ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

Dixit AI: An OpenAI Clip-based Hint Selection Player

Shi-Tao Chang

Jericho Senior High School, 99 Cedar Swamp Rd, Jericho, NY, USA, 11753

Abstract: *Artificial Intelligence (AI) has the potential to outperform humans in even the most strategic, creativity-based games. Previous work that incorporated AI players for such games—like Dixit—involving text generation has explored hint generation but never the implementation and thorough application of said generated text. To address these limitations, we built upon existing datasets of hints and established a game interface upon which we used CLIP by Open AI. We also use Jupyter Notebook and Conda to recognize images, convert them to vectors, compare the cosine similarity, and select the optimal hint thereafter for use in gameplay. We find that the AI player is successful in a significant percentage of the rounds in picking optimal hints. Future studies may address the generation of new hints to expand the dataset of hints from which the AI player can draw.*

Keywords: *Artificial Intelligence, AI players, Dixit, text-generation, hint selection, game interface, CLIP, OpenAI, cosine similarity*

1. Introduction

Artificial intelligence (AI) has achieved remarkable advances, unlocking novel applications that reach beyond traditional fields into areas like creativity and gameplay [1][2]. A growing area of interest is how AI can enhance creative, interactive experiences, particularly through language and image interpretation. For example, AIs designed for language and image recognition, such as ChatGPT and DALL-E, demonstrate the capacity to engage users creatively by generating responses or visual content based on user prompts [3]. While these tools are widely popular, their potential in interactive and game-based applications remains underexplored.

One domain that highlights the potential for AI-driven creativity is gameplay, especially in games that require both linguistic and visual reasoning. For instance, Dixit, a card game that combines elements of storytelling and image association, presents a unique challenge for AI, demanding a nuanced understanding of both visual cues and language-based hints. Unlike previous AI research, which has often focused on predicting which card matches a player's hint, our project seeks to develop an AI capable of engaging in comprehensive gameplay, requiring a deeper level of joint reasoning across image and language.

Our project therefore centers on advancing AI capabilities in creative gameplay by focusing on Dixit, which requires players to creatively interpret and communicate through images. This endeavor is motivated by the dual objectives of enriching interactive AI and pushing the boundaries of language-image understanding. By delving into a lesser-studied aspect of AI—integrative language and image processing within the context of gameplay—we aim to contribute valuable insights to the broader field of AI in creative applications.

2. Problem Setup

Dixit is an image-based card game where players interpret abstract illustrations to provide hints and make associations. Designed by Jean-Louis Roubira in 2008, the game uses 84 cards, each with a unique, ambiguous image that players interpret in their own way. The game involves multiple rounds where players take on different roles, creating a dynamic environment that challenges both verbal and visual creativity.

In each round, one player serves as the storyteller, selecting a card from their hand and providing a hint inspired by its image. The other players then choose cards from their hands that they believe could fit this hint. After shuffling and revealing all selected cards, players (except the storyteller) vote to

ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

identify the storyteller's card. The scoring system incentivizes the storyteller to create hints that are neither too straightforward nor too obscure.

2.1 AI-Driven Problem Objectives

To model a comprehensive AI player for Dixit, three main tasks emerge:

- **Hint Generation by the Storyteller**: This is the core task of our project, aiming to automate the generation of hints based on the card's visual features. The hint should strike a balance—specific enough for some players to recognize, but ambiguous enough to prevent unanimous guessing.

- **Card Selection Based on the Hint**: Each player (non-storyteller) must select a card that aligns with the storyteller's hint, maximizing the chance it will receive votes from others.

- **Voting on Played Cards**: Once cards are revealed, each player votes on the card they believe corresponds to the storyteller's hint. This task requires players to evaluate and rank options based on the initial hint.

While previous research has explored some aspects of these tasks, our research focuses specifically on the hint generation task, a complex and underexplored area in AI [4]. We aim to enhance AI's ability to interpret and generate associations across visual and linguistic elements, contributing a novel approach to interactive and creative AI applications.

2.2 Dataset Utilization

The goal of hint production in Dixit is to generate text descriptions for card images that prompt players to select the intended card without making the hint too explicit. This challenge lies in balancing clarity and ambiguity: hints should evoke the imagery or associated concepts of the card, trying to push players to pick the correct choice without being overly obvious.

A comprehensive evaluation of hint quality would involve testing against human players in full gameplay, but the latter is complex and resource-intensive. Instead, we are focusing on evaluating hint generation in isolation, relying on qualitative analysis rather than quantitative metrics because there is no established "gold standard" for hint effectiveness in this context. We qualitatively compare hints generated by our models with those produced by humans in the dataset, comparing and analyzing interpretive nuances and effectiveness.

2.3 Approach to Evaluation

Evaluating the quality of hints quantitatively is challenging due to the absence of an objective metric or benchmark. Instead, we perform qualitative comparisons by examining hints from the dataset that we know have generated high engagement—those that received the maximum votes (accounting for player count and voting limitations). Although maximizing votes might suggest an effective hint, we acknowledge the scoring nuances in Dixit; hints that all players guess correctly result in zero points for the storyteller.

In our study we are prioritizing the interpretive quality and engagement level of the hints, comparing our model's output to these high-scoring human hints.

By focusing on replicating effective human hints, our goal is to enhance AI-driven hint generation that aligns closely with the cognitive creativity involved in Dixit gameplay.

3. Approach

3.1 Intro to CLIP

CLIP (Contrastive Language-Image Pre-training) is a multimodal model developed by OpenAI, designed to associate images and text through shared representations for both modalities. Specifically, the model consists of two neural networks (transformers): An image encoder and a text encoder. The image encoder takes an image and maps it to a 512-dimensional vector, representing the image. The second, "text encoder", takes the text and maps it to another 512-dimensional vector. The core functionality of CLIP revolves around measuring the similarity between an image and a piece of text as

ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

the dot product (also known as scalar product, or inner product) between the vectors to which the CLIP encoders map them.

Mathematically, let EImage and EText be the image and the text encoder, respectively. The similarity sim between an image and a piece of text is computed as

$$u = E_{image}(Image), \quad v = E_{text}(Text), \quad sim(Image, Text) = \cos(u, v)$$

where u is the 512-dimensional vector ("embedding") representing the image, v is the embedding of the text, and cos(u,v) is the cosine between the two vectors, computed as

$$\cos(u, v) = \frac{u \cdot v}{||u|| ||v||}, \quad u \cdot v = \sum_{i=i}^{512} u_i v_i, \quad ||u|| = \sqrt{\sum_i u_i{}^2} .$$

CLIP was trained on a large dataset of paired images and text scraped from the Internet, with the goal of making the similarity for related image/text pairs higher than the similarity for any unrelated pair.

For our project, we use an open-source implementation of CLIP, called OpenCLIP, and rely on pre-trained models (encoders) available as part of the open-source project [5].

3.2 Relevance of CLIP to Our Goal

We can formulate task A, our focus generating a hint for a given card(image), as an optimization problem where the goal is to select the hint that maximizes similarity to the card. Let the card (image) in question be C. We can consider a set of possible hints (text strings) t1,...,tn and select the hint as

$$hint = argmax\_i \cos(E\_Image\ C), E\_text\ (t\_i))$$

In other words, we aim to choose the hint whose CLIP similarity to the card is the highest. The key questions we will now consider are: how can we construct the set of possible "candidate" hints? And are there reasonable baseline approaches we can consider that do not involve CLIP?

3.2.1 Dataset-Driven Baseline Approach

We consider a "baseline" approach that does not use any machine learning and directly uses the dataset. For each card, we identify the top k hints from the dataset--these are the hints that, when paired with the card during gameplay, received the highest number of votes from human players. When generating a hint, this approach simply selects one of these top-performing hints for the card.

This method works like a player who relies on memory. On one hand, this strategy might produce good scores: something that has worked in the past should work in the future. However in a repeated play (suppose you play against this player many times) this strategy would produce diminishing results since other players would eventually learn the exact (and limited) set of descriptions this player produces, associate the cards with these descriptions, and avoid voting for those cards. It also requires extensive gameplay data and lacks flexibility as it cannot create hints for cards that are underrepresented in the dataset. Moreover, it is a non-creative method: by reproducing past hints, it may become predictable over repeated games, allowing other players to adapt by associating specific hints with specific cards.

3.2.2 CLIP- Based Approach

To overcome the limitations of the dataset-driven baseline, we propose a CLIP-based approach that allows for the construction of new hints. The idea is to use a predefined vocabulary consisting of single words from the standard English dictionary, or from the set of words used in past Dicit games-which is easier and more manageable. We will delve more into it in the Methodology section.

Our focus in this research is on the AI's ability to generate hints for a given card image. The task involves selecting or generating a text description that closely matches the images in a way that aligns with how human players would interpret it, balancing between ambiguity and clarity to optimize the game's scoring mechanics.

Inherent to the nature of Dixit is the game mechanic of guessing the card's image through text descriptions, which translates into the AI's generative and descriptive capabilities nicely. The hint generation process goes as follows: we compute CLIP similarity between each word and the card for which we need to produce a hint and record the highest scoring (most similar) word. Then, we try to add one more word to it.We iteratively build on this initial word by adding others, forming two-word combinations, three-word combinations, and so on. For instance, if the vocabulary consists of: "dog", "cat", "black", "white", and "flowers", then the first stage requires computing 5 CLIP similarities; let's

ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

say that “cat” is the word that got selected in the first round. The second stage requires computing 4 similarities, namely, between the card image and the phrases “cat dog”, “cat black”, “cat white”, and “cat flowers”. These might not look like normal English expressions, but they are valid phrases—or more precisely valid combinations of words, which Dixit rules allow us to use.

We can continue with multiple generations until we construct a combination of k words. Now, we can either output the longest combination (the last one); the combination among these k that has the highest score; or we can randomly select one of these as our hint.

Previous work by Vatsakis et al. assembled a list of text-based hints for various cards, along with their corresponding accuracies. Each row contains columns that reveal the number of votes possible, the number of votes cast, and the ratio between the two. If the ratio between the two is the greatest possible ratio for the number of voters for the card, then the hint is considered to be ideal. Vatsakis et al. focused on the complementary task of card selection.

Unlike Vatsakis et al., who rely on established voting data for evaluation of their model, our research faces challenges due to the lack of a clear “gold standard” for hint generation. The subjective nature of hints, combined with the game's reliance on human intuition and cultural references, make it difficult to objectively measure the success of AI-generated hints.

4. Methods

4.1 Tools and Programming Language

- **Python 3 (Anaconda Distribution)**:

○ The primary programming language for this study was Python 3, managed through the Anaconda distribution, due to its comprehensive libraries for data science, machine learning, and natural language processing.

- **Integrated Development Environments (IDEs)**:

The following IDEs were utilized to facilitate code development and execution:

○ Jupyter Notebook: Employed for iterative coding, data visualization, and step-by-step testing.

○ Replit.com: An online IDE offering a collaborative environment for real-time code development.

○ Visual Studio Code (VS Code): Used for its advanced debugging capabilities, extensive extensions, and seamless integration with GitHub for version control

4.2 Data Sources

- **Dixit Cards**:

○ The primary programming language for this study was Python 3, managed through the Anaconda distribution, due to its comprehensive libraries for data science, machine learning, and natural language processing.

- **Gameplay Data**:

○ **Vatsakis et al. Dataset**: The primary dataset used was obtained from the study by Vatsakis et al., which encompasses records of human-played Dixit games. This dataset includes detailed information, such as hints provided for each card, voting outcomes, and game results. It was essential in implementing the Top K method and establishing a baseline for hint selection.

4.3 AI Model: CLIP by OpenAI

- **CLIP (Contrastive Language-Image Pretraining)**:

○ Developed by OpenAI, CLIP is a multimodal AI model that links images with corresponding textual descriptions. It includes two neural networks—an image encoder and a text encoder—both of which project inputs into a shared 512-dimensional embedding space.

○ **Implementation**: In this study, the pre-trained CLIP model was employed to identify Dixit card

ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

images and to generate or evaluate potential hints based on cosine similarity scores between image and text embeddings. This approach enabled the comparison of them against historical data.

4.4 Baseline Approach

- **Concept**:

○ The Top K method was used as a baseline approach for hint selection. By analyzing the Vatsakis et al. dataset, the most successful historical hints (i.e., those with the highest vote counts) were identified for each Dixit card.

- **Implementation**:

○ For each card, we selected the top 5 hints based on their historical success rate. This method served as a simple yet effective baseline, leveraging existing data to produce the most likely successful hints. The selected hints were then used to evaluate the AI model's performance against a human-like baseline.

- **Limitations**:

○ While the Top K method provides a reliable baseline, it relies heavily on historical data and lacks the ability to generate new, creative hints. It essentially memorizes past gameplay rather than adapting to new scenarios, which is a key limitation compared to the CLIP-based approach.

4.5 Game Code and Storage

- **Game Code:**

○ The implementation of the game, including the Top K method and CLIP-based hint generation, was executed using Python. Logic, data handling, and model integration were this framework.

- **Version Control:**

○ **GitHub:** All code and related files were stored and managed via GitHub for version control.

4.6 Evaluation Metrics

- **Requirement of Hint Selection:**

○ The primary metric for evaluating the model's performance was the accuracy of hint selection—how frequently the AI correctly identified or generated the most appropriate hint for a given Dixit card.

- **Comparison**:

○ The AI’s performance was benchmarked against the Top K method, as well as the average performance of human players from the Vatsakis et al. dataset. This dual comparison provided a comprehensive understanding and assessment of how well the AI performs in a real-world, game-playing scenario.

5. Results

Consider the following Dixit game round. The selected Dixit card is 35, and the player count is four. Below is the card image (see Figure 1).

ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

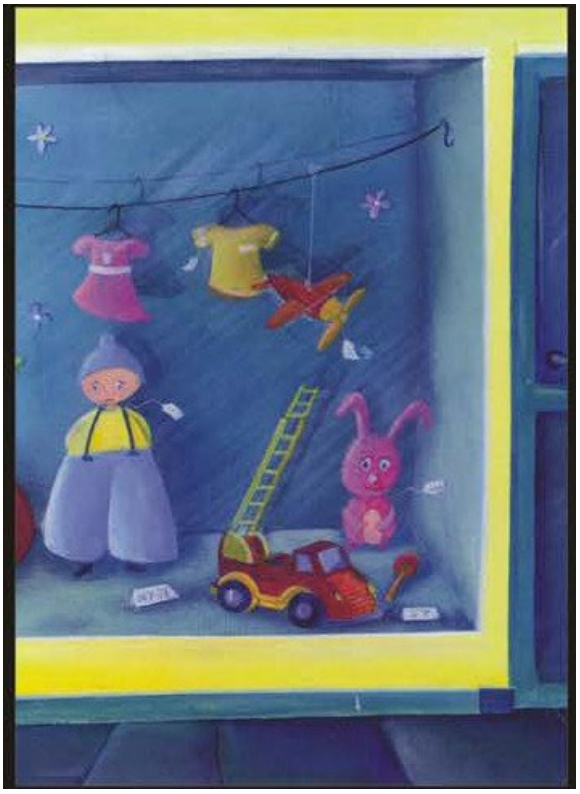

Figure 1: Dixit Card 35.

The AI-Player's generated hint is "tough decision," with 66% of guesses. Two-thirds of the players guessed in this case, marking it as the best possible outcome.

5.1 Statistics

The AI model's performance was evaluated based on its ability to consistently identify the most optimal hints for Dixit cards. The primary goal was to assess its effectiveness in generating and selecting hints that align with the historical data provided in the dataset.

5.1.1 Dataset Overview

The dataset used for training and testing the AI model included hints for 84 unique Dixit cards, distributed as follows:

- 41,442 hints for games with four players
- 15,539 hints for games with five players
- 35,999 hints for games with six players

These hints were drawn from the X_train.csv dataset, which comprised a total of 92,980 hints across different player configurations.

The AI's performance in selecting top hints was evaluated over 100+ rounds of playtesting. Key findings include:

- Hint Selection Accuracy:
- The AI model demonstrated 100% accuracy in selecting the most successful hints from the dataset, indicating that it consistently identified the top-performing hints based on the voter ratios provided for each card.

5.2 Game Outcomes

During playtesting, the AI won the majority of games, showcasing not only its precision in hint selection but also its overall competitiveness in gameplay. This confirms the model's ability to generate

ISSN 2616-5775 Vol. 7, Issue 11: 102-108, DOI: 10.25236/AJCIS.2024.071114

effective hints that align with the gameplay strategy needed to win. The ability of the Dixit AI player to consistently pick out top hints for its respective cards proves its usefulness and accuracy; its nature to select and present the best hints from the available dataset makes it ideal for hint generation and selection. Due to the required ratios needing to be ideal in terms of voters, only select hints were selected. The AI player also selected the best possible hints based on the voter ratio for each card, drawing from the dataset as intended. There were 84 Dixit cards provided; the X_train.csv dataset we used to get hints contained 41,442 hints for four players; 15,539 hints for five players; and 35,999 hints for six players. From these hints and 100+ rounds of playtesting, we determined that the AI successfully selected optimal hints 100% of the time and won a majority of the time.

6. Conclusion

The current implementation of the AI player demonstrates significant potential in automating hint selection in Dixit gameplay, with a notable accuracy in aligning hints with visual content. However, certain limitations persist. Primarily, the AI's reliance on a static dataset constrains its functionality over extended gameplay, leading to repetitive hint usage and diminishing effectiveness. Expanding the dataset to include a broader range of hints and incorporating dynamic hint generation could enhance the AI's adaptability and performance. Additionally, the authenticity and cultural relevance of hints need careful consideration, especially given that they originate from an external dataset (e.g., Vatsakis et al.). Future research should focus on developing models capable of generating novel hints, and utilizing advanced language models to introduce variability and creativity. These advancements could pave the way for a more robust and versatile AI, capable of engaging in richer and more human-like interactions during gameplay.

6.1 Limitations

Admittedly, the AI player has its limits. Because it was evaluated on a limited dataset, its training may not be sufficient to carry it victory in every match it plays.

The baseline evaluation approach is also limited—it accesses hints from a dataset, it may have limited functionality after several games, as hints for cards may become exhausted; the opponents may be faced with the same hint more than once. Additionally, the dataset itself is limited; the dataset could be more reliable if it contained more hints to select from, in terms of the baseline approach. Another issue that may raise concerns is the authenticity of the hints and their wording, due to their origination from a foreign entity, Vatsakis et al. Altogether, the baseline approach could use more dynamic functionality and the ability to generate hints itself, rather than taking from a dataset.

Overall, the model was instrumental in calculating the various successes of the card similarities. Future studies can continue its development by generating various descriptive hints for the AI player to pick from, adding to its dataset based on its potential success rates. This way, there'll be more variation and less of a chance for the player to "adapt" and see repeat hints used by the AI player.

References

[1] Gupta, A., Anpalagan, A., Guan, L. and Khwaja, A. S. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. Array, 10, 100057. https://doi.org/10.1016/j.array.2021.100057

[2] Bohr, A. and Memarzadeh, K. (2020). The rise of artificial intelligence in healthcare applications. Elsevier. https://doi.org/10.1016/b978-0-12-818438-7.00002-2

[3] Xue, Z., Xu, C. and Xu, X. (2023, July). Application of ChatGPT in natural disaster prevention and reduction. https://doi.org/10.1016/j.nhres.2023.07.005

[4] Vatsakis, D., Blom, P. M. and Spronck, P. (2022). An Internet-assisted Dixit-playing AI [Paper presentation]. 17th International Conference on the Foundations of Digital Games. https://doi.org/10.1145/3555858.3555863

[5] Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L. and Jitsev, J. (2024, July 13). Reproducible scaling laws for contrastive language-image learning. arXiv.org. https://arxiv.org/abs/2212.07143