IMPROVEMENT OF CLUSTERING ALGORITHMS BY IMPLEMENTATION OF SPELLING BASED RANKING

Evan Bryer[1], Theppatorn Rhujittawiwat[1], John R. Rose[1] and Colin F. Wilder [2]

[1]*College of Engineering and Computing, University of South Carolina, USA*

[2]*Center for Digital Humanities, University of South Carolina, USA*

ABSTRACT

The goal of this paper is to modify an existing clustering algorithm with the use of the Hunspell spell checker to specialize it for the use of cleaning early modern European book title data. Duplicate and corrupted data is a constant concern for data analysis, and clustering has been identified to be a robust tool for normalizing and cleaning data such as ours. In particular, our data comprises over 5 million books published in European languages between 1500 and 1800 in the Machine-Readable Cataloging (MARC) data format from 17,983 libraries in 123 countries. However, as each library individually catalogued their records, many duplicative and inaccurate records exist in the data set. Additionally, each language evolved over the 300-year period we are studying, and as such many of the words had their spellings altered. Without cleaning and normalizing this data, it would be difficult to find coherent trends, as much of the data may be missed in the query. In previous research, we have identified the use of Prediction by Partial Matching to provide the most increase in base accuracy when applied to dirty data of similar construct to our data set. However, there are many cases in which the correct book title may not be the most common, either when only two values exist in a cluster, or the dirty title exists in more records. In these cases, a language agnostic clustering algorithm would normalize the incorrect title and lower the overall accuracy of the data set. By implementing the Hunspell spell checker into the clustering algorithm, using it to rank clusters by the number of words not found in their dictionary, we can drastically lower the cases of this occurring. Indeed, this ranking algorithm proved to increase the overall accuracy of the clustered data by as much as 25% over the unmodified Prediction by Partial Matching algorithm.

KEYWORDS

Pre-Processing, Clustering, Cleaning, Data Mining, Spellchecking

1. INTRODUCTION

Clustering can be a powerful tool for cleaning repetitive string data. Able to catch many common errors such as typos, word swaps, punctuation mistakes, it is an efficient and robust tool so long as the correct version of your string exists in your data set. However, most of these algorithms will simply normalize the most common member of the cluster onto all values in the cluster. While this is a good approach, as it can often be assumed a misspelling will occur less frequently than the correct version within highly repetitive data, this assumption may not always be true. In cases where it is in fact false, the normalization process further lowers the accuracy of the data set. Furthermore, when there are only two members of a cluster, the algorithm often does little more than guess at which is the proper value. In attempts to improve the ability for clustering to be used for data cleaning, we have implemented a ranking system for values in the cluster based on the number of misspelled words that exist in the string.

Our research team makes use of a large dataset of book catalog records for early modern Europe, approximately six million records in total. This dataset has been created and maintained by the Online Computer Library Center (OCLC). A more detailed history of this organization as well as our relationship with them can be found in our previous paper (Bryer et al, 2021). The data contained within the MARC records (in MARC 21 format) stores the metadata of a book, all of the bibliographic and publishing information without containing any of the actual data from the book. This includes many valuable fields such as book title, author, publisher, publishing location, year of publication, as well as many others. The focus here will be on book title data.

Because this metadata has been collected throughout upwards of 18,000 libraries, it is only natural that many of the records contain duplicative data. Due to the libraries in our data set originating from multiple countries throughout Europe, as well as encompassing books from 1500 to 1800, this duplicative data contains many inconsistencies that make a simple deduplication process difficult. As language progressed over time, the spellings of many words changed, as well as differing between regions of a country. Additionally, a book reprinted in a different country may make some slight changes to the spelling of a title. These factors, along with the human error of typos during digitization, makes for a messy set of data that requires extensive processing to clean. However, this repetition makes clustering a good fit for this data set.

2. RELATED WORKS

The plight for a method to ensure clean data is far from recent in the field of both Computer Science as well as Bibliography. Methods of cleaning data are numerous, as well as those which normalize data. Specifically for bibliographic data like ours, cleaning and dedupliacting records are a common concern for ensuring the information extracted from data sets is valid. Indeed, some studies have identified duplication problems in OCLC's catalog (O'Neill, 1993; Taniguchi, 2013), where our data originates.

One particularly pervasive group of researchers in the field of bibliography, and by extension the cleaning of the bibliographic data they study, is Mikko Tolonen and his colleagues. In Tolonen, Mikko, et al (Mikko et al, 2019), they used cleaning in a way similar to our goal of taking information like titles and authors and attempting to fix any errors within them. As it is not the central point of their paper, they do not go in depth about this cleaning process, unfortunately. In another paper, Tolonen, Mikko, et al. (Mikko et al, 2019) also used some innovative methods of cleaning their data. Their paper describes their usage of string distance ranking algorithms and utilizing additional information such as number of pages to rank the matches. The idea is similar to how we intend to rank our clusters, but in a less broadly applicable way. While the use of page count can be extremely valuable for book matching, as the total will likely vary little in republications, it would not be universally applicable as few of our records contain this information.

The ability of clustering to normalize data is equally valuable as its ability to clean data, particularly when studying such a large period of time. Language is ever evolving, and changed spellings for words could drastically impact the results of the queries we perform. This is an issue tackled in many papers. For example, the work of Marjanen, Jani, et al. (Jani et al., 2019) describes how they normalize special characters and spelling errors. They even go a step beyond simple clustering and employ a method to normalize synonymous words to lower the variability of titles. This would certainly improve the ability for querying such large data sets, allowing for a broader search based on the idea a title represents than the exact match for it.

Various ways of cleaning and normalizing data exist, however not many are capable of doing both with the accuracy of clustering in data sets such as ours. This is primarily due to three factors, the multilingual nature of our data set, the fact many words in the data set have multiple valid spellings, and the fact titles may use names and other words not normally understood by spell checkers. As such, there are many papers exploring the forms and usage of a clustering based approach. Some of these works use the vast amount of readily available bibliographic data to test their developments, which is a very similar application to what we seek. For example, in the work of Li, Jingxuan, and Tao Li (Jingxuan & Tao, 2010), they address existing forms of clustering algorithms and additionally create their own algorithms to test against pre-existing methods. In their paper, they use an example data set consisting of books and keywords. Their method, called Hierarchical Co-Clustering (HCC), utilizes two features: hierarchical clustering and co-clustering. Hierarchical clustering enables browsing and navigation, while co-clustering clusters different types of data simultaneously by using information about their relationship. HCC begins with singleton clusters and merges the two closest clusters until only one remains, all while using the agglomerative hierarchical clustering algorithm as the framework. By comparing the HCC algorithm to more common techniques such as K-Means and Single Linkage Hierarchical Clustering (SHLC), they can show how much more effective their method is. The work of Waluyo, Arif Mardi, Eko Prasetyo, and Arif Arizal (Arif et al., 2018) also sought to improve on the method of clustering. Their work introduces a method of clustering book titles based on the K-Means average of distance dissimilarity. This method produced excellent results for categorizing books based on shared features, but the issue they attempted to solve required far fewer clusters than title normalization. While there are a fair number of genres a book can fit into, this number is far lower than the number of unique clusters we find while looking for semantic identicality. The method can efficiently group books into categories such as law, economics, accounting, engineering, management, and information technology, as demonstrated by their tests utilizing a data set of 500 book titles from the Library of Bhayangkara Surabaya. Peter Kiraly (Péter, 2019) attempted to validate the metadata of records, ensuring that linked titles existed in records, that ISBN and ISSN fields had the correct identifiers, and that the control field was properly formatted. Some works used the petition around medoids (PAM) algorithm to improve clustering performance such as the works of De Amorim, Renato Cordeiro et al. (De Amorim et al., 2013) and Zampeieri, Marcos et al. (Zampieri & de Amorim, 2014). This technique can significantly improve the performance when the optimal parameters are known. It is relatively similar to our method of choice but it requires much computational resources and the optimal parameters. There were many great insights about the problem of duplicate MARC data within this paper, focusing mainly on classifying records as either valid or invalid.

Many researchers also use machine learning techniques to tackle this problem. Ozsarfati, Eran, et al. (Eran et al., 2019) explored a variety of machine learning algorithms to classify the genres of books based on their titles. This paper analyzes six different machine learning techniques: Naive Bayes, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Bi-Directional LSTM (Bi-LSTM). Their paper assessed 207,575 book titles, attempting to categorize them into 32 genres. If expanded from attempting to classify genres to extracting a larger variety of semantic information from titles, it could be possible for a similar method to be used to compare titles in a way pure string comparison could not expand to. Bleiweis (Avi, 2017) presents a model capable of combining a word vector set of a varying number of elements in order to represent a verse or a sentence, with an iterative distance metric that evaluates similarity in pairs of non-conforming verse matrices. The work in this paper investigates linguistic structure in raw texts and uses Word2Vec to explore clustering and search tasks to train the underlying word representations. Representing the text in this way improves the performance of semantic clustering as well as searching searching, an extremely valuable improvement given the time costs of these procedures.

3. METHODOLOGY

This paper builds on earlier analysis of popular clustering algorithms which sought to determine which performed the best for data analogous to ours. Prediction by Partial Matching (PPM) clustering has been shown to outperform the other algorithms and gave us a good place to start with our data cleaning. However, the glaring issues of cases where the incorrect title would be normalized onto correct titles was a cause for concern. To attempt to improve the accuracy of choosing the correct title to normalize, leaving it to the most common value in the cluster would not suffice. As such, we implemented a spell-checking system into the clustering algorithm.

PPM clustering is a method of clustering based on the PPM data compression algorithm. The PPM compression algorithm is a commonly used algorithm in RAR and 7z compressed files, and employs the use of a statistical model to predict what the next character is most likely to be. To use this to compute clusters, we calculate the distance that exists between the two compressed strings. Following the logic of OpenRefine’s PPM clustering, a data management tool created by Google, we can compute this distance relatively easily. To get the distance between two strings, A and B, we need to look at how concatenating the strings affects the length of the compressed string. Because PPM compression compresses duplicated text data very effectively, the difference between compressing string A and the string A+A would be very slight, where “+” is the concatenation of the two. Of course, that means that if B was an identical string to A, the compression of A+B would be the same as A+A. The more that string A and B differ, the greater the difference in length of the compressed concatenation will be. Because of this, we can calculate the distance using comp(A + B) + comp(B + A)/(comp(A + A) + comp(B + B)), where comp() returns the length of the compressed string. By comparing the lengths, we get a much more time efficient algorithm than if we were to compute the distance between the two compressed strings using something such as Levenshtein Distance. We can convert this into a score by taking this value, D, and computing 10*(D-1). This is because, if all strings are identical, it would give us a D of 1, and as the strings get less and less similar, D fractionally increases. With this score, we can now compute clusters using PPM compression by selecting a distance cutoff for values to be grouped. For this, we will analyze a distance of 1 and 2. (Chen et al., 2004)

After creating and verifying the functionality of this PPM clustering algorithm, the next step was to decide on a method of spell checking the titles to create the ranking. We decided with Hunspell, the spell checking library used by LibreOffice, OpenOffice, Mozilla Firefox, Google Chrome, as well as many other programs. Being a well trusted library with an extensive dictionary selection, it was a perfect choice for a multilingual dataset like the one we are working with. The test data sets we created included English, French, German and Latin idioms and proverbs, all of which are publicly available dictionaries provided by Hunspell.

To rank the members of the clusters, we process each of the titles in the clusters found by PPM clustering. For each word in the title, we check if it exists in Hunspell, and if not, we increase the score of that title by 1. After each title in the cluster is processed, we know that the least number of words absent in the dictionary belong to the title with the lowest score. Doing it this way opposed to simply counting the number of correct words accounts for additional words in titles, such as “a” or “the” which would artificially increase the score of titles which may have misspelled words that do not outweigh the additional words. It also allows for correctly spelled words that do not exist in the dictionaries to be inconsequential, as the score increase is identical between each of the titles.

To accurately compare the performance of the algorithm against the original PPM clustering algorithm, we created two test data sets. While the preferable way to test these data sets would be to use the data provided by OCLC, as it is the end goal for these algorithms to be used on, however they have a fatal flaw of not having a “base truth”. Because we cannot resolve the correct title based on only the possibly dirty title, without vetting the data by hand, this solution is not feasible. The scale of the data makes manually cleaning even a portion of it difficult as well, as trying to clean enough records for it to be a representative sample would take far more time than reasonable. Given this, the creation of test data sets that mirror the data in as many ways as possible is necessary to properly gauge the ability of each algorithm to be used to clean data.

The first data set will be one preferable for our new algorithm, and is the same data set we used in our last paper to compare the different clustering algorithms (Bryer et al, 2021). The data set is made up of approximately 1,000 idioms and proverbs taken from English, French, German and Latin. Idioms and proverbs were a choice made to mirror both the average length of book titles, as well as the age of the titles we will be using this to clean. Because many idioms and proverbs were created using the version of languages that existed around the time of our books being published, we would not run into the issue of modern spellings that have changed skewing the test data in favor of our spell checking, as the dictionaries will be biased towards the more modern forms of the language.

We then corrupted these using the most commonly observed inaccuracies in our records as well as noted in other scholar’s data (O’Neill, 1993). Specifically, we randomly corrupted letters, as well as swapping words with those adjacent to it. We created two copies of our idioms and proverbs, the first copy being “mild” corruption, and the second being “severe” corruption. In the first copy, we had a 7% chance for it to be randomly swapped with any alphanumeric character. In the severe case, for any letter, there was a 10% chance of it being changed, and for any word there was a 6% chance of it being swapped with the adjacent word.

While this mock data set worked well to compare the quality of multiple different clustering algorithms, the fact it has 3 copies of each title, with one being correct and all three being different, it is not an impartial data set to be used against our new algorithm. Because the unmodified algorithm will favor the most common value in a cluster, having three distinct titles puts it at a distinct disadvantage. This disadvantage would likely not be nearly as severe in the real use case, as generally typos and misspellings would be less common in a repetitive data set than the correct version of the value. To account for this, we needed to create a more fair test data set for this test, one that would more accurately represent the repetition found in our real data set. To do this, we randomly duplicated entries in the mock data set. Looking at the OCLC data set, we can see that approximately one third of the entries are unique titles. Because correct titles are likely to appear more often when an entry is duplicated, we randomly duplicated correct titles in our mock data set with a probability of 33%, and had corrupted duplicates at a rate of half that. Further to the benefit of the original PPM algorithm, we corrupted the duplicated titles separately, as few duplicate records in the OCLC data set would contain the same typo.

4. RESULTS

To compare the results of the improved clustering algorithm implementing spell check, we will compare it against the version of this algorithm not using spell check. In Table 1, we can observe how each performed against the first test data sets. This was examined by comparing the full strings, if the string was not identical to the correct string it was marked as a failure.

The increase of accuracy is extremely significant between the unmodified clustering algorithm and our new algorithm. Especially in the case of using a radius of two, we find that there is a bigger difference between the improved algorithm and the unmodified algorithm than there is between the unmodified algorithm and the original. This trend of significant increases does not disappear with the next test, visible in Table 2. We tested the improvement based on characters, instead of full string matching. This would reward the algorithms for improving the overall accuracy of the strings, even if the most accurate string was not the one selected. While this would not be the preferable case, it is still an important test as any improvements to the data set are significant.

Table 1. String based comparison for data set 1

String Based	Original	Unmodified PPM	Improved PPM
Radius of 1	0.422	0.473	0.545
Radius of 2	0.422	0.551	0.705

Table 2. Character based comparison for Data Set 1

Character Based	Original	Unmodified PPM	Improved PPM
Radius of 1	0.954	0.956	0.959
Radius of 2	0.954	0.965	0.976

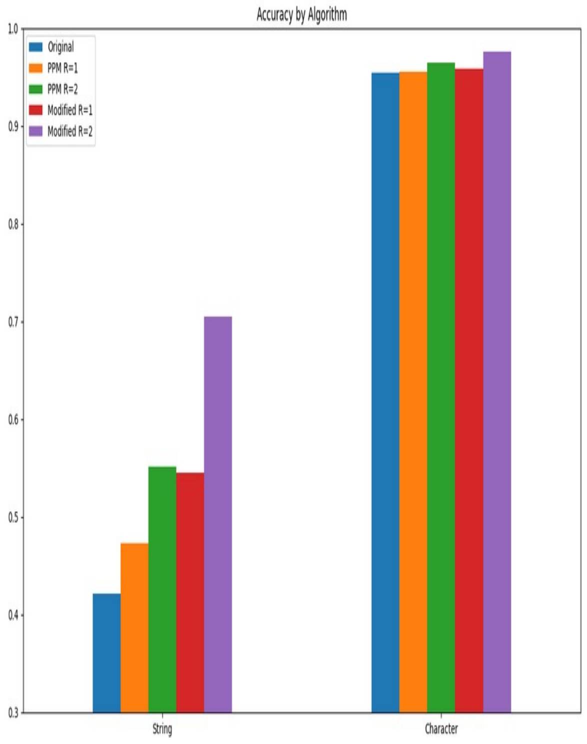
Once again, we observe a significant increase in the accuracy, matching the increase between the original and the unmodified clustering algorithm. A visualization of these results can be seen in Figure 1. We then ran these same tests on our second test data set. Table 3 shows the results of the string-matching analysis, and Table 4 shows the character based.

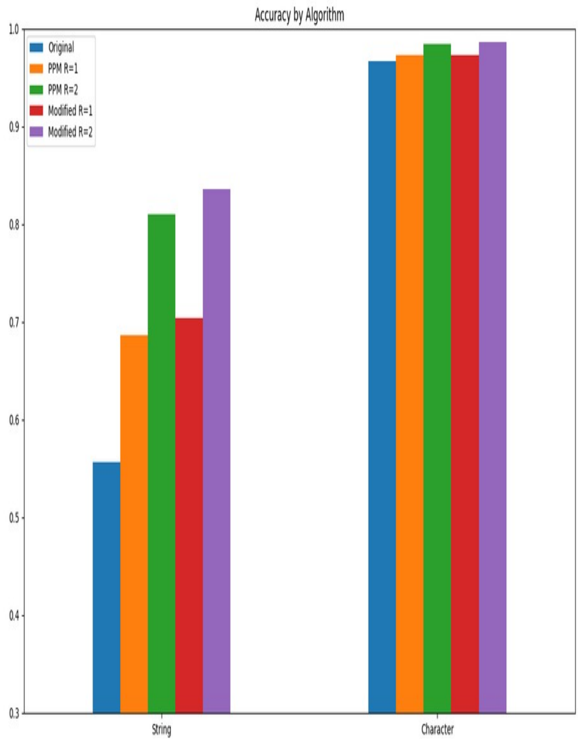
Table 3. String based comparison for Data Set 2

String Based	Original	Unmodified PPM	Improved PPM
Radius of 1	0.557	0.686	0.704
Radius of 2	0.557	0.810	0.836

Table 4. Character based comparison for Data Set 2

Character Based	Original	Unmodified PPM	Improved PPM
Radius of 1	0.967	0.973	0.973
Radius of 2	0.967	0.984	0.986





As expected, the increases between the unmodified algorithm and our original algorithm is much smaller than it was previously, as can be seen in Figure 2. However, the increase in accuracy is still significant, even when compared under more favorable conditions for the unmodified algorithm. For more in depth statistics about the performance of each, Table 5 and Table 6 show the positives and negatives, both true and false, for each data set.

Table 5. Positives and Negatives for Data Set 1

Positive/Negative	True Positive	True Negative	False Positive	False Negative
Unmodified (R=1)	478	954	320	1275
Unmodified (R=2)	833	834	440	920
Modified (R=1)	614	1226	52	1135
Modified (R=2)	1241	1120	158	508

Table 6. *Positives and Negatives for Data Set 2*

Positive/Negative	True Positive	True Negative	False Positive	False Negative
Unmodified (R=1)	616	1885	143	999
Unmodified (R=2)	1116	1837	191	499
Modified (R=1)	572	1994	36	1043
Modified (R=2)	1126	1924	106	489

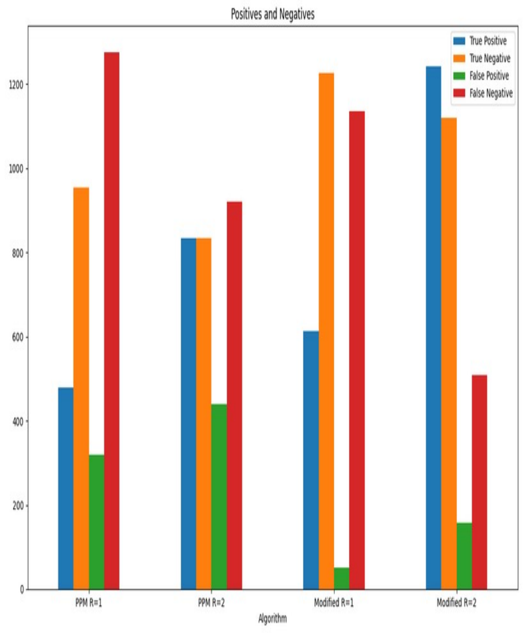


Figure 3. Visualization of True/False for Data Set 1

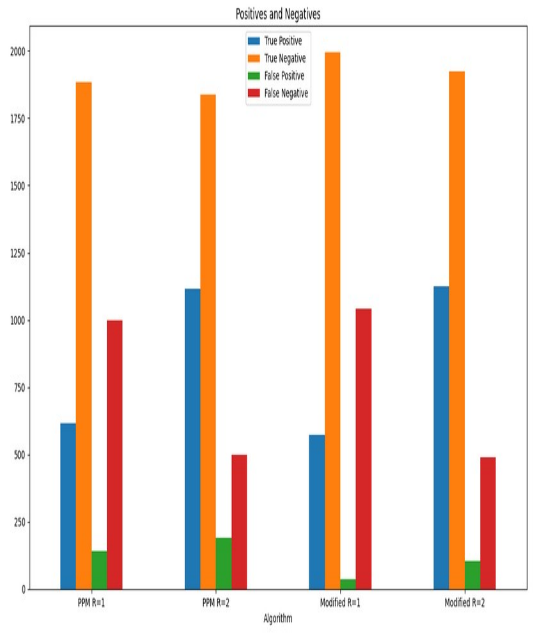


Figure 4. Visualization of True/False for Data Set 2

Using these values, we can calculate the precision, recall and F-Scores, shown in Table 7 and Table 8.

Table 7. Precision and Recall for Data Set 1

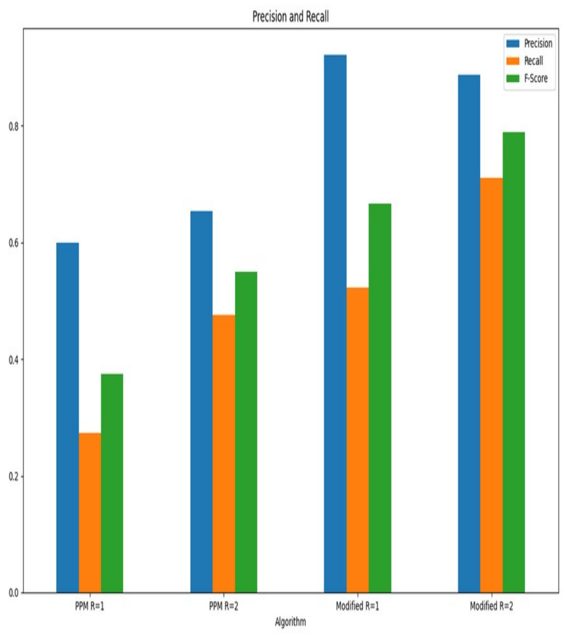
Precision and Recall	Precision	Recall	F-Score
Unmodified (R=1)	0.599	0.273	0.375

Unmodified (R=2)	0.654	0.475	0.550
Modified (R=1)	0.921	0.522	0.666
Modified (R=2)	0.887	.710	0.789

Table 8. Precision and Recall for Data Set 2

Precision and Recall	Precision	Recall	F-Score
Unmodified (R=1)	0.812	0.381	0.519
Unmodified (R=2)	0.854	0.691	0.764
Modified (R=1)	0.941	0.354	0.514
Modified	0.914	0.697	0.790

–



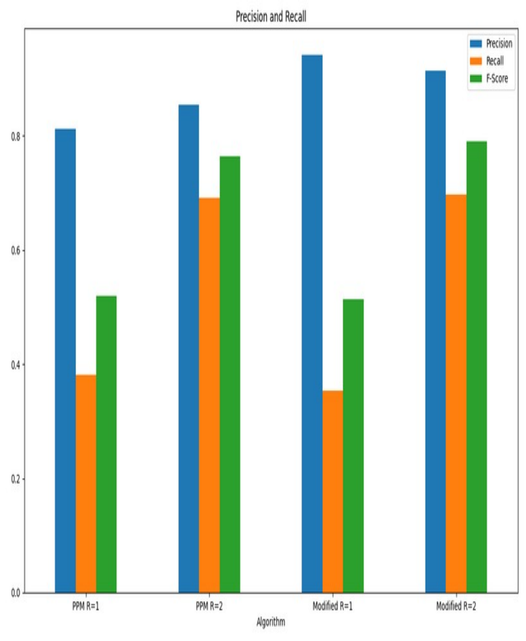


Figure 5. Visualization of Recall for Data Set 1

Figure 6. Visualization of Recall for Data Set 2

On the first data set, we see significant improvements in all three in our improved PPM algorithm. Interestingly, however, there is one case where the original algorithm outperformed ours, using a radius of one for data set 2. Our algorithm produced far fewer false positives, at the cost of fewer true positives and more false negatives. While the difference in F score between the two's performance in this case is minimal, it is an interesting trade off that shows the potential use case for choosing these parameters when cleaning a data set. If cleaning highly sensitive data where false positives are a worst-case outcome, the updated algorithm consistently produces the fewest false positives when using a radius of one.

To further test the efficacy of implementing a spelling based ranking algorithm to pre-existing clustering algorithms, we chose two other algorithms that we have previously experimented with. While PPM was the best performing algorithm prior to the implementation, that may not necessarily be the case for after the ranking is implemented. Additionally, it is possible that the implementation of the algorithm could have different impacts on different algorithms, depending on how those algorithms form clusters. In general, it is expected that it will have a similar impact on any of the algorithms that it is attached to, as it simply augments which member of the cluster is normalized as opposed to changing how the clusters themselves are formed. As such, it should not cause an algorithm to outperform one that previously performed better, when applied to both.

To test how the implementation of the ranking algorithm would impact other clustering methods, Levenshtein and Fingerprinting were chosen as the ideal candidates. Levenshtein works very similarly to PPM and was the second-best performing algorithm when tested without ranking. This makes it the most likely of any other algorithm to be able to perform better than PPM, both in speed and accuracy. Fingerprint offered an algorithm that worked significantly different from Levenshtein and PPM in how the clusters are formed. Implementing ranking to this would allow us to see if changing the method in which clusters are formed would impact the extent to which ranking could improve the final accuracy.

Table 9. Precision and Recall for Differing Algorithms Unranked

Precision and Recall	True Positive	True Negative	False Positive	False Negative
Fingerprint	251	1054	222	1500
Levenshtein	431	960	316	1320
PPM	538	810	466	1213

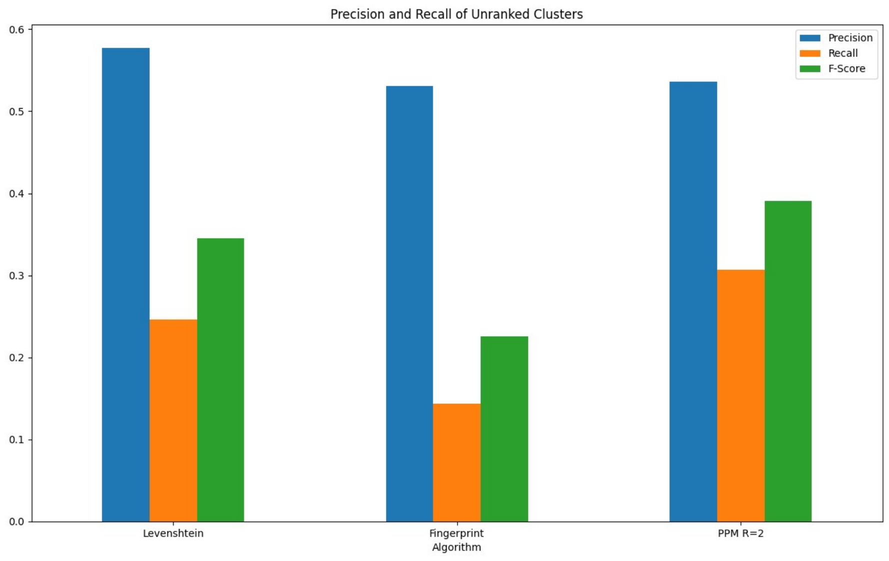


Figure 7. Visualization of Precision and Recall for Differing Unranked Algorithms

Once the ranking algorithm is applied to each of the clustering algorithms, their values all improve, as shown in Table 10.

Table 10. Precision and Recall for Differing Ranked Algorithms

Precision and Recall	True Positive	True Negative	False Positive	False Negative
Fingerprint	419	1194	82	1332
Levenshtein	673	1172	104	1078
PPM	1008	1150	126	743

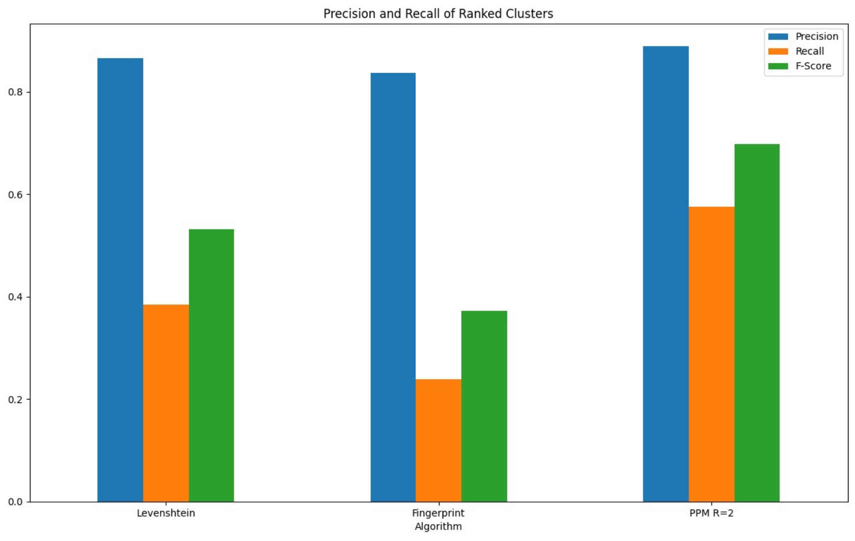


Figure 8. Visualization of Precision and Recall for Differing Ranked Algorithms

From these statistics, we can see that the implementation of the spelling based ranking algorithm increased the final accuracy of clustering by over 50% in all cases. Additionally, PPM showed the greatest increase in F-Score out of the three clustering algorithms we tested. Because PPM performed the best prior to ranking the choices, it follows that the clusters formed from this algorithm were the most accurate. As such improving the normalization of these clusters provided the greatest benefit to the result of the algorithm. Therefore, the F-Score increase was the greatest of the three. However, this does not explain why the increase in Levenshtein's final accuracy was lower than the increase in fingerprinting's final accuracy, as Levenshtein performed better than fingerprinting prior to ranking the clusters.

The reason for this lies in the recall scores for the two algorithms. The number of true positives found with the fingerprinting algorithm nearly doubled, where the number for Levenshtein only increased by approximately 50%, where their change in false negatives were nearly identical. The reason for this is how small the radius on fingerprinting is. Because far more of the clusters created through fingerprinting only included two forms of the data point, as opposed to Levenshtein who's clusters were far more likely to include all three versions of the data points. This is very impactful, as when only two versions of the data point exist, the algorithms have no choice but to randomly select one of the two to be the one normalized onto the other values. If three exist, the most common version of the data point can be normalized, which would improve the overall accuracy of the algorithm. Implementing the ranked system gives the algorithms a way to determine which of the two to select, so cases where only two members exist in a cluster are far less problematic, and the algorithm with more of these is improved more.

This logic explains why the unranked Levenshtein algorithm has a higher precision than the unranked PPM algorithm as well. Because PPM creates more, larger clusters, each case in which the wrong value is normalized is much more impactful than the cases in which smaller clusters have this problem. Despite PPM having more true positives, it has equally more false positives due to the incorrect values being normalized so often. This drawback would generally make PPM a much more destructive algorithm to use for data cleaning, more false positives being normalized and dirtying the data quite significantly. Once the clusters are ranked, however, the rate of false positives drops by nearly 270%. However, as it is still casting a very wide net for creating clusters, it still presents the highest quantity of false positives of the three algorithms when used with a radius of 2. Indeed, even with a radius of one, PPM has comparable true positives and negatives to fingerprinting, but still has more false positives.

5. CONCLUSION

To get the most valuable information as possible out of the expansive OCLC data set, it is imperative that we normalize and clean the data as thoroughly as possible. Due to being compiled over many years, by many librarians and separately digitized, naturally there are errors of all sorts throughout. As found in our last paper, due to the duplicative nature of many of the entries in this data set, using clustering for the purpose of cleaning excelled in its ability to improve the accuracy of our data.

By implementing the Hunspell library into the best performing clustering algorithm we tested, we are able to significantly improve it over its base form for the purposes of text data cleaning. Hunspell and its ability to link dictionaries from multiple languages is especially important for our use case. Given that we must be able to account for misspelled words regardless of the language of the title, so long as it falls within the set of languages the data provides us with, being able to check every pertinent language at once greatly increases the efficiency of ranking clusters.

Because testing our improved algorithms against the original version of each clustering algorithm resulted in higher accuracy scores in each data set tested, it is a safe interpolation that it will perform significantly better on increasing the overall accuracy of the OCLC data set than the unmodified clustering algorithms. Even in cases where we must prioritize avoiding false positives for clustering over overall improved accuracy, using a radius of one in the modified algorithm produced far fewer false positives, and therefore would be a better algorithm to apply for this case.

The one drawback of this method of clustering is the increase in time taken to process and clean the data. Naturally adding a spell checking and ranking step to this process would increase the run time, and through our tests we found the increase in run time to hover around 35%. Clustering is already a relatively time intensive process, so this increase could indeed be an important factor when performing data cleaning on extremely large data sets. However for our cases, the benefits of vastly increased accuracy improvements outweighs the increased cost of time.

Methods of expanding on this research would either focus on decreasing the time cost of adding the spell checking functionality, or improving the ability for clusters to be ranked. The first method would likely rely on heuristics, such as only calculating the maximum length two strings could differ to fall within the radius for PPM clustering before the compression, to reduce the number of unnecessary compressions made. The second may implement other methods into the ranking system, not only relying on individual word spelling. This may take the form of grammar checking, or comparison against a database of book titles.

REFERENCES

Bryer et al. (2021). Analysis of Clustering Algorithms to Clean and Normalize Early Modern European Book Titles. *In Proceedings of the 4th International Conference on Software Engineering and Information Management (ICSIM '21).*

O'Neill, E. (1993). Characteristics of Duplicate Records in OCLC's Online Union Catalog. *Library Resources & Technical Services, 37,* 59-71.

Chen et al. (2004). The Similarity Metric. *IEEE Transactions on Information Theory*. *50*. 3250-3264.

Péter K. (2019). Validating 126 million MARC records. *In Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*. 161–16.

Taniguchi, S. (2013). Duplicate Bibliographic Record Detection with an OCR-Converted Source of Information. Jo*urnal of information Science, Volume 39*. 153-168.

Mikko et al. (2019). A quantitative approach to book-printing in Sweden and Finland, 1640–1828. *Historical Methods: A Journal of Quantitative and Interdisciplinary History Volume 52, No. 1*, 57–78

Mikko et al. (2019). Scaling Up Bibliographic Data Science. *In DHN, 450–456*

Jani et al. (2019). A National Public Sphere? Analyzing the Language, Location, and Form of Newspapers in Finland, 1771–1917. *Journal of European Periodical Studies,Volume 4, No.1*, 54–77

Jingxuan L. & Tao L. (2010). HCC: a hierarchical co-clustering algorithm. *In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 861–862

Arif et al. (2018). Clasification System of Library Book Based on Similarity of the Book Title Using K-means Method (Case Study Library of Bhayangkara surabaya). *Journal of Electrical Engineering and Computer Sciences, Vol 3 Number 1*

Eran et al. (2019). Book Genre Classification Based on Titles with Comparative Machine Learning Algorithms. *In 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 14–20

A Hierarchical Book Representation of Word Embeddings for Effective Semantic Clustering and Search. *In ICAART, Volume 2*, 154–163.

De Amorim et al. (2013). Effective spell checking methods using clustering algorithms. *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, 172-178.

Zampieri, M. & de Amorim, R. (2014). Between sound and spelling: combining phonetics and clustering algorithms to improve target word recovery. *International conference on natural language processing,* 438-449.