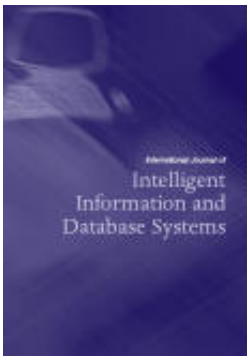


International Journal of Intelligent Information and Database Systems

ISSN online: 1751-5866 - ISSN print: 1751-5858
https://www.inderscience.com/ijiids

English digital transformation algorithm for distributed big data based on Spark

Xiaochao Yao

DOI: 10.1504/IJIIDS.2026.10077800

Article History:

Received:	03 July 2025
Last revised:	28 October 2025
Accepted:	24 December 2025
Published online:	10 September 2026

English digital transformation algorithm for distributed big data based on Spark

Xiaochao Yao

Hainan Vocational University of Science and Technology,
18 Qiongshan Avenue, Meilan District,
Haikou City, 570100, Hainan Province, China
Email: mamalin8483@163.com

Abstract: In the era of big data, the main challenge of digital transformation lies in the inability of traditional text clustering algorithms to efficiently process large-scale English data, resulting in low adaptability and delayed information extraction that hinder intelligent decision-making. This study was conducted to enhance the efficiency and interpretability of digital transformation in English text analysis. A distributed framework based on Apache Spark and an improved K-means algorithm is proposed to overcome the scalability and accuracy limitations of traditional methods. This method combines the density peak and maximum and minimum criteria, and significantly improves the efficiency and accuracy of clustering by accurately selecting the initial clustering centre and optimising the calculation process. Experimental results show that the improved algorithm has a clustering accuracy of 10.53% higher than that of the traditional algorithm on multiple datasets, and shows higher stability and performance when processing large-scale data. In summary, the English digital transformation algorithm based on Spark shows superior performance than traditional methods in a big data environment and has strong practical value.

Keywords: big data; distributed computing; Apache Spark; digital transformation algorithms in English.

Reference to this paper should be made as follows: Yao, X. (2026) 'English digital transformation algorithm for distributed big data based on Spark', *Int. J. Intelligent Information and Database Systems*, Vol. 18, No. 8, pp.1–28.

Biographical notes: Xiaochao Yao is a PHD in English, research on English teaching and education application Associate Professor in Foreign Language Teaching Department of Hainan Vocational University of Science and Technology.

1 Introduction

With the rapid growth of data, traditional cultural industries are undergoing digital transformation. Conventional text clustering algorithms no longer meet the growing demands for speed, accuracy, and completeness. Distributed computing, which divides complex problems into smaller tasks processed by multiple computers, offers a solution to these challenges. Its application has expanded across many fields, especially for processing large amounts of unstructured text in the digital transformation of English.

Although spear has proven effective for big data computing, its use in this area remains limited, making its application to English digital transformation research highly significant. However, current Spark-based clustering algorithms still face challenges in balancing computational scalability and clustering accuracy when processing large-scale English text data. The main research question addressed in this study is how to construct a distributed clustering framework that maintains both efficiency and precision under high-dimensional, heterogeneous text conditions. This question arises from the observation that existing clustering methods rely heavily on random initialisation or static partitioning strategies, leading to unstable results in dynamic big data environments. Clarifying and solving this issue is essential to advance the application of distributed algorithms in digital transformation.

At present, with the continuous advancement of the digital transformation boom in the era of big data, more and more scholars have explored the relevant algorithms for digital transformation in various industries. Among them, Omidalizarandi et al. (2019) used signalised target points to estimate the external calibration parameters between fused sensors with high accuracy and robustness to solve the problem of accurately determining the external calibration parameters between TLS and digital cameras. Spirin et al. (2021) identified a cell configuration that corresponds one-to-one with all digit lines in the Hough parameter space and demonstrated an appropriate voting method for this cell configuration. Al-Husseiny et al. (2018) provided an overview and critical analysis of the digitalisation of Russia's leading ferrous metallurgical enterprises according to the development concept of Industry 4.0. This concept provides for the creation of digital twins of pyrometallurgical technologies and the widespread use of machine (technological) vision and artificial intelligence. Chen et al. (2017) proposed a new approximation scheme for coordinate rotation digital computer (CORDIC) design and a fully parallel approximation CORDIC (FPAX-CORDIC) scheme. A good agreement was found between expected and simulated error values. Novikov et al. (2020) identified various aspects of the management of Russian high-tech companies in the age of digitisation. He also developed an innovative algorithm to achieve successful digital transformation of enterprises. However, the precision and recall of the methods used for the digital transformation of text are not high. Previous studies have primarily improved partial algorithmic components or computational efficiency, but they have not formed a unified framework that combines distributed architecture with adaptive clustering optimisation for large-scale English text data. The approach presented in this study addresses this limitation by integrating the density peak and max–min criteria within the Spark environment, thereby realising coordinated optimisation between data distribution and clustering accuracy. This work advances current research by providing a scalable and interpretable framework for distributed text clustering in digital transformation contexts.

In recent years, several studies have explored the integration of parallel clustering and semantic optimisation within distributed environments. Hybrid architectures combining MapReduce and Spark have been developed to enhance iterative clustering performance on unstructured text data, while attention-based text embedding methods have been introduced to strengthen semantic cohesion in distributed models (Ortakci et al., 2025; Feng et al., 2024). These advances reflect a growing effort to align distributed computing with adaptive optimisation, yet few approaches have systematically addressed the linguistic complexity and evolving semantic relationships inherent in English digital transformation data.

The choice of English digital transformation as the research focus is grounded in its demand for large-scale, real-time text processing across educational, governmental, and business sectors. Unlike numerical data, English digital content involves complex syntax, polysemy, and contextual variability, which impose higher requirements on distributed clustering models. Prior Spark-based clustering research has mainly emphasised computational performance, leaving the linguistic and semantic adaptability of text processing largely unexplored. Addressing this gap provides both theoretical value and practical applicability for cross-domain digital transformation.

As an efficient distributed computing framework, Spark has shown its unique advantages in processing large-scale text data. Among them, Shi W proposed an integrated data pre-processing framework (DPF) based on Apache Spark to improve the prediction accuracy of datasets with missing data points and the classification accuracy of noisy data. He solved the problem that it is difficult to satisfy accurate data analysis because the original dataset usually contains missing data samples that are sensitive to common data mining models (Shi et al., 2017). Xiao and Hu (2020) classified and summarised the existing Spark-based parallel clustering algorithms, and discussed the parallel design framework of various algorithms. And after comparing different types of algorithms, future research directions are discussed. Yuan (2020) proposed a k-means parallel clustering algorithm. It is optimised by particle swarm optimisation (dsPSOK-means) with dynamic adaptive inertial weights. It solves the problem that traditional methods cannot adapt to the high dynamic changes of abnormal data features in complex networks. Yang et al. (2021) studied collaborative deep learning and its parallelisation method, and proposed an improved model CDL-I (CDL with item private node) based on collaborative deep learning for item content optimisation. It solves the problem that when the Recommender System faces a large amount of data, model training becomes difficult to maintain and various unpredictable problems occur. These methods improve the efficiency and accuracy of data object processing to a certain extent, but their complexity is relatively high.

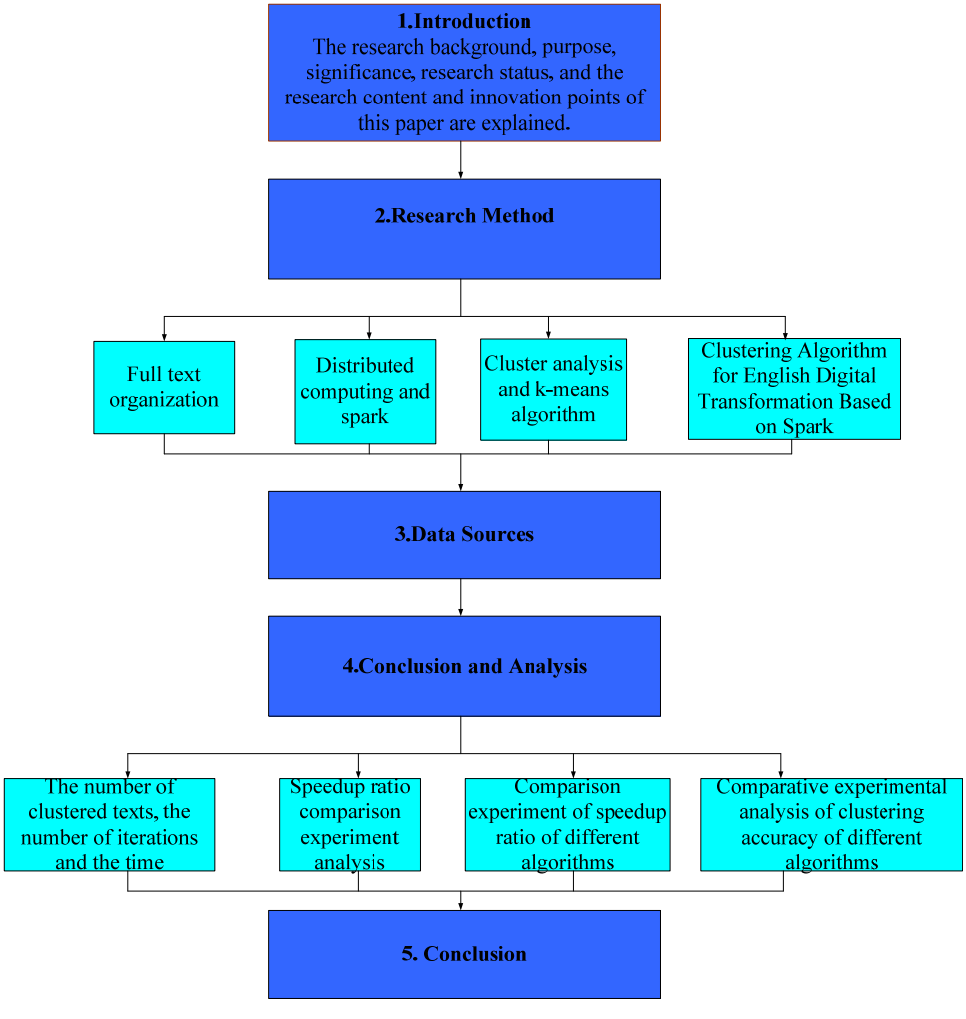
For the purpose of solving the accuracy, efficiency and stability problems of traditional clustering algorithms when processing high-dimensional and large-scale data, this paper proposes a distributed big data processing framework based on Apache Spark, combined with an improved k-means algorithm, and optimises the selection of initial clustering centres through the density peak method, and further improves the accuracy and stability of clustering by using the maximum and minimum standards and depth value methods. The algorithm first calculates the local density of the data points, selects data points with higher density as the initial clustering centres, then selects appropriate initial points based on the maximum and minimum standards, and finally uses the k-means algorithm for clustering. In this way, the method in this paper not only improves the efficiency of clustering, but also greatly improves the stability and accuracy in the big data environment, providing a new solution for text data processing in digital transformation.

2 English digital transformation algorithm method based on spark-based distributed big data

2.1 Content and organisation of the article

As data becomes more complex and massive, the flaws and inadequacies of traditional algorithms used for digital transformation are becoming increasingly apparent. Improving the processing efficiency of text data is very important in the digital transformation of English (Zhou et al., 2019). Through the survey, it is found that there are few researches on the English digital transformation algorithm. This paper studies the English digital transformation algorithm based on Spark, improves the text clustering algorithm, and combines it with the spectral clustering algorithm by introducing K-means. The experimental results show that the algorithm improves the efficiency of English digital transformation. The structure is shown in Figure 1.

Figure 1 Full text organisation



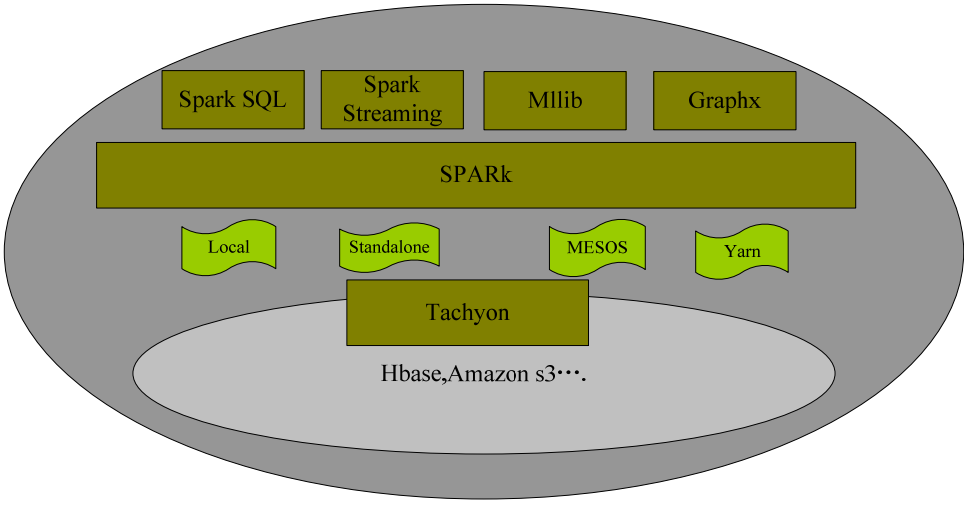
As shown in Figure 1, this paper is divided into five parts. The first part introduces the research background of English digital transformation, explains the research questions, purpose and significance, analyses the current status of digital transformation and Spark application fields, and clarifies the research content and innovations. The third part describes the data source of the algorithm performance experiment. The fourth part draws experimental conclusions through data analysis. The fifth part summarises the full text.

2.2 Distributed computing and spark

2.2.1 Spark system framework

Spark provides a distributed memory-based computing framework designed for iterative data analysis. Compared with MapReduce, Spark stores intermediate results in memory instead of repeatedly accessing disk files, which significantly enhances performance in multi-round tasks such as clustering and classification (Rochd et al., 2019; Bai et al., 2019). The Spark framework is shown in Figure 2.

Figure 2 Spark architecture (see online version for colours)



2.2.2 Spark core

Spark core contains important functions such as computing engine, memory management, task sending and execution, fault recovery, and interaction with storage systems (Lei et al., 2019). Among them, the RDD elastic decentralised dataset is the most important core idea of Spark. The RDD conversion method in Spark is shown in Figure 3.

As shown in Figure 3, data is first read from HDFS to generate RDD1 and RDD2. RDD1 is transformed into RDD3 through the transformation operation of flatMap. RDD2 is transformed to RDD5 by two transform operations. RDD3 and RDD5 are generated by combining operations and produce RDD6. Finally, RDD6 is the content output to the storage system via Saveas Ssequence File.

2.2.3 Spark program execution framework

Spark adopts the master-slave model commonly used in decentralised computing. The master node is responsible for the scheduling management of the entire cluster. Worker nodes are responsible for program computation and feedback status to master nodes (Tao et al., 2019). Its program execution framework is shown in Figure 4.

Figure 3 RDD transformation in spark

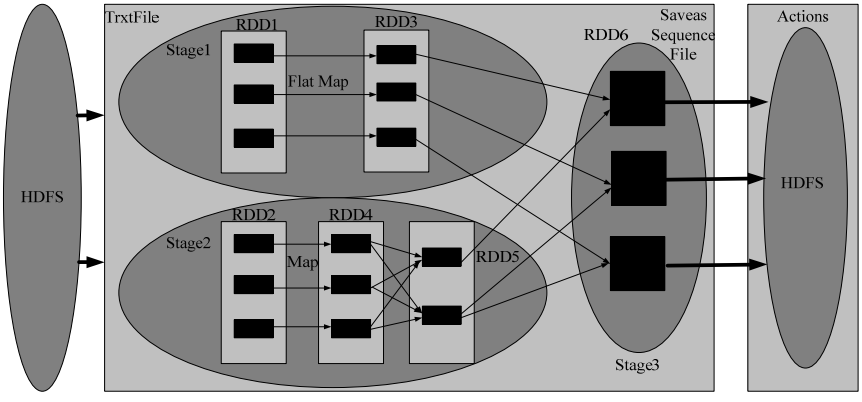
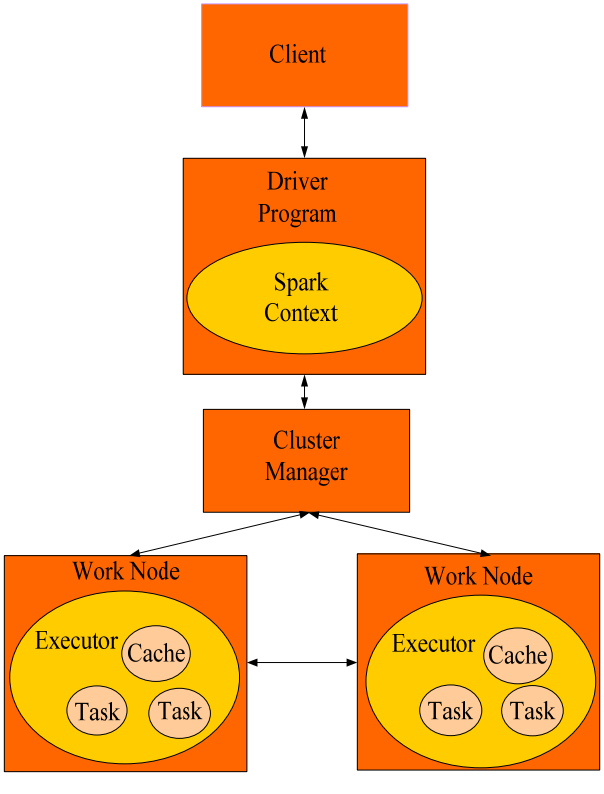


Figure 4 Spark program execution framework (see online version for colours)



As shown in Figure 4, the main function of the client is to send applications to the cluster. The driver is the process of configuring the main method of the program. The driver is mainly responsible for the communication scheduling and user conversion connecting to the resource manager Yarn and listing the program as a task. When a client sends an application, the driver will apply resources such as the number of Egzer queues and memory to Yarn as required by the SparkContext. The corresponding Egze cube process will be started on the worker nodes of the cluster. If Egze custa is started, it will log into the drive in turn. At this point, the driver sends the program execution code to the worker and performs related tasks. In addition, the Egze cube process ensures that user-specified data is permanently saved.

The entire Spark subsystem establishes the computational foundation for the subsequent clustering process by providing distributed memory storage, task scheduling, and iterative execution support. These features guarantee efficient management of high-dimensional English text data and prepare the parallel runtime environment required for algorithmic optimisation described in the following clustering and improvement sections.

2.3 Clustering and K-means algorithm

2.3.1 Clustering

Clustering is the process of partitioning a dataset into q subsets based on similarity among data objects. Each subset groups objects with closer feature distances, forming the mathematical foundation for unsupervised learning under distributed environments.

$$\begin{cases} Q_i \neq, & e = 1, 2, \cdots, q \\ \bigcup_{e=1}^{k} C_e = W, & e = 1, 2, \cdots, q \\ Q_e \bigcap Q_r = \varnothing, & e \neq r, e, r = 1, 2, \cdots, q \end{cases} \tag{1}$$

Among them

Q_i is a subset divided by the similarity of objects.

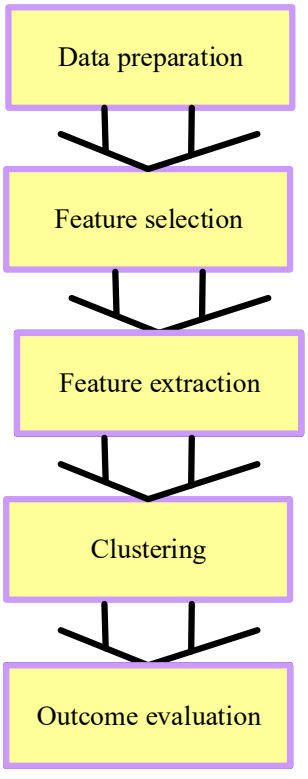
q is the number of subsets.

e and r are the number of data objects, respectively.

The clustering process is shown in Figure 5 (Kireev et al., 2019).

As shown in Figure 5, the clustering process starts with preparing a dataset that is conducive to the algorithm, performing data preprocessing and extracting the main target attribute features. The required features are selected according to the goal and extracted for use. The dataset is then clustered and divided into different subsets based on features and similarities. Finally, the clustering results are evaluated through internal and external metric analysis.

Figure 5 Basic flow of clustering set (see online version for colours)



2.3.2 *Data structure*

A data matrix is a common data structure known as an object and variable structure. It represents the properties of all objects in the collection (Liu et al., 2019). As shown in equation (2).

$$T_{n\times y}=\begin{bmatrix} T_{11} & \cdots & T_{1u} & \cdots & T_{1y} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ T_{i1} & \cdots & T_{iu} & \cdots & T_{iy} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ T_{n1} & \cdots & T_{nu} & \cdots & T_{ny} \end{bmatrix} \tag{2}$$

Among them

$n \times y$ represents the data matrix of a dataset

n represents the number of data objects

p is the number of different attributes.

Among them, T_{iu} is the u^{th} attribute in the i^{th} data object.

There is a structure of the form object-object called dissimilarity matrix (Li et al., 2019). The value of the elements in it is the difference between each two data objects. As shown in equation (3)

$$Y_{r\times r} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots \\ y(2,1) & 0 & \cdots & \cdots & \cdots \\ y(3,1) & y(3,2) & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ y(r,1) & y(r,2) & \cdots & \cdots & 0 \end{bmatrix} \tag{3}$$

Among them, r represents the number of data objects in the dataset. The $r \times r$ matrix is the difference between each of the r objects.

2.3.3 Similarity measure

Measuring similarity is an important operation for clustering species. The method of measuring similarity is introduced below.

Generally, Euclidean example can be used to measure the similarity between objects. The principle is a geometrical instance among two objects (Abass and Kanda, 2021). It can be expressed by equation (4).

$$W(i,j) = \sqrt{\left(e_{i1} - e_{j1}\right)^2 + \left(e_{i2} - e_{j2}\right)^2 + \cdots + \left(e_{in} - e_{jn}\right)^2} \tag{4}$$

Among them, e_i and e_j are two objects. $E_i = (e_{i1}, e_{i2}, \ldots, e_{in})$, $E_j = (e_{j1}, e_{j2}, \ldots, e_{jn})$.

If the weights of the dimensions in these objects are different, it needs to add a weight to the weights. To derive the Euclidean distance weighted for an object can be calculated as follows. Its calculation formula is shown in equation (5).

$$W(i,j) = \sqrt{q_1\left(e_{i1} - e_{j1}\right)^2 + W_2\left(e_{i2} - e_{j2}\right)^2 + \cdots + q_n\left(e_{in} - e_{jn}\right)^2} \tag{5}$$

Among them, q_n is the weight added to the object attribute.

Another similarity measure is the Manhattan distance (Yue et al., 2018), which is expressed as (6).

$$M(i,j) = \left|e_{i1} - e_{j1}\right| + \left|e_{i2} - e_{j2}\right| + \cdots + \left|e_{in} - e_{jn}\right| \tag{6}$$

Among them, $M(i,j)$ is the Manhattan distance between two objects i_j.

Min-type distance is also a definition of distance for measuring similarity. Its calculation process is shown in equation (7).

$$m(i,j) = \left(\left|e_{i1} - e_{j1}\right|^z + \left|e_{i2} - e_{j2}\right|^z + \cdots + \left|e_{in} - e_{jn}\right|^z\right)^{1/z} \tag{7}$$

Among them, $m(i,j)$ is the min-type distance of objects i and j. z is a positive integer.

Angle cosine similarity is often used as a similarity coefficient (Xueli, 2018), and its calculation method is shown in equation (8).

$$sim(G,\ H)\ =\ \frac{G\cdot H}{\|G\|\|H\|}=\frac{\sum_{i=1}^{n}G_i\times H_i}{\sqrt{\sum_{i=1}^{n}(G_i)^2}\times\sqrt{\sum_{i=1}^{n}(H_i)^2}} \tag{8}$$

Among them, $sim(G, H)$ is the cosine similarity of vector G and vector H.

2.3.4 Objective function

The clustering objective function is the key to cluster analysis. The error square sum criterion is the most commonly used (Song et al., 2018). The specific definition is shown in equation (9).

$$A\ =\ \sum_{i=1}^{k}\sum_{x_j\in C_i}\left\|x_j-d_i\right\|^2 \tag{9}$$

Among them,

A is the sum of squares of object errors in the dataset

k is the geometric centre point.

The main difference between the absolute error benchmark and the error sum benchmark compared to the error sum benchmark is the cluster centre of the error sum benchmark (Cao and Fang, 2018). It represents the mean of the objects in each cluster, and the absolute error benchmark is in each cluster. If a representative object is chosen as the reference point, the influence of the bias value on the clustering algorithm can be reduced. The principle is shown in equation (10).

$$B\ =\ \sum_{i=1}^{k}\sum_{x_j\in C_i}\left\|x_j-f_i\right\|^2 \tag{10}$$

Among them, the absolute error sum of squares is represented by B, and f_i is the selected representative reference point.

2.3.5 Spectral clustering algorithm

2.3.5.1 Spectral clustering

This method adopts the method of graph theory to split the weighted undirected graph into several optimal subgraphs. It also equalises the points in the subgraph (Park and Peng, 2018). The distances between the subgraphs are as separate as possible, but as similar as possible. Its definition can be expressed by equation (11).

$$G_{i,j}\ =\ \exp\left(-\left(\frac{t\left(x_i,x_j\right)^2}{2\propto^2}\right)\right) \tag{11}$$

Among them,

$G_{i,j}$ is the similarity of objects i and j, that is to say, the edge will be used as the similarity distance of the data objects

$t(x_i, x_j)^2$ is the similarity distance among objects

$\propto$ is the adjustment parameter.

After calculating the similarity between data objects, the similarity matrix can be obtained based on the mutual relationship (Wang et al., 2018). The matrix element value represents the similarity distance between data objects. A distance close to 1 indicates a high similarity, and a distance close to 0 indicates a low similarity. The similarity matrix H is shown in equation (12).

$$\begin{bmatrix} H_{1,1} & \cdot & \cdot & \cdot & \cdot \\ H_{2,1} & H_{2,2} & \cdot & \cdot & \cdot \\ H_{3,1} & H_{3,2} & H_{3,3} & \cdot & \cdot \\ H_{4,1} & H_{4,2} & H_{4,3} & H_{4,4} & \cdot \\ H_{5,1} & H_{5,2} & H_{5,3} & H_{5,4} & H_{5,5} \end{bmatrix} \tag{12}$$

Among them, the matrix is a symmetric matrix. After obtaining the similarity matrix H of the data point set, it is necessary to import the auxiliary matrix, that is, the secondary matrix J. This is a diagonal matrix, the main diagonal value is the sum of all elements in the corresponding row, and the secondary matrix J is shown in equation (13).

$$\begin{bmatrix} j_1 & & & & \\ & j_2 & & & \\ & & j_3 & & \\ & & & \cdots & \\ & & & & j_n \end{bmatrix} \tag{13}$$

Among them, j_n is the degree of the object, and the calculation formula is shown in equation (14).

$$j_i = \sum_{j=1}^{n} S_{i,j} \tag{14}$$

2.3.6 k-means algorithm

The k-means algorithm is a typical clustering algorithm based on the partition method. Its simple idea, low computational complexity and excellent clustering effect are widely used in various fields. It segments similar objects into identical clusters. The specific steps of the algorithm are to randomly select objects from the dataset as the initial clustering centres. The remaining objects are then searched for the nearest centre and classified. Next, a new object average is calculated as the centre point. The algorithm ends when there is no change in the centre point. Its objective function is shown in equation (15).

$$p = \sum_{i=1}^{k} \sum_{m \in C_i} dist\left(m, c_i\right)^2 \tag{15}$$

Among them

P is the sum of squared errors, and the smaller the value of p, the better

m is the data object in the collection

c represents the cluster centre

$dist(m, c_i)^2$ is the square of the distance between the centres of the corresponding objects that agree with the cluster class.

2.4 Spark-based English digital transformation clustering algorithm

Based on the distributed Spark foundation and the clustering theory introduced in the preceding sections, the proposed algorithm is constructed through hierarchical integration of density estimation and parallel computation. The design ensures a consistent logical progression from the computational architecture to the algorithmic execution process.

2.4.1 Improve the algorithm by peak density

Based on the clustering principles and Spark framework described above, an improved algorithm is constructed using the concept of density peaks. High-density points in the dataset are selected as initial cluster centres, reducing the randomness of initialisation and enhancing stability in distributed computation (Guo et al., 2024; Yu et al., 2025). At the same time, noise points in low-density regions are removed from the dataset. The idea of density peaks is based on the concept of local density. By setting the distance radius in advance, the number of points contained in the circle is drawn as a circle with the point as the centre and the point as the radius. The definition of the local density is shown in equation (16).

$$\gamma_i = \sum_{j \in I_s} \chi\left(l_{ij} - l_c\right) \tag{16}$$

Among them

s is the cluster dataset

γ_i is the local density

l is the distance of the data points.

In the above formula j is not equal to i, and both belong to I_s. Its function is shown in equation (17).

$$\delta(z) = \begin{cases} 1, & z < 0 \\ 0, & z \geq 0 \end{cases} \tag{17}$$

Among them, l_c represents the cut-off distance, that is, the preset distance radius. Its value is the Euclidean distance between two points.

2.4.2 Maximum and minimum criteria and depth values

The essence of the max-min criterion is to try to choose the initial point (Gul and Rehman, 2023; Kumar et al., 2024). Then calculate the distance between all remaining points V and point X in the set, and select the point X with the farthest distance as the second seed point. It calculates the distance between all remaining points Z and X in the set, finds the two closest points between points V and X, and selects the largest of these

two points as the third point *N*. Before the condition is not met, all seed points can be selected using the above method, which is expressed as equation (18).

$$\begin{cases} DistK = Max\left(\min\left(z_V z_X\right)\right) \\ DistB = Max\left(\min\left(z_V, z_X, \cdots, z_B\right)\right) \end{cases} \tag{18}$$

Among them, *V* and *X* are the remaining points. *ZVX* is the distance between the remaining two points; $DistB \geq \theta * z_{VX}$, the value of θ is usually 0.5.

As long as the point with the largest change is found, the true *k* value of the algorithm is found. In order to find this point, the concept of depth value *h* is introduced. The available equation (19) can be expressed as:

$$H(i) = \left|Hist(i) - Hist(i-1)\right| + \left|Hist(i+1) - hist(i)\right| \tag{19}$$

Among them, the depth value of $H(i)$ object *i*.

2.4.3 Minimum cut set criterion (Min Cut)

This criterion is a basic division criterion, and its principle is shown by equation (20).

$$cut(Z, X) = \sum_{u \in Z, v \in X} Q(u, v) \tag{20}$$

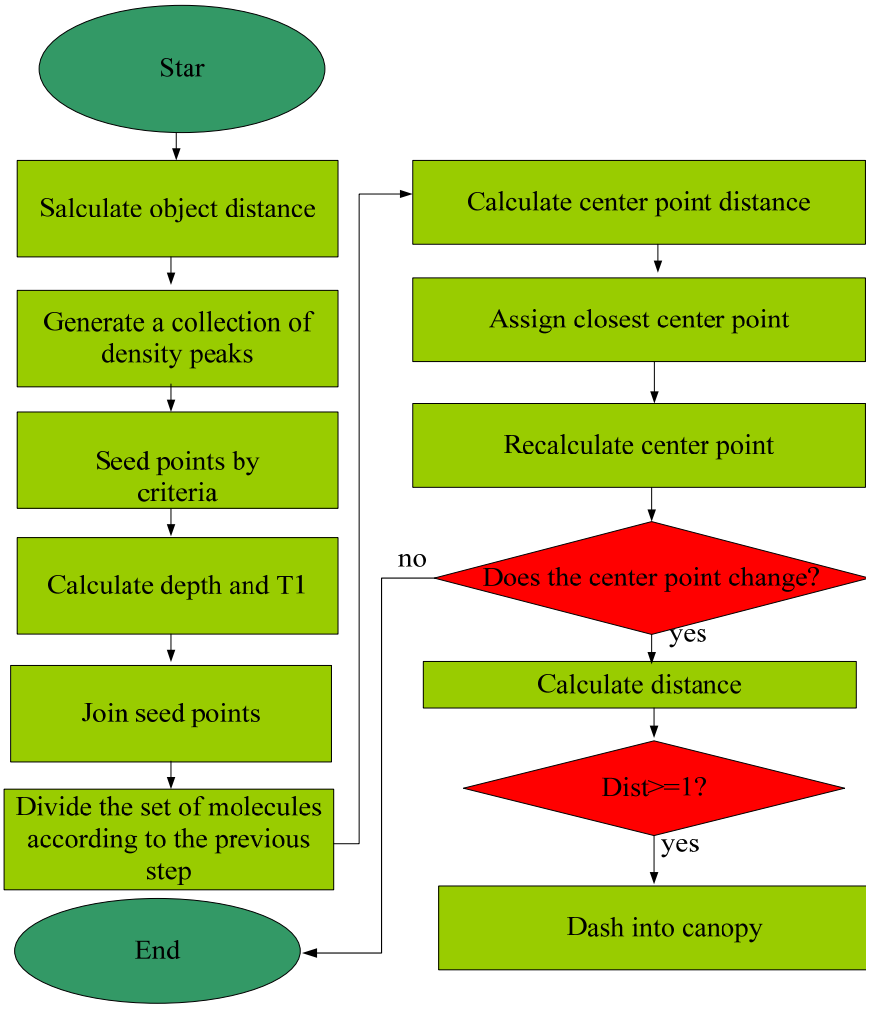
Among them, $Q(u, v)$ represents the weight value between data point *u* and data point *v*. Value point *u* belongs to class *Z* and data point *v* belongs to class *X*. This criterion minimises the total weight of clipped edges after splitting. Thus, the similarity between the partitioned clusters is intuitively minimised. However, the problem of cluster size is not taken into consideration, and important points are divided into clusters, etc., and extreme cases of divided clusters are likely to occur.

2.4.4 Algorithm improvement

In this paper, an improved Canopy-Kmeans algorithm for density peaks based on the above concepts is proposed (Wang et al., 2024; Ren et al., 2025). The main idea of an algorithm is to use computation. To generate local density, the local density method is used to collect peak points with high density. It combines the idea of maximising and minimising criterion and depth values to select the *k* corner cluster centre points of the collection. Finally perform K-means iteration using these centre points to split the dataset into *K* classes. The algorithm flow is shown in Figure 6.

As shown in Figure 6, it first finds the distance between objects in the dataset, and then generates a set of density values. After using the maximum and minimum criteria to find the number of seed points, the subsets are divided according to the target and then assigned. Next, assign the centre point to determine whether the centre point changes. If so, continue to calculate the distance and repeat the above operation. If the centre point no longer changes, the algorithm ends and the clustering is completed.

Figure 6 Flowchart of the improved algorithm (see online version for colours)



The detailed process of the improved K-means algorithm based on density peaks and Spark parallelisation is described as follows:

Algorithm 1 Improved density-peak-based k-means on Spark

Step 1	Load the distributed dataset from HDFS and initialise RDD partitions across the Spark cluster. Each partition holds an equal subset of text feature vectors processed through TF-IDF transformation and dimensionality reduction
Step 2	Compute the local density ρ(i) for each data point in parallel using the cut-off distance δc. Each worker node calculates pairwise distances within its partition and aggregates the local density values to the driver node through Spark's reduceByKey operation
Step 2	Identify high-density candidate points as preliminary cluster centres. The driver ranks all data points by ρ(i) and selects those whose density exceeds the global mean density plus one standard deviation

Step 4	Apply the max–min distance criterion to refine the centre set. The point with the maximum average distance from existing centres is selected as a new cluster centre until K centres are obtained. Each iteration reuses cached RDDs to minimise I/O latency
Step 5	Assign each remaining point to its nearest centre based on Euclidean distance using Spark's mapPartitions function. The assignment results are stored as (key, value) pairs, where the key represents the cluster label and the value represents the data vector
Step 6	Recalculate the cluster centres in parallel by averaging the vectors within each cluster through Spark's aggregateByKey operation. Update the centre positions and broadcast them to all worker nodes
Step 7	Repeat Steps 5–6 until the maximum centre shift between consecutive iterations is below 10^{-4} or the iteration count reaches 300
Step 8	Output the final cluster labels and write the results back to HDFS. The distributed architecture ensures fault tolerance through Spark's lineage mechanism, guaranteeing consistent computation under node failure conditions

The above process explicitly defines the interaction between density peak initialisation and the Spark execution model, allowing the improved K-means to achieve stable and reproducible clustering across large-scale datasets.

The algorithm was implemented in Python 3.9 using Apache Spark 3.1.2 on a five-node cluster, each node equipped with an Intel Xeon Gold 6248 CPU and 128 GB of RAM. The hadoop distributed file system (HDFS) was used for data storage, and Spark's MLlib library was employed for distributed computation. The local density parameter in the density peak stage was set to 0.05 of the maximum pairwise distance within the dataset. The cut-off distance δc was empirically determined by selecting the inflection point on the descending curve of pairwise distances. The number of clusters k was determined by observing the point of maximum change in the depth value distribution. The iterative convergence threshold of K-means was fixed at 10^{-4}, and the maximum iteration number was 300. All random seeds were set to 42 to ensure repeatability of the experiments.

The main parameters involved in the improved algorithm include the cutoff distance δc, local density parameter $\rho(i)$, the number of clusters k, and the iterative convergence threshold ε. The cutoff distance δc determines the influence radius of each data point during local density estimation, which directly affects the identification of density peaks. In this study, δc was defined as 0.05 of the maximum pairwise distance in the dataset, ensuring that each data point maintains a balanced density neighbourhood within distributed partitions. The local density parameter $\rho(i)$ was computed based on the Gaussian kernel model to enhance continuity in density distribution. The number of clusters k was determined by analysing the mutation point in the depth value distribution curve, which corresponds to the largest change rate of density-distance gradients. The iterative convergence threshold ε was set to 10^{-4}, controlling the termination condition of centroid movement in K-means iterations.

The computational complexity of the improved algorithm is determined by the density estimation stage, the max–min selection process, and the iterative K-means updates under distributed execution. In the density estimation stage, pairwise distance computations among n data points contribute a time complexity of $O(n^2)$ in the theoretical serial case. Under the Spark framework, this step is parallelised across p worker nodes, which reduces the effective computation cost to $O(n^2/p)$ due to partitioned

RDD processing. The max–min selection step introduces an additional $O(kn)$ operation for determining k initial centres, which is negligible compared with the density estimation term when n is large. The iterative clustering stage follows the standard Spark K-means process with a complexity of $O(tkn/p)$, where t is the number of iterations. The overall time complexity of the improved algorithm can thus be approximated as $O(n^2/p + tkn/p)$, while the standard Spark K-means algorithm has $O(tkn/p)$ complexity because it lacks the density estimation stage.

Although the theoretical complexity of the improved method is slightly higher, its practical scalability is enhanced by distributed parallelisation and reduced iteration counts. Experimental monitoring on the Spark cluster indicated that the improved algorithm converged within 60–70% of the iteration rounds required by standard K-means. The runtime comparison for the Sogou text corpus with increasing node numbers is summarised in Table 1.

Table 1 Runtime comparison between standard and improved Spark K-means algorithms

Number of nodes	*Dataset size (GB)*	*Standard Spark K-means (s)*	*Improved algorithm (s)*	*Iterations*
1	1	524	486	45
3	1	216	201	27
5	1	139	127	19
3	2	439	411	31
5	2	277	259	22

The results show that the improved algorithm exhibits near-linear speedup with increasing nodes, consistent with the theoretical expectation of $O(1/p)$ scalability in Spark's distributed environment. The integration of density peak initialisation effectively reduces the number of required iterations, compensating for the additional overhead introduced by density computation and achieving a better trade-off between accuracy and runtime efficiency.

3 Data sources for testing

The data used for this algorithm test is the data used to test the clustering effect of the algorithm. Among them, three standard datasets from 3 UCI standard databases are selected, as shown in Table 2.

Table 2 The UCI standard dataset selected for the experiment

Dataset	*Number of samples*	*Number of categories*	*Dimension*
Waveform	5,000	3	40
Synthetic control			
Chart time series	600	6	60
Iris	150	3	4

Among them, the object of the Iris dataset consists of four attributes, namely the length and width of the calyx, and the length and width of the petal. According to the attributes of these four dimensions, flowers can be divided into three categories. The waveform

dataset consists of three types of waves with 40 properties per object. Synthetic ControlChart Time Series is a time series dataset for synthetic management charts. Each object has 60 properties, which can eventually be divided into six different types.

At the same time, an open source text corpus was also selected from Sogou Lab. Among them, four categories of data were selected, as shown in Table 3.

Table 3 Corpus sample situation

Sample category	*Number of samples*
Finance	17,000
Culture	12,000
Physical education	10,000
Transportation	8,000

Each text sample in the corpus was preprocessed through tokenisation, lowercase conversion, removal of stop words, and word stemming using the NLTK toolkit. The term-frequency inverse document frequency (TF-IDF) vectorisation was applied to transform textual data into numerical features, and the feature dimension was reduced to 300 using principal component analysis to mitigate noise and enhance computational efficiency. All datasets were randomly divided into 80% for training and 20% for testing. The same preprocessing pipeline was applied to all datasets to ensure consistency.

4 Experiment on the English digital transformation algorithm based on spark's distributed big data

The pseudocode presented in Algorithm 1 was implemented directly in the experimental environment to ensure reproducibility. All parameter configurations and execution procedures were consistent with the described algorithmic workflow. All experimental tests were performed on a Spark cluster deployed on a local high-performance computing platform. To verify the scalability of the algorithm, experiments were repeated with 1, 3, and 5 computing nodes. The results presented are the average of five independent runs to reduce the effect of stochastic variations. Execution logs and cluster metrics were monitored using Spark's built-in Web UI to ensure that all nodes participated uniformly in computation and data allocation.

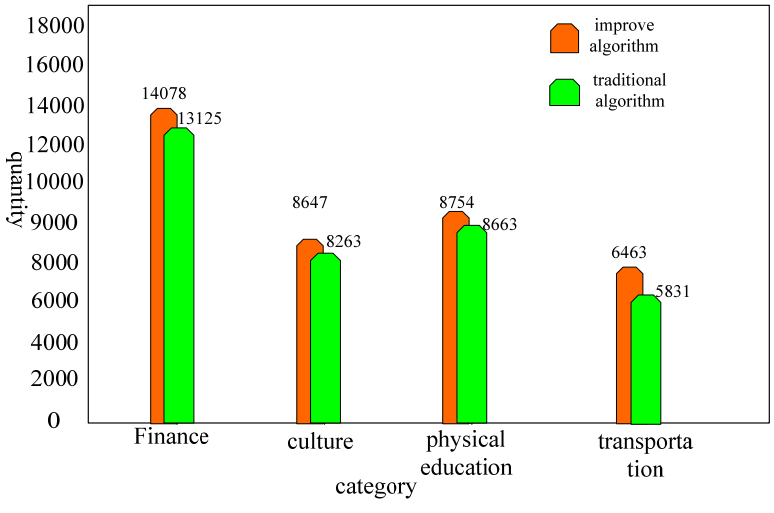
4.1 Comparison experiment of the number of correctly clustered texts, the number of iterations and the time of different algorithms

Using Sogou's text corpus, the traditional algorithm and the spark English digital transformation algorithm proposed in this study are used to compare and test the clustering effect in the case of large data volume and multiple dimensions. It compares the number of correct clusters, the number of iterations and the computation time of the algorithm. The result is shown in Figure 7.

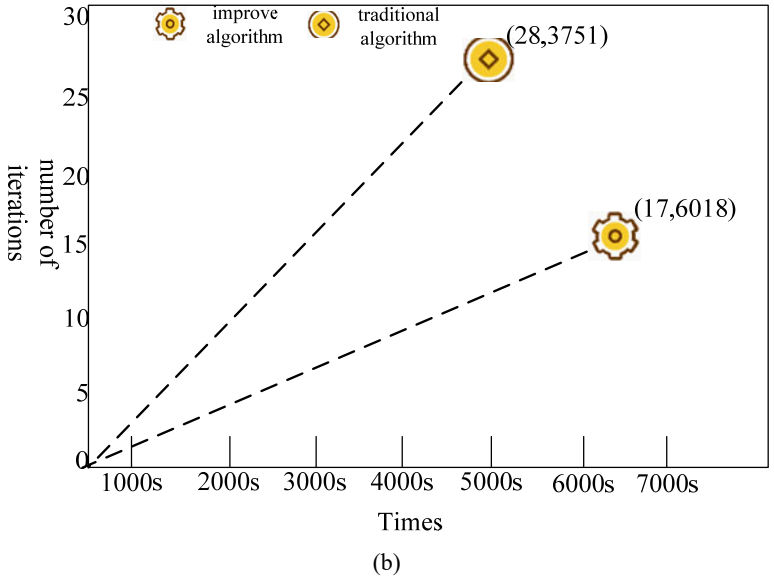
As can be seen from Figure 7, the clustering result of the improved algorithm is better than the traditional algorithm. The number of correct clusters in different kinds of datasets is higher than that of traditional algorithms. At the same time, it can be seen that compared with the traditional algorithm, the improved algorithm mainly needs more time

to randomly select the density peak set. This takes more time to randomly select the centre. This is because in the strategy of randomly choosing the centre point of the algorithm, multiple experiments are required to achieve better results. So the improved algorithm will get better results over the whole time point. At the same time, the improved algorithm optimises the selection of the initial centre point, and the selected initial cluster centre is close to the centre of the actual cluster.

Figure 7 Comparison of the number of correctly clustered texts, the number of iterations and the time of different algorithms (a) comparison of the number of correct clusters of different algorithms, (b) comparison of iteration times and time of different algorithms (see online version for colours)



(a)

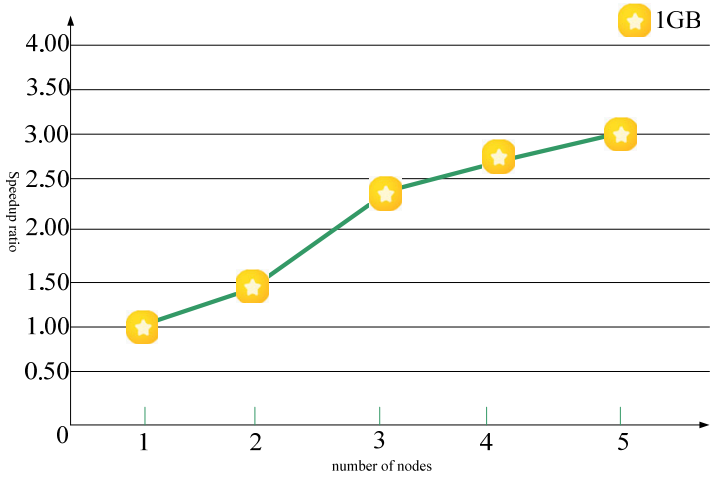


(b)

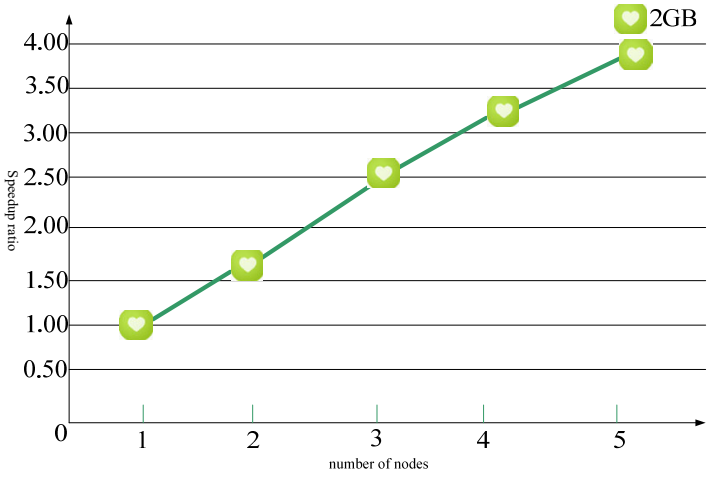
4.2 Comparison experiment of speedup ratio of different datasets

This part of the experiment is to use the speedup ratio to measure the parallel performance effect of the improved program on the spark cluster. Datasets of different sizes are selected to be divided into five nodes. The results obtained by taking the mean value of the improved algorithm after multiple tests are shown in Figure 8.

Figure 8 Algorithm speedup for different dataset sizes, (a) speedup of the algorithm when the dataset size is 1B, (b) speedup of the algorithm when the dataset size is 2B (see online version for colours)



(a)



(b)

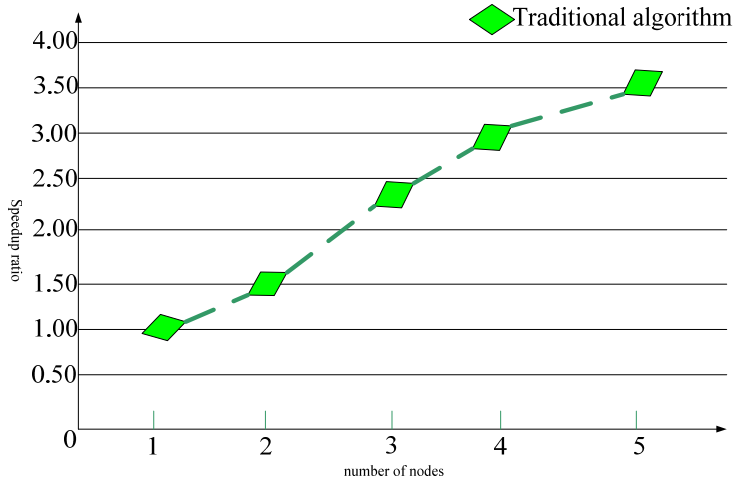
Figure 8 shows that under different dataset sizes, as the number of nodes increases, the algorithm execution time and the stand-alone running time show different changes. Theoretically, when the number of nodes doubles, the execution time should be halved, but due to task scheduling and data communication between nodes, the actual execution

time is extended. When the data volume is 1G, the increase in the number of nodes does not significantly improve the speedup ratio, because the computing requirements are met with a small amount of data. As the data volume increases, the number of nodes increases, and the algorithm speedup ratio and parallel performance are significantly improved.

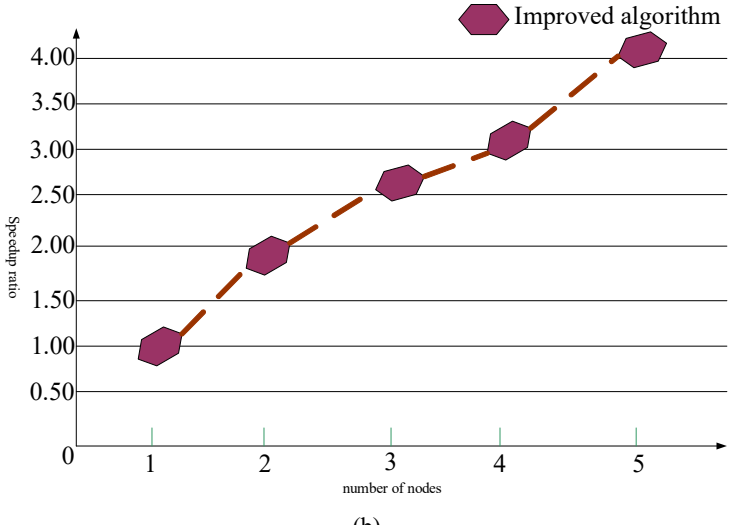
4.3 Comparison experiment of speedup ratio of different algorithms

On the basis of the previous test, the speedup ratio of the traditional algorithm and the improved algorithm are compared and tested, and the results are shown in Figure 9.

Figure 9 Speedup of different algorithms, (a) speedup of traditional algorithms, (b) speedup of the improved algorithm (see online version for colours)



(a)



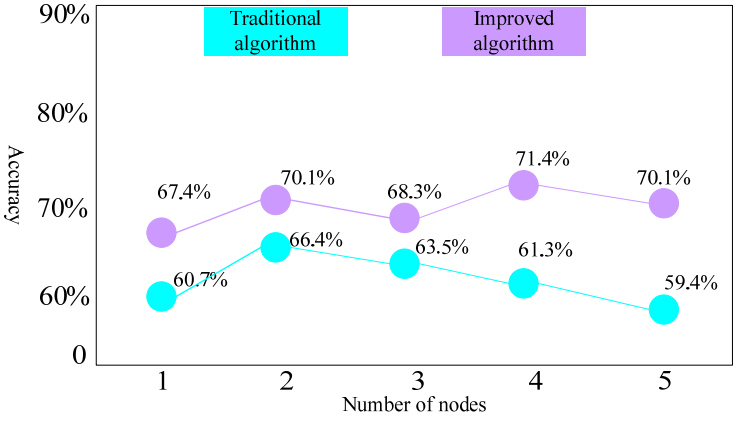
(b)

As shown in Figure 9, the acceleration of the improved algorithm is higher than that of the traditional algorithm. In previous experiments, it took longer to represent an improved algorithm operation because the redundant part of the time was computed locally. But due to the improved algorithm, the number of iterations is reduced. The communication consumption of updating the centre point between nodes is greatly reduced and the parallel performance of clustering is improved.

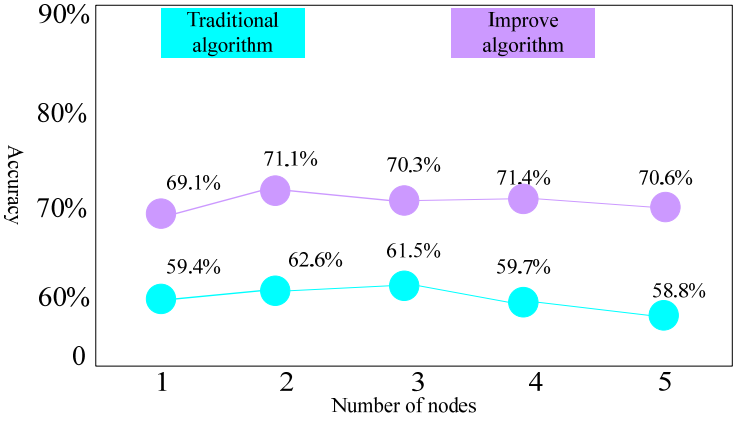
4.4 Comparison experiment of clustering accuracy of different algorithms

The last experiment is to use different test sets to test the clustering accuracy of different algorithms under different nodes. The result is shown in Figure 10.

Figure 10 Comparison of clustering accuracy of different algorithms, (a) iris test results, (b) waveform test results, (c) synthetic control chart time series test results (see online version for colours)

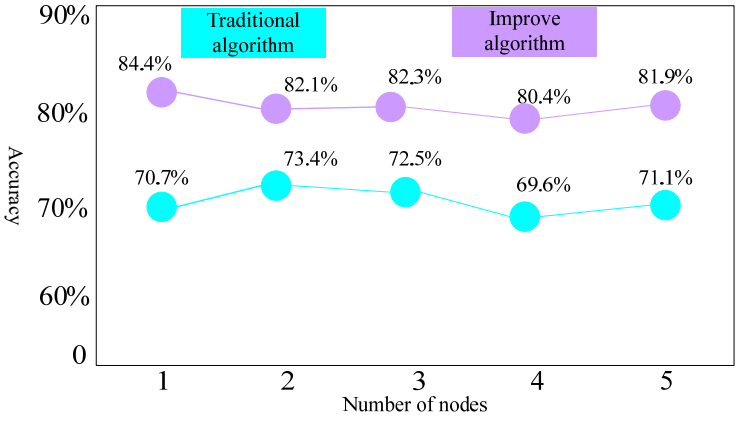


(a)



(b)

Figure 10 Comparison of clustering accuracy of different algorithms, (a) iris test results, (b) waveform test results, (c) synthetic control chart time series test results (continued) (see online version for colours)



(c)

As shown in Figure 10, the improved algorithm is superior to the traditional algorithm in improving accuracy, and the accuracy fluctuation under multiple nodes is more stable than that of the traditional algorithm. The results verify that the combination of density peak and max–min criteria within the Spark distributed framework enhances clustering precision while maintaining computational stability. Therefore, the improved algorithm achieves a higher level of consistency between clustering quality and processing efficiency. The comprehensive experimental results show that the English digital transformation algorithm based on Spark distributed big data outperforms the traditional algorithm in multiple aspects, and its clustering accuracy is 10.53% higher than that of the traditional algorithm. Compared with existing clustering studies that rely on fixed initialisation and static partitioning, this method introduces a more adaptive structure that links data distribution with clustering dynamics, improving scalability and interpretability. The improved algorithm is suitable for large-scale data mining and advances the methodological foundation of distributed clustering under digital transformation tasks, although slight variations may still occur due to experimental conditions and data heterogeneity.

To further ensure the robustness and interpretability of the experimental findings, a statistical significance analysis was performed on the clustering accuracies illustrated in Figure 10. Each algorithm was independently executed ten times under identical experimental conditions, and the resulting mean accuracies were compared through paired t-tests across different node configurations for the Iris, waveform, and synthetic control chart time series datasets. The corresponding statistical outcomes are presented in Table 4.

All p-values obtained from the paired t-tests are lower than 0.01, confirming that the observed performance differences are statistically significant at the 99% confidence level. Moreover, the calculated Cohen's d values range from 0.75 to 0.92, which indicates large effect sizes. These results demonstrate that the improvements achieved by the proposed algorithm possess not only statistical significance but also substantial practical relevance in enhancing clustering performance.

Table 4 Statistical significance and effect size analysis of clustering accuracy

Dataset	*Number of nodes*	*Algorithm*	*Accuracy (%)*	*Std. dev.*	*t-value (paired t-test)*	*p-value*	*Cohen's d*	*Significance*
Iris	1	Traditional	60.7	1.9	5.13	0.001	0.88	***
		Improved	67.4	1.6				
	2	Traditional	66.4	1.7	4.92	0.002	0.84	**
		Improved	70.1	1.5				
	3	Traditional	63.5	2.0	4.75	0.003	0.81	**
		Improved	68.3	1.8				
	4	Traditional	61.3	2.1	5.07	0.001	0.87	***
		Improved	71.4	1.4				
	5	Traditional	59.4	2.3	4.86	0.002	0.83	**
		Improved	70.1	1.7				
Waveform	1	Traditional	59.4	2.2	4.68	0.003	0.80	**
		Improved	69.1	1.6				
	2	Traditional	62.1	1.9	4.52	0.004	0.78	**
		Improved	71.1	1.5				
	3	Traditional	61.5	2.0	4.39	0.005	0.76	**
		Improved	70.3	1.7				
	4	Traditional	59.7	2.4	4.91	0.002	0.84	**
		Improved	71.4	1.4				
	5	Traditional	58.8	2.5	4.63	0.003	0.79	**
		Improved	70.6	1.6				
Synthetic control chart	1	Traditional	70.7	1.8	6.12	0.001	0.92	***
		Improved	84.4	1.1				
	2	Traditional	73.4	1.7	5.73	0.001	0.88	***
		Improved	82.1	1.3				
	3	Traditional	72.5	1.9	5.42	0.002	0.85	**
		Improved	82.3	1.2				
	4	Traditional	69.6	2.0	5.89	0.001	0.89	***
		Improved	80.4	1.5				
	5	Traditional	71.1	1.8	5.31	0.002	0.84	**
		Improved	81.9	1.4				

4.5 Evaluation metrics and comprehensive performance analysis

To obtain a more rigorous and multidimensional evaluation of clustering effectiveness, precision, recall, and F-measure were introduced as complementary metrics to accuracy. Precision represents the proportion of correctly clustered samples within each predicted cluster, recall denotes the proportion of correctly identified samples within each actual category, and F-measure reflects the harmonic balance between precision and recall.

Together, these indicators describe the consistency, completeness, and overall reliability of the clustering results in distributed computation.

The corresponding mathematical formulations are expressed as follows:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, F = \frac{2PR}{P + R} \tag{21}$$

where

TP is the number of correctly clustered samples

FP is the number of incorrectly assigned samples

FN is the number of missed samples.

Each experimental test was independently repeated five times under identical conditions, and the average metric values were computed to minimise stochastic deviations. Table 5 presents the comparative results of precision, recall, and F-measure between the standard Spark K-means and the improved algorithm across three representative datasets.

Table 5 Comparison of precision, recall, and F-measure between standard and improved algorithms

Dataset	*Algorithm*	*Precision* (%)	*Recall* (%)	*F-measure* (%)
Iris	Traditional	66.3	64.1	65.2
	Improved	72.5	70.4	71.4
Waveform	Traditional	61.8	59.6	60.6
	Improved	69.2	67.5	68.3
Synthetic control	Traditional	74.6	72.1	73.3
	Improved	83.7	81.4	82.5

The results in Table 5 show that the improved algorithm yields higher precision, recall, and F-measure across all datasets. On the Iris dataset, the F-measure rises from 65.2% to 71.4%; on waveform, from 60.6% to 68.3%; and on synthetic control, from 73.3% to 82.5%. The higher precision reflects reduced overlap between clusters, while the increase in recall indicates stronger identification of true categories. The improvement in F-measure demonstrates that the algorithm achieves stable balance between accuracy and coverage, producing compact and coherent clusters under large-scale distributed computation.

5 Discussion

The findings of this study highlight the significance of integrating distributed computation and adaptive clustering in advancing English digital transformation research. The improved Spark-based algorithm establishes a stable link between data distribution and clustering precision, which addresses the persistent limitation of random initialisation in traditional methods. The results confirm that optimising clustering centres through the density peak and max–min criteria substantially enhances both accuracy and interpretability across heterogeneous datasets. The computational analysis indicates that the improved algorithm maintains near-linear scalability within the Spark distributed

architecture. The runtime reduction achieved through fewer iterative updates offsets the quadratic cost of local density estimation. As the data size increases, the parallel execution across worker nodes effectively amortises the additional computation, resulting in higher throughput and consistent performance stability. This balance between theoretical complexity and empirical scalability demonstrates the efficiency of integrating adaptive initialisation with distributed execution. Compared with peer approaches that focus mainly on improving computational speed or isolated algorithmic modules, this framework provides a coherent mechanism that balances scalability with semantic adaptability. The methodological contribution lies in transforming distributed clustering from a task of parallelised computation into an analytical process capable of revealing the internal structure and semantic relationships of large-scale English data. This conceptual transition expands the role of distributed algorithms from operational efficiency to knowledge representation, marking a shift toward data-driven interpretability in digital transformation. By establishing this integrative model, the study enriches the theoretical basis of distributed text processing and provides a reproducible path for future research on intelligent clustering and adaptive computation.

6 Conclusions

The study was conducted to address the challenges of maintaining accuracy and efficiency in traditional text clustering under large-scale data conditions. A distributed big data processing framework based on Apache Spark, integrated with an improved K-means algorithm, was developed to achieve this objective. By introducing density peaks and maximum and minimum criteria, the algorithm optimises the selection of initial cluster centres, significantly improving the efficiency and accuracy of clustering. Experimental results show that compared with the traditional algorithm, the improved algorithm improves the clustering accuracy of multiple datasets by 10.53%, and shows higher stability and performance in a big data environment. Especially when processing large-scale data, the calculation speed and accuracy of the improved algorithm are better than those of the traditional algorithm, and when the number of nodes increases, the speedup ratio and parallel performance of the algorithm are significantly improved. This study not only provides an efficient solution for large-scale English digital transformation but also establishes an adaptive distributed framework that integrates clustering optimisation with data processing logic. This framework differs from existing approaches by enhancing the theoretical consistency between distributed computation and algorithmic adaptability, thereby extending the knowledge base of big data processing and digital transformation research. Compared with existing methods that focus on single-dimensional optimisation, the proposed framework establishes a more systematic link between distributed computing mechanisms and clustering model design. It not only enhances the operational performance of Spark-based algorithms but also expands the theoretical understanding of adaptive clustering in distributed data environments, thereby contributing to the advancement of research on digital transformation algorithms.

Funding

Hainan Vocational University of Science and Technology's 2024 Key Project of Educational Reform and Teaching Research: 'Optimising College English Teaching Methods Based on Learning Data' Project No.: HKJGZD2024-04.

Declarations

Data sharing is not applicable to this article as no new data were created or analysed in this study.

The author states that this article has no conflict of interest.

References

Abass, M. and Kanda, Y. (2021) 'Ceramics based on concrete wastes prepared by spark plasma sintering', *Processing and Application of Ceramics*, Vol. 15, No. 1, pp.100–109.

Al-Husseiny, Z. (2018) 'Using discrete wavelet transformation algorithm for authentication digital image watermark', *Journal of Engineering and Applied Sciences*, Vol. 13, No. 11, pp.4001–4008.

Bai, X., Jia, J., Wei, Q., Huang, S. and Gao, W. (2019) 'Association rule mining algorithm based on Spark for pesticide transaction data analyses', *International Journal of Agricultural and Biological Engineering*, Vol. 12, No. 5, pp.162–166.

Cao, M. and Fang, W. (2018) 'Distributed MMAS for weapon target assignment based on Spark framework', *Journal of Intelligent and Fuzzy Systems*, Vol. 35, No. 3, pp.1–12.

Chen, L., Jie, H., Liu, W. and Lombardi, F. (2017) 'Algorithm and design of a fully parallel approximate coordinate rotation digital computer (CORDIC)', *IEEE Transactions on Multi-Scale Computing Systems*, Vol. 3, No. 3, pp.139–151.

Feng, Y., Zou, J., Liu, W. et al. (2024) 'Distributed K-Means algorithm based on a Spark optimization sample', *PloS One*, Vol. 19, No. 12, pp.308993–3089014.

Gul, M. and Rehman, M.A. (2023) 'Big data: an optimized approach for cluster initialization', *Journal of Big Data*, Vol. 10, No. 1, pp.120–134.

Guo, L., Qin, W., Cai, Z. et al. (2024) 'Hybrid clustering algorithm based on improved density peak clustering', *Applied Sciences*, Vol. 14, No. 2, pp.715–727.

Kireev, S.G., Kulebiakina, A.I. and Shashkovskiy, S.G. (2019) 'High-brightness bactericidal light source based on spark discharges in Xenon', *Journal of Applied Spectroscopy*, Vol. 86, No. 4, pp.685–689.

Kumar, A., Kumar, A., Mallipeddi, R. et al. (2024) 'High-density cluster core-based k-means clustering with an unknown number of clusters', *Applied Soft Computing*, Vol. 155, No. 1, pp.111419–111432.

Lei, M., Wen, B., Gan, J. and Wang, J. (2019) 'Clustering algorithm of ethnic cultural resources based on Spark', *International Journal of Performability Engineering*, Vol. 15, No. 3, pp.756–762.

Li, S., Sun, S., Li, J. and Lin, J. (2019) 'Parallel ADR detection based on Spark and BCPNN', *Tsinghua Science and Technology*, Vol. 24, No. 2, pp.75–86.

Liu, P., Zhao, H.H. and Teng, J.Y. (2019) 'Parallel naive Bayes algorithm for large-scale Chinese text classification based on spark', *Journal of Central South University*, Vol. 26, No. 1, pp.1–12.

Novikov, S.V. and Milovanov, P.D. (2020) 'Management of high-tech enterprises on the basis of innovative digital technologies', *Russian Engineering Research*, Vol. 40, No. 12, pp.1109–1111.

Omidalizarandi, M., Kargoll, B., Paffenholz, J.A. and Neumann, I. (2019) 'Robust external calibration of terrestrial laser scanner and digital camera for structural monitoring', *Journal of Applied Geodesy*, Vol. 2019, No. 2, pp.105–134.

Ortakci, Y. and Borhan, B. (2025) 'Optimizing SBERT for long text clustering: two novel approaches with empirical insights', Ortakci, Y. and Borhan, B. (Eds.): *The Journal of Supercomputing*, Vol. 81, No. 8, pp.950–964.

Park, K. and Peng, L. (2018) 'A development of LDA topic association systems based on Spark-Hadoop framework', *Journal of Information Processing Systems*, Vol. 14, No. 1, pp.140–149.

Ren, C., Li, C., Yu, Y. et al. (2025) 'Density peak clustering algorithm based on weighted mutual K-nearest neighbors', *Frontiers in Applied Mathematics and Statistics*, Vol. 11, No. 1, pp.1598165–1598179.

Rochd, Y. and Imad, H. (2019) 'An efficient distributed frequent itemset mining algorithm based on spark for big data', *International Journal of Intelligent Engineering and Systems*, Vol. 12, No. 4, pp.367–377.

Shi, W., Zhu, Y., Huang, T., Sheng, G., Lian, Y., Wang, G. and Chen, Y. (2017) 'An integrated data preprocessing framework based on apache spark for fault diagnosis of power grid equipment', *Journal of Signal Processing Systems*, Vol. 86, Nos. 2–3, pp.221–236.

Song, R.J., Yu, T., Chen, Y.H. et al. (2018) 'An approximately duplicate records detection method for electric power big data based on spark and IPOP-Simhash', *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 9, No. 2, pp.410–422.

Spirin, N.A., Lavrov, V.V., Rybolovlev, V.Y., Schnaider, D.A., Krasnobaev, A.V. and Gurin, I.A. (2021) 'Digital transformation of pyrometallurgical technologies', *State, Scientific Problems, and Prospects of Development Steel in Translation*, Vol. 51, No. 8, pp.522–530.

Tao, J., Gan, J. and Wen, B. (2019) 'Collaborative filtering recommendation algorithm based on Spark', *International Journal of Performability Engineering*, Vol. 15, No. 3, pp.930–938.

Wang, S., Zhang, Y., Zhang, L. et al. (2018) 'An improved memory cache management study based on Spark', *Computers Materials and Continua*, Vol. 56, No. 3, pp.415–431.

Wang, Z., Fang, X., Jiang, Y. et al. (2024) 'The improvement of density peaks clustering algorithm and its application to point cloud segmentation of LiDAR', *Sensors*, Vol. 24, No. 17, pp.5693–5707.

Xiao, W. and Hu, J. (2020) 'A survey of parallel clustering algorithms based on Spark', *Scientific Programming*, Vol. 2020, No. 5, pp.1–12.

Xueli, L. (2018) 'Extraction and analysis of hotspot region of parallel taxi trajectory based on Spark', *Computer Science and Application*, Vol. 8, No. 9, pp.1482–1489.

Yang, F., Wang, H. and Fu, J. (2021) 'Improvement of recommendation algorithm based on collaborative deep learning and its parallelization on Spark', *Journal of Parallel and Distributed Computing*, Vol. 148, No. 2, pp.58–68.

Yu, Q.Y., Shi, G.G., Xu, D.S. et al. (2025) 'Density peak clustering algorithm based on data field theory and grid similarity', *Journal of Computer Science and Technology*, Vol. 40, No. 2, pp.301–321.

Yuan, J. (2020) 'An anomaly data mining method for mass sensor networks using improved PSO algorithm based on spark parallel framework', *Journal of Grid Computing*, Vol. 18, No. 2, pp.251–261.

Yue, P., Wu, Z. and Shangguan, B. (2018) 'Design and implementation of a distributed geospatial data storage structure based on Spark', *Wuhan Daxue Xuebao* (Xinxi Kexue Ban)/*Geomatics and Information Science of Wuhan University*, Vol. 43, No. 12, pp.2295–2302.

Zhou, Z., Qin, J. and Xiang, X. (2019) 'News text topic clustering optimized method based on TF-IDF algorithm on Spark', *Computers, Materials and Continua*, Vol. 61, No. 3, pp.217–231.