Remote Viewing Experiment: Automated Analysis Framework

Overview

This project introduces an automated, objective methodology for conducting and analyzing remote viewing experiments. The framework addresses traditional challenges in remote viewing research, specifically:

- Subjective human judgment in evaluating viewer accuracy
- Lack of reproducibility in analysis
- Scalability limitations
- Potential experimental bias

Background

Traditional Remote Viewing Experiments

In traditional remote viewing experiments:

1. A sender selects a target image
2. A remote viewer attempts to describe the target without seeing it
3. Human judges evaluate the accuracy by comparing the viewer's description to the target
4. Success is determined through statistical analysis of matching accuracy

Current Limitations

- Human judgment introduces subjectivity
- Inconsistent evaluation criteria between different judges
- Time-consuming analysis process
- Difficult to replicate results across different studies
- Potential for unconscious bias in evaluation

Proposed Methodology

Automated Analysis Framework

Our method introduces two parallel analysis approaches:

1. **NLP-Based Analysis**
    - Standardized descriptions for target and decoy images
    - Semantic similarity comparison between viewer descriptions and image descriptions
    - Objective scoring based on linguistic similarity metrics
2. **CNN-Based Visual Analysis**
    - Computer vision analysis of image features
    - Direct comparison of visual similarities between images

  - Clustering-based validation of image relationships

Validation Strategy

The framework's validity is established by comparing:

- NLP-generated similarity matrices
- CNN-generated similarity matrices
- Visualization through dendrograms and heatmaps
- Correlation between linguistic and visual clustering patterns

Key Advantages

1. **Objectivity**: Removes human bias from evaluation process
2. **Reproducibility**: Standardized analysis methods ensure consistent results
3. **Scalability**: Automated analysis enables larger-scale experiments
4. **Validation**: Dual-analysis approach (NLP and CNN) provides robust validation
5. **Efficiency**: Reduces time and resources needed for analysis

Experimental Process

1. Image Pool Preparation

- Collection of target and decoy images
- Development of standardized descriptions
- Processing images for CNN analysis

2. Remote Viewing Session

- Target selection from image pool
- Remote viewer provides description
- Recording of viewer's description

3. Automated Analysis

- NLP processing of viewer descriptions
- CNN analysis of image features
- Generation of similarity matrices
- Creation of dendrograms and heatmaps

4. Statistical Analysis

- Comparison of similarity scores
- Clustering analysis
- Statistical significance testing

Technical Implementation

1. Image Description Standardization

- Selected 20 diverse images for initial validation

- Images chosen to represent varied characteristics:
  - Textures
  - Objects
  - Emotional content
  - Color schemes
- Created standardized descriptions:
  - 20 descriptors per image
  - Consistent complexity level across all images
  - Descriptions cover multiple aspects (visual, emotional, contextual)

2. NLP Analysis Implementation

SBERT Encoding Approaches

Approach Comparison

Two potential methods were considered for encoding image descriptions:

1. **Individual Descriptor Encoding**

   - *Process*: Encode each of the 20 descriptors separately
   - *Advantages*:
     - More granular representation of each descriptor
     - Maintains full semantic meaning of individual descriptors
     - Allows for descriptor-level similarity analysis
     - Better handles descriptors that might conflict or contradict
   - *Disadvantages*:
     - Results in 20 separate embedding vectors per image
     - More computationally intensive
     - Requires additional aggregation strategy
     - May lose contextual relationships between descriptors

2. **Combined Descriptor Encoding**

   - *Process*: Encode all 20 descriptors as one comma-separated text
   - *Advantages*:
     - Single embedding vector per image
     - Captures potential relationships between descriptors
     - More efficient computation
     - Simpler similarity comparison between images
   - *Disadvantages*:
     - May dilute the importance of individual descriptors
     - Could hit token length limits for transformer models
     - Risk of losing fine-grained semantic details
     - Potential for descriptor order to affect encoding

Implementation Decision

The combined descriptor encoding approach was selected for implementation:

- **Implementation Details**:
  - Uses all-MiniLM-L6-v2 SBERT model
  - Processes each image's 20 descriptors as a single text input
  - Generates one feature vector per image
  - Stores descriptors in combined_descriptors.txt

- **Key Implementation Benefits**:
  - Simplified similarity computation between images
  - More efficient processing pipeline
  - Maintains contextual relationships between descriptors
  - Single embedding vector per image enables straightforward clustering

- **Processing Flow**:
  1. Load combined descriptors from text file
  2. Encode each combined description using SBERT
  3. Generate similarity matrix using correlation distance
  4. Create hierarchical clustering using Ward linkage
  5. Visualize results through dendrograms and heatmaps

3. CNN Analysis Implementation

Feature Processing and Similarity Analysis

1. **Feature Vector Processing**
   - Features extracted from last maxpool layer of AlexNet
   - Each feature vector flattened to 1D array
   - Features stored as individual .pt files for reusability

   ```
   feature_vector = torch.load(feature_path).flatten().numpy()
   ```

2. **Similarity Computation**
   - Direct cosine similarity calculation between feature vectors
   - Chosen over correlation distance for better interpretability
   - Range: [-1, 1] where:
     - 1 indicates perfect similarity
     - 0 indicates orthogonality
     - -1 indicates opposite features

   ```
   similarity_matrix = cosine_similarity(cnn_feature_vectors)
   ```

Visualization Implementation

1. **Dendrogram Generation**
    - Uses Ward's linkage method for hierarchical clustering
    - Displays relationships between images based on visual features
    - Horizontal layout for better label readability
    - Parameters:

    ```
    dendrogram(
        linkage_matrix_ward_cnn,
        labels=cnn_image_labels,
        leaf_rotation=0,
        leaf_font_size=10
    )
    ```

2. **Heatmap Visualization**
    - Ordered to match dendrogram clustering
    - Diagonal values masked to focus on inter-image relationships
    - Color scaling:
        - vmin=-1, vmax=1 for full cosine similarity range
        - Centered at 0 for balanced visualization
        - Uses coolwarm colormap for intuitive interpretation
    - Parameters:

    ```
    sns.heatmap(
        ordered_similarity_matrix,
        annot=True,
        fmt='.2f',
        mask=mask,
        vmin=-1,
        vmax=1,
        center=0
    )
    ```

Key Implementation Decisions

1. **Similarity Metric Choice**
    - Cosine similarity preferred over correlation distance
    - Reasons:
        - More interpretable range (-1 to 1)
        - Standard in computer vision tasks
        - Better at capturing visual feature relationships
        - Scale-invariant comparison

2. **Visualization Choices**

  - Clean, minimal dendrogram design
  - Masked diagonal in heatmap
  - Two-decimal precision for similarity scores
  - Removed axis labels for cleaner presentation

3. **Data Organization**

  - Consistent ordering between dendrogram and heatmap
  - Image labels matched to actual filenames
  - Hierarchical structure preserved in visualizations

Analysis Benefits

1. **Visual Feature Comparison**

  - Direct comparison of image content
  - No reliance on semantic descriptions
  - Captures subtle visual similarities

2. **Objective Measurement**

  - Consistent feature extraction
  - Standardized similarity computation
  - Reproducible results

3. **Complementary to SBERT Analysis**

  - Provides visual perspective alongside semantic analysis
  - Enables validation of semantic relationships
  - Helps identify cases where visual and semantic similarities diverge

Future Directions

- Expansion of image dataset
- Refinement of similarity metrics
- Integration of additional analysis methods
- Development of real-time analysis capabilities

Impact

This methodology represents a significant advancement in remote viewing research by:

- Establishing objective evaluation standards
- Enabling larger-scale studies
- Providing reproducible results
- Creating a foundation for more rigorous scientific investigation

Visualization Methodology

Distance Matrix Generation Approaches

1. **Correlation-based Approach** (Original sbert-analysis.py)
    - Uses correlation distance metric
    - Focuses on pattern similarity
    - May emphasize relative relationships between features

    ```
    condensed_dist_matrix_cnn = pdist(sbert_feature_vectors,
    metric="correlation")
    ```

2. **Cosine Similarity Approach** (Improved working-dendrogram.py)
    - First computes cosine similarity
    - Better suited for semantic text embeddings
    - More interpretable for text-based comparisons

    ```
    similarity_matrix = cosine_similarity(embeddings_whole)
    distance_matrix = 1 - similarity_matrix
    ```

Implementation Decision

The cosine similarity approach was chosen as optimal for our experiment because:

- More appropriate for semantic text comparisons
- Standard metric in NLP tasks
- Better interpretation of similarity scores
- Directly comparable to human intuition about text similarity

Additional Analysis Features

The improved implementation includes:

- Top-N most related pairs identification
- Detailed similarity score output
- Multiple clustering method options
- Enhanced validation capabilities for remote viewing matches

Clustering and Visualization Implementation Decisions

Hierarchical Clustering Method Selection

We evaluated several hierarchical clustering approaches before selecting Ward's method:

1. **Single Linkage**
    - *Method*: Measures distance between closest points of clusters
    - *Characteristics*:
        - Creates long, chain-like clusters

    - Sensitive to noise and outliers
    - Good at finding elongated clusters
  - *Why Not Chosen*:
    - Too sensitive for semantic analysis
    - Creates unbalanced clusters
    - Doesn't reflect natural semantic groupings

2. **Complete Linkage**

   - *Method*: Uses maximum distance between points in clusters
   - *Characteristics*:
     - Creates compact, spherical clusters
     - Less sensitive to outliers
     - Tends to break large clusters
   - *Why Not Chosen*:
     - Too conservative for semantic relationships
     - May miss subtle semantic connections
     - Can create artificially small clusters

3. **Average Linkage**

   - *Method*: Uses mean distance between all pairs
   - *Characteristics*:
     - Compromise between single and complete
     - Moderately robust to outliers
     - Creates medium-sized clusters
   - *Why Not Chosen*:
     - Lacks clear theoretical justification for semantic data
     - Doesn't consider variance within clusters
     - Can be inconsistent with semantic relationships

4. **Ward's Method** (Chosen Approach)

   - *Method*: Minimizes variance within clusters
   - *Characteristics*:
     - Creates compact, well-defined clusters
     - Considers cluster structure holistically
     - Tends to create balanced clusters
   - *Why Chosen*:
     - Best reflects semantic relationships
     - Creates interpretable groupings
     - Maintains cluster cohesion
     - Ideal for semantic embedding spaces

Dendrogram Implementation

Several visualization options were considered:

1. **Color-Coded by Height**

  - Shows different levels of hierarchy in different colors
  - *Why Not Chosen*:
    - Too visually complex
    - Distracts from core relationships

2. **Multiple Threshold Lines**

  - Shows different possible clustering levels
  - *Why Not Chosen*:
    - Clutters visualization
    - Complicates interpretation

3. **Simple Structure with Single Threshold** (Chosen Approach)

  - Clean visualization
  - Single red threshold line
  - Clear branch structure
  - *Why Chosen*:
    - Maximizes interpretability
    - Focuses on key relationships
    - Matches semantic analysis needs

Heatmap Implementation

Considered various approaches for similarity visualization:

1. **Full Matrix with Diagonal**

  - Shows all similarities including self-similarity
  - *Why Not Chosen*:
    - Diagonal values dominate color scale
    - Reduces contrast for important comparisons

2. **Binary Threshold Visualization**

  - Shows only similarities above threshold
  - *Why Not Chosen*:
    - Loses granular relationship information
    - Too simplistic for semantic analysis

3. **Masked Diagonal with Normalized Scale** (Chosen Approach)

  - Hides diagonal values
  - Uses centered color scale
  - Shows full range of similarities
  - *Why Chosen*:
    - Focuses on inter-image relationships
    - Maximizes visual contrast
    - Preserves all relevant information
    - Easier to interpret semantic relationships

Integration of Methods

The combination of Ward's clustering, simple dendrogram, and masked heatmap provides:

1. Robust semantic clustering
2. Clear visualization of relationships
3. Detailed similarity information
4. Balance between detail and interpretability

This integrated approach best serves our remote viewing analysis by:

- Identifying meaningful semantic groups
- Showing hierarchical relationships
- Quantifying similarities between descriptions
- Supporting objective analysis of viewer accuracy

CNN Implementation Details

AlexNet Architecture and Feature Extraction

1. **Why AlexNet?**
   - Proven architecture for image feature extraction
   - Well-understood feature hierarchy
   - Pre-trained on diverse ImageNet dataset
   - Efficient computation and robust features
   - Strong performance in transfer learning tasks

2. **Hook Implementation for Feature Extraction**

```python
class FeatureExtractor:
    def __init__(self, model):
        self.model = model
        self.features = None
        self.hook = model.features[12].register_forward_hook(self.hook_fn)
```

   - Hooks are PyTorch mechanisms that:
     - Intercept intermediate layer outputs
     - Allow access to feature maps without modifying model
     - Enable extraction of specific layer representations
   - Layer 12 (last maxpool) chosen because:
     - Contains high-level semantic features
     - Balances abstraction and detail
     - Provides compact representation

3. **Feature Processing Pipeline**

```
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                       std=[0.229, 0.224, 0.225])
])
```

- Standardize image sizes
- Apply ImageNet normalization
- Convert to tensor format
- Ensure consistent processing

Dual Heatmap Visualization Strategy

1. **Raw Similarity Heatmap**
    - Shows unmodified cosine similarity values
    - Range: [-1, 1]
    - Centered at 0
    - Advantages:
        - Preserves actual similarity measures
        - Shows true relationship strengths
    - Limitations:
        - May be hard to visually interpret
        - Subtle differences less apparent

2. **Interpolated Similarity Heatmap**

```
min_val = ordered_similarity_matrix[~mask].min()
max_val = ordered_similarity_matrix[~mask].max()
interpolated_matrix = (ordered_similarity_matrix - min_val) / (max_val -
min_val)
```

    - Rescales values to [0, 1] range
    - Based on actual data range
    - Benefits:
        - Enhanced visual contrast
        - Easier pattern recognition
        - Better group identification
    - Implementation:
        - Masks diagonal values
        - Preserves relative relationships
        - Shows original values in annotations

3. **Visualization Parameters**

- Side-by-side display for comparison
- Consistent ordering with dendrogram
- Masked diagonal entries
- Coolwarm colormap for intuitive interpretation
- Annotations show original values

4. **Analysis Benefits**

- Complementary views of similarity structure
- Balance between accuracy and interpretability
- Facilitates pattern discovery
- Supports both detailed and overview analysis

Comparison Analysis: CNN vs SBERT Features

Implementation Strategy

1. **Dimension Matching**

- Ensure equal number of samples between CNN and SBERT features
- Use sequential sampling to maintain consistency
- Preserve image-label correspondence across both analyses

2. **Visualization Framework**

- Three separate comparative visualizations:
    1. Side-by-side dendrograms
    2. Raw similarity heatmaps
    3. Interpolated similarity heatmaps
- Consistent layout and scaling for direct comparison

3. **Correlation Analysis**

- Pearson correlation: Measures linear relationship
- Spearman correlation: Measures monotonic relationship
- P-values for statistical significance

```
correlation_pearson, p_value_pearson = pearsonr(condensed_dist_matrix_cnn,
condensed_dist_matrix_sbert)
correlation_spearman, p_value_spearman =
spearmanr(condensed_dist_matrix_cnn, condensed_dist_matrix_sbert)
```

Visualization Components

1. **Dendrogram Comparison**

- Purpose:
    - Compare hierarchical relationships
    - Identify clustering patterns

        - Validate structural similarities
    - Implementation:
        - Ward linkage for both methods
        - Consistent leaf rotation and font size
        - Aligned distance scales

2. **Raw Similarity Heatmaps**

    - Purpose:
        - Compare actual similarity values
        - Identify relationship strengths
        - Validate measurement consistency
    - Features:
        - Masked diagonal values
        - Consistent color scaling (-1 to 1)
        - Original similarity scores

3. **Interpolated Heatmaps**

    - Purpose:
        - Enhanced pattern visualization
        - Relative relationship comparison
        - Easier interpretation of groupings
    - Implementation:

    ```
    interpolated_matrix = (similarity_matrix - min_val) / (max_val -
    min_val)
    ```

    - Benefits:
        - Normalized scale (0 to 1)
        - Preserved relative relationships
        - Better visual contrast

Analysis Benefits

1. **Validation**

    - Cross-validation between visual and semantic features
    - Confirmation of relationship patterns
    - Identification of discrepancies

2. **Interpretation**

    - Multiple perspectives on relationships
    - Both absolute and relative comparisons
    - Clear visualization of similarities and differences

3. **Quality Control**

  - Verification of feature extraction
  - Validation of clustering approach
  - Confirmation of analysis consistency

Implementation Details

- Consistent preprocessing
- Matched sample sizes
- Aligned visualization parameters
- Standardized analysis metrics

Automated Image Selection Using Hierarchical Clustering

Overview

To automatically select a diverse subset of images from a large database, we use hierarchical clustering with Ward's method. This approach was validated using the CIFAR-100 dataset and can be applied to any image collection.

Selection Process

1. **Feature Extraction**
   - Run images through AlexNet's convolutional layers
   - Extract features from last maxpool layer
   - Flatten features into vectors

2. **Similarity Matrix Construction**

```
# Compute cosine similarity between all feature vectors
similarity_matrix = cosine_similarity(cnn_features)
```

3. **Hierarchical Clustering**

```
# Convert similarities to distances
distances = 1 - similarity_matrix

# Perform Ward's hierarchical clustering
linkage_matrix = linkage(distances, method='ward')

# Cut tree to get desired number of clusters
clusters = fcluster(linkage_matrix, n_clusters=100, criterion='maxclust')
```

4. **Representative Selection**
   - For each cluster, select the image closest to cluster center
   - This ensures selected images are:

        - Maximally different from each other
        - Representative of their local neighborhood
        - Well-distributed across the feature space

Implementation Methods

1. **Ward's Method**

    - Minimizes within-cluster variance
    - Creates compact, balanced clusters
    - Better than single/complete linkage for this task
    - Used in CIFAR-100 validation

2. **Center Selection Strategy**

```
for cluster_id in unique_clusters:
    cluster_indices = np.where(clusters == cluster_id)[0]

    # Find image closest to cluster center
    cluster_similarities = similarity_matrix[cluster_indices][:,
cluster_indices]
    mean_similarities = np.mean(cluster_similarities, axis=1)
    center_idx = cluster_indices[np.argmax(mean_similarities)]

    selected_indices.append(center_idx)
```

3. **Validation Approach**

    - Visualize similarity matrix
    - Check cluster distributions
    - Analyze intra/inter-cluster distances
    - Compare selected images visually

CIFAR-100 Validation Results

- Successfully selected diverse subsets (20-100 images)
- Maintained class distribution
- Achieved good coverage of feature space
- Validated through visual inspection and similarity metrics

Advantages of This Approach

1. **Automatic Selection**

    - No manual intervention needed
    - Scales to any dataset size
    - Consistent, reproducible results

2. **Diversity Guarantee**

   - Each selected image represents a distinct cluster
   - Maximum separation between selected images
   - Good coverage of full dataset

3. **Quality Control**

   - Center selection avoids outliers
   - Ward's method ensures balanced clusters
   - Easy to validate through visualization

Application to New Datasets

1. Extract CNN features from all images
2. Compute similarity matrix
3. Perform hierarchical clustering
4. Select cluster centers
5. Validate selections through visualization

This method can automatically select any number of representative images from a larger dataset while ensuring maximum diversity and representativeness.