus2[:1])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1)]]
```

In [136…

```
coherence_values2 = []
model_list2 = []
for num_topics in range(2,52,5):

    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus2,
                    id2word=id2word2,
                    num_topics=num_topics,
                    # random_state=100,
                    # update_every=1,
                    # chunksize=100,
                    # passes=10,
                    # alpha='auto',
                    # per_word_topics=True
    )

    model_list2.append(lda_model)
    coherencemodel2 = CoherenceModel(model=lda_model, texts=fenci2, dictionary=id2word2, cohe
    coherence_values2.append(round(coherencemodel2.get_coherence(),3))
```

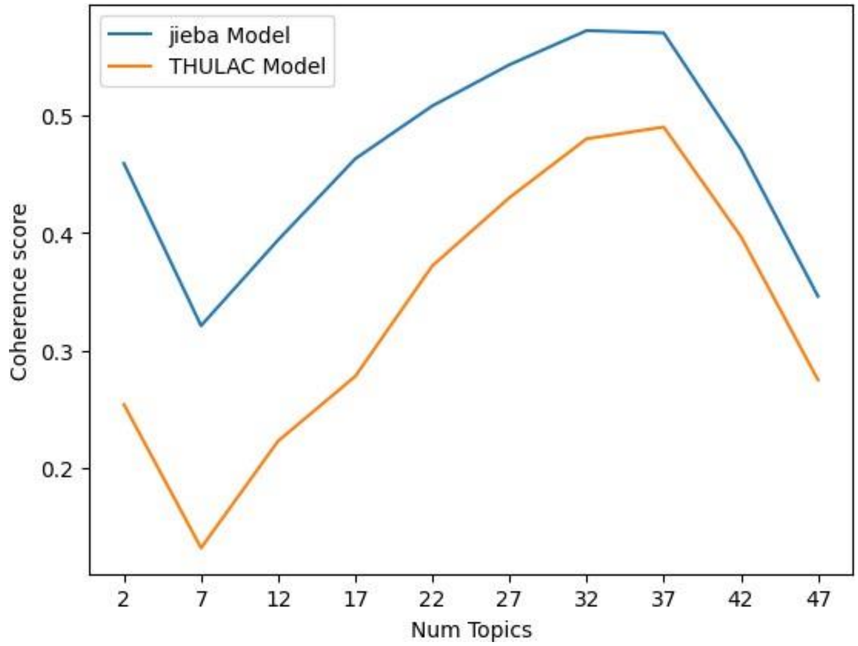
In [142…

```
x = range(2,52,5)
for m, cv in zip(x, coherence_values2):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.254
Num Topics = 7  has Coherence Value of 0.132
Num Topics = 12  has Coherence Value of 0.223
Num Topics = 17  has Coherence Value of 0.278
Num Topics = 22  has Coherence Value of 0.372
Num Topics = 27  has Coherence Value of 0.43
Num Topics = 32  has Coherence Value of 0.48
Num Topics = 37  has Coherence Value of 0.49
Num Topics = 42  has Coherence Value of 0.397
Num Topics = 47  has Coherence Value of 0.275
```

In [143…

```
plt.plot(x, coherence_values,label='jieba Model')
plt.plot(x,coherence_values2,label='THULAC Model')
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.xticks(x)
plt.legend()
# plt.savefig('Competition.png')
plt.show()
```



In [145…

```
optimal_model2 = model_list2[7]
model_topics2 = optimal_model2.show_topics(formatted=True)
pprint(optimal_model2.print_topics(num_words=12))
```

```
[(7,
  '0.147*"相" + 0.067*"晚" + 0.028*"修" + 0.023*"清风" + 0.020*"马" + 0.018*"著" + '
  '0.013*"山色" + 0.013*"明年" + 0.012*"寒食" + 0.011*"扇" + 0.010*"北风" + 0.010*"均"'),
 (25,
  '0.154*"开" + 0.044*"鸟" + 0.027*"自" + 0.022*"鸡" + 0.021*"深处" + 0.020*"遥" + '
  '0.020*"功" + 0.015*"独立" + 0.013*"绝" + 0.013*"遭" + 0.012*"海上" + 0.012*"眉"'),
 (9,
  '0.141*"来" + 0.095*"愁" + 0.092*"风" + 0.060*"香" + 0.058*"清" + 0.053*"能" + '
  '0.034*"含" + 0.015*"露" + 0.013*"上" + 0.011*"引" + 0.009*"浅" + 0.007*"咽"'),
 (17,
  '0.118*"长" + 0.045*"常" + 0.038*"久" + 0.036*"苦" + 0.028*"江南" + 0.024*"光" + '
  '0.021*"西风" + 0.020*"乡" + 0.018*"八" + 0.015*"离别" + 0.014*"恐" + 0.014*"只"'),
 (28,
  '0.149*"花" + 0.097*"难" + 0.089*"罗" + 0.063*"闻" + 0.052*"吹" + 0.045*"秋" + '
  '0.043*"酒" + 0.022*"发" + 0.021*"言" + 0.018*"居" + 0.013*"逐" + 0.011*"已"'),
 (8,
  '0.111*"初" + 0.051*"偏" + 0.036*"变" + 0.033*"种" + 0.025*"齐" + 0.023*"叶" + '
  '0.019*"殷勤" + 0.018*"把" + 0.017*"封" + 0.010*"弃" + 0.006*"远近" + 0.005*"羽客"'),
 (22,
  '0.218*"更" + 0.075*"去" + 0.062*"空" + 0.058*"莫" + 0.047*"听" + 0.028*"般" + '
  '0.020*"龙" + 0.017*"必" + 0.016*"好" + 0.013*"虎" + 0.013*"唯" + 0.011*"迟迟"'),
 (4,
  '0.118*"雨" + 0.055*"书" + 0.051*"会" + 0.045*"觉" + 0.034*"将" + 0.032*"散" + '
  '0.023*"暂" + 0.022*"复" + 0.021*"驻" + 0.014*"腰" + 0.012*"何须" + 0.011*"停"'),
 (30,
  '0.096*"君" + 0.076*"正" + 0.068*"行" + 0.051*"胜" + 0.042*"泪" + 0.042*"残" + '
  '0.039*"在" + 0.037*"百" + 0.027*"竟" + 0.025*"烧" + 0.017*"杯" + 0.008*"欢"'),
 (6,
  '0.137*"共" + 0.043*"登" + 0.041*"穷" + 0.027*"先生" + 0.023*"兴" + 0.016*"勿" + '
  '0.015*"四海" + 0.013*"仙" + 0.012*"驾" + 0.012*"呈" + 0.011*"颜色" + 0.011*"咫尺"'),
 (2,
  '0.128*"云" + 0.061*"南" + 0.042*"低" + 0.028*"渐" + 0.023*"黄金" + 0.022*"雾" + '
  '0.020*"动" + 0.019*"伊" + 0.018*"赏" + 0.015*"扫" + 0.014*"由" + 0.012*"融"'),
 (11,
  '0.157*"入" + 0.072*"情" + 0.052*"白云" + 0.048*"愿" + 0.042*"明月" + 0.035*"全" + '
  '0.030*"何人" + 0.026*"迎" + 0.023*"碧" + 0.016*"窗" + 0.009*"隐" + 0.002*"秦川"'),
 (16,
  '0.218*"时" + 0.070*"醉" + 0.055*"旧" + 0.054*"寄" + 0.052*"真" + 0.045*"逢" + '
  '0.031*"四" + 0.025*"令" + 0.024*"绕" + 0.017*"卧" + 0.015*"稀" + 0.014*"今朝"'),
 (29,
  '0.078*"事" + 0.072*"便" + 0.067*"皆" + 0.054*"处" + 0.050*"名" + 0.034*"多" + '
  '0.022*"过" + 0.019*"行人" + 0.018*"系" + 0.017*"合" + 0.014*"石" + 0.013*"乐"'),
 (15,
  '0.204*"家" + 0.091*"重" + 0.071*"坐" + 0.034*"生" + 0.029*"到" + 0.024*"病" + '
  '0.024*"遇" + 0.022*"访" + 0.018*"学" + 0.018*"乍" + 0.015*"艳" + 0.012*"响"'),
 (5,
  '0.114*"寻" + 0.047*"忘" + 0.041*"歌" + 0.034*"天地" + 0.032*"条" + 0.015*"短" + '
  '0.009*"才" + 0.008*"啸" + 0.007*"桃源" + 0.006*"夸" + 0.002*"滔滔" + 0.002*"枚"'),
 (14,
  '0.068*"双" + 0.052*"太" + 0.050*"西" + 0.049*"斜" + 0.040*"惊" + 0.031*"穿" + '
  '0.023*"苔" + 0.023*"唱" + 0.019*"落日" + 0.013*"熏" + 0.012*"阙" + 0.011*"灭"'),
 (27,
  '0.156*"应" + 0.087*"春" + 0.076*"作" + 0.066*"半" + 0.050*"住" + 0.049*"留" + '
  '0.042*"思" + 0.017*"通" + 0.016*"信" + 0.015*"阴" + 0.010*"聊" + 0.007*"辟"'),
 (33,
  '0.146*"天" + 0.071*"夜" + 0.057*"千" + 0.027*"倚" + 0.019*"背" + 0.019*"尊" + '
  '0.013*"主人" + 0.011*"也" + 0.010*"室" + 0.009*"瀑布" + 0.009*"兵" + 0.009*"进"'),
 (36,
  '0.176*"似" + 0.052*"争" + 0.047*"身" + 0.040*"十" + 0.038*"向" + 0.028*"同" + '
  '0.023*"从" + 0.021*"曾" + 0.013*"销" + 0.013*"车" + 0.013*"镜" + 0.012*"永"')]
```

In [147…

```python
print('Perplexity: ', optimal_model2.log_perplexity(corpus2))
coherence_model_lda2 = CoherenceModel(model=optimal_model2, texts=fenci2, dictionary=id2word2
coherence_lda2 = coherence_model_lda2.get_coherence()
print('Coherence Score: ', coherence_lda2)
```

```
Perplexity:  -41.387757370441165
Coherence Score:  0.49027544019686625
```

AA

Conclusion

From both graph, coherence score (higher is better), and perplexity (smaller is better), we can see that jieba is better than THULAC. Thus, our decision to use jieba as words segmentation for preprocessing before LDA is correct.

Summary and Discussion

In this project, we tried to build a database of Chinese poem, and use NLP models to analyze the data. We use LDA to analyze bag of words and compare different text segementation's contribute to LDA. As a result, we find out that jieba is the best text segementation library that we can directly use. Though there are errors in text segementation and stop words for Chinese poem is not mature, we can discover some interesting point through our result.

In the future, we can implement golve model while translating our database into English prompts and have deeper research on how to analyze the theme of Chinese poems and the similarity between different words. Also, we can try to mix two segmentation model together or train our own segmentation and update our own stop words database to improve the model.

APPENDIX B: GitHub repository link

https://github.com/IanMei/Signature-Work