AI Assessment

Contents

1 Word Clustering in Literary Texts: A Co-occurrence and Graph-based Approach

1.1 Introduction

This project investigates how unsupervised learning can be used to group words based on their meanings by analyzing how often they appear together in a set of literary texts. We use both direct co-occurrence patterns and graph-based refinements (via Dijkstra's algorithm) to understand how different ways of measuring word distance affect the structure and clarity of the resulting groups.

1.2 Data Collection and Preprocessing

1.2.1 Text Selection

Five novels from Project Gutenberg were chosen: Siddhartha, This Side of Paradise, Ulysses, The Great Gatsby, and The Mystery of the Blue Train. These texts were selected for their diverse language and modern writing style to ensure a rich variety of vocabulary.

1.2.2 Data Cleaning and Preprocessing

The raw text was cleaned using a custom pipeline:

- **PDF Extraction:** Extracted using `PyPDF2`.
- **Cleaning:** Used regular expressions to remove Gutenberg Project headers / footers, URLs, digits, and non-ASCII characters, as described by Goodfellow et al. [2016].
- **Normalization:** Converted all text to lowercase and expanded contractions using the`contractions` library.
- **Tokenization and Lemmatization:** Split text into sentences and words with NLTK, then lemmatized using WordNetLemmatizer with POS tagging.
- **Stopword Removal:** Removed common and custom stopwords.

`This thorough cleaning ensures that the analysis focuses on meaningful content words, which is critical for the quality of subsequent clustering.`

1.3 Selection of Target Words

From the cleaned and lemmatized corpus, the 100 most frequent content words were selected, balanced between nouns, verbs, adjectives and adverbs using POS tagging and frequency filtering. This list, denoted as L, excludes function words and focuses on semantically rich vocabulary.

Sample Output: 'amory', 'man', 'time', 'siddhartha', 'thing', 'day', ... 'ask'
This careful selection ensures that the clustering reflects semantic relationships rather than syntactic or functional similarities.

1.4 Constructing the Co-occurrence Matrix

1.4.1 Definition and Construction

Sentences were segmented using `nltk.sent_tokenize()`, and each sentence was treated as a context window for co-occurrence counting. The co-occurrence matrix CC was constructed such that C_{ij} records the number of sentences in which both w_i and w_j appear. Diagonal entries were set to zero to prevent self-similarity from biasing the clustering.

1.4.2 Distance Transformation

To convert co-occurrence counts into a distance metric suitable for clustering, the following formula was used:

$$D_{ij} = \frac{1}{C_{ij} + 1}$$

This ensures that pairs of words that co-occur frequently have small distances, while infrequent or non-co-occurring pairs have large distances. Momtazi et al. [2010] Adding 1 prevents division by zero.
Mathematical Rationale:
This transformation is monotonic and ensures all distances are finite and positive, which is required for most clustering algorithms.

1.4.3 Visualization and Interpretation

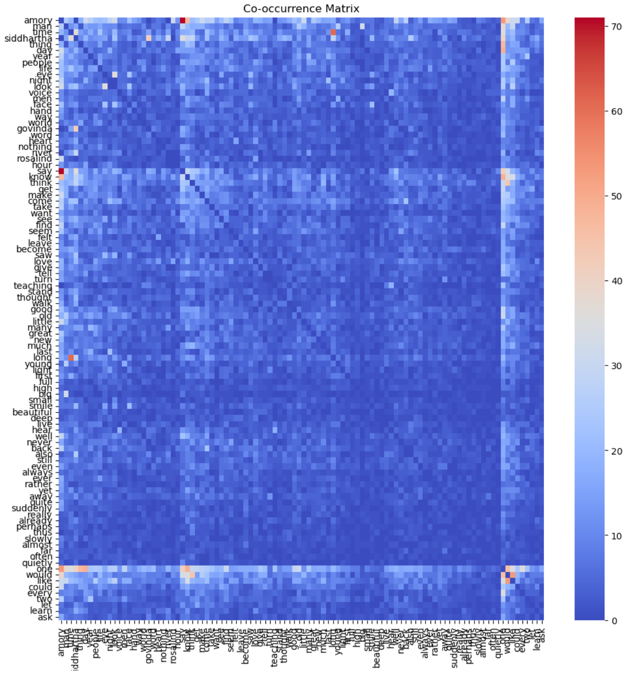

Figure 1: Word Co-occurrence Matrix Heatmap

The heatmap reveals that most word pairs have low co-occurrence (darker blue), while a few pairs, likely related in meaning or context, have higher co-occurrence (lighter or red spots).
Observation: Clusters of higher co-occurrences may indicate topical or semantic groupings.

1.5 Unsupervised Clustering and Results

1.5.1 Clustering Algorithms

Three unsupervised clustering algorithms were applied:

- **DBSCAN**: Density-based clustering, robust to noise and outliers. `eps=0.5`, `min_samples=5` (empirically chosen for best separation).
- **Agglomerative (Hierarchical)**: Groups words based on average linkage. `n_clusters=5`, `linkage='average'`.
- **KMeans**: Partitioning method, applied after embedding words into 2D space using Multi-Dimensional Scaling (MDS). `n_clusters=5`, applied after MDS projection as discussed byTanaka-Ishii and Iwasaki [1997]

1.5.2 Silhouette Scores and Quantitative Evaluation

```
Silhouette Scores:
```

Clustering Algorithm	Before Dijkstra	After Dijkstra
KMeans	-0.005	0.038
Agglomerative Clustering	0.136	0.460

Table 1: Silhouette Scores Before and After Dijkstra Correction

Interpretation: Higher silhouette scores after Dijkstra correction indicate improved cluster cohesion and separation, especially for Agglomerative clustering.

1.5.3 Results and Visualizations

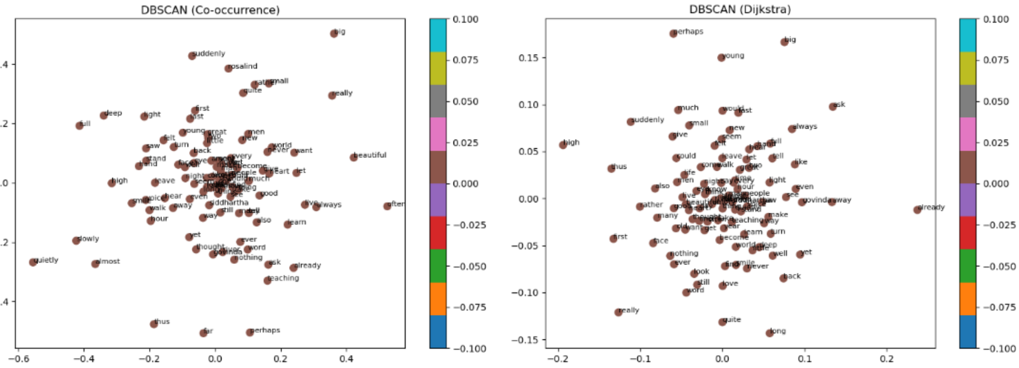

Figure 2: Clustering of Word Embeddings (KMeans)

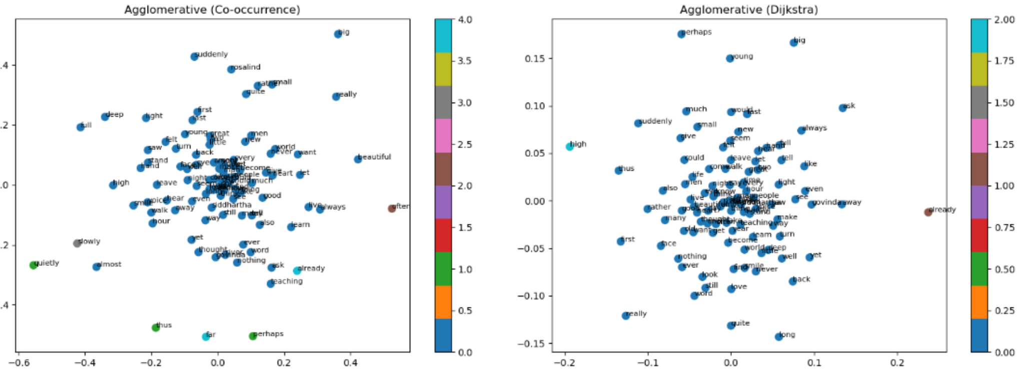

Figure 3: Clustering of Word Embeddings (DBSCAN)

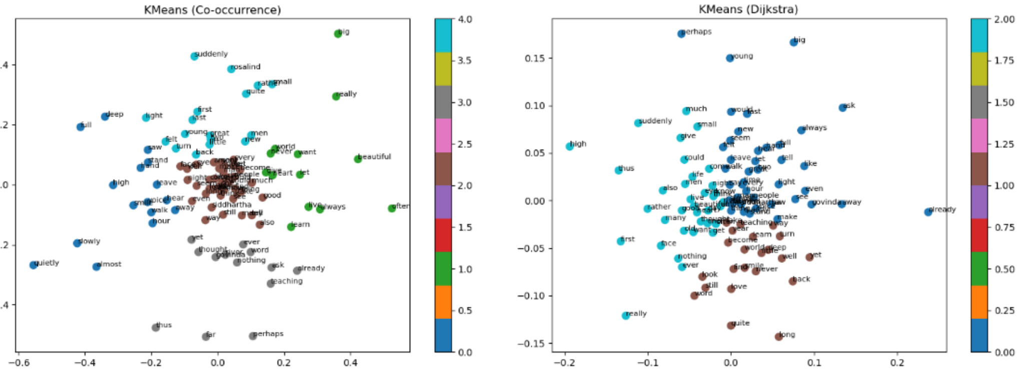

Figure 4: Clustering of Word Embeddings (Agglomerative)

Clustering Algorithm	Performance Before Dijkstra Correction	Visual Evidence Before Dijkstra	Performance After Dijkstra Correction	Visual Evidence After Dijkstra
Agglomerative Clustering	Several visible groupings with thematic cohesion (e.g., "father", "day", "man").	Formed visibly distinct groups, reflecting semantic coherence.	Significant improvement: Indirectly related words (e.g., “river” and “boat” via “water”) were now clustered together.	Tight, non-overlapping groupings, showing improved separation and coherence.
KMeans	Produced evenly-sized, artificially shaped clusters.	Loosely grouped points with overlapping boundaries, indicating weak semantic separation.	Redistributed words across clusters, with some closer words clustered more appropriately.	Redistribution of cluster sizes but still with low silhouette score, suggesting forced partitioning.
DBSCAN	Formed one large cluster capturing most words, resulting in semantic dilution.	A dense, singular cluster with minimal differentiation, unsuitable for silhouette evaluation.	Continued forming a large single cluster with minimal improvement in semantic differentiation.	Consistently produced a single large cluster, showing no improvement post-correction.

Table 2: Clustering Algorithm Performance and Visual Evidence Before and After Dijkstra Correction

1.6 Graph-based Correction via Dijkstra’s Algorithm

1.6.1 Motivation and Method

Direct co-occurrence distances may not capture indirect semantic relationships. For example, if "river" and "boat" are both related to "water" but rarely co-occur directly, their semantic proximity is underestimated. A weighted undirected graph was constructed, where nodes are words and edges exist between words with positive co-occurrence, weighted by their distance. Dijkstra’s algorithm was used to compute the shortest path distances between all pairs of words. Dijkstra [1959], Tanaka-Ishii and Iwasaki [1997]

1.6.2 Results and Interpretation

- **Cluster Membership:** After Dijkstra correction, words that are indirectly related through intermediary words are more likely to be clustered together.
- **Visualization:** Both scatter plots and bar charts show that Dijkstra correction leads to more semantically coherent clusters and changes in cluster sizes.

1.6.3 Mathematical Reasoning

The use of shortest-path distances aligns with the intuition that semantic similarity can be transitive: if "A" is close to "B" and "B" to "C", then "A" and "C" should also be considered related. This graph-based correction captures indirect relationships that direct co-occurrence cannot.
Let $d(s, v)$ denote the minimum distance from source node s to node v. We initialize:

$$d(s) = 0, \quad d(v) = \infty \quad \forall v \in V \setminus \{s\}$$

At each step, select the unvisited node u with the smallest tentative distance $d(u)$, and for each neighbor v of u:

$$d(v) \leftarrow \min\left(d(v),\ d(u) + w(u, v)\right)$$

1.7 Conclusion

This project demonstrates the value of combining co-occurrence statistics, graph algorithms, and unsupervised learning to cluster words in literary texts. The methodology is robust and reveals nuanced semantic structures in language data. Dijkstra correction, in particular, enhances the quality and interpretability of clusters by capturing indirect semantic relationships.

2 Report on Investigating Logistic Regression and Neural Networks on a Nonlinear Dataset

2.1 Introduction

This report investigates how well Logistic Regression and Neural Networks perform on a two-dimensional dataset with a nonlinear decision boundary. The boundary is defined by the equation:

$$y = ax^2 + x$$

where x is a randomly sampled value from the range $[-2, 2]$, and the corresponding y value is computed based on the equation. Points on one side of the boundary are assigned to Class A, while points on the other side are assigned to Class B. The primary objective of this experiment is to evaluate the performance of Logistic Regression and Neural Networks on this dataset as we vary the following factors:

- The boundary curvature parameter a.
- The number of data points in the dataset.
- The balance between Class A and Class B.
- The size of the Neural Network architecture.

We assess model performance using accuracy as the main evaluation metric, while also looking at confusion matrices and ROC curves for additional insights Bishop [2006].

2.2 Data Generation

The dataset is generated based on the quadratic boundary $y = ax^2 + x$, where a controls the curvature of the boundary. The value of x is randomly selected from the interval $[-2, 2]$. A small random noise is added to y to introduce variability and make the boundary slightly irregular. This dataset is then used to train and test the models, with the points being classified into Class A or Class B based on their position relative to the boundary.

Additionally, we introduce class imbalance by varying the ratio of Class A and Class B samples. The class balance can significantly influence model performance, especially for Logistic Regression, which tends to be sensitive to class imbalances Goodfellow et al. [2016]. The dataset also varies in size, with experiments conducted using 100, 300, and 500 samples.

2.3 Experimental Setup

For the experiment, we define the following parameters:

- **Boundary Curvature (a):** We test three values for a: 0.5, 1.0, and 2.0. These values represent different degrees of boundary nonlinearity.
- **Number of Samples:** Datasets with 100, 300, and 500 samples are created to study how the size of the dataset affects performance.
- **Class Balance:** We experiment with class ratios of 0.3, 0.5, and 0.7, where each ratio represents the proportion of data points assigned to Class A and the rest to Class B.
- **Neural Network Size:** We test several Neural Network architectures, including simple models like (5,) and (10,) as well as more complex architectures such as (10, 10) and (10, 15, 15).

For each combination of these parameters, Logistic Regression and Neural Networks (implemented using scikit-learn) are trained and tested. Model performance is primarily measured by accuracy, but additional metrics such as confusion matrices and ROC curves are also examined James et al. [2013].

2.4 Mathematical Explanation

Logistic Regression is a linear classifier that assumes the decision boundary between classes is linear. The model attempts to find a hyperplane (or line in 2D) that best separates the classes based on the features. The equation of the decision boundary for Logistic Regression is:

$$\sigma(w_1x_1 + w_2x_2 + b) = 0$$

where σ is the sigmoid function and w_1, w_2, b are the learned parameters. Logistic Regression struggles with complex, nonlinear boundaries such as the quadratic $y = ax^2 + x$, as the model cannot capture the curvature.

Neural Networks, on the other hand, consist of layers of interconnected nodes (neurons), where each neuron applies a non-linear activation function (like ReLU or Sigmoid). This non-linearity allows the model to approximate highly complex decision boundaries:

$$y = \sigma(Wx + b)$$

where σ is the activation function, W is the weight matrix, and b is the bias. Neural Networks can effectively learn and represent nonlinear decision boundaries, making them well-suited for this problem Goodfellow et al. [2016].

2.5 Comparison of Logistic Regression and Neural Network Performance

As expected, Logistic Regression performs well when the decision boundary is nearly linear (small a) but struggles as the boundary becomes more nonlinear. Neural Networks, on the other hand, consistently outperform Logistic Regression across all settings, particularly when the boundary becomes more complex. However, Neural Networks do not show significant improvement beyond a certain level of complexity, suggesting that a relatively simple network is sufficient for this task Bishop [2006], James et al. [2013].

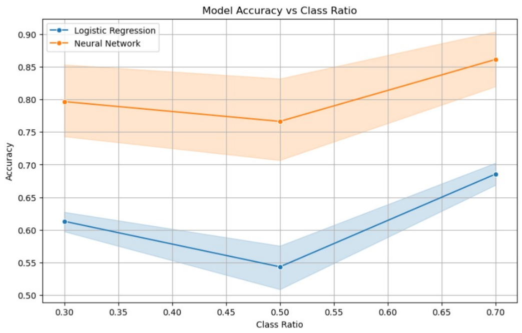

Figure 5: Model accuracy for Logistic Regression vs Neural Network along the different class ratio

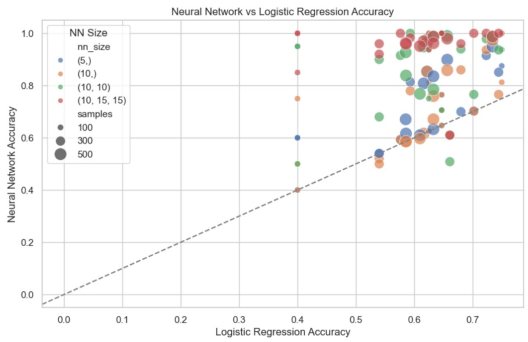

Figure 6: Trend of model accuracy with different class ratio

2.5.1 Confusion Matrices and ROC Curves

Confusion matrices and ROC curves were generated for both models across different configurations. The confusion matrices help identify false positives and false negatives, while the ROC curves provide a more granular view of performance by illustrating the trade-off between true positive rate and false positive rate.

In terms of ROC curves, the Neural Network shows a higher Area Under the Curve (AUC), indicating better discriminatory power compared to Logistic Regression, especially as the decision boundary becomes more complex James et al. [2013].

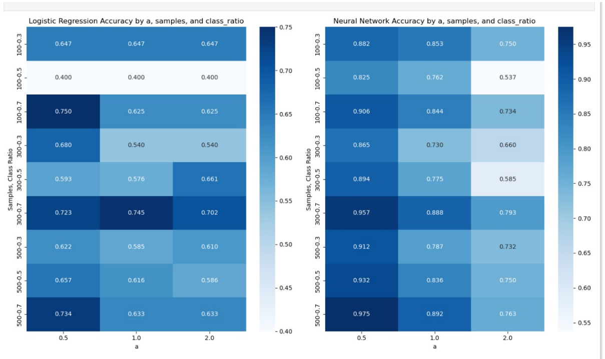

Figure 7: Confusion matrix of both models with varying a, class ratios and samples

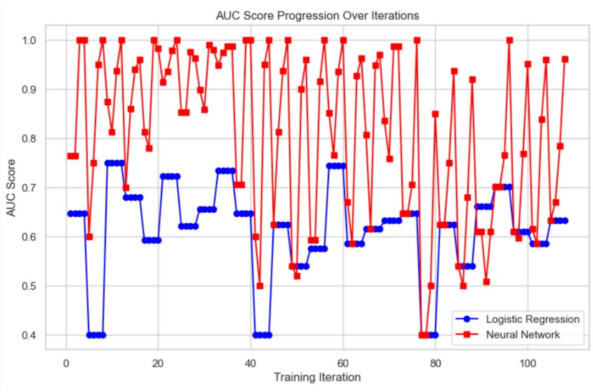

Figure 8: AUC score progression over iterations

2.6 Results

Factor	**Logistic Regression**	**Neural Network**
Effect of Nonlinearity (parameter a) (a)	Performs reasonably when a is small (e.g., 0.5), but accuracy drops significantly as a increases due to inability to capture complex relationships.	Effectively learns nonlinear boundaries even with simple architectures (e.g., (5,)); deeper networks (e.g., (10,10), (10,15,15)) improve performance further. Bishop [2006]
Network Architecture	Not applicable.	More complex architectures show better performance, but gains diminish after a certain size, indicating saturation in learning capacity. Goodfellow et al. [2016]
Sample Size	Performs poorly with small datasets (e.g., 100 samples), especially for complex boundaries; generalization improves with more samples (e.g., 300, 500).	Similar trend; performs better with increased sample size due to better generalization. James et al. [2013]
Class Imbalance	Sensitive to imbalance (e.g., class ratio = 0.3), biased toward majority class.	More robust, particularly with deeper networks, but still affected under extreme imbalance. Goodfellow et al. [2016]

Table 3: Performance Comparison of Logistic Regression and Neural Networks under Varying Conditions

2.7 Conclusion

This experiment demonstrates the limitations of Logistic Regression when confronted with nonlinear decision boundaries, such as the quadratic boundary defined by $y = ax^2 + x$. As the curvature parameter a increases, the model's performance rapidly degrades. In contrast, Neural Networks are able to capture the complexity of the decision boundary, showing superior performance across different values of a.

Both models perform better with larger datasets, and Neural Networks are more resilient to class imbalance. However, adding more complexity to the network architecture beyond a certain point does not significantly improve performance for this problem, suggesting that simpler architectures may be sufficient.

Ultimately, for this nonlinear classification task, Neural Networks outperform Logistic Regression due to their ability to model complex, nonlinear decision boundaries. However, simple neural network architectures are sufficient, and performance tends to plateau as network complexity increases.

2.8 Further Work

- **Hyperparameter Tuning:** Future experiments should focus on hyperparameter tuning for both Logistic Regression and Neural Networks, particularly for regularization parameters and network training settings like learning rates and epochs.
- **Data Augmentation:** Investigating the effects of data augmentation techniques could improve model performance, especially in cases of class imbalance.
- **Ensemble Methods:** Combining the results of multiple models through ensemble methods like Random Forests or Gradient Boosting Machines may yield better results on nonlinear datasets.

3 Should Large Language Models Be Granted Rights or Person-like Characteristics?

The issue of whether large language models (LLMs), like ChatGPT, should be granted rights or be treated as having personhood is both a thought-provoking philosophical question and a practical concern. As artificial intelligence (AI) continues to evolve, LLMs showcase impressive abilities to generate human-like language, sparking discussions about whether they deserve ethical or legal recognition similar to humans. This topic has even been addressed in legal contexts, such as when the European Parliament considered AI rights in its 2017 report on the civil law framework for robotics [Parliament, 2017]. Despite the ability of these models to imitate human thinking and dialogue, they lack essential traits like consciousness, emotions, and self-awareness qualities that are necessary for personhood and moral responsibility. As a result, LLMs do not have rights and are not likely to be granted any in the near future.

A key argument against giving rights to large language models (LLMs) is their lack of sentience. Legal and philosophical theories, like Immanuel Kant's view on moral agency, suggest that rights are given to those capable of rational thinking, self-governance, and moral reasoning. Modern legal systems also link personhood to the ability to take on responsibilities and engage in moral decision-making. Sentience, or the capacity for subjective experiences and inner motivations, is considered essential for being morally significant. Since LLMs don't have self-awareness or the ability to feel emotions or pain, they don't meet this vital requirement for having rights. Philosophers like John Searle [Searle, 1980] argue that AI, regardless of how sophisticated it is, doesn't truly understand language but simply manipulates symbols, as explained by his Chinese Room argument.

Rights are typically given to beings with subjective experiences, like humans and some animals, and since LLMs don't have these qualities, they don't qualify. Unlike humans, who feel emotions and have intrinsic desires, LLMs only generate responses based on statistical predictions, without real understanding or intent [Tegmark, 2017]. Even the most advanced AI systems, like GPT-4, merely process and rearrange human knowledge without creating new insights or experiencing emotions.

Granting personhood to large language models (LLMs) could lead to serious ethical and legal issues. If an AI were treated as a legal person, the question arises: who would be held accountable for its actions? Currently, AI models are owned and operated by companies, meaning they function more as tools rather than independent beings. For example, OpenAI's GPT-4 is governed by strict usage rules, and any misuse of it is the responsibility of the human users or developers. A past incident, such as when Microsoft's AI chatbot Tay started producing offensive content in 2016, showed that the developers were ultimately held accountable [Bostrom and Yudkowsky, 2014].

Some experts suggest that as AI evolves, it could begin showing behaviours that challenge our traditional understanding of thinking and agency. The idea of "digital sentience" posits that future AI systems might reach a point where their inner workings start to resemble human-like consciousness. However, critics argue that just because an AI shows advanced behaviours doesn't mean it should be considered a person.

To conclude, while large language models (LLMs) show remarkable language abilities, they do not have the fundamental characteristics needed for personhood. They lack the capacity for subjective experience, moral responsibility, and independent decision-making, so granting them rights would not be appropriate. Instead, ethical discussions should centre on how humans utilize AI, focusing on fairness, transparency, and accountability in its use.

References

Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

Nick Bostrom and Eliezer Yudkowsky. *The Ethics of Artificial Intelligence.* Cambridge University Press, 2014.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1: 269–271, 1959.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning.* Springer, 2013.

Saeed Momtazi, Franz Naumann, and Dietrich Klakow. A comparative study of word co-occurrence for term clustering in language model-based sentence retrieval. *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010.

European Parliament. Civil law rules on robotics. European union report, European Union, 2017.

John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–457, 1980.

Kumiko Tanaka-Ishii and Hiroshi Iwasaki. Clustering co-occurrence graph based on transitivity. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997.

Max Tegmark. *Life 3.0: Being Human in the Age of Artificial Intelligence.* Knopf, 2017.