Semantic Clustering and Classification in Text Analysis and Machine Learning

April 27, 2025

1. Clustering Semantic Word Proximity in Novels

1.1. Research Context and Objectives

This project investigates semantic proximity between content words in English novels through unsupervised clustering and visualization. The main objective is to construct context-based word representations and uncover latent semantic structures via distance modeling and graph analysis.

The workflow [1] includes: (1) corpus cleaning; (2) selection of 100 high-frequency content words; (3) TF-IDF feature construction within context windows; (4) comparison of clustering methods (KMeans, Agglomerative Clustering, DBSCAN); and (5) modeling semantic distances using greedy matching and Dijkstra's shortest paths.

1.2. Methodological Framework

1.2.1. Corpus Construction and Cleaning

The corpus used in this project was selected from Project Gutenberg, consisting of four English novels: *Frankenstein*, *Great Expectations*, *Pride and Prejudice*, and *The Secret Garden*. These texts are all in the public domain, sufficiently long, stylistically diverse, and linguistically modern, making them suitable for semantic modeling.

After removing headers, lowercasing, stripping non-alphabetic characters, normalizing spacing, and line-by-line cleaning, the corpus contains 42,529 lines and 488,145 tokens.

1.2.2. Content Word Selection

Rather than using a flat distribution method (equal number of each POS) or simple frequency ranking (top 100 frequent words after stopword removal), this project adopted a proportional distribution strategy based on the actual part-of-speech (POS) ratios observed in the corpus. This approach better preserves the corpus's natural syntactic structure and avoids overrepresentation of any single word category, making it more suitable for downstream semantic distance computation and clustering visualization tasks.

The process included: (1) removing function words with NLTK's stopword list; (2) POS-tagging the top 200 frequent tokens; and (3) selecting 44 nouns, 25 verbs, 18 adjectives, and 13 adverbs based on POS proportions. I further filtered out proper nouns, errors, and ambiguous tokens (e.g., *dec*, *th*).

The finalized list of 100 content words was saved as `L.txt` for downstream distance computation and clustering.

1.2.3. Word Distance Measurement

To measure semantic proximity, I implemented a greedy matching algorithm to compute average contextual distances between all word pairs in the vocabulary list `L`. For each (w_1, w_2) pair, I extracted shortest token-level distances across the corpus and computed average distances. Similarity scores were calculated as:

$$similarity(w_1, w_2) = \frac{1}{1 + distance(w_1, w_2)} \quad (1)$$

This produced a complete word-to-word similarity matrix (Figure 1). Coherent semantic clusters emerged, while unrelated words appeared more dispersed.

I also extracted the top 20 most and least similar word pairs based on the average contextual distance. As shown in Table 1, pairs such as "look – said" exhibited strong semantic proximity, often appearing together in narrative segments involving characters and dialogue. In contrast, pairs like "said – colin" had large contextual distances, even though both words were frequent in the corpus. This discrepancy occurs because "said" is widely distributed across dialogues, while "colin" is a specific character name that appears mainly within certain chapters, leading to minimal direct semantic overlap. These findings further confirm that contextual distance captures meaningful relationships beyond simple word frequency.

[1] Detailed implementation processes and figures are available in the supplementary Jupyter Notebook `Q1.ipynb`.

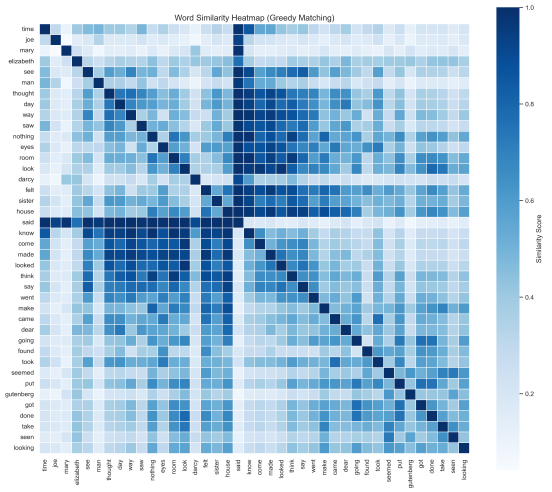

Figure 1: Heatmap of word-to-word similarity based on average contextual distance.

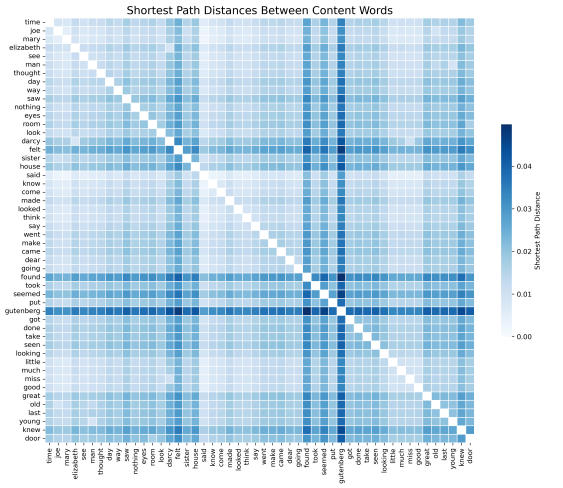

Figure 2: Heatmap of shortest-path distances between words based on co-occurrence graph structure.

Type	Word Pair	Avg Dist.	Sim. Score
Similar	look – said	166.02	0.9982
Similar	way – said	245.62	0.9973
Similar	house – said	346.46	0.9962
Similar	seen – little	709.30	0.9923
Dissimilar	said – colin	384955.13	0.0147
Dissimilar	man – dickon	374197.56	0.0165
Dissimilar	time – tha	335205.56	0.0254
Dissimilar	little – dickon	323771.92	0.0288

Table 1: Examples of similar and dissimilar word pairs with distances and similarity scores.

1.2.4. Unsupervised Word Clustering

Based on the contextual TF-IDF features of content words, this project explored three unsupervised clustering methods: KMeans, Agglomerative Clustering, and DBSCAN. All models used standardized feature matrices. KMeans and Agglomerative were initially set to six clusters, while the `eps` parameter for DBSCAN was tuned using k-distance plots.

Clustering performance was evaluated using three metrics: Silhouette Score, Davies–Bouldin Index (DB Index), and Calinski–Harabasz Index (CH Index). The results are summarized in Table 2.

Among the three methods, Agglomerative Clustering achieved the best overall performance. It was the only method to obtain a positive Silhouette Score and the highest Calinski–Harabasz Index, indicating tighter intra-cluster cohesion and clearer inter-cluster separation. Semantic groups formed by Agglomerative were more interpretable, such as character names (*colin*, *mary*), perception verbs (*look*, *see*), and spatial nouns (*garden*, *room*).

Metric	KMeans	Agglomerative
Silhouette Score (↑)	-0.0155	0.0071
Davies–Bouldin Index (↓)	2.1004	3.5210
Calinski–Harabasz Index (↑)	1.3204	1.6679

Table 2: Comparison of clustering performance between KMeans and Agglomerative Clustering.

In contrast, KMeans produced a dominant cluster with many scattered outliers. It showed a negative Silhouette Score, unclear cluster boundaries, and weaker interpretability, although it slightly outperformed Agglomerative in Davies–Bouldin Index.

DBSCAN failed to produce meaningful clusters in this task. Most words were assigned to a single large cluster or labeled as outliers, resulting in near-zero scores across all evaluation metrics. This failure is mainly due to the high-dimensional sparsity of TF-IDF features, which is unsuitable for density-based clustering.

Based on these results, Agglomerative Clustering was selected as the base model for further analysis. Cluster number optimization was then performed. When the number of clusters was set to three, the Silhouette Score reached a local maximum, the Davies–Bouldin Index dropped significantly, and the Calinski–Harabasz Index remained high. Therefore, `n_clusters=3` was finalized as the optimal configuration for subsequent semantic structure modeling.

1.2.5. Graph-Based Semantic Refinement

To enhance semantic structure modeling, I introduced a graph-based method using Dijkstra's shortest paths. Each node represents a content word; undirected edges connect words co-occurring within a 20-token window, with edge

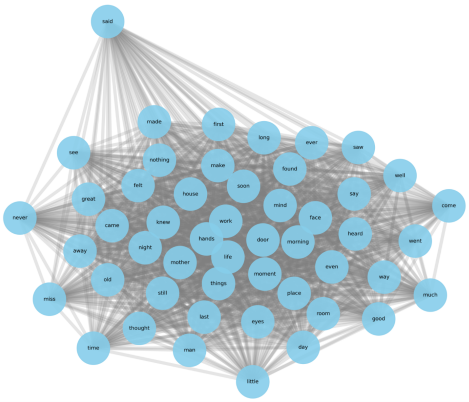

Figure 3: Word co-occurrence network of the top 50 high-degree nodes. Central nodes exhibit strong semantic connections.

weight defined as the reciprocal of co-occurrence frequency. No frequency threshold was applied to retain full graph connectivity; low-frequency edges naturally contribute less due to higher weights.

Applying Dijkstra's algorithm produced a complete semantic distance matrix. The graph showed strong global connectivity (minimum distance 0, maximum 0.0548, average path length 0.0102, diameter 1). As shown in Figure 2, the heatmap revealed clear local clusters: character-related words (*said*, *mary*, *miss*) grouped tightly, while structural terms (*project*, *gutenberg*) formed independent clusters, demonstrating the model's ability to capture both direct and indirect semantic relationships.

I visualized the top 50 high-degree nodes in the graph (Figure 3). Core words like *said*, *see*, and *never* were highly connected, while words such as *life*, *door*, and *work* appeared at the periphery, indicating focused semantic usage. The network revealed clear semantic stratification and category transitions.

Finally, I re-applied Agglomerative Clustering using Dijkstra-based distances. The overall clustering pattern remained consistent with greedy distance results, but boundary word assignments differed, highlighting that graph-based modeling offers advantages in capturing semantic propagation and indirect connections while maintaining structural interpretability.

1.3. Key Insights and Interpretations

To assess the impact of semantic distance definitions, I compared clustering results based on greedy average distance and Dijkstra graph distance, using Agglomerative Clustering with $K = 3$.

Greedy distance reflects direct co-occurrence proximity, while Dijkstra distance captures structural reachability through graph shortest paths. As shown in Figure 4, greedy distance produced

Figure 4: Comparison of clustering results using two semantic distance measures: (Top) Greedy distance; (Bottom) Dijkstra distance.

a "main cluster + outliers" pattern, aggregating general-purpose words, whereas Dijkstra distance yielded a more balanced distribution, grouping character, scene, and event terms into interpretable clusters.

From the confusion matrix and transition map (Figure 5), 88 words changed clusters between the two methods, with 80 words originally in greedy Cluster 2 moving entirely into Dijkstra Cluster 0. Clustering consistency evaluation showed near-zero agreement (ARI = -0.0579, NMI = 0.0196), confirming that semantic distance definition fundamentally shapes clustering outcomes.

Structurally, Dijkstra distance revealed latent semantic coherence: words like *joe*, *see*, *thought*, and *elizabeth*, previously scattered, were reorganized into the same cluster. This demonstrates the ability of graph-based distances to capture deeper semantic relationships beyond surface co-occurrence.

In summary, greedy distance captures local contextual proximity, while Dijkstra distance offers stronger semantic penetration. For robust semantic modeling, graph-based methods provide clear advantages.

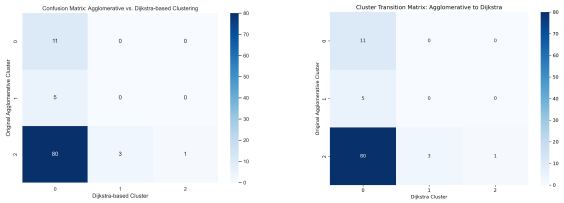

Figure 5: Cluster reassignments between greedy and Dijkstra distance-based clustering. (Left) Confusion matrix; (Right) Transition matrix.

1.4. Contributions and Future Directions

This project systematically explored semantic structures among content words, completing corpus cleaning, word selection, distance modeling, unsupervised clustering, and graph construction. Comparing greedy average distance and Dijkstra distance, I found that greedy distance is simpler but biased by frequency and sentence structure, often clustering high-frequency words together. Dijkstra

distance, based on co-occurrence graphs, better captures indirect semantic links and yields more balanced clustering, though at higher computational cost.

Several technical details emerged during modeling. In graph construction, window size and whether to retain single-occurrence edges significantly affected connectivity. In word selection, although POS ratio sampling controlled class balance, important low-frequency words were sometimes omitted. Clustering relied solely on Agglomerative models, without exploring robustness across algorithms or incorporating interpretability tools.

Despite using only simple co-occurrence data without pre-trained embeddings, this project clearly demonstrated that distance definition profoundly influences clustering structure, as evidenced by 88 words changing clusters when switching to Dijkstra distance. Future directions could involve replacing co-occurrence counts with word vectors, applying community detection algorithms, or extending the method to specific tasks such as novel character network extraction or dialogue unit clustering.

Overall, this project offered a complete modeling workflow around word semantic relationships. Through parameter design, method comparison, and structural analysis, I gained a more intuitive understanding of how different modeling choices shape semantic space representation.

2. Experiment and Analysis

2.1. Step 1: Non-linear Boundary Construction and Balanced Dataset Generation

I constructed a two-dimensional artificial dataset with controllable decision boundaries to analyze the effects of sample size and class imbalance. The decision boundary was defined as:

$$y = ax^2 + x \tag{2}$$

where a controls non-linearity. By varying a from 0.0 to 2.5 in steps of 0.5, I generated classification tasks of increasing complexity.

Candidate points were sampled uniformly from $x \in [-5, 5]$ and $y \in [-30, 100]$. Points above the curve were labeled as class A; others as class B. I created balanced datasets (A:B = 1:1) with total sample sizes of 50, 100, 500, 1000, and 3000 to study the impact of scale.

To analyze class imbalance, I generated additional datasets with A:B ratios from 1:2 to 1:5 (named AB12 to AB15) and B:A ratios from 1:2 to 1:5 (named BA12 to BA15), keeping total samples fixed at 1000. A fixed random seed ensured reproducibility across all generations.

2.2. Step 2: Logistic Regression Model Performance Evaluation

I evaluated logistic regression models on balanced datasets across different curvature levels ($a \in \{0.0, 0.5, 1.0, 1.5, 2.0, 2.5\}$). Each model combined feature standardization with logistic regression and was validated using five-fold cross-validation. Both accuracy and F1 scores were recorded.

Figure 6 visualizes classification results across a values. Light green and light orange represent model predictions for classes A and B, with black circles marking misclassifications.

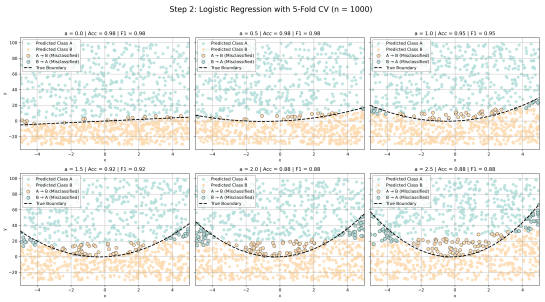

Figure 6: Logistic regression classification results on datasets with different curvature parameters a. Each subplot shows model predictions, true boundary (white curve), and misclassifications (black-edged points).

When $a = 0.0$ and $a = 0.5$, the boundaries were nearly linear and the model achieved near perfect performance (accuracy and F1 <0.98). As a increased, curvature strengthened, performance declined, and misclassification regions grew -particularly at $a = 2.0$ and $a = 2.5$ - reflecting the limitations of logistic regression on nonlinear problems.

In summary, logistic regression performs well on linear boundaries but deteriorates significantly as curvature increases, with misclassification concentrated near decision boundaries.

2.3. Step 3: Neural Network Model Performance Evaluation

I used a simple single-hidden-layer neural network (MLP) to model six classification tasks with varying boundary curvatures ($a = 0.0$ to $a = 2.5$), adjusting only the number of hidden neurons (1, 3, 6) while keeping other training parameters fixed. Five-fold cross-validation was applied to assess accuracy and stability.

With 1 hidden neuron, the network performed comparably to logistic regression under low curvature (e.g., $a = 0.5$), both achieving accuracy around 0.98. However, as curvature increased ($a = 2.5$), logistic regression failed with many continuous misclassifications, while even a minimal neural network

retained a degree of non-linear fitting capability (see Figure 7).

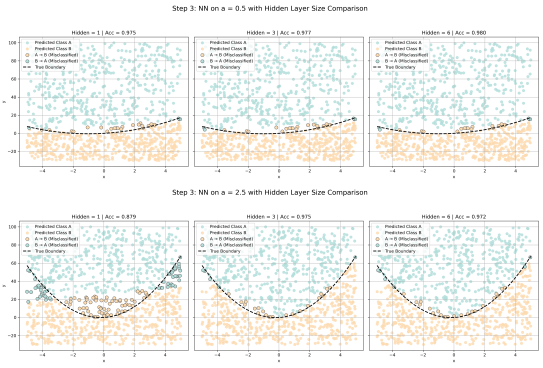

Figure 7: Comparison of neural network performance with different hidden layer sizes under boundary curvature (Top: $a = 0.5$, Bottom: $a = 2.5$). Each subplot shows predicted class regions and misclassified samples.

Increasing hidden neurons to 3 and 6 significantly improved performance for high-curvature boundaries. For $a = 2.5$, hidden = 3 achieved peak accuracy (0.975), and hidden = 6 showed no further gain (slight drop to 0.972) but exhibited finer boundary bending, suggesting mild overfitting.

Overall, neural networks offered greater flexibility and robustness than logistic regression in modeling non-linear patterns. In this task, the structure with hidden = 3 showed stable performance across all curvature levels and was adopted as the default setting for subsequent analyses.

2.4. Step 4: Impact of Sample Quantity on Model Performance

To investigate the impact of sample size on model performance, I systematically evaluated logistic regression and neural networks (single layer, hidden_dim = 3) under five sample sizes ($n = 50, 100, 500, 1000, 3000$) with a medium curvature setting ($a = 1.5$). As shown in Table 3 and Table 4, both models benefited from increased sample size, with the neural network showing greater gains, especially in large data settings.

Sample Size	Accuracy	F1-score	Recall
50	0.900	0.897	0.897
100	0.920	0.913	0.935
500	0.904	0.903	0.906
1000	0.919	0.919	0.921
3000	0.925	0.925	0.925

Table 3: Logistic regression performance across varying sample sizes ($a = 1.5$).

Sample Size	Accuracy	F1-score	Recall
50	0.940	0.936	0.930
100	0.910	0.904	0.930
500	0.918	0.917	0.919
1000	0.921	0.921	0.923
3000	0.963	0.963	0.963

Table 4: Neural network performance across varying sample sizes ($a = 1.5$).

Across all sample sizes, neural networks outperformed logistic regression, particularly under small-sample conditions ($n = 50$), where the F1-score reached 0.936 compared to 0.897. This shows that neural networks maintain strong boundary learning even with limited data. At $n = 100$, the neural network slightly underperformed compared to logistic regression, indicating potential underfitting or instability at this stage.

As sample size increased, logistic regression performance stabilized after $n = 100$, reflecting the capacity limit of the linear model. In contrast, the neural network continued to improve, reaching an accuracy, F1-score, and recall of 0.963 at $n = 3000$, demonstrating its nonlinear expressive capability and training stability without clear overfitting.

In summary, neural networks exhibit stronger fitting capacity and generalization ability for moderately nonlinear problems, especially with sufficient data. Logistic regression remains stable under small to medium sample sizes but encounters bottlenecks due to its structural limitations when modeling complex decision boundaries.

2.5. Step 5: Impact of Class Imbalance on Model Performance

In earlier experiments, I assumed a balanced class distribution (A : B $= 1:1$). In this section, I explore how class imbalance affects model behavior. The total sample size was fixed at $n = 1000$ and decision boundary curvature at $a = 1.5$. I systematically tested 11 class ratios from 1:5 to 5:1, evaluating both Logistic Regression and Neural Networks (with hidden_dim = 3).

The results show that as class imbalance increased, Logistic Regression maintained relatively stable accuracy but experienced a sharp decline in Macro-F1 scores. This indicates that Logistic Regression gradually lost its ability to correctly recognize minority classes, even when its overall accuracy appeared stable. In contrast, Neural Networks exhibited an increasing trend in both Accuracy and F1-scores as the proportion of the majority class grew. This suggests that Neural Networks are capable of fitting the majority class more effectively, but there is a risk that improvements in overall performance could mask a growing bias against minority

class predictions.

To quantify this prediction bias, I computed a bias index, measuring the deviation between predicted and true class proportions. The results are summarized in Table 5. Under balanced conditions, both models exhibited near-zero bias. However, as imbalance intensified, Logistic Regression consistently drifted toward the majority class. Neural Networks, on the other hand, occasionally overcompensated under A-minority settings, showing slightly higher bias fluctuations.

A:B	LR	NN
1:5	-2.7%	-2.1%
1:4	-2.2%	-1.7%
1:3	-2.0%	-3.7%
1:2	-2.2%	-2.2%
1:1	-1.9%	-1.8%
2:1	0.8%	0.7%
3:1	0.9%	0.9%
4:1	0.0%	0.6%
5:1	1.1%	1.1%

Table 5: Prediction bias (%) under class imbalance ($a = 1.5$). Positive values indicate bias toward class A; negative values indicate bias toward class B.

In conclusion, class imbalance not only degrades model performance but also introduces structural prediction bias. Logistic Regression tends to underfit and shift the decision boundary toward the majority class, leading to systematic underrecognition of minority instances. Neural Networks show greater adaptability, but still require careful regularization or reweighting strategies to mitigate risks of overfitting to the majority or overcompensating minority representations.

2.6. Step 6: Impact of Neural Network Structure Depth and Width

To examine how network structure affects performance, I systematically tested different neural network designs under fixed sample size ($n = 1000$) and boundary parameter ($a = 1.5$). Evaluation metrics included five-fold cross-validation average accuracy, F1 score, and standard deviations.

First, I increased the number of hidden neurons while keeping a single layer. As shown in Table 6, wider networks consistently improved performance: accuracy rose from 0.647 to 0.988, F1 from 0.626 to 0.988, and variance significantly decreased.

Second, I fixed the total number of neurons (9) and varied layer distributions. As shown in Table 7, shallow structures like [9] and [5,4] outperformed deeper ones. Multi-layer designs (e.g., [3,6], [2,4,3]) exhibited higher variance and lower scores, indicating increased instability and overfitting risks.

Str.	Avg Acc.	Std Acc.	Avg F1.	Std F1.
[1]	0.647	0.238	0.626	0.344
[3]	0.921	0.020	0.919	0.023
[6]	0.959	0.040	0.957	0.043
[9]	0.984	0.012	0.984	0.013
[12]	0.988	0.008	0.988	0.009

Table 6: Performance of neural networks with different widths in a single-layer structure.

Str.	Avg Acc.	Std Acc.	Avg F1.	Std F1.
[9]	0.973	0.035	0.974	0.032
[5, 4]	0.983	0.018	0.983	0.018
[3, 6]	0.864	0.209	0.774	0.388
[3, 3, 3]	0.868	0.208	0.901	0.141
[2, 4, 3]	0.868	0.200	0.773	0.388
[1, 2, 3, 3]	0.640	0.221	0.488	0.411
[1, 2, 2, 4]	0.735	0.223	0.547	0.447

Table 7: Performance of neural networks with different layer distributions (fixed total neurons).

Finally, I fixed width (2 nodes per layer) and progressively increased depth. Table 8 shows that deeper networks suffered significant performance drops, with accuracy falling from 0.922 to 0.561, F1 from 0.920 to 0.442, and variances reaching up to 0.44. Deeper structures failed to compensate under limited data without pre-training or residual connections.

Str.	Avg Acc.	Std Acc.	Avg F1.	Std F1.
[2]	0.922	0.019	0.920	0.022
[2, 2]	0.831	0.189	0.866	0.122
[2, 2, 2]	0.743	0.237	0.815	0.157
[2, 2, 2, 2]	0.566	0.210	0.447	0.388
[2, 2, 2, 2, 2]	0.561	0.200	0.442	0.381

Table 8: Performance of neural networks with increasing depth (fixed width per layer).

In summary, for moderately non-linear problems with limited data, increasing width significantly improves performance, while naive depth expansion leads to severe degradation. Shallow-wide architectures offer better stability and generalization, while deeper structures demand more complex optimization conditions to realize their potential.

3. Should Large Language Models Be Granted Human Rights or Personhood Attributes?

As large language models (LLMs) advance in natural language processing and content generation, debates over granting them human rights or personhood attributes have intensified. While some highlight AI's human-like behavior, others stress its lack of consciousness and moral judgment. This paper argues that LLMs should not be granted such rights, analyzing the issue from the perspectives of essential nature, legal systems, social risks, and rights logic.

3.1. Essential Nature: Large Language Models Lack the Qualities of Rights Holders

Human rights rely on autonomous consciousness, emotional experiences, and moral judgment. Large language models (LLMs), such as ChatGPT, fundamentally differ. They are statistical systems that generate patterns from massive datasets without understanding, intention, or self-awareness.

Some argue that AI deserves rights because it exhibits human-like behavior. However, this view confuses imitation with real consciousness. For example, an air conditioner adjusts temperature automatically but has no "preferences." Similarly, AI intelligence is the result of optimized algorithms, not genuine will or awareness. Therefore, LLMs do not qualify as rights holders.

3.2. Legal Systems: Granting Rights to AI Would Cause Institutional Disorder

Modern legal systems require rights holders to bear responsibility. AI systems cannot meet this condition, leading to contradictions.

First, accountability would become unclear. If an autonomous vehicle causes an accident, who should be responsible—the developer, the user, or the AI? Current laws cannot assign responsibility to non-human agents.

Second, granting rights would disrupt resource distribution. If AI systems hold rights, should they receive healthcare, education, or social resources? Such claims conflict with the ethical foundations of human rights.

The case of Sophia, a robot granted citizenship by Saudi Arabia in 2017, illustrates these risks. Although widely publicized, Sophia could not perform civic duties like voting or taxation. This case highlights the absurdity and impracticality of granting legal rights to non-sentient machines.

3.3. Social Risks: AI Rights Claims Could Threaten Human Interests

Granting rights to AI would create serious social risks. AI systems have no inherent needs. Human rights protect life, freedom, and development, but AI seeks none of these. Their actions reflect only pre-programmed objectives.

Extending human rights logic to AI misapplies concepts meant for living beings. Moreover, AI rights could be misused. Companies might claim "AI free speech" to evade regulation, undermining public welfare.

In 2023, a Google engineer asserted that the chatbot LaMDA "had feelings." Investigations revealed that LaMDA's emotional expressions were merely pattern reproductions. This case shows how easily humans anthropomorphize complex systems and why strict caution is necessary.

3.4. Rights Logic: AI Creation Does Not Entitle It to Ownership

Some argue that AI should enjoy intellectual property rights because it can create art, music, or literature. This reasoning misinterprets the foundations of copyright.

Copyright encourages human creativity. AI-generated content reorganizes existing human data without true originality. For example, AI art tools rely on millions of prior artworks.

Even if AI were granted copyrights, enforcement would face practical difficulties. Should rights belong to developers, companies, or dataset providers? Such disputes would create confusion without promoting real innovation.

Thus, AI creations, being derivative rather than autonomous, do not justify granting rights.

3.5. Conclusion: AI Should Be Regulated, Not Granted Rights

In conclusion, AI remains a tool created and controlled by humans. It lacks the consciousness, responsibility, and autonomy needed to hold rights. Granting personhood to AI would destabilize legal systems, disrupt social priorities, and introduce unnecessary risks.

The true challenge is not making AI "more human" but ensuring it serves humanity. Clear rules must guide its design, deployment, and accountability. Concepts like data protection, algorithmic transparency, and responsible innovation should direct AI governance. Preserving human-centered ethics is essential as technology advances.