Problem Chosen	2023 ShuWei Cup Summary Sheet	Team Control Number
C		2023091819503

The Detection of AI-generated Text Based on SVM and Factor Analysis

Summary

AI technology is advancing rapidly, and the widespread attention to large language models has brought about significant concerns due to the potential misuse, prompting a growing focus on scientifically and effectively detecting AI-generated text and text generation patterns.

To facilitate further research, we quantified text generation rules and constructed a set of indicators to describe text features. Within the syntax dimension, we measured the writing style by using average sentence length as an indicator. In the semantic dimension, text richness was measured, considering vocabulary richness and using self-BLEU as a metric from the lexical and sentence perspectives, respectively.

For Problem One, we leveraged factor analysis models to establish a detection model for AI-generated text. To obtain text generation rules, we utilized the ChatGPT API by formulating requests for text generation patterns based on the problem descriptions. The resulting 20 texts in Appendix 1 were split into training and testing sets for subsequent model training and evaluation.

The results indicate that, compared to human-generated text, AI-generated text exhibits lower vocabulary richness, higher similarity between sentences, and no significant difference in average sentence length. There is a positive correlation between word limit, generation frequency, and average sentence length, as well as vocabulary richness. Additionally, these factors show a negative correlation with self-BLEU. In comparison between formal and colloquial text styles, the academically-oriented text style tends to have longer average sentence lengths, lower vocabulary richness, and higher self-BLEU scores.

For Problem Two, which is essentially a multivariate analysis problem, we applied factor analysis to fit the relationship between text features and generation rules. Factor analysis, as a method for latent variable modeling, offers advantages such as data dimensionality reduction and high accuracy in the context of multivariate analysis problems.

For Problem Three, essentially a binary classification problem, we applied a separable support vector machine to distinguish AI-generated text from human-generated text. As the training set's sample points were not linearly separable, we mapped them into a higher-dimensional feature space to make them linearly separable. Following the differentiation of AI-generated text, we utilized the methods from question two to determine their generation rules.

For Problem Four, we represented formulas in LaTeX code and applied the previously developed AI text detection model for plagiarism detection. For images, we classified them, extracted features for each class, and stored them in a database for plagiarism detection.

Keywords: LLM; Factor Analysis; SVM; Text Analysis

Contents

1 Introduction

Science and technology have completely mastered the objective world, and as a result, this world may lose, and indeed has already lost, external and antagonistic forces. It will also transform into an intermediary for human self-realization as a consequence.

——**Herbert Marcuse**

1.1 Problem Background

AI has developed rapidly these years. ChatGPT, a type of Large Language model(LLM), is the cutting edge of the applications of AI. ChatGPT imitates the mechanism of human's brain, based on deep learning model, Self-supervised Learning and Transfer Learning. It can generate accurate and multiple texts through learning numerous sample sets. However, the misuse of NLG models raises social concerns in many domains and the amount of the texts based on AI is extremely large. Thus, to establish general arithmetic to detect AI-generated texts is of great significance.

1.2 Restatement of the Problem

Considering the background and restricted conditions provided, we decide to solve the following problems:

1. **Problem One:** Based on word-number constrain, text-generation times, translation condition and language style, we divide the AI-generated texts into different patterns. Compute the statistics in different patterns of texts.

2. **Problem Two:** For passages provided in Appendix II, we need to judge what generation pattern each passage is though the statistics features of each patterns.

3. **Problem Three:** For passages in Appendix III, we need to judge whether they are generated by LLM. If they are, as the same approaches in Problem One, what generation pattern of each passage is.

4. **Problem Four:** Establish models to analyze whether the images, formulas and models in Appendix IV are in suspect of academic misconduct.

1.3 Our Work

1. **Problem One:** Set the independent variable based on word count limit, regeneration frequency, translation condition and language style. Ask ChatGPT to rewrite the articles according to the rules. Set several indicators to portray the features of artificial texts and various types of AI-generated texts. We explore the relationship between the generation rules and text features.

2. **Problem Two and Three:** For each passage provided in Appendix II and paragraphs provided in Appendix III, we construct SVM and factor analysis model to distinguish the AI-generated texts and human-authored texts and judge the generation patterns of AI-generated texts furthermore.

3. Problem Four: the main idea is to abstract the features of the formulas and images. As for formulas, we transform them to text through Latex code and apply the model in

problem two and three. For images, we abstract the features of values, ratios and shapes of images to further compare the with other images in database.

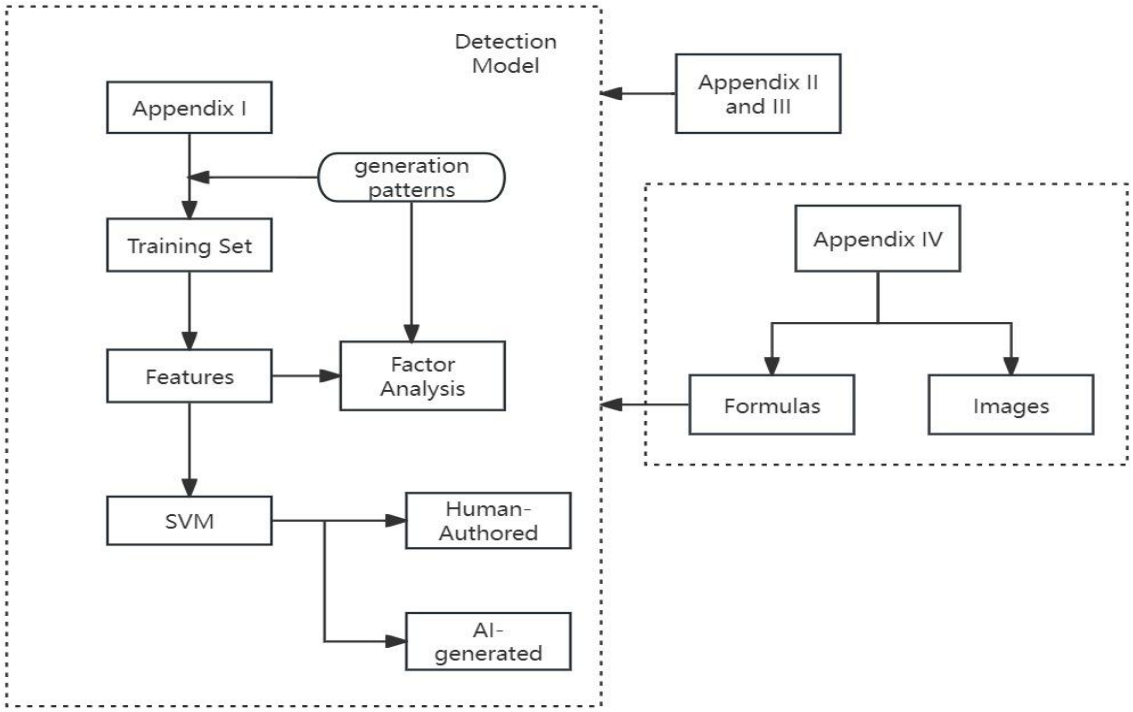

Figure 1: Overall Structure of Our Model

2 Assumptions and Justification

2.1 Assumption

1. **Assumption 1:** All the passages and paragraphs are generated by ChatGPT-3.5. **Reason:** Since different LLM generates different texts, the features of the generated texts by different LLM differ. To make the detection model reliable, the generation model of training sets and test sets should be the same.

2. **Assumption 2:** The twenty passages in Appendix I is typical and it is able to present the overall characteristics of Appendix II to IV. **Reason:** The problem inquire us to use Appendix I as a training set to judge whether the texts in Appendix II to IV are generated by LLM and infer the generation patterns.

3. **Assumption 3:** Our research thoroughly extracts textual information. **Reason:** The extent of extracting textual information is crucial for the representatives and reliability of the models we obtain.

4. **Assumption 4:** The language style of text can be broadly categorized into colloquial and formal. **Reason:** The use of language is typically dictated by the context, with a preference for simplicity and clarity in informal settings and adherence to formal standards in formal contexts.

2.2 Justification

1. **Generation rule:** the request to LLM on the language style, the regeneration times and the translation situation.

2. **Generation pattern:** different combination of generation rules is a certain generation pattern.

3. **Text feature:** the statistics of the texts in training set, based on the TF-IDF vectors.

3 Notations

The key mathematical notations used in this paper are listed in Table 1.

Table 1: Notations used in this paper

Symbol	Description
X	Text feature metric, $X = (x_1, x_2, x_3, x_4)$
x_i	The i-th text feature of texts
Y	Generation rule metric, $Y = (y_1, y_2, y_3)$
y_i	the i-generation rule of texts
$\boldsymbol{a}_i$	a vector composed of statistical features of i th text
T	training set of SVM
ω	a column vector that renders the training set linearly separable.
y_i	categorical variable of $i\,th\,text$
M^+	the sample sets when $y_i = 1$
M^+	the sample sets when $y_i = -1$
conv(M^+)	convex hulls of the sample sets M^+
conv(M^-)	convex hulls of the sample sets M^-
$\boldsymbol{\Lambda}$	Loading metric

4 Sample acquisition and Metric Construction

4.1 Sample Acquisition: Text Generator

We establish a crawler programme to collect the twenty blogs in Appendix I. Use OpenAI API to establish a text generator. The generator can control the generation patterns of generated texts through controlling the generation rules provided in the problems. Then, we obtain various versions of the twenty passages rewritten by ChatGPT-3.5 of different rules. The rules containing regenerating frequency from zero to five, word constrain of 200 and 500 words, translation conditions of translated, non-translated and language style from academic expression and oral expression.

4.2 Metric construction

4.2.1 Generation pattern

We define the text generation pattern as a vector composed of four text generation rules. We represent the text generation rules numerically as follows:

a) The word count limit (WCL) $\in \{200, 500\}$, where WCL=200 indicates a word count limit set to 200 words, and WCL=500 indicates a word count limit set to 500 words;

b) The regeneration frequency (RF) $\in \{0, 1, 2, 3, 4, 5\}$, where RF=0 represents the original text, and RF takes other values to indicate the number of clicks on the regenerate button;
c) The translation condition (TC) belongs to $\in \{0, 1\}$, where TC=1 indicates that it has been translated, and TC=-1 indicates that it has not been translated.
d) The language style (LS) belongs to $\in \{0, 1\}$, where LS=1 indicates an academic language style, and LS=0 indicates a requirement for a colloquial language style.

We use the generated rule vector $X = (x_1, x_2, x_3, x_4)$ to represent the generation pattern, where x_1, x_2, x_3, x_4 respectively represent WCL, RF, TC, LS. a_{ij} represents the value of the j-th generation rule for the i-th text, forming a vector represented as $\boldsymbol{a}_{\cdot j} = (a_{1j}, a_{2j}, \cdots, a_{nj})$.

4.2.2 Text Information Extraction

We use the Term Frequency-Inverse Document Frequency (TF-IDF) method to digitalize texts, thereby extracting information from each text and obtaining TF-IDF vectors. TF-IDF is a numerical statistic that reflects the importance of a word in a document relative to a corpus. TF-IDF is commonly used in information retrieval and text mining to measure the significance of a word in a document with respect to a larger collection of documents.

Term Frequency (TF) measures how often a term (word) appears in a document. It is calculated as the ratio of the number of times a term occurs in a document to the total number of terms in the document. A word is more important if it appears frequently within a document.

Inverse Document Frequency (IDF) measures how important a term is across the entire corpus. It is calculated as the logarithm of the ratio of the total number of documents in the corpus to the number of documents containing the term. A word is more significant if it is rare across the entire corpus.

The TF-IDF score for a term *t* in a document *d* is then given by the product of TF and IDF:

$$\begin{aligned} \text{TF-IDF}(t,d,D) &= \text{TF}(t,d) \times \text{IDF}(t,D) \\ &= \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \\ &\times \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t+1}\right) \end{aligned} \tag{1}$$

The TF-IDF score is higher for terms that are frequent in the document and rare in the corpus, making it a useful metric for determining the relevance of a term to a specific document in the context of a larger collection.

When processing Chinese text, we first segment the text into word chunks and then use the aforementioned method to obtain the TF-IDF vectors for the Chinese text.

4.2.3 Feature Description

We construct the feature indicators of two dimensions, syntax and semantics. For syntax, which reveals the style of a text, we select length of sentences; for semantics, which reveals the diversity of a text, we select the self-BLEU for sentences and the word richness, as Table

1 shows.

Table 1: Feature Description Framework

Perspectives	Dimensions	features
Syntax	Language style	Mean length of sentences
Semantics	Diversity	Word richness
		Self-BLEU

The feature indicators can be computed by following formulas:

1. Mean length of sentences equals to the word count of a text divided by the sentence count.
2. Word richness equals to the vocabulary size divided by word count.
3. Self-BLEU is an indicator to measure the diversity of sentences. Self-BLEU equals to the average of BLEU values between every two sentences. We calculate BLEU of sentence a and sentence b as following formulas:

a) Compute precision: n-gram is a way to segment a sentence into fragments containing n words. For each n-gram, calculate the number of matching n-grams in two sentences and sum them up. Then, divide by the total number of n-grams in two sentences.

$$\text{Precision} = \frac{\text{Number of matching n-grams}}{\text{Total number of n-grams in the two sentences}}$$

b) Compute Brevity Penalty: If the system-generated translation is shorter than the reference translation, a brevity penalty is applied. The brevity penalty is the system-generated translation length divided by the length of the closest reference translation.

$$\text{Brevity Penalty} = \begin{cases} 1, & \text{if the length of } b \geq \text{length of } a \\ e^{(1-\frac{\text{length of } sentence\ a}{\text{length of } sentence\ b})}, & \text{if the length of } b < \text{length of } a \end{cases}$$

c) Compute BLEU Score: We combine precision and brevity penalty to compute the BLEU score. BLEU considers precision for different n-grams, typically using 1-gram to 4-gram.

$$\text{BLEU} = \text{Brevity Penalty} \times \exp\left(\frac{1}{4}\sum_{n=1}^{4}\log(\text{Precision}_n)\right)$$

5 Model I: Multivariate Analysis Model

Problem two and three ask us to find out what the generation patterns of AI-generated text are. We should establish association analysis model to explore the relationships between the generation rules and the text features. We have set a series of metrics to portray the feature of the passages in Appendix I. The problem converts to the text feature metrics and the generation rules.

The classical model to explore relationship between two groups of variables is regression analysis model. The establishment of our association begins with the regression model.

5.1 Multivariate Linearly Regression Model

Let $a_{ij}(i = 1,2, ..., n, j = 1,2, ..., r)$ denote the j-th generation rule for the i-th text, and $b_{ik}(i = 1,2, ..., n, k = 1,2, ..., f)$ denote the k-th feature for the i-th text. Vector $\boldsymbol{a}_{i\cdot}$ represents the vector composed of generation rules for the i-th text, and vector $\boldsymbol{b}_{i\cdot}$ represents the vector composed of features for the i-th text. Vectors $\boldsymbol{a}_{\cdot j}$ represents the vectors composed of the j-th generation rules, and vector $\boldsymbol{b}_{\cdot k}$ represents the k-th features across different texts. Text feature matrix $\boldsymbol{B} = [b_{ij}]$ is composed of the features for each text. We focus on the study of the j-th generation rule for each text. The multivariate linear regression model can be expressed as:

$$\begin{cases} a_{ij} = \beta_0 + \beta_{1j} b_{i1} + \beta_{2j} b_{i2} + \cdots + \beta_{fj} b_{if} + \varepsilon_{ij} \\ \varepsilon_{ik} \sim (0, \sigma_k^2) \end{cases}, \ i = 1, 2, \cdots, n \tag{2}$$

where $\beta_{ij}(i = 1,2, ..., n)$ represents the regression coefficients, and ε_{ij} is the residual, following a normal distribution. Expressing the above equation in matrix form, we have:

$$\begin{cases} \boldsymbol{a}_{\cdot j} = \boldsymbol{B}\boldsymbol{\beta}_j + \boldsymbol{\varepsilon}_j \\ \boldsymbol{\varepsilon}_j \sim (0, \sigma_j^2 \boldsymbol{E}_n) \end{cases} \tag{3}$$

Here, $\boldsymbol{E}_n$ represents the identity matrix. $\boldsymbol{\beta}_j = (\beta_{1j}, \beta_{2j}, ..., \beta_{fj})$ is the vector of regression coefficients, and $\boldsymbol{\varepsilon}_j = (\varepsilon_{1j}, \varepsilon_{2j}, ..., \varepsilon_{nj})$ is the vector of residuals. We utilize the least squares method to estimate the parameter vector $\boldsymbol{\beta}_j$. That is, when selecting appropriate estimates $\hat{\boldsymbol{\beta}}_j$, i.e., $\beta_{ij} = \hat{\beta}_{ij}(i = 1, 2, ..., n)$, the sum of squared errors (RSS) is minimized:

$$\min RSS = \sum_{i=1}^{n} \varepsilon_{ij}^2 = \sum_{i=1}^{n} (a_{ij} - \beta_0 + \beta_{1j} b_{i1} + \beta_{2j} b_{i2} + \cdots + \beta_{fj} b_{if})^2 \tag{4}$$

The goal is to minimize the sum of squared residuals by choosing suitable estimates for the regression coefficients.

Taking the derivative of RSS with respect to β_k and setting it to zero to find the extremum conditions, we obtain the normal equations in matrix form:

$$\frac{\partial RSS}{\partial \boldsymbol{\beta}_j} = -2\boldsymbol{B}^T(\boldsymbol{x}_j - \boldsymbol{B}\boldsymbol{\beta}_j) = 0 \tag{5}$$

Rearranging the extremum conditions in matrix form yields the normal equations:

$$\boldsymbol{A}^T \boldsymbol{A}\boldsymbol{\beta}_k = \boldsymbol{A}^T \boldsymbol{b}_{\cdot k} \tag{6}$$

Solving the above equations provides the vector of regression coefficients, $\hat{\boldsymbol{\beta}}_j$. This model is applied to calculate the relationship between text features and generation rules for each text. Consequently, the formula for determining the text generation pattern is given by:

$$\boldsymbol{x}_{\cdot j} = \boldsymbol{Y}\hat{\boldsymbol{\beta}}_j \tag{7}$$

Here, vector $\boldsymbol{x}_{\cdot j}$ represents the vector composed of the j-th generation rule for each different text to be tested, and matrix $\boldsymbol{Y} = [y_{ij}]$ is composed of the text features for each

text to be tested.

Multivariate linear regression is effective in fitting problems involving multiple variables. However, the RSS values calculated by this model may not fall within a satisfactory range. This is because the regression model is well-suited for addressing problems involving manifest variables and numerical variables, but it lacks explanatory power for latent variables such as language style and categorical variables like translation conditions. In light of this limitation, we have established a factor analysis model to investigate the impact of generation patterns on text features.

5.2 Factor Analysis Model

Because of the limitation, based on multivariate linearly regression model, we promote it to analysis model.

5.2.1 Establishment of Factor Analysis Model

Similar to the previous model, we begin by studying the j-th rule of the i-th text. We assume that each j-th rule of every text can be linearly represented by text features, $x_1, x_2, ..., x_j$. These four common factors represent the generation rules of word count limit, regeneration frequency, translation conditions, and language style. The representation is expressed as:

$$x_j = \mu_j + \rho_{j1} y_1 + \rho_{j2} y_2 + \cdots + \rho_{jf} y_f + \nu_j,\ j = 1, 2, \cdots, r \tag{8}$$

Here, y_{ik} is the k-th feature of the i-th text. $\rho_{ijk}(i = 1, ..., n, k = 1, ..., f, j = 1, ..., 4)$ represents the loading coefficient, indicating the impact of the j-th generation rule on the k-th feature of the i-th text. μ_{ik} is the total mean value of the i-th text, and ν_{ik} is the unique factor representing the part of the k-th feature of the i-th text that cannot be represented by these four common factors. The above equation can be rewritten in matrix form:

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\Lambda Y} + \boldsymbol{\nu} \tag{9}$$

Here,

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix},\ \boldsymbol{\mu}_j = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_r \end{bmatrix},\ \boldsymbol{\Lambda}_j = \begin{bmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1f} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2f} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{r1} & \rho_{r2} & \cdots & \rho_{rf} \end{bmatrix}, \boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_f \end{bmatrix},\ \boldsymbol{\nu}_j = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_r \end{bmatrix}$$

The factor loading ρ_{ijk} represents the correlation coefficient between the i-th text and the j-th generation rule, reflecting the importance of the correlation between the i-th variable and the j-th common factor. The absolute value of the loading coefficient signifies the closeness of the correlation. Therefore, the loading factor matrix $\boldsymbol{\Lambda}_j$ is the correlation matrix between $\boldsymbol{Y}_j$.

5.3 Solution Using Principal Component Analysis

5.3.1 Solution of the Correlation Matrix

The Spearman rank correlation coefficient is a non-parametric statistical method used to measure the correlation between two variables. It is particularly suited for assessing the

correlation between ordered variables and is based on ranking data, independent of the distribution characteristics of the data. This method effectively circumvents the issue of different value ranges for each feature, eliminating the need for normalization steps. The following are the steps to calculate the Spearman rank correlation coefficient for any two sets of feature indicators:

1) Calculate the rank differences for two sets of data:

First, assign new ranks to the ordered data sets based on their magnitudes for 1, 2, ..., n. In cases where values are equal, assign the average rank. Next, compute the rank differences d_i by subtracting the new ranks of the two data sets.

2) Calculate the Spearman rank correlation coefficient between two objects:

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{10}$$

where n is the sample size. The magnitude of the Spearman rank correlation coefficient allows the evaluation of the degree of association between two objects. $r_s \in [-1,1]$, and when $r_s > 0$, there is a positive correlation between the two evaluated objects; when $r_s < 0$, there is a negative correlation. The closer the absolute value of r_s is to 1, the stronger the correlation between the evaluated objects.

3) Significance level test:

Formulate the null hypothesis H_0: $r_s = 0$ and the alternative hypothesis H_1: $r_s \neq 0$.

i. For sample sizes less than 30, validate the significance level of the Spearman rank correlation coefficient by consulting the critical values table for Spearman rank correlation (see Appendix 3). This table includes the correspondence between Spearman rank correlation coefficients, significance levels, and sample sizes less than 30. By finding the critical value closest to and less than $|r_s|$, the corresponding significance level can be determined, allowing for the rejection of the null hypothesis H_0.

ii. For sample sizes greater than 30, the test statistic needs to be examined. Calculate the corresponding p-value and determine the associated significance level.

By that way, we obtain the Spearman rank correlation coefficient between text features $y_{\cdot j}(j = 1, 2, \cdots, f)$. Furthermore, we can obtain a correlation coefficient metric $\boldsymbol{R}$.

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & \cdots & r_{1f} \\ \vdots & \ddots & \vdots \\ r_{f1} & \cdots & r_{ff} \end{bmatrix} \tag{11}$$

Here, r_{ij} for $i, j = 1,2, ..., f$ is the Spearman rank correlation coefficient between the i-th feature and the j-th feature.

Denote the eigenvalue of correlation coefficient metric $\boldsymbol{R}$ as $\lambda_1 > \lambda_2 > ... > \lambda_f$ and denote the orthogonal eigenvectors of $\boldsymbol{R}$ as $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, ..., \boldsymbol{\eta}_f$. Then, the factor loading metric $\boldsymbol{\Lambda}$ is represented as:

$$\boldsymbol{\Lambda} = [\sqrt{\lambda_1}\boldsymbol{\eta}_1, \sqrt{\lambda_2}\boldsymbol{\eta}_2, \cdots, \sqrt{\lambda_f}\boldsymbol{\eta}_f] \tag{12}$$

Now, we establish the relationship between text features and generation rules. We can judge the generation patterns through inputting the every f features of the text to be tested.

6 Model 2: Binary Classification model

Problem three asks us to distinguish between AI-generated text and human-authored text. The detection task is typically approached as a binary classification problem, with the objective of capturing textual, features that differentiate between human-written and LLM-generated texts. In this study, we employ Support Vector Machines (SVM) to achieve the purpose. SVM is a machine learning model used for classifying sample sets. The primary process of SVM involves solving for a hyperplane that maximally separates two classes of sample points.

The original text from Appendix I and AI-generated text serve as the training set T. We use a categorical variable $y_i \in \{1, -1\}$ to label the two classes of text; for human-authored text, $y_i = 1$, and for AI-generated text, $y_i = -1$. We denote the input space as ω, which contains the sample points $\boldsymbol{a}_i (i = 1, 2, \cdots, l)$. Each text is treated as a sample point, where $\boldsymbol{a}_i$ is a vector composed of statistical features of i th texts. Subsequently, the training set T is represented as

$$T = \{[\boldsymbol{a}_1, y_1], [\boldsymbol{a}_2, y_2], \cdots, [\boldsymbol{a}_l, y_l]\} \in (\Omega \times Y)^l \tag{13}$$

Based on the training set, we solve for the decision function $g(\boldsymbol{x})$ and use it as the argument for the sign function to obtain the classification function $f(\boldsymbol{x})$.

$$f(\boldsymbol{x}) = \text{sign}(g(\boldsymbol{x})) \tag{14}$$

Here, $\boldsymbol{x}$ is a vector composed of the components of the sample to be tested. The classification function allows us to derive the corresponding Y value for any given text, thereby determining the text type.

The fundamental SVM type is the linearly separable SVM, characterized by the strictest conditions. To address more intricate challenges, we employ the separable SVM, deemed the most fitting solution for our specific problem.

6.1 Linearly Separable SVM

We define the sample set composed of human-authored text as M^+ and the sample set composed of AI-generated text as M^-. Thus, we have

$$M^+ = \{\boldsymbol{\alpha} | y_i = 1, [\boldsymbol{\alpha}_i, y_i] \in T\},\ M^- = \{\boldsymbol{\alpha} | y_i = -1, [\boldsymbol{\alpha}_i, y_i] \in T\} \tag{15}$$

Simultaneously, we define the convex hulls of the sample sets M^+ and M^- as conv(M^+) and conv(M^-), respectively.

$$
\begin{aligned}
conv(M^+) &= \left\{ \boldsymbol{\alpha} = \sum_{j=1}^{N_+} \lambda_j \boldsymbol{\alpha}_j \mid \sum_{j=1}^{N_+} \lambda_j = 1; \lambda_j \geq 0;\ \boldsymbol{\alpha}_j \in M^+ \right\} \\
conv(M^-) &= \left\{ \boldsymbol{\alpha} = \sum_{j=1}^{N_-} \lambda_j \boldsymbol{\alpha}_j \mid \sum_{j=1}^{N_-} \lambda_j = 1; \lambda_j \geq 0;\ \boldsymbol{\alpha}_j \in M^- \right\}
\end{aligned} \tag{16}
$$

Here, N^+ is the number of sample points in the set M^+, and N^- is the number of sample points in the set M^-.

A linearly separable SVM is based on a model designed for linearly separable training sets. We define linear separability as

$$
\begin{aligned}
\forall \boldsymbol{\alpha} \in M^+,\ (\boldsymbol{\omega} \cdot \boldsymbol{\alpha}_i) + b \geq \varepsilon \\
\forall \boldsymbol{\alpha} \in M^-,\ (\boldsymbol{\omega} \cdot \boldsymbol{\alpha}_i) + b \leq -\varepsilon
\end{aligned} \tag{17}
$$

Here, $\boldsymbol{\omega}$ and b are an n-dimensional column vector and a constant, represented as $\boldsymbol{\omega} \in \mathbf{R}^n, b \in \mathbf{R}$, respectively, satisfying the conditions mentioned above. Clearly, the linear separability of the training set T is a necessary and sufficient condition when the convex hulls of M^+ and M^- are disjoint.

We express the hyperplane as $[\boldsymbol{\omega} \cdot \boldsymbol{a}_i] + b = 0$ and define the canonical hyperplane as the one satisfying the equation

$$
\begin{cases}
y_i[(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b] \geq 0, \\
\min\limits_{i=1,\cdots,l} |(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b| = 1, \quad i = 1, 2, \cdots, l
\end{cases} \tag{18}
$$

Choosing the canonical hyperplane satisfying the marginal conditions, denoted as P: $(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b = 0$, ensures that P satisfies the following conditions:

$$
\begin{cases}
(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b \geq 1, y_i = 1 \\
(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b \leq -1, y_i = -1
\end{cases} \tag{19}
$$

We define an ordinary support vector Machine as a sample point $\boldsymbol{a}_i$ satisfying $[\boldsymbol{\omega} \cdot \boldsymbol{a}_i] + b = \pm 1$. The distances from the sample sets M^+ and M^- to the canonical hyperplane are denoted as D^+ and D^-, respectively:

$$
D^+ = \min_{y_i = 1} \frac{|(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b|}{||\omega||} = \frac{1}{||\omega||}, \quad D^- = \min_{y_i = -1} \frac{|(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b|}{||\omega||} = \frac{1}{||\omega||} \tag{20}
$$

To maximize the sum of distances from the sample sets M^+ and M^- to the canonical hyperplane, we seek the solution to

$$
\begin{aligned}
&\min\ D^+ + D^- = \frac{1}{2}||\boldsymbol{\omega}||^2 \\
&\text{s.t. } y_i\{[\boldsymbol{\omega} \cdot \boldsymbol{a}_i] + b\} \geq 1,\ i = 1, 2, \cdots, l
\end{aligned} \tag{21}
$$

Solving the optimization problem under the aforementioned linear constraints, we employ the Lagrange multiplier method. The Lagrangian function is written as

$$L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\omega}||^2 + \sum_{i=1}^{l} \alpha_i \{1 - y_i[(\boldsymbol{\omega} \cdot \boldsymbol{a}_i) + b]\} \tag{22}$$

Here, $\alpha_i (i = 1, 2, \cdots, l)$ represents the Lagrange multipliers. With the definition of the dual, we transform the above problem into the Lagrangian dual problem:

$$\begin{aligned} &\max \ -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (\boldsymbol{a}_i, \boldsymbol{a}_j) + \sum_{i=1}^{l} \alpha_i \\ &\text{s.t.} \begin{cases} \sum_{i=1}^{l} y_i \alpha_i = 0 \\ \alpha \geq 0, \ i = 1, 2, \cdots, l \end{cases} \end{aligned} \tag{23}$$

According to the Karush-Kuhn-Tucker (KKT) conditions, α_i is positive only when $\boldsymbol{a}_i$ is a support vector; otherwise, it is zero. Solving the optimization problem in (4), we obtain the optimal solution $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \cdots, \alpha_l^*)^T$. Selecting one positive component α_j^*, we derive the optimal solution b^*.

$$b^* = y_j - \sum_{i=1}^{l} y_i \alpha_i{}^* (\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j) \tag{24}$$

Thus, we obtain the classification function

$$f(\boldsymbol{x}) = \text{sign}\left[\sum_{i=1}^{l} y_i \alpha_i{}^* (\boldsymbol{\alpha}_i \cdot \boldsymbol{x}) + b^*\right] \tag{25}$$

The assumption of a linearly separable Support Vector Machine is excessively stringent, rendering our training set unsuitable for this assumption. Consequently, we seek a solution to transform the original training set into a new set that adheres to the linearly separable assumption.

6.2 Separable Support Vector Machine

The linearly separable Support Vector Machine requires that the training set T be linearly separable. For training sets with high interactivity among sample points, and non-linearly separable characteristics, we address this by mapping the sample points from the input space to a feature space, transforming them into a linearly separable training set $\widetilde{T}$ for classification. The fundamental idea of separable SVM is to uncover non-linear relationships in low-dimensional space, rendering the problem linearly separable in a higher-dimensional space.

Building upon the discussion of linearly separable SVM in the preceding text, the process of establishing a separable SVM model is outlined as follows:

1) Establishing the Feature Space。

To make the training set linearly separable, we establish a mapping relationship φ from the input space Ω to the high-dimensional Hilbert space H, represented as φ: $\Omega \to H$. We refer to H as the feature space, transforming the training set from the input space into a new training set in the feature space:

$$\begin{aligned}\widetilde{T} &= \{[\widetilde{\boldsymbol{a}}_1, y_1], [\widetilde{\boldsymbol{a}}_2, y_2], \cdots, [\widetilde{\boldsymbol{a}}_l, y_l]\} \\ &= \{[\varphi(\boldsymbol{a}_1), y_1], [\varphi(\boldsymbol{a}_2), y_2], \cdots, [\varphi(\boldsymbol{a}_l), y_l]\} \in (H \times Y)^l\end{aligned} \tag{26}$$

This transformation ensures that the new training set is linearly separable in the Hilbert space.

2) Establishing the Kernel Function

A kernel function is a mathematical function used to map input data to a high-dimensional space, defining the inner product of any sample point in the high-dimensional feature space H. We employ the Fourier kernel function for model construction:

$$K(a_i, a_j) = \sum_{k=1}^{n} \frac{1-q^2}{2[1-2q \cdot \cos(a_{ik} - a_{jk}) + q^2]} \tag{27}$$

In the feature space H, we obtain a hyperplane $[\boldsymbol{\omega} \cdot \varphi(\boldsymbol{x})] + b = 0$ that divides the transformed training set $\widetilde{T}$ into two parts. Thus, the original problem is transformed into the linearly separable Support Vector Machine problem for the new training set $\widetilde{T}$, and equation (1) is modified as follows:

$$\begin{gathered}\min \ \frac{1}{2}||\boldsymbol{\omega}||^2 \\ s.t. \ y_i\{[\omega \cdot \varphi(a_i)] + b\} \geq 1, \ i = 1, 2, \cdots, l\end{gathered} \tag{28}$$

Therefore, the Lagrangian dual problem for the linearly separable Support Vector Machine can be rewritten as:

$$\begin{gathered}\max \ -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\boldsymbol{a}_i, \boldsymbol{a}_j) + \sum_{i=1}^{l} \alpha_i \\ s.t. \begin{cases} \sum_{i=1}^{l} y_i \alpha_i = 0 \\ \alpha \geq 0, \ i = 1, 2, \cdots, l \end{cases}\end{gathered} \tag{29}$$

Solving the aforementioned optimization problem to obtain the optimal solution $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*, \cdots, \alpha_l^*]^T$, selecting one positive component α_j^*, we have

$$b^* = y_j - \sum_{i=1}^{l} y_i {\alpha_i}^* K(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j) \tag{30}$$

Subsequently, rewriting the classification function based on the above, we get

$$f(\boldsymbol{x}) = \operatorname{sign}\left[\sum_{i=1}^{l} y_i {\alpha_i}^* K(\boldsymbol{\alpha}_i, \boldsymbol{x}_j) + b^*\right] \tag{31}$$

With this function, AI-generated text and human-authored text can be classified accordingly.

7 Detection for Formula and Images

For Question Four, we interpret it as providing a method to calculate the correlation between formulas and images, in order to compare whether works suspected of plagiarism exhibit high correlation and determine if plagiarism exists.

Since a comprehensive text correlation algorithm model has already been provided in the previous sections, we contemplate the possibility of converting formulas and images into text and data language and applying the models from the first three sections. Specifically:

For formulas, considering each individual mathematical symbol, we assign a linguistic definition to each symbol. By doing so for each mathematical symbol, the formula can be textualized. Subsequently, cosine similarity can be calculated to obtain the final correlation model.

For images, due to the numerous types, we will provide textualization methods for only the three models mentioned in the question:

For Model 1, it can be understood as a directed graph of a neural network. By assigning a number to each circle and writing out each path's data from left to right and top to bottom in a directed manner, the model can be successfully textualized.

For Model 2, it can be understood as a function graph. Taking an appropriate length as the standard distance, we can discretize the graph by writing out the points on some functions. The judgment criteria for correlation can be calculated based on the proportions, coordinate values, derivatives, and curvature of the discretized points.

For Model 3, it can be understood as a clustering connection graph. We first extract the position and radius information of the major clusters (blue dots) and then extract the position and radius information of the smaller points (red dots). By observing their interconnection information, we can textualize these aspects and calculate the similarity from these five dimensions.

8 Solution of Problems

8.1 Statistic regularly

We have computed the text features and have conducted the qualitive relation between text features and generation rules.

The results indicate that, compared to human-generated text, AI-generated text exhibits lower vocabulary richness, higher similarity between sentences, and no significant difference in average sentence length. There is a positive correlation between word limit, generation frequency, and average sentence length, as well as vocabulary richness. Additionally, these factors show a negative correlation with self-BLEU.

In comparison between formal and colloquial text styles, the academically-oriented text style tends to have longer average sentence lengths, lower vocabulary richness, and higher self-BLEU scores.

8.2 Classification Model

Applying SVM, we divide the 20 passages in Appendix I into training set with 15 passages and 5passages. We solve the optimization problem (29) and obtain the optimal solution, thereby we gain the classification function. Through the function, we divide the paragraphs in Appendix III into AI-generated text and human-authored text.

8.3 Multivariable Analysis Model

We calculated the Spearman correlation coefficient matrix among text features and subsequently obtained the loading matrix $\mathbf{\Lambda}$as shown in the table below. Based on this, we established the relationship between text features and generation rules. By extracting features from the text to be tested, we derived the text generation rules for the text under examination.

Table 2: Loading Metric $\mathbf{\Lambda}$

	y_1	y_2	y_3
x_1	0.506	0.4312	-0.4213
x_2	0.3966	0.4591	-0.4587
x_3	0.8833	0.3819	-0.2718
x_4	0.7932	-0.3044	0.3078

This allowed us to establish the connection between text features and generation rules. Through feature extraction from the text to be tested, we were able to determine the text generation rules for the text under examination.

9 Model Evaluation and Further Discussion

9.1 Strengths

1. Factor analysis can effectively reduce the dimensionality of text features. By utilizing factor variables as new explanatory variables in modeling, the approach offers enhanced interpretability.

2. In the realm of AI text classification, SVM (Support Vector Machine) is widely adopted as the classifier of choice due to its low time complexity. One notable advantage of SVM, relative to other classifiers, is its capability to achieve precise classification with sample-targeted specificity by selecting an appropriate kernel function.

3. For plagiarism detection in non-textual content, we propose a viable approach. Converting graphical representations/formulas into textual formats allows their integration into text-based methods for a holistic comparison of the overall plagiarism level across an entire document.

9.2 Weaknesses

1. Insufficient sample size renders the trained model non-representative.

2. Regarding the fourth question, we are unable to provide a universal method for image extraction. Instead, feature extraction can only be tailored to specific types of images.

9.3 Further Discussion

In further research, it is imperative to establish statistical measures that more accurately reflect text features for analysis. The advancement of large language models has made it increasingly challenging to distinguish between AI-generated text and human-authored text. Consequently, the construction of metrics that more effectively highlight the distinctions between these two types of text becomes particularly crucial.

It is advisable to employ larger sample sizes for model training to enhance the accuracy of model parameters and decision outcomes. The samples should encompass a broader range of domains, styles, and structures, aiming to represent the diverse data used to train large language models. Ideally, the training set for the detection model should include as many samples as possible from the training data of large language models. Handling such a vast amount of information requires the utilization of deep learning, involving the development of new AI models specifically designed to detect instances of Large Language Models (LLM).

10 Conclusion

Our AI-generated text detection model is based on both binary classification and multivariate analysis models. It classifies AI-generated text, determining the generation patterns for each text. For non-textual information, we endeavor to convert it into text whenever possible for the application of the AI text model. In cases where conversion into text is challenging, we attempt to extract other textual information for plagiarism detection.

According to our established model, there are four pieces of AI-generated text in Appendix III. The generation patterns for the AI-generated text in Appendix III and the text in Appendix II are detailed in the table below.

Table 3: generation patterns in Appendix II

text	generation frequency	language style	translation condition	word count limit
1	3	academic	yes	200
2	4	academic	no	200
3	2	oral	yes	500
4	2	oral	yes	500
5	1	oral	no	200
6	1	oral	no	500
7	2	academic	no	200
8	2	academic	no	200
9	2	oral	yes	500
10	3	oral	no	500

Table 4: the display of AI-generated texts and the corresponding generation patterns

paragraph number	generation frequency	language style	translation condition	word count limit
1	2	academic	yes	yes
5	1	academic	yes	yes
7	1	academic	no	yes
10	2	academic	no	no

References

[1] Herbert Marcuse. Collected Works of Herbert Marcuse (Volume 2): Towards a Critique of Society [M]. Translated by Gao Haiqing and Tao Tao, Beijing: People's Publishing House, 2019.

[2] Tang R, Chuang Y N, Hu X. The science of detecting llm-generated texts[J]. ar**v preprint ar**v:2303.07205, 2023.

Appendices

Appendix 1

Introduce: the main code by python

```
import sklearn.feature_extraction.text as ft
import jieba
import pandas as pd
import copy
from tqdm import tqdm
import openai
import json
import os
import sklearn.feature_extraction.text as ft
import jieba
import numpy as np
import nltk
import re
from nltk.translate.bleu_score import sentence_bleu

with open("rawtext.txt", "r", encoding='utf-8') as f:    #打开文本
    rawtext = f.read()      #读取文本

#%%整合出人类文本集
text_raw=rawtext.split("||")
for i in range(len(text_raw)):
    t=text_raw[i].split("\n")
    t=[j for j in t if len(j)>1]
    t="||\n".join(t)
    text_raw[i]=t
    del t
#%%%将篇幅过大的文本拆分成段落，最终得到 92 个样本
for i in range(len(text_raw)):
    t=text_raw[i].split("\n")
    t=[j.rstrip("||") if len(j)<200 else j for j in t ]
    t="\n".join(t)
    text_raw[i]=t
    del t
text_raw="||".join(text_raw)
text_raw=text_raw.split("||")
text_raw=[i for i in text_raw if len(i)>150 and len(i)<1500]
```

```
#%%%存储
print(text_raw)
#%%askgpt
def askgpt(q):
    # 目前需要设置代理才可以访问 api
    os.environ["HTTP_PROXY"] = "http://10.1.179.67:7890"
    os.environ["HTTPS_PROXY"] = "http://10.1.179.67:7890"

    openai.api_key = "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"

    rsp = openai.ChatCompletion.create(model="gpt-3.5-turbo",
                                       messages=[{
                                           "role":
                                           "system",
                                           "content":
                                           "You are a good assistant"
                                       }, {
                                           "role": "user",
                                           "content": q
                                       }])
    answer = rsp.get("choices")[0]["message"]["content"]
    return answer
#%%% 遍历区间
words_constraints = [200, 500]
languages_list = ["中文", "英文"]
styles_list1 = [
    "你是一个科普作家，需要让文字轻松活泼，语言风格尽可能口语化。", "你是该领域的科研专家，在撰写一篇学术论文，需要让语言风格冷静客观"
]
styles_list2 = [
    "As a research expert in the field, you are drafting a paragraph for an academic paper,requiring a language style that is as scholarly, rigorously objective, and compliant with academic standards as possible.",
    "You're a science popularizer, aiming for a breezy and lively tone, with language as casual as possible."
]
regenerate_times = range(0, 5)
answers_list = []
#%%% 中文
for s in tqdm(text_raw):
    for words in tqdm(words_constraints):
        for style in tqdm(styles_list1):
            for i in tqdm(regenerate_times):
                q = f"{style}请你用中文重写以下文字,\
                    形成一段单词数不超过{words}的文字。如果字数不足{words}词，请你进行扩写到{words}词；\
                    如果字数超出{words}词，请你进行缩写到{words}词。\
                    切记，不计标点，空格，你生成的文字单词数尽量接近{words}
```

```
词，误差不超过 50 词。    \n {s}"

                        answer = askgpt(q)
                        answers_list.append(answer)


#%%% 英文

answers_list = []
for s in tqdm(text_raw):
    for words in tqdm(words_constraints):
        for style in tqdm(styles_list2):
            for i in tqdm(regenerate_times):
                q = f"{style}Please rewrite the following text in English, creating a paragraph with a word count equal to {words}. If the word count is below {words} words, \
                please expand it to reach {words} words; if it exceeds {words} words, please condense it to {words} words. \
                Remember, punctuation and spaces are not counted,\
                and strive to make the word count of your generated text as close to {words} words as possible, \
                with an error margin not exceeding 50 words.\n {s}"

                answer = askgpt(q)
                answers_list.append(answer)



#%% rule
answers=text_raw
rule1=pd.DataFrame(columns=["label","regenerate_times","language","words_constraint","style"])
rule2=pd.DataFrame(columns=["label","regenerate_times","language","words_constraint","style"])
#中文
for words in tqdm(words_constraints):
    for j in tqdm(range(2)):#style
        for i in tqdm(regenerate_times):

rule1.append(dict(zip(list(rule1.columns)[:-1],[1,i,0,words,j])),ignore_index=True)
#英文
for words in tqdm(words_constraints):
    for j in tqdm(range(2)):#style
        for i in tqdm(regenerate_times):

rule2.append(dict(zip(list(rule2.columns)[:-1],[1,i,1,words,j])),ignore_index=True)
#%% text-analysis
#%%%将文档集转化为词频矩阵
def Chinese_countarray(X):#X is a list of strings(文档集)
    tokenized_text = []
```

```
    for i in X:
        tokenized_text.append(" ".join(jieba.cut(i)))
    svac=ft.CountVectorizer()
    X=svac.fit_transform(tokenized_text)
    vac=svac.get_feature_names_out()
    x=X.toarray()
    return vac,x
def English_countarray(X):#X is a list of strings(文档集)
    tokenized_text = []
    svac=ft.CountVectorizer()
    X=svac.fit_transform(tokenized_text)
    vac=svac.get_feature_names_out()
    x=X.toarray()
    return vac,x
#%%% 词汇丰富度
def abundant(x):#x 为一个文档集的词频矩阵
    xvactotal=x.sum(axis=1)
    xvactype=np.count_nonzero(x,axis=1)
    y=xvactype/xvactotal
    return y
#%%%% test
_,x=Chinese_countarray(text_raw)
x_abundant=abundant(x)
#%%% 分句
def Chinese_sent_tokenizer(s):#s is a text(in str)
    sentences = re.split('(。|！ |\!|\.|？ |\?)',s)
    new_sents = []
    for i in range(int(len(sentences)/2)):
        sent = sentences[2*i] + sentences[2*i+1]
        new_sents.append(sent)
    sentences_number=len(new_sents)
    return sentences_number,new_sents

def English_sent_tokenizer(s):#s is a text(in str)
    sentences=nltk.sent_tokenize(s)#return a list of sentences
    sentences_number=len(sentences)
    return sentences_number,sentences


#%%% self-BLEU
def get_bleu_score(sentence, remaining_sentences):
    lst = []
    for i in remaining_sentences:
        bleu = sentence_bleu(sentence, i)
        lst.append(bleu)
    return lst


def calculate_selfBleu(sentences):
```

```
    '''
    sentences - list of sentences generated by NLG system
    '''
    bleu_scores = []

    for i in sentences:
        sentences_copy = copy.deepcopy(sentences)
        remaining_sentences = sentences_copy.remove(i)
        print(sentences_copy)
        bleu = get_bleu_score(i,sentences_copy)
        bleu_scores.append(bleu)

    return np.mean(bleu_scores)
#%%%%test
ss,sss=English_sent_tokenizer(s)
ss,sss1=English_sent_tokenizer(s1)

print(calculate_selfBleu(sss)-calculate_selfBleu(sss1))
```

Appendix 2

Introduce: Spearman's rank correlation critical values table

n	Significance level for a two-tailed test.			
	0.1	0.05	0.02	0.01
4	1	\	\	\
5	0.9	1	1	\
6	0.829	0.886	0.943	1
7	0.714	0.786	0.893	0.929
8	0.643	0.738	0.833	0.881
9	0.6	0.7	0.783	0.833
10	0.564	0.648	0.745	0.794
11	0.536	0.618	0.709	0.755
12	0.503	0.587	0.671	0.727
13	0.484	0.56	0.648	0.703
14	0.464	0.538	0.622	0.675
15	0.443	0.521	0.604	0.654
16	0.429	0.503	0.582	0.635
17	0.414	0.485	0.566	0.615
18	0.401	0.472	0.55	0.6
19	0.391	0.46	0.535	0.584
20	0.38	0.447	0.52	0.57
21	0.37	0.435	0.508	0.556
22	0.361	0.425	0.496	0.544
23	0.353	0.415	0.486	0.532
24	0.344	0.406	0.476	0.521
25	0.337	0.398	0.466	0.511
26	0.331	0.39	0.457	0.501
27	0.324	0.382	0.448	0.491
28	0.317	0.375	0.44	0.483
29	0.312	0.368	0.433	0.475
30	0.306	0.362	0.425	0.467
35	0.283	0.335	0.394	0.433
40	0.264	0.313	0.368	0.405
45	0.248	0.294	0.347	0.382
50	0.235	0.279	0.329	0.363
60	0.214	0.255	0.3	0.331
70	0.19	0.235	0.278	0.307
80	0.185	0.22	0.26	0.287
90	0.174	0.207	0.245	0.271
100	0.165	0.197	0.233	0.257

Zar, J. H. (1972). Significance testing of the Spearman rank correlation coefficient. Journal of the American Statistical Association, 67, 578.