EasyChinese: Simplifying the Chinese Writing System Using Clustering Algorithms and Ambiguity Metrics

Kai Xu, Advised by Robert Frank

Yale University

Abstract

Chinese is especially difficult to learn to read and write due to its logographic writing system, which requires memorizing thousands of characters, rather than a simpler alphabetic or phonetic system in most major languages. Meanwhile, writing Chinese in purely phonetic notation (e.g., Pinyin, Zhuyin) will result in semantic ambiguity due to many characters sharing the same pronunciation but having different meaning (homophones). In light of these challenges, this paper presents a new notation system for Chinese that represents every character by a combination of phonetic and semantic category, ensuring ease of reading but also allowing for semantic disambiguation. The creation of semantic categories is mostly approached as a clustering problem, where each Chinese character is represented as a vector embedding. Various clustering algorithms are tested, their strengths and weaknesses for this problem analyzed. This paper concludes by proposing novel methods for outlier detection, consensus clustering, and human-assisted clustering that enhance clustering results.

1 Introduction

Chinese is one of the most difficult languages to learn to read and write, largely because it requires memorizing thousands of characters. This characteristic of Chinese makes learning it time-consuming, posing a barrier to many students. Overseas Chinese communities often speak the language fluently due to its use in family settings but struggle with reading and writing due to a lack of engagement with characters in daily life. In contrast, for most other major languages, one can quickly learn to read and write by mastering a concise alphabet or syllabary (e.g., Spanish, English, Korean, etc.). Simplifying the Chinese writing system would not only make the language more accessible but also facilitate written communication among Chinese speakers from different backgrounds.

Phonetic notations for Chinese, such as Pinyin and Zhuyin, are useful tools for teaching the language, but are seldom used for writing. This is partly because many characters have the same pronunciation but different meanings (homophones), and relying solely on phonetic notation can result in semantic confusion. In light of this problem, this paper proposes a

new way of notating Chinese: replacing every character by a combination of pronunciation and semantic disambiguator.

For example, consider the characters for the numbers 1, 4, 9 in Chinese: 一 (yī), 四 (sì), 九 (jiǔ). Now, consider the characters for "clothes," "pants", "skirt": 衣 (yī), 裤 (kù), 裙 (qún). It would make sense for the numbers to belong to the same semantic category, and the garments to be in another semantic category. We can let # denote number-related characters and let ⇑ denote garment-related characters, calling these symbols our semantic disambiguators. The numbers can henceforth be written as #yī, #sì, #jiǔ, and the garments as ⇑yī, ⇑kù, ⇑qún. Even though the characters for "one" and "clothes" share the same pronunciation, they can now be semantically distinguished by the symbol, in a way that notating purely pronounciation could not.[1]

The question of assigning Chinese characters to different semantic categories thus becomes a clustering problem, where each Chinese character is a unique data point.[2] In this study, Chinese characters are primarily represented as word embeddings, although this is by no means the only representation that many of the discussed techniques can be used on. The first main section of this paper evaluates the performance of standard clustering algorithms such as centroid-based, density-based, distribution-based, and hierarchical clustering algorithms for this task. This section also introduces novel outlier removal methods for improving clustering results. The latter main section of this paper introduces methods that incorporate human feedback to improve clustering results, attempting to address the ways in which purely algorithmic approaches may not always reflect human preferences.

2 Pinyin with Grouped Words Significantly Reduces Ambiguity

A commonly mentioned problem of writing Chinese with purely Pinyin is that it can result in a great degree of semantic ambiguity, as multiple Chinese characters can share the same Pinyin. Here, we assess the degree to which this statement is valid. We also find that grouping Pinyin syllables of the same word reduces ambiguity to a considerable extent.

We consider two methods of writing Pinyin: The first is writing Pinyin with spaces between every syllable, so the phrase 我很喜欢冰淇淋 (I really like ice-cream) becomes "wǒ hěn xǐ huān bīng qí lín." The second method is writing Pinyin with grouped words, so the same phrase becomes "wǒ hěn xǐhuān bīngqílín."

Now, suppose we were to replace each space-separated Pinyin segment with the most common character or word with the same Pinyin.[3] We are interested in the percentage of characters or words that are correctly replaced, which we call *confidence*. To answer this question, we use a corpus created by Beijing Language and Culture University containing the frequency of over 1 million Chinese words (Baolin 2019). We also use a corpus created by Jun

[1]Creating symbols for semantic categories and notating pronunciation in a visually appealing way is a worthwhile artistic endeavor. However, this paper is mainly concerned with the information being conveyed.

[2]Beyond language simplification, semantic clustering of Chinese characters may be useful for other purposes such as education and information retrieval.

[3]Due to lack of available data, this study does not consider the effects of polyphones, or characters with multiple pronunciations.

Da containing the frequency of 9933 characters. We find that the confidence when writing Pinyin with spaces between every syllable is 75.3%, which is insufficient for practical semantic disambiguation. However, context provided by preceding and subsequent characters means that the actual percentage of time a reader is expected to correctly interpret a character without confusion is higher than this percentage suggests.

However, adding grouped words when writing Pinyin with grouped words raises the confidence to 94.6%. We can further decompose this value into the confidence of words of different character lengths:

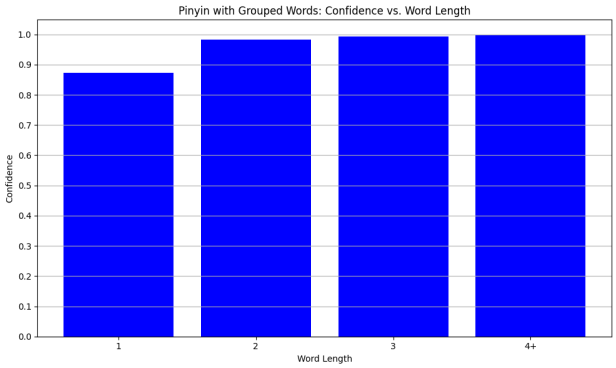

Figure 1: Grouping Pinyin syllables of the same word together significantly increases confidence and reduces ambiguity. The longer the word, the less likely it is to be ambiguous.

With grouped words, even the confidence of single characters is higher than the confidence of writing Pinyin with spaces between every syllable. That is because the tendency of some characters to be found solely or primarily in grouped words leaves less character options for the single-character words. Altogether, grouping words in Pinyin reduces semantic ambiguity significantly, with the actual rate of understanding likely higher than 94.6% due to the context of neighboring words.

3 Adding Radicals to Pinyin Virtually Eliminates Ambiguity

The idea of notating Chinese characters as a combination of combination of semantic category and pronunciation is nothing new—in fact, it forms the structural basis of existing Chinese characters. Many Chinese characters are structurally composed of a semantic component and a phonetic component, also known as a radical. The well-known *Kangxi Dictionary* maps each character to one of 214 radicals.

This raises a question: what if we instead notate each Chinese character as a combination of Kangxi radical and pronunciation? One potential source of ambiguity could be when two characters have the same pronunciation and radical. To quantify this effect, we apply the same framework: First, we assume that every Chinese character was replaced by Pinyin and semantic disambiguator with word grouping. So for example, the phrase 我很喜欢冰淇淋 becomes "戈wǒ 彳hěn 口xǐ欠huān 冫bīng水qí水lín." Now, if we were to replace each

space-separated word in this new notation with the most common character combination with that radical-pinyin combination, our confidence is the percentage of characters that are correctly replaced.

Using radical data from hanziDB (*HanziDB* 2023), we find that overall confidence is 99.7%, making Pinyin and radical with grouped words a very semantically unambiguous writing system. Even the radical may be redundant if a given word has only one interpretation. For instance, both the words "xīhuān" and "bīngqílín" have one only one character interpretation, hence radicals are unnecessary when writing them. While the words "wǒ" and "hěn" correspond to several characters, the most common character with those pronunciations are much more common than the rest. In such cases, the radical can be excluded unless referring to a less common character with the same Pinyin. Hence, Pinyin is almost always enough, and radicals should be added if Pinyin alone is ambiguous.

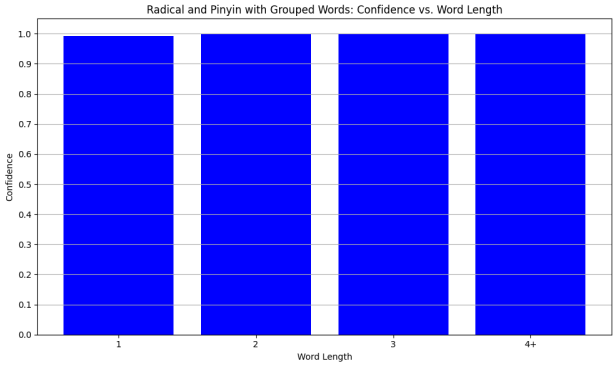

Figure 2: Confidence is extremely high when notating Chinese using Pinyin and radical with grouped words.

4 Creating Semantic Categories with Clustering Algorithms

For unsupervised clustering of Chinese characters, we represent each Chinese character as a 200-dimensional word embedding, as pre-trained by the Tencent AI Lab using the directional skip-gram model (Tencent AI Lab 2021). Word embeddings are normalized to unit length before applying clustering algorithms; this is achieved by dividing each word embedding vector by its magnitude.

There are several unique considerations for semantic clustering of Chinese characters. Semantic similarity of characters within a cluster should be high, and semantic similarity of characters across clusters should be low. To quantify how well a character i fits into its own cluster as opposed to other clusters, we can use the silhouette score $s(i)$, defined as follows:

$$s(i) = \frac{a(i) - b(i)}{\max(a(i), b(i))}$$

where $a(i)$ is the mean cosine distance (defined as 1 minus the cosine of the angle between two vectors) of character i to all other characters within its cluster, and $b(i)$ is the smallest

mean cosine distance of character i to all characters in any other cluster (Rousseeuw 1987).

The silhouette score measures how well an individual character fits into its cluster. To assess overall clustering quality, let us define two metrics: the first is the mean silhouette score of all characters. The second is the mean silhouette score of all clusters, where the silhouette score of a cluster is defined as the mean silhouette score of characters within that cluster. We can further examine the distribution of cluster silhouette scores, checking for consistency.

Another consideration with the semantic clustering of Chinese characters is that cluster sizes should be fairly similar. To quantify this effect, we can examine the variance of cluster sizes, along with other measures of spread.

Subsequently, we assess the performance of DBSCAN, K-means, Gaussian mixture models, and agglomerative clustering for the semantic categorization of Chinese characters, using the sklearn Python library (Pedregosa et al. 2011).

4.1 Baseline: Kangxi Radicals

Even though adding radicals to Pinyin virtually eliminates ambiguities, radicals do not provide the most meaningful semantic categorization of Chinese characters. If we treat all the characters of a particular radical as a cluster, we find that both silhouette scores across characters and clusters are low, with significant inconsistencies in cluster size. This is unsurprising, given the lack of sophisticated data analysis techniques when Chinese characters were created, and the shift in the meaning of characters over time.

Radical-based Clustering

	Character Silhouette	Radical Silhouette	Radical Size
count	7542	212	212
mean	−0.176	−0.176	35.575
std	0.11	0.102	65.293
min	−0.845	−0.432	1
25%	−0.244	−0.235	4
50%	−0.175	−0.198	10
75%	−0.111	−0.136	29
max	0.495	0.31	412

Figure 3: Summary of results when clustering Chinese characters by their Kangxi radicals.

The lack of semantic similarity in the radicals is revealed in the numbers from 1 to 10: 一二三四五六七八九十. All of these characters are very similar semantically, yet we find that they are split among 7 different radicals. With the exception of DBSCAN and single-linkage clustering, all the other tested clustering algorithms provided improved semantic clusterings.

4.2 DBSCAN

DBSCAN is a density-based clustering algorithm that assigns points in densely connected regions into the same cluster, marking the points that are far from these densely connected regions as outliers (Ester et al. 1996). DBSCAN works with any distance metric; since we are working with word embeddings, we use cosine distance. DBSCAN takes two parameters: One of them is the minimum number of samples in a cluster, which we set equal to 2 to ensure no cluster has only one character without making further assumptions about cluster

size. The other parameter, ϵ, determines how closely points must be to each other to be considered part of the same cluster. We adjust ϵ to be equal to 0.1775 to generate 214 clusters, the same as the number of Kangxi radicals.

DBSCAN, Outliers Excluded

	Character Silhouette	Cluster Silhouette	Cluster Size
count	876	214	214
mean	0.332	0.506	4.093
std	0.285	0.188	9.957
min	−0.478	−0.156	2
25%	0.116	0.412	2
50%	0.36	0.528	2
75%	0.558	0.632	3
max	0.916	0.914	102

Figure 4: Summary of DBSCAN results.

Applying DBSCAN, we find the vast majority of characters are classified as outliers. Furthermore, the number of characters per cluster is extremely positively skewed, with the majority of clusters having only 2 or 3 characters but the largest cluster having 102. The larger clusters tend to have lower silhouette scores, likely because DBSCAN is density-based, allowing characters with low semantic similarity can still be in the same cluster if they are connected by a dense region of characters in the vector space. Altogether, DBSCAN yields suboptimal results if used as the sole algorithm for semantic clustering of Chinese characters.

However, that is not to say that DBSCAN is altogether useless for this purpose. The small clusters created by DBSCAN often contain very closely related characters, such as 北南西东 (cardinal directions) or 低高 (low, high). This could be helpful for identifying "inseparable" character subgroups, which could be used to evaluate other clustering metrics.

4.3 K-Means

K-means is a clustering algorithm that represents each cluster by a centroid. Each data point is assigned to the cluster with the closest centroid in terms of Euclidean distance. The goal is to minimize the sum of squared Euclidean distances between each point and the centroid of its cluster (MacQueen et al. 1967). The number of clusters is a parameter that we set equal to 214 to facilitate comparison with Kangxi radical clusters.

K-Means, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	0.036	0.046	47.944
std	0.097	0.083	28.025
min	−0.245	−0.111	5
25%	−0.03	−0.009	27
50%	0.023	0.037	40.5
75%	0.087	0.091	61.75
max	0.562	0.378	177

Figure 5: Summary of k-means results.

Silhouette scores with k-means are generally quite high across both individual characters and clusters. Cluster sizes are also generally fairly uniform and reasonable despite the outliers. Visual examination of the clusters reveals a moderately high degree of intra-cluster semantic similarity. For example, consider the cluster with a comparatively high silhouette

score of 0.134 that includes the following characters: 四五六七十三八二初月年第九半一几末节旧女两期零日历笫男天新注. Most of these terms are number-related, but some are semantically distinct, such as 日月 (calendar dates), 男女 (genders), or 新 (new). It is plausible that these terms often co-occur with numerical characters, but they are still very semantically distinguishable. Altogether, k-means sheds light on relevant semantic categories that Chinese characters can be categorized under, but its results still require considerable manual tweaking before being considered usable.

One can optimize the number of clusters, at least theoretically, by plot the mean character silhouette score against the number of clusters while running k-means with varying cluster counts. An optimal cluster count should yield a higher silhouette score than its neighbors. To test this concept, we apply k-means to a smaller set of embeddings for only the characters in the HSK levels 1 to 6 (Hanban/Confucius Institute Headquarters 2023), considering cluster sizes ranging from 2 to 150.

Confidence is very high at 99.6% overall, suggesting that k-means clustering provides very semantically unambiguous categories. This can be broken down into 99.1% for one-character words, 99.9% for two-character words, 99.9% for three-character words, and 100.0% for words with four or more characters.

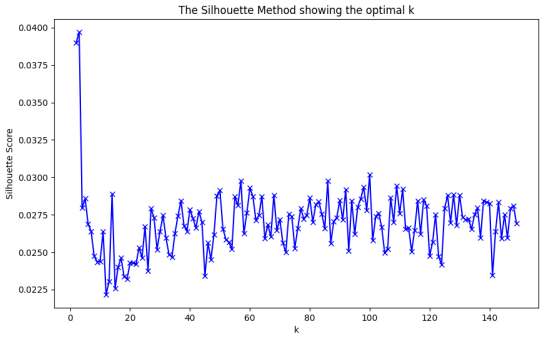

Figure 6: Plotting the mean silhouette score against the number of clusters shows that some cluster counts should strictly be preferred to neighboring values.

4.4 Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering framework where individual clusters are merged to form larger clusters within a hierarchy. The decision on which clusters to merge is determined by a dissimilarity measure between pairs of clusters (Murtagh and Contreras 2012). Different linkage criteria can be applied, resulting in various hierarchical clustering algorithms. The four linkage criteria tested in this study are “single,” “complete,” “average,” and “Ward.”

Single-Linkage Clustering, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	−0.177	0.003	47.944
std	0.071	0.029	686.25
min	−0.462	−0.181	1
25%	−0.219	0	1
50%	−0.174	0	1
75%	−0.131	0	1
max	0.248	0.241	10040

Average-Linkage Clustering, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	−0.003	0.089	47.944
std	0.099	0.099	317.969
min	−0.322	−0.045	1
25%	−0.064	0.004	2
50%	−0.015	0.065	3
75%	0.044	0.132	11.75
max	0.618	0.536	4193

Complete-Linkage Clustering, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	0.016	0.024	47.944
std	0.108	0.08	71.462
min	−0.456	−0.094	1
25%	−0.057	−0.036	12.25
50%	0.004	0.007	26.5
75%	0.081	0.06	51.75
max	0.614	0.458	713

Ward-Linkage Clustering, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	0.004	0.028	47.944
std	0.125	0.117	29.258
min	−0.446	−0.174	7
25%	−0.08	−0.052	26.25
50%	−0.012	0.006	40
75%	0.074	0.078	59.75
max	0.721	0.594	167

Figure 7: Summary of hierarchical clustering results.

Of the linkage criteria tested, the "single" criterion produced the worst results, with almost all clusters containing only one character. This is because the single-linkage approach tends to create elongated chains of data points without considering cluster size and similarity across all points within a cluster. Consequently, larger clusters are more likely to accumulate additional points, creating to a "rich get richer" scenario. The "average" linkage criterion also performed poorly, with substantial variation in cluster sizes and low intra-cluster similarity. In contrast, the "complete" and "Ward" linkage criteria both yielded better results, with complete-linkage generating slightly higher silhouette scores but more variability in cluster sizes compared to Ward-linkage. The improved performance of these algorithms can be attributed to the fact that Ward-linkage minimizes the variance in distances among points within a cluster, and complete linkage applies stringent criteria for adding a new point to a cluster, ensuring it is similar to even the most dissimilar point in the cluster.

Confidence is very high at 99.5% overall with Ward hierarchical clustering. This can be broken down into 98.8% for one-character words, 99.9% for two-character words, 99.9% for three-character words, and 100.0% for words with four or more characters.

4.5 Gaussian Mixture Model

Gaussian mixture model (GMM) clustering is an algorithm that represents each cluster as a multivariate Gaussian distribution. A point belongs to the cluster with the Gaussian distribution that the point has the greatest probability of being generated from (Reynolds et al. 2009). One can control the number of clusters as a parameter; we set it equal to 214 to facilitate comparisons with Kangxi radical clusters. One can also specify the covariance parameters of the mixture components; we allow each component to have a full covariance matrix, and different components to have different covariance matrices.

GMM, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	0.029	0.039	47.944
std	0.097	0.086	24.913
min	−0.262	−0.106	8
25%	−0.034	−0.013	31.25
50%	0.016	0.025	42
75%	0.076	0.072	57.75
max	0.554	0.436	161

Figure 8: Summary of Gaussian mixture model results.

When applied to Chinese character embeddings, Gaussian mixture model clustering yields remarkably similar cluster sizes. Silhouette scores for both individual characters and clusters are slightly lower than for k-means. However, simply looking at silhouette scores may be reductive—further examination of the results shows that most of the clusters have semantic similarities comparable to clusters generated by k-means.

Confidence is very high at 99.7% overall, suggesting that GMM clustering provides very semantically unambiguous categories. This can be broken down into 99.2% for one-character words, 99.9% for two-character words, 99.9% for three-character words, and 100.0% for words with four or more characters.

5 Improving Clustering Results

This section discusses methods to improve the results of clustering algorithms for both the specific task of Chinese character classification, as well as broader clustering applications. Methods include outlier detection before and after clustering, as well as consensus clustering, where the results of multiple clustering iterations are leveraged to create more representative clusters.

5.1 Pre-Clustering Outlier Detection

Many clustering algorithms provide a measure of how well a point fits into a cluster. In k-means, this is determined by distance to a cluster's centroid, while in Gaussian mixture models, it is determined by the probability of a point belonging to a cluster based on the cluster's Gaussian distribution parameters. As measured in this way, some points may be deemed outliers based on poor fit to their respective cluster. Other metrics such as silhouette score can also measure how well a point fits into its respective cluster.

However, here we explore the idea of creating a formula that identifies outliers before clustering. Outlier status is determined by proximity to other points, where a lower proximity to other points implies a greater outlier status. We aim to quantify proximity to other points as follows:

Let n denote the number of points in our dataset minus one. For some point in the dataset, let $d_1, d_2, \ldots, d_n \in \mathbb{R}_0^+$ denote the distance of that point to every other point in the dataset, as measured by some distance measure such as Euclidean distance, cosine distance, etc. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function of these distances. We call f a *monotonic outlier measure* if for any $i \in \{1, 2, \ldots, n\}$, $f(d_1, \ldots, d_{i-1}, d_i', d_{i+1}, \ldots, d_n) \geq f(d_1, \ldots, d_{i-1}, d_i, d_{i+1}, \ldots, d_n)$ holds for all $d_i' > d_i$. A point is considered an outlier if f exceeds some threshold $t \in \mathbb{R}$.

We find that many functions are monotonic outlier measure, potentially serving as useful determinants of a point's outlier status, with some examples as follows:

- Distance to the nearest point, or to the k-nearest points.
- Distance to the furthest point, or to the k-furthest points.
- Mean distance to all other points, or any norm of the distances to all other points.
- Any polynomial function of distances that does not have negative coefficients.

There are many more examples of valid monotonic outlier measures. We propose a novel monotonic outlier measure that gives increasing weight to closer points and geometrically decreasing weight to further points.

Let $d'_1, d'_2, \ldots, d'_n$ denote the distances $d_1, d_2, \ldots, d_n$ arranged in ascending order. Let $r \in [0, \infty)$ be a parameter that we can vary. We define the *geometric ranked outlier score* as the following:

$$\frac{\sum_{i=1}^{n} r^{i-1} d'_i}{\sum_{i=1}^{n} r^{i-1}}$$

By adjusting r, we can decide whether to give more weight to the distances of closer or further points when determining outlier status. If r equals 0, this formula returns the distance of the closest point. If r equals ∞, it returns the distance of the furthest point. And if r equals 1, it returns the mean of the distances to all other points.

Geometric Ranked Outlier Score

r = 0		r = 0.5		r = 1		r = 2		r = ∞	
Hi-scores	Lo-scores	Hi-scores	Lo-scores	Hi-scores	Lo-scores	Hi-scores	Lo-scores	Hi-scores	Lo-scores
蝘 0.544	呀 0.021	讯 0.549	五 0.058	尬 0.808	青 0.584	尬 1.06	巴 0.807	所 1.069	贝 0.809
讯 0.539	啊 0.021	蝘 0.549	啊 0.062	蝘 0.791	烈 0.583	所 1.039	贝 0.808	尬 1.069	巴 0.811
摄 0.532	么 0.024	摄 0.542	四 0.064	曝 0.787	崇 0.582	并 1.027	昂 0.81	并 1.063	狂 0.814
疫 0.523	吗 0.024	饰 0.537	呀 0.065	贷 0.785	伯 0.582	等 1.017	绝 0.811	对 1.041	绝 0.815
睹 0.51	五 0.049	疫 0.534	六 0.068	章 0.78	阳 0.581	及 1.014	狂 0.811	等 1.04	什 0.817
供 0.51	六 0.049	睹 0.521	七 0.068	稿 0.777	华 0.58	蝘 1.008	括 0.812	趴 1.04	括 0.818
诞 0.506	哎 0.049	供 0.521	三 0.072	饰 0.776	童 0.578	髦 1.007	什 0.814	及 1.029	昂 0.819
髦 0.506	唉 0.049	髦 0.514	八 0.079	钙 0.774	立 0.578	对 1.004	藏 0.815	赔 1.026	昏 0.82
饰 0.505	二 0.053	继 0.513	二 0.103	骡 0.772	尤 0.578	唠 1.002	然 0.816	蝘 1.026	然 0.823
轨 0.501	三 0.053	诞 0.512	南 0.105	辑 0.772	毕 0.577	瞩 0.998	昏 0.818	髦 1.022	格 0.827
鲷 0.498	四 0.054	轨 0.511	九 0.107	髦 0.772	乞 0.577	奠 0.995	介 0.819	订 1.021	隐 0.827
瘾 0.497	男 0.054	瘾 0.51	么 0.108	考 0.771	云 0.576	办 0.995	隐 0.82	污 1.021	藏 0.829
厢 0.496	女 0.054	厢 0.508	阿 0.109	播 0.771	典 0.575	诞 0.994	沙 0.823	其 1.02	涛 0.829
辐 0.489	象 0.057	鲷 0.507	北 0.11	睹 0.766	平 0.575	均 0.993	典 0.824	正 1.019	乐 0.829
拟 0.486	像 0.057	矩 0.506	十 0.112	癌 0.765	明 0.571	娱 0.993	格 0.824	办 1.019	典 0.83
仿 0.486	北 0.058	辐 0.505	呢 0.117	弊 0.765	空 0.571	侃 0.992	百 0.824	奠 1.019	乞 0.831
继 0.485	南 0.058	拟 0.502	吗 0.124	债 0.764	然 0.567	其 0.992	疯 0.824	城 1.015	介 0.831
著 0.484	七 0.059	濒 0.499	孙 0.125	题 0.763	绝 0.565	污 0.992	乞 0.825	份 1.015	食 0.831
投 0.484	痛 0.062	著 0.493	哎 0.125	煎 0.763	乌 0.561	著 0.991	迷 0.826	界 1.014	疯 0.832
濒 0.482	疼 0.062	妇 0.491	西 0.126	娱 0.762	隐 0.561	赔 0.99	桑 0.826	唠 1.011	沧 0.833

Figure 9: Geometric ranked outlier score applied to 2666 characters in HSK 1 to 6. Changing the value of r changes the characters with the greatest and least outlier status.

Applying the geometric ranked outlier score to the embeddings of 2,666 characters in HSK levels 1 to 6, using cosine distance as the metric, yields insightful patterns. At lower

values of r, characters with the lowest outlier scores predominantly correspond to numbers, interjections (e.g., 啊, 呀), and cardinal directions, which are straightforward to categorize visually. In contrast, characters with the highest outlier scores often are structurally complex and visually challenging to categorize. Importantly, results at $r = 1$ are less coherent compared to those at lower r values, indicating that determining an optimal r value between 0 and 1 for a specific dataset is essential.

Removing outliers prior to clustering allows clusters to be modeled on a set of less ambiguous "core" points, resulting in more meaningful clusters for certain applications. After the clusters are created, outliers can be reassigned to these clusters of core points.

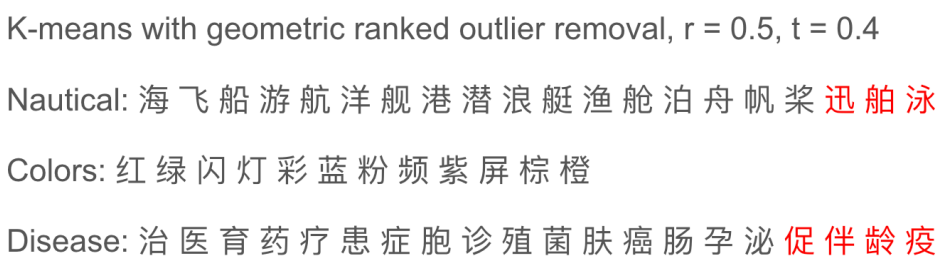

Figure 10: Example clusters from running k-means with outlier removal, setting r to 0.5 and our outlier threshold t to 0.4. Outliers are marked in red.

When applying this approach to the k-means algorithm, outlier points, which are less closely related to other points, are not used to determine the positions of cluster centroids. For example, in the nautical cluster, the outlier 泳 (swim) is somewhat related to nautical terms but not as closely as most other non-outlier characters like 船 (boat) and 海 (ocean). In contrast, some clusters, such as the one for colors, have no outlier terms, as all characters within these clusters are quite cohesively related.

5.2 Consensus Clustering and Consensus Outlier Detection

Say we run multiple clustering algorithms, or run a non-deterministic clustering algorithm multiple times. We can use this information across different clusterings to provide us with better overall clusterings, i.e., consensus clustering. This information can also give us insight on outliers, as well as the degree of confidence in certain clusterings versus others.

Let m denote the total number of clusterings, and n denote the number of points. Let $I_{ij,k}$ be an indicator variable that equals 1 if points i and j are assigned to the same cluster by the k-th clustering, and 0 if they are assigned to different clusters by the k-th clustering. Let w_k denote the weight we want to give to the k-th clustering, satisfying $w_1 + w_2 + \ldots + w_k = 1$ and $w_k \geq 0$. We define P as a n-by-n matrix such that the following holds:

$$P_{ij} = \sum_{k=1}^{m} w_k I_{ij,k}$$

We note that P is in turn a similarity matrix, and $1 - P$ is a distance matrix. This gives us a world of possibilities. To demonstrate this idea, we run k-means on Chinese character embeddings 100 times with 214 clusters. We give equal weighting to all runs of k-means,

setting w_k equal to $1/n$. This turns P into a matrix representing the probability of characters being in the same cluster during a particular run of k-means.

Same-cluster probabilities:

	一	二	日	月	猫	狗
一	1	0.95	0.39	0.42	0	0
二	0.95	1	0.43	0.46	0	0
日	0.39	0.43	1	0.87	0	0
月	0.42	0.46	0.87	1	0	0
猫	0	0	0	0	1	0.91
狗	0	0	0	0	0.91	1

Figure 11: Pairwise same-cluster probabilities after running k-means on Chinese character embeddings 100 times with 214 clusters. The displayed results is a submatrix of P, showing the relationships between the following characters: 一 (one), 二 (two), 日 (day), 月 (month), 猫 (cat), 狗 (dog). From the probabilities, we observe that the pairs 一 and 二, 日 and 月, and 猫 and 狗 are all closely related. The numbers and times are somewhat related, and the animals are not closely related to any other characters.

One clustering approach is to run a hierarchical clustering algorithm on $1 - P$, which would group points with greater semantic similarity lower in the hierarchy before grouping points with less semantic similarity higher in the hierarchy. We apply complete-linkage hierarchical clustering with 214 clusters to $1 - P$, where P is the pairwise same-cluster probability matrix from from 100 iterations of k-means.

Consensus K-Means via Complete-Linkage, 214 Clusters

	Character Silhouette	Cluster Silhouette	Cluster Size
count	10260	214	214
mean	0.164	0.211	47.944
std	0.272	0.197	181.28
min	-0.589	-0.105	3
25%	-0.027	0.067	18
50%	0.118	0.159	29
75%	0.335	0.282	44
max	0.946	0.909	2658

Figure 12: Results of consensus k-means via complete-linkage hierarchical clustering.

Applying consensus k-means via complete-linkage hierarchical clustering, we notice a problem: the largest cluster has 2658 characters. This occurred because 214 clusters turned out to be too low of a cluster count for complete-linkage hierarchical clustering to ensure consistently sized clusters; a less stringent clustering threshold would alleviate this problem. The next-largest cluster had 192 characters, similar to the largest cluster size for regular k-means. The mean silhouette score for characters and clusters is lower than for k-means, likely due to the skewing effect of the very large cluster, which had a low cluster silhouette score of -0.05. However, we also find that the median, 75th percentile, and maximum silhouette scores across both clusters and characters is significantly greater for consensus k-means via complete-linkage hierarchical clustering than for both regular k-means and complete-linkage hierarchical clustering on cosine distances. This suggests that disregarding the anomalously

large cluster, the consensus-based approach generates more cohesive clusters than both regular k-means and regular complete-linkage hierarchical clustering, improving upon the results of both algorithms.

This is by no means the only way to use the matrix P for clustering. For instance, the GloVe algorithm for generating word embeddings is trained on a word co-occurrence matrix, which the matrix P essentially reflects. Hence, one can train new embeddings on the matrix P and apply clustering algorithms on these new embeddings.

Next, we explore the possibility of using the matrix P to detect outliers. For these purposes, an outlier can be defined as a point that does not get assigned to the same cluster as many other points during clustering. Since $1 - P$ is a pairwise distance matrix, we can apply the principles of monotonic outlier measures to this matrix as we have for pre-clustering outlier detection. We test this idea on the subset of Chinese characters in HSK 1 to 6, defining P as the pairwise same-cluster probability for k-means as measured over 100 iterations. Applying a geometric ranked outlier score on the distance matrix $1 - P$ with a ratio $r = 0.5$, we find that the character with the greatest and least outlier scores as measured in this fashion have a large overlap with those identified from pre-clustering outlier detection with cosine distances.

Geometric Ranked Different-Cluster Outlier Score

r = 0.5

Hi-scores		Lo-scores	
蓄	0.782	四	0.011
触	0.781	黑	0.012
介	0.765	绿	0.012
烟	0.76	猴	0.015
兜	0.753	三	0.015
兼	0.752	蓝	0.016
勤	0.752	紫	0.016
促	0.749	矛	0.016
闭	0.743	橙	0.017
贡	0.743	傅	0.017
串	0.743	剑	0.017
乒	0.741	阿	0.017
缺	0.741	霍	0.018
篮	0.738	白	0.018
疫	0.737	色	0.018
运	0.737	红	0.018
变	0.736	黄	0.018
结	0.734	刀	0.018
巧	0.733	杜	0.019
忌	0.731	九	0.021

Figure 13: Geometric ranked outlier score applied to pairwise different-cluster probabilities of 2666 characters in HSK 1 to 6, derived from 100 iterations of k-means. Note the overlap with pre-clustering outlier detection.

5.3 Human-In-The-Loop Clustering

While convenient, algorithmic clusterings often may not align with human preferences. This is particularly relevant for clustering Chinese characters, as human considerations for semantic similarity may differ from vector embedding similarity measures. At the same time, manual clustering can be tedious and slow, especially with large datasets. Thus, there exists

a demand for a hybrid approach, where humans can evaluate existing clusters, and clustering patterns can be iteratively updated based on this feedback. Humans are good at pattern recognition and can quickly discern which characters do not fit into a particular cluster. They can also quickly discern whether a cluster generally makes sense or not. These abilities should be taken into account when designing a system that integrates human feedback to improve clustering results.

We note a certain class of cluster algorithms—constrained clustering algorithms—that allow the incorporation of must-link and cannot-link constraints between points. Two points in a must-link relationship must be placed in the same cluster, and two points in a cannot-link relationship cannot be placed in the same cluster. If we are working with a clustering algorithm that does not allow constraints, a must-link relationship can be mimicked by increasing the pairwise similarity between two points in the similarity matrix, and a cannot-link relationship can be mimicked by decreasing the pairwise similarity.

We propose a clustering evaluation and improvement mechanism that allows the reviewer to specify a NOT-IN relationship between a character and the rest of the characters in a cluster. We establish cannot-link relationships between each point labeled NOT-IN and all other points in the cluster that are not labeled NOT-IN. This ensures that NOT-IN points are separated from the rest. Additionally, for the remaining points in these clusters (those not labeled NOT-IN), we create must-link relationships between all pairs in these points, indicating that they should be in the same cluster. We repeat the process of creating constraints and regenerating clusters until arriving at a satisfactory final clustering.

We test this approach with hierarchical clustering. We keep our dataset limited to 9 characters: 猫 (cat), 狮 (lion), 虎 (tiger), 狗 (dog), 狼 (wolf), 狐 (fox), 鸡 (chicken), 鸭 (duck), 鷹 (eagle). Our goal is to create three clusters: one for the dog family, one for the cat family, and one for the birds.

Figure 14: Iterative feedback clustering on complete-linkage hierarchical clustering, with red characters denoting NOT-IN.

After three rounds of iterative feedback, the clusterings converged to the intended result. Nevertheless, three feedback iterations is quite high for such a small dataset, and the number of required feedback iterations is likely to increase with dataset size. Future research will likely focus on testing iterative feedback on different clustering algorithms, as well as modeling alternative ways of providing feedback in terms of must-link and cannot-link relationships.

6 Conclusion

Our findings show that simplifying the Chinese writing system is a tractable problem. The existing writing system can be simplified while minimizing ambiguity by writing Pinyin with grouped words and using Kangxi radicals to further disambiguate if necessary. To create semantic disambiguators with greater semantic meaning than Kangxi radicals, we can use clustering algorithms on vector embeddings of Chinese characters. For this task, k-means, Gaussian mixture models, and Ward hierarchical clustering are particularly useful. Clusters can further be improved using a variety of outlier removal techniques, in addition to integrating the results of multiple clustering iterations through consensus clustering. Finally, this study demonstrates the possibility of incorporating human feedback to improve existing clustering results, ensuring that a data analysis problem for human beings can incorporate the feedback of its users.

7 Acknowledgements

I (Kai Xu) extend my sincere gratitude to Professor William Lauenroth for his patient guidance and thoughtful revisions throughout the course of this project.

8 References

Baolin, Zhang (May 31, 2019). *Global Chinese Interlanguage Texts Corpus*. Beijing Language and Culture University Language Resource High-Quality Innovation Center. URL: http://yuyanziyuan.blcu.edu.cn/en/info/1050/1297.htm.

Ester, Martin et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *kdd*. Vol. 96. 34, pp. 226–231.

Hanban/Confucius Institute Headquarters (2023). *Hanyu Shuiping Kaoshi (HSK)*. Standardized Mandarin proficiency test. URL: https://www.chinesetest.cn/.

HanziDB (2023). *Chinese Character Database*. HanziDB. URL: https://www.hanzidb.org (visited on 12/11/2023).

MacQueen, James et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, pp. 281–297.

Murtagh, Fionn and Pedro Contreras (2012). "Algorithms for hierarchical clustering: an overview". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1, pp. 86–97.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Reynolds, Douglas A et al. (2009). "Gaussian mixture models." In: *Encyclopedia of biometrics* 741.659-663.

Rousseeuw, Peter J (1987). "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20, pp. 53–65.

Tencent AI Lab (2021). *Tencent AI Lab Embedding Corpora for Chinese and English Words and Phrases*. URL: https://ai.tencent.com/ailab/nlp/en/embedding.html.