Visual Sentence Complexity Prediction

Ursu Andrei
andrei.ursu@s.unibuc.ro

December 29, 2024

Abstract

This project proposes several supervised machine learning models with the aim of determining the visual complexity of a sentence. The dataset contains a column for text and a column for scores, with the scores ranging between -1 and 2. The best models for this regression task have proven to be K-NN and a neural network built from scratch.

1 Introduction

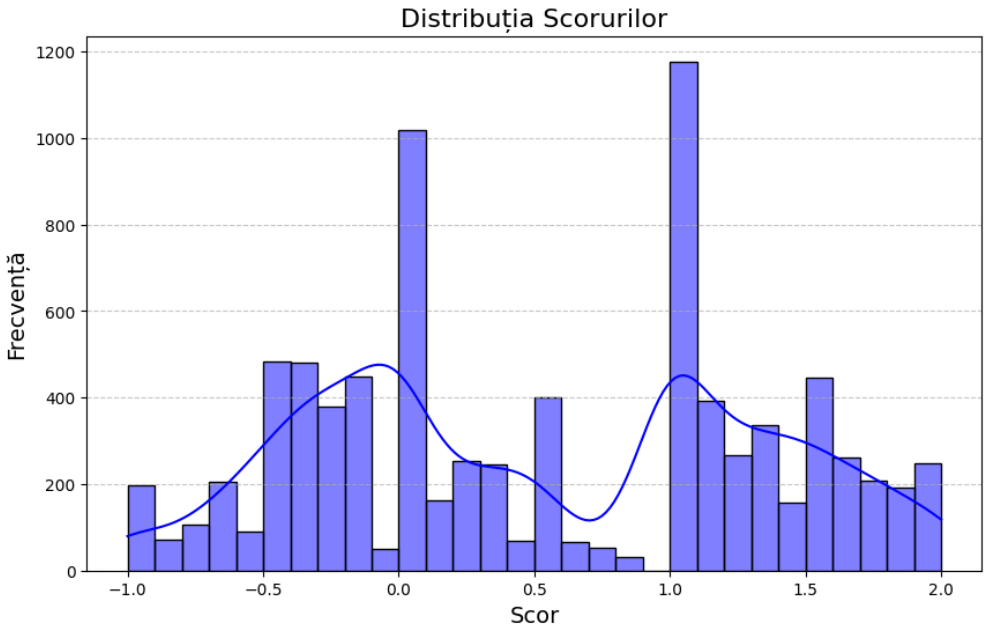
Visual complexity refers to the cognitive load required to process and understand textual content. This can include factors like sentence structure, word choice, syntax, and overall readability. High visual complexity in text might hinder comprehension, while overly simplified text might lack depth or precision, making this balance a critical aspect in fields like education, accessibility, and content optimization. Textual visual complexity can be influenced by several linguistic features such as: Lexical Features, Syntactic Features, Semantic Features and also Stylistic Features.

2 Methodology

In this section presents the preprocessing method that obtained the best results as well as a short description for each model that I used.

2.1 Dataset

The training dataset contains 8000 labeled samples with scores ranging from -1 to 2. The validation dataset provides extra 500 samples with the respective scores also ranging from -1 to 2. And finally, the test dataset contains 500 unlabeled samples.



This graph shows the distribution of scores within the dataset, plotted as a histogram with a kernel density estimation (KDE) curve overlaid. The scores range between -1 and 2, as expected, with distinct peaks at specific intervals. Also, a high frequency of scores is centered around 0 and 1, suggesting that the dataset has a significant concentration of samples in these ranges. The peaks imply that the dataset may have been designed or labeled with a bias toward these values, possibly reflecting common text patterns or linguistic features that correspond to "neutral" and "moderately complex" text, respectively. Scores near the extremes, particularly closer to -1 and 2, have relatively lower frequencies. These values likely represent outliers or rare cases of very simple or highly complex text. The KDE curve (smooth blue line) provides a good approximation of the underlying probability distribution, reinforcing the presence of multiple modes.

2.2 Data Preprocessing Pipeline

Our preprocessing methodology emphasizes feature extraction while minimizing noise in the visual complexity dataset. The pipeline consists of several key steps: text cleaning, linguistic processing, and numerical transformation.

The initial cleaning phase removes all non-alphabetic characters and spaces, establishing a standardized text format. We leverage spaCy's Natural Language Processing framework for tokenization and linguistic analysis, specifically targeting nouns, verbs, adjectives, and adverbs. This selective approach ensures the retention of semantically significant components. To enhance data quality, we eliminate common stopwords using NLTK's English stopword dictionary, effectively reducing linguistic redundancy.

Text normalization is achieved through lemmatization, converting words to their root forms while preserving semantic meaning. The final transformation utilizes TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, generating numerical feature vectors that capture word importance. Our TF-IDF implementation extracts 5000 features, incorporating both unigrams and bigrams to capture individual word significance and phrasal relationships.

2.3 Methodological Approaches

Our analysis employs various machine learning techniques, beginning with classical approaches to establish baseline performance.

Linear Regression

Description: A fundamental regression model that establishes linear relationships between dependent and independent variables, optimizing coefficients to minimize mean squared error.
Strengths: High interpretability, efficient computation, straightforward implementation.
Constraints: Limited to linear relationships, susceptible to outlier influence.

Random Forest Regressor

Description: An ensemble method utilizing multiple decision trees, each trained on random data subsets. Predictions are derived through aggregation of individual tree outputs.
Strengths: Exceptional generalization, inherent feature importance analysis, robust nonlinear modeling.
Constraints: Reduced interpretability, computational intensity scales with tree quantity.

Decision Tree Regressor

Description: Hierarchical model creating data partitions based on feature conditions, with predictions derived from branch-specific averages.
Strengths: High interpretability, natural handling of nonlinearity, scale-invariant operation.
Constraints: Tendency toward overfitting without proper parameter constraints.

Support Vector Regressor (SVR)

Description: Regression variant of SVM, optimizing for maximum acceptable error margin while maintaining function simplicity.
Strengths: Effective nonlinear modeling through kernel functions, robust generalization.
Constraints: Computational complexity, sensitivity to hyperparameter selection.

Gradient Boosting Regressor

Description: Sequential ensemble technique combining weak learners, typically decision trees, with each iteration focusing on previous error correction.
Strengths: Superior predictive performance, robust error handling, effective complexity modeling.
Constraints: Extended training periods, parameter optimization complexity.

XGBoost Regressor

Description: Advanced gradient boosting implementation incorporating regularization, tree pruning, and complexity control mechanisms.
Strengths: Exceptional computational efficiency, superior performance metrics, scalability to large datasets.
Constraints: Complex hyperparameter optimization requirements.

K-Nearest Neighbors Regressor (K-NN)

Description: Instance-based algorithm predicting values through local neighborhood analysis in the training set.
Strengths: Implementation simplicity, minimal training requirements, effective for moderate datasets.
Constraints: Performance degradation with high dimensionality, scale sensitivity.

Neural Network Architecture

We implemented a deep learning approach utilizing a fully connected neural network optimized for regression tasks. The model emphasizes prediction accuracy through careful architectural design and optimization strategies.

- **Network Structure:**
  - **Input Layer:** Dimensionality matched to `input_dim`
  - **Dense Layers:** Sequential arrangement of 700, 600, 500, and 400 neurons
  - **Activation Function:** ReLU implementation post-first layer
  - **Output Layer:** Single-neuron regression endpoint
- **Training Configuration:**
  - **Loss Function:** Mean Squared Error optimization
  - **Optimizer:** Adam implementation with 0.001 learning rate
- **Implementation Details:**
  - **Data Management:** Training-validation partitioning with PyTorch tensor conversion
  - **Training Protocol:** 10-epoch optimization with continuous loss monitoring
  - **Performance Metric:** Spearman correlation for ranking alignment assessment

Experimental Results

This section presents an analysis of our experimental outcomes, highlighting both successful and unsuccessful approaches. For Linear Regression, Random Forest Regressor, and Decision Tree Regressor models, we use **Spearman's Rank Correlation** as our primary **score** metric. The Gradient Boosting Regressor, XGBoost Regressor, and K-NN models were evaluated using the **6000 features** preprocessing pipeline.

Evaluation Metrics

We employed the following metrics to evaluate and compare model performance:

1. Mean Absolute Error (MAE)

Definition: MAE quantifies prediction accuracy by calculating the average absolute difference between predicted and actual values, treating all deviations equally.
Formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where y_i represents the actual value, $\hat{y}_i$ denotes the predicted value, and n is the sample size.
Characteristics:

- Maintains the original scale of measurement
- Provides direct interpretation in terms of the target variable

2. Mean Squared Error (MSE)

Definition: MSE evaluates prediction accuracy by computing the average squared deviation between predicted and actual values, giving higher weight to larger errors.
Formula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Characteristics:

- Particularly effective at identifying large prediction errors
- Provides heightened sensitivity to outliers

3. Spearman's Rank Correlation

Definition: This metric evaluates the strength and direction of monotonic relationships between predicted and actual values, operating on rank-transformed data.
Formula:

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

where d_i represents the rank difference between corresponding observations, and n denotes the sample size.
Interpretation Range: $-1 \leq r_s \leq 1$

- $r_s = 1$: Indicates perfect positive monotonic correlation
- $r_s = -1$: Indicates perfect negative monotonic correlation
- $r_s = 0$: Indicates absence of monotonic relationship

4. Kendall's Tau Correlation

Definition: This non-parametric measure assesses ranked correlation by analyzing the proportion of concordant versus discordant pairs in the dataset.
Formula:

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)}$$

where C represents concordant pairs, D represents discordant pairs, and n is the total number of observations.
Interpretation Range: $-1 \leq \tau \leq 1$

- $\tau = 1$: Signifies complete rank agreement
- $\tau = -1$: Signifies complete rank disagreement
- $\tau = 0$: Signifies no rank correlation

2.4 Linear Regression

Features	4000	5000	6000	7000	8000
Score	0.3241	0.3310	**0.3519**	0.3502	0.3392

Table 1: Linear Regression

2.5 Random Forest Regressor

Features	4000	5000	6000	7000	8000
Score	0.4132	0.4337	**0.4782**	0.4431	0.4415

Table 2: Random Forest Regressor

2.6 Decision Tree Regressor

Features	4000	5000	6000	7000	8000
Score	0.3619	**0.3688**	0.3592	0.3433	0.3106

Table 3: Decision Tree Regressor

2.7 Support Vector Regressor (SVR)

All the tables shown include the **6000 features** preprocessing. Since the **rbf kernel** obtained the best results, it was used in all the tables below.

C controls the balance between training error and model complexity (high C $\rightarrow$ less regularization) and **gamma** determines the complexity of the decision boundary by controlling the reach of influence for support vectors (high gamma $\rightarrow$ more complex boundaries)

1. Mean Absolute Error (MAE)

	gamma = 0.01	gamma = 0.05	gamma = 0.1	gamma = 0.5	gamma = 1
C= 0.01	0.9221	0.9174	0.9112	0.9122	0.9203
C= 0.05	0.9008	0.8944	0.8913	0.8897	0.8851
C= 0.1	0.8513	0.8503	0.8470	0.8427	0.8462
C= 0.5	0.7964	0.7925	**0.7901**	0.7914	0.7912
C= 1	0.8103	0.8128	0.8155	0.8148	0.8110

Table 4: MAE for SVR

2. Mean Squared Error (MSE)

	gamma = 0.01	gamma = 0.05	gamma = 0.1	gamma = 0.5	gamma = 1
C= 0.01	0.8441	0.8218	0.8077	0.8051	0.8072
C= 0.05	0.8004	0.7977	0.7829	0.7891	0.7997
C= 0.1	0.6182	0.6154	0.6011	0.6003	0.6067
C= 0.5	0.5733	0.5799	0.5618	**0.5616**	0.5681
C= 1	0.4824	0.4997	0.5021	0.4808	0.4813

Table 5: MSE for SVR

3. Spearman's Rank Correlation

	gamma = 0.01	gamma = 0.05	gamma = 0.1	gamma = 0.5	gamma = 1
C= 0.01	0.3514	0.3572	0.3632	0.3710	0.3641
C= 0.05	0.3788	0.3821	0.3883	0.4041	0.4121
C= 0.1	0.4443	0.4512	0.4517	0.4429	0.4466
C= 0.5	0.4903	0.5084	0.5351	**0.5782**	0.5412
C= 1	0.4824	0.4997	0.5021	0.4808	0.4813

Table 6: Spearman's Rank Correlation for SVR

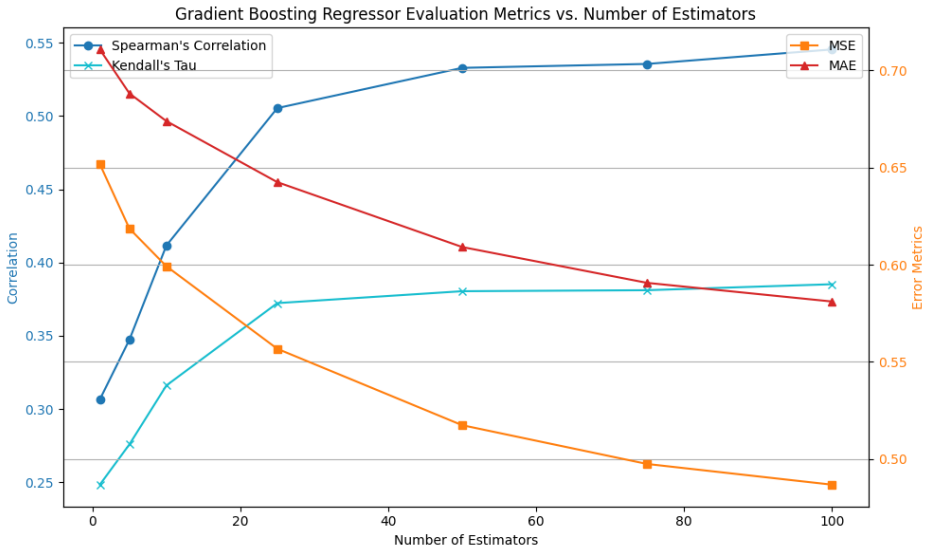
4. Kendall's Tau Correlation

	gamma = 0.01	gamma = 0.05	gamma = 0.1	gamma = 0.5	gamma = 1
C= 0.01	0.1447	0.1690	0.1711	0.1708	0.1689
C= 0.05	0.1633	0.1712	0.1764	0.1829	0.1764
C= 0.1	0.2015	0.2115	0.2271	0.2317	0.2285
C= 0.5	0.2341	0.2373	0.2419	**0.2679**	0.2622
C= 1	0.2176	0.2209	0.2175	0.2054	0.2091

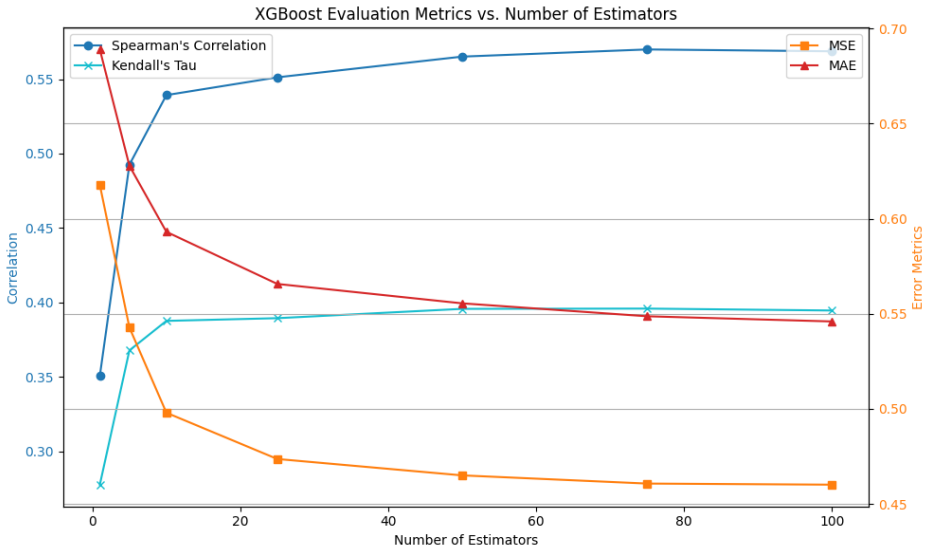
Table 7: Kendall's Tau Correlation for SVR

2.8 Gradient Boosting Regressor

The **number of estimators** specifies the total number of boosting rounds, or equivalently, the number of decision trees added sequentially to the ensemble.

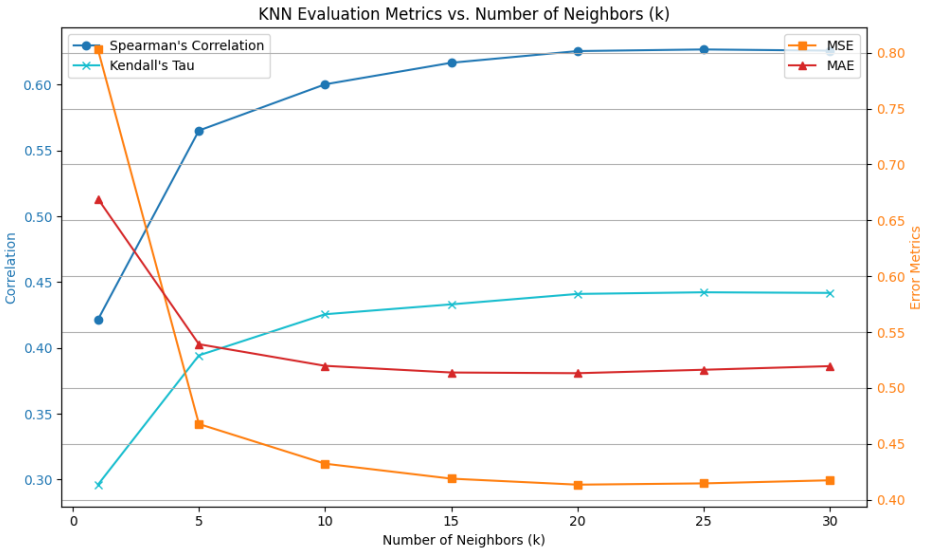


2.9 XGBoost Regressor



2.10 K-Nearest Neighbors Regressor (K-NN)

Here, I am using the **cosine** metric and the **distance** weights .Also, **K** represents the number of nearest neighbors to consider when making a prediction for a new data point.



2.11 Custom Neural Network

Learning Rate	0.001	0.002	0.003	0.004	0.005
Score	0.6218	**0.6391**	0.6205	0.6154	0.6049

Table 8: Custom Neural Network

3 Conclusion

In conclusion, the best preprocessing method turned out to be the simplest of all. This paper demonstrates that even models that might initially seem simplistic can achieve reasonable results in the context of a fairly complex task like visual sentence complexity prediction.

A Appendix: Analysis of Suboptimal Approaches

This section examines our experimental approaches that demonstrated limited effectiveness in practice.

A.1 Word2Vec Implementation Analysis

Our custom Word2Vec preprocessing implementation consisted of:

1. **Text Sanitization**: Implementation of character filtering and case normalization protocols
2. **Word Segmentation**: Deployment of `word_tokenize` for lexical unit extraction
3. **Embedding Training**: Development of Word2Vec model for vector space representation
4. **Sentence Representation**: Computation of averaged word vectors for sentence-level encoding

Identified Limitations:

1. **Contextual Degradation**: Vector averaging results in loss of sequential information
2. **Vocabulary Constraints**: Incomplete coverage of terms in the embedding space

3. **Phrase Processing**: Inadequate handling of multi-token expressions
4. **Resource Requirements**: Significant computational demands during model training

The method demonstrates efficiency in processing but falls short in preserving semantic relationships.

A.2 Basic TF-IDF Analysis

Our fundamental TF-IDF implementation incorporated the following preprocessing stages:

- **Punctuation Management**: Systematic removal of punctuation through regex patterns
- **Numerical Transformation**: Conversion of numerical values using `num2words` library
- **Stop Word Filtration**: Implementation of `nltk.corpus` for common word removal
- **Morphological Reduction**: Application of `WordNetLemmatizer` for base form extraction

Methodological Constraints:

- **Processing Intensity**: Significant computational overhead in numerical conversion and lemmatization
- **Semantic Fidelity**: Loss of meaning through aggressive punctuation removal
- **Lemmatization Accuracy**: Context-dependent errors in word form reduction
- **Language Restrictions**: Framework optimized solely for English text processing
- **Numerical Precision**: Potential ambiguities in number-to-text conversion processes

A.3 Advanced TF-IDF Implementation

- **Text Preprocessing**: Comprehensive cleaning protocol including case normalization and artifact elimination
- **Linguistic Analysis**: Integration of spaCy for advanced text processing
- **Feature Construction**: Combined TF-IDF vectorization with supplementary complexity metrics

Technical Limitations:

- **Computational Demands**: Resource-intensive processing pipeline
- **Framework Dependencies**: Heavy reliance on multiple external libraries
- **Vocabulary Coverage**: Insufficient stop word coverage across contexts
- **Complexity Assessment**: Incomplete capture of textual complexity dimensions
- **Adaptation Constraints**: Limited flexibility for domain-specific modifications

A.4 Graph Complexity Analysis

This implementation quantifies structural complexity through the following metrics:

- **Vertex Enumeration**: Quantification of graph size through node count
- **Hierarchical Analysis**: Maximum depth calculation for directed acyclic structures
- **Distance Metrics**: Maximum geodesic path length determination
- **Connection Density**: Average node degree measurement
- **Path Analysis**: Mean shortest path length computation

Implementation Limitations:

- **Component Handling**: Suboptimal management of disconnected subgraphs through zero-value assignments, potentially misrepresenting structural characteristics
- **Structure Constraints**: Depth calculations restricted to DAG structures, limiting broader applicability
- **Edge Analysis**: Uniform edge treatment potentially oversimplifying complex relationships
- **Statistical Representation**: Mean-based metrics potentially obscuring significant structural variations
- **Performance Scaling**: Computational complexity challenges in processing large-scale or dense networks