Detecting Rhyming Words

Simran Karnani[1] , **Stravos Makrodimitris**[1] , **Arman Naseri Jahfari**[1] , **Tom Viering**[1] , **Marco Loog**[1]

[1]TU Delft

Abstract

Rhyming words are one of the most important features in poems. They add rhythm to a poem, and poets use this literary device to portray emotion and meaning to their readers. Thus, detecting rhyming words will aid in adding emotions and enhancing readability when generating poems. Previous studies have been done on the topic of poem generation. However, those works did not put too much emphasis on the rhyme detector. Thus, this research will solely focus on rhyme detection and its evaluation. The aim of this research is to determine the most accurate way of detecting whether two English words rhyme. English rhyming words will be detected using combinations of features. Five features are used: edit distance, hamming distance, jaccard similarity, longest common substring, and vowel and consonant weights. We also experiment with two methods of retrieving phonemes: using the entire phoneme translation, and using part of the phoneme translation. We find that using only hamming distance and jaccard similarity with part of the phoneme translation, we can already obtain an accuracy of 90.05% with a log loss of 0.25 when trained on a balanced dataset. The reason for this remains unclear because there is no clear separation between the two classes.

1 Introduction

Deep learning has revolutionized natural language processing by generating realistic pieces of texts. It can predict the next set of words, summarize articles, and generate automatic responses [17]. Deep learning can also be taken a step forward to generate poems. In order to generate poems, a rhyme detector is necessary. This is because many poem generators first generate a pair of rhyming words and then generate the remaining of the sentence. Thus in order for the generator to generate a rhyming pair, it must be able to detect rhyming words[7].

Rhyme detection is an easy task for humans, but difficult for machines. Rhyme is necessary in creative pieces, such as poems. Thus, in order to generate poems, a rhyme detector is necessary. Although poem generators already exist, they underperform in terms of emotion and readability [12]. In order to enhance the readability, poets use rhyming words, either within the lines of a poem or at the end of the lines. Rhyming words add rhythm to a poem, which keeps the poem in harmony[4]. Rhythm is a literary device that sets poems apart from other prose. It sets the tone for the poem and can generate emotion or enhance ideas [14]. Rhyme also helps poets to draw an image for the reader and can create internal rhyme to bring about meaning, emotions, and feelings [8].

There are different types of rhymes, such as end rhyme, double rhyme, first syllable rhyme and many more. This research will focus on end rhymes. End rhymes can be further subdivided into perfect rhymes and imperfect rhymes also known as slant rhymes. In perfect rhymes, the last stressed vowel and all sounds following it are identical for both words (example: conviction, prediction) [15]. Slant rhymes, on the other hand, have similar, but not identical sounds or emphasis. This includes words that end with similar consonant sounds (example: pact, slacked), words whose last syllable contains an assonance (example: unpack, detach), and words whose last syllable contains final consonants that have consonance (example: country, conta) [2]. Both of these have their own purpose in poems and help the poet to portray different emotions and illustrations.

The aim of this research is to determine the most accurate way of detecting whether two English words rhyme. To do this, the following questions will be answered:

- What (pronunciation) dictionary should be used? Two words can simply be compared using their spellings or pronunciations. When using spelling, the English alphabets can be used. When comparing using pronunciation, pronunciation dictionaries, CMU or IPA, can be used.
- Should the entire word be used or only part of the word that contains the rhyme? End rhymes usually have similar endings, thus the second method will only compare the ends of the words.
- What combination of features should be used? The five features that will be used are: edit distance, hamming distance, jaccard similarity, longest common substring, and vowel and consonant weights.
- Are more rhyming or non-rhyming pairs detected?

A machine learning algorithm will be used to detect rhyme. As mentioned above, perfect rhymes have an identical match-

ing, but imperfect rhymes do not. Thus, there is not always a 100% similarity from the last stressed syllable onward. Additionally, the vowel stress may also differ for slant rhymes. Therefore, a machine learning algorithm is required to learn the different patterns.

The remaining of the paper is structured in the following manner. Section 2 contains some related works. Descriptions of the pronunciation dictionaries and features are in Sections 3 and 4 respectively. Section 5 contains an explanation of the method used. The results will be presented in Section 6 and further discussed in Section 7. Section 8 concludes the research and recommends some future works. Lastly, Section 9 includes a part about responsible research and how this experiment can be reproduced.

2 Related Work

Previous works in the field of rhyme detection have used datasets of poems or rap music and trained their models with one feature. These works paid more attention on generating poems or rap lyrics and determining the position of the rhymes, thus smaller evaluations were performed on the rhyme detection. The main focus of this paper will be on evaluating rhyme detection using a dictionary of rhyming and non-rhyming pairs.

DeepLyricist [10] is a tool that generates rap lyrics by detecting rhyme, idiom, structure, and novelty. They use longest common substring, which is further explained in Section 4, to detect rhyme and calculate the rhyme density. However, to do this, they remove all consonants from the word and find the longest substring of vowels. As they discussed, the lack of consonants in between vowels may be why the rhyme density was not accurate. Furthermore, the model was trained on a dataset of rap lyrics, many of which did not contain rhyme. Therefore, it was less likely for the model to learn the parameters for rhyme.

Hirjee and Brown [9] used a probabilistic method to detect rhyming phrases. This method calculated a rhyme score based on the vowel score, end consonant alignment score, and metrical stress marking score. These scores are calculated based on how probable their matches are in rhyming phrases. The model is trained on a datset of rhyming rap lyrics and instead of using the entire word for rhyme detection, only the vowels, stress markings and last consonants of the last stressed syllable are used.

Furthermore, the work of Kesarwani [11], uses vowel and consonant weights (described in Section 4) to detect rhyming pairs. The model is trained on a set of rhyming poems to detect the type of rhyme (perfect or imperfect) and rhyming position (internal rhyme, consecutive end rhyme or alternative end rhyme). When trained on 50 poems and tested on a different set of 50 poems, the model scored an accuracy of 96.51%.

Lastly, in the Deep-speare study [12], the cosine distance was used to detect rhyming words. The model was trained on a set of quatrains and used a rule-based method, where a pair is considered to be rhyming if it has a cosine similarity ≥ 0.8. This resulted in many rhyming errors. For example, 'supply' and 'sigh' have a cosine distance 0.836 when using their model, despite being a non-rhyming pair.

English Alphabets	CMU	IPA
syrups	['S', 'ER1', 'AH0', 'P', 'S']	ˈsɜ·əps
musketeer	['M', 'AH2', 'S', 'K', 'AH0', 'T', 'IY1', 'R']	ˌməskəˈtɪr

Figure 1: Examples of CMU phonemes and IPA phonemes along with their metric stress markings. Primary stress is represented in red and secondary stress in blue.

Some of these features and methods worked well, while others did not. In this paper, a combination of these methods and features will be experimented with to determine which accurately detects rhyming pairs.

3 Pronunciation Dictionaries

The two dictionaries that were used to get the pronunciation of words are: Carnegie Mellon University (CMU) Pronouncing Dictionary and International Phonetic Alphabet (IPA). These two dictionaries contain a large number of English words, however they do not contain all the words. Thus, a combination of both dictionaries is used to increase the dataset of words and their pronunciations.

3.1 CMU Dictionary

The Carnegie Mellon University (CMU) Pronouncing Dictionary is a machine readable pronouncing dictionary for North American English. It contains over 134,000 words which are mapped to their pronunciation in the ARPAbet phoneme set, a standard for English pronunciations [13]. The phoneme set contains 39 phonemes: 24 consonants and 15 vowels. The vowels carry a metric stress marking which indicates whether they have a primary stress (1), secondary stress (2), or no stress (0) (Figure 1). Thus for each word, its phoneme translation is returned, which consists of the speech sounds, as well as the emphasis placed on each syllable pronounced [9]. Some words have multiple phoneme translations, depending on the context in which it is used. However, the dictionary does not state the context of the word, thus for this research, the first phoneme translation is used.

3.2 IPA

The International Phonetic Alphabet (IPA) is an alphabetic system of phonetic notions based on Latin characters, which is very well referenced by [6]. The English language consists of 44 phonemes, 24 consonants and 20 vowels. A vertical line at the top (') indicates primary stress and a vertical line at the bottom (,) indicates secondary stress (Figure 1). The stress symbols are placed before the stressed syllable in a word [16]. IPA contains mappings of words to their pronunciations in multiple languages.

4 Features

In order to train a model to detect rhyming words, the rhyming pair must first be transformed into a format that can

be easily understood by a machine. For this research, five different similarity techniques were used as features for the models. Besides for these techniques, a sixth feature, the length of the longest phoneme translation, is also used. Each of these features depends on a pair of words to determine how similar two pronunciations are. The five similarity techniques will be described in this section.

4.1 Edit Distance

The Edit Distance finds the minimum number of operations (remove, insert, replace) that need to be performed to transform one list into another. In figure 2, 'P' in the second list can be replaced with 'T', and 'L' can be removed, to obtain the first list. Therefore giving an edit distance of 2.

Thunder: ['T', 'AH1', 'N', 'D', 'ER0']
Plunder: ['P', 'L', 'AH1', 'N', 'D', 'ER0']

Figure 2: Phonetic translation of 'Thunder' and 'Plunder' expressed using the CMU dictionary

4.2 Hamming Distance

The Hamming Distance is a metric for comparing two strings or lists. Both lists of phonemes must be of equal length, hence the shorter list is padded. Since we are detecting end rhymes, the padding is done at the beginning, to align the ends of the words. Hamming Distance counts the number of positions at which the elements differ.

4.3 Jaccard Similarity

The Jaccard Similarity compares the members of two sets to see which members are shared and which are distinct. Like the Hamming Distance, both lists must be of the same size, thus the shorter one is padded at the beginning. Jaccard similarity coefficient score is used to calculate the jaccard similarity. It compares the phonemes of word A to those of word B and returns a score between 0 and 1. Figure 3 displays an example of jaccard similarity on the lists of phonemes for words A and B. A higher score means that the two sets are more similar. Jaccard similarity is calculated using the following formula:

$$d(A, B) = \frac{A \cap B}{A \cup B}$$

A = ['L', 'K', 'I', 'EY1']
B = ['K', 'I', 'K', 'EY1']
d(A, B) = 0.25

Figure 3: Example of Jaccard Similarity using Jaccard similarity coefficient score

4.4 Longest Common Substring

The longest Common Substring problem is to find the longest substring that is present in two strings [1]. This can be applied to lists as well, as seen in figure 2, where the longest common sublist is ['AH1', 'N', 'D', 'ER0'].

4.5 Vowel and Consonant Weights

Rhyme is defined by word pronunciations, but end rhymes are also defined by the stress position. Thus, stress similarity and phoneme comparisons are made. This approach was inspired by Kesarwani [11]. Since this research focuses on end rhymes, the two pronunciations are aligned from the last phonemes. At each position, both phonemes are compared. Table 1 presents the scores for each type of pairing (scores are determined by Kesarwani [11]). If only one phoneme is present at that position, no score is given. The scores at each position are summed to give the final score. We added an additional scoring for vowels that match and have different stresses. This is for slant rhymes since those words do not always have the same stress marking.

	Type of match	Score
n	Vowel-constant mismatch	0.0
nv	Vowel mismatch	0.2
nc	Constant mismatch	0.4
-yv	Vowel match but stress mismatch	0.5
yv	Vowel match without stress	0.6
yc	Constant match	0.8
***yv**	Vowel match with stress	1.0

Table 1: Scores for the type of match (Vowel and Constant Weights)

5 Methodology

The following steps were carried out to build a model that detects whether two words rhyme:

1. Data collection and cleaning
2. Retrieving phonemes
3. Extracting features
4. Train model
5. Evaluate classifier

5.1 Data collection and cleaning

Since the available dictionaries of rhyming words were very small (about 100 pairs), a self made dictionary was used. Using a webscraper[1], the website www.rhymer.com[2] was scraped for end rhymes. To obtain a relatively large and diverse dataset, one syllable and two syllable end rhymes were retrieved. 78,475 words were collected, each having a list of one- and two- syllable end rhymes. The rhyming pairs that did not have pronunciations in the CMU Dictionary or IPA American English Dictionary were removed, thus resulting in 44,660 words, with lists of rhyming words.

In order to make a dictionary of non-rhyming pairs, the rhyming pairs dictionary was used, and the rhyming words lists were swapped around. These non-rhyming words were filtered to remove any words that rhyme with the main word. An example is displayed in Figure 4.

The dataset of 44,660 words with large lists of rhyming words made computation very expensive. Thus, a smaller

[1]https://bitbucket.org/TommieV/sintscrape/src/master/
[2]https://www.rhymer.com/

Word	Rhyming	Non-Rhyming
Cat	Mat Hat Begat Format	Blew Flu Horseshoe Bamboo
Corn	Born Horn Acorn Firstborn	Mat Hat Begat Format
Tissue	Blew Flu Horseshoe Bamboo	Born Horn Acorn Firstborn

Figure 4: Example of non-rhyming dictionary

dataset was used to train the model. The first 25,000 words made up the rhyming dictionary while the last 25,000 made up the non-rhyming dictionary. Each of these words had a list of about 5 rhyming or non-rhyming words. In previous experiments [12], they discovered that having false words that were not part of the rhyming category can improve the performance of the model, hence the last 25,000 words were used for the non-rhyming dictionary. To sum up, the smaller dataset contains a total of 248,905 rhyming and non-rhyming pairs.

5.2 Retrieving phonemes

In the case that CMU and/or IPA dictionaries are used, this step is conducted. As previously mentioned, the IPA dictionary has phonemes for multiple languages, including both American English and British English. Since the CMU dictionary only has phonemes for American English, American English phonemes will be used for both dictionaries to keep them compatible. CMU phoneme translations were obtained using the nltk package [3]. The IPA American English dictionary was retrieved from the IPA dictionary.[3]

End rhymes have identical or similar pronunciations towards the end of the words. This is why we apply two methods. Method A uses the full pronunciation of the words and method B only uses part of the pronunciation. The part of words that are similar or identical are from the last stressed syllable onward [12]. Thus method B only compares the parts of the pronunciation from the last primary stressed syllable onward. If there is no primary stress, the secondary stress is used. An example of both methods is presented in Figure 5.

5.3 Extracting features

In order to determine whether two words rhyme, features are extracted from their pronunciations. The five features used to train the model are:

1. Edit distance
2. Hamming distance
3. Jaccard similarity

[3]https://github.com/open-dict-data/ipa-dict

	English alphabet	CMU	IPA
Full phoneme (Method A)	musketeer	['M', 'AH2', 'S', 'K', 'AH0', 'T', 'TY1', 'R']	məskəˈtɪr
Part phoneme (Method B)	eer	['TY1', 'R']	ˈtɪr

Figure 5: Example of full phonemes and part of the phoneme in CMU and IPA

4. Longest Common substring
5. Vowel and consonant weights

Hamming distance[4], jaccard similarity[5], and vowel and consonant weights compare phonemes at each index. Edit distance[6] and longest common substring[7] do not compare phonemes at indices. Instead they look at the entire pronunciation and make comparisons, which may be helpful for slant rhymes.

The longest common substring returns a sublist of common phonemes. Thus the length of this sublist is used to obtain a score. The edit distance, longest common substring, and vowel and consonant weights are normalized by dividing the score by the length of the longest phoneme translation in the word pair. Jaccard similarity and hamming distance already give a score between 0 and 1.

Combinations of these features will be experimented with to determine which set of metrics should be used to determine whether two words rhyme.

5.4 Train model

Due to the large dataset, SKLearn's SGDClassifier implementation was used to train the model. The classifier uses the loss function, log, for logistic regression, with regularization term l1, which can shrink some model parameters towards the zero vector. The regularization parameter, alpha, is set to 0.001. This parameter was selected through trial and error. The values 0.0001, 0.1, 0.001, and 1 were tried with five different feature combinations: all features, jaccard and hamming, edit, longest common substring, and vowels and consonant weights, longest common substring and vowels and consonant weights, and jaccard and longest common substring. The value 0.001 resulted in the best accuracy and log loss. This parameter is used to prevent the model from overfitting to the training data. Moreover, this parameter is also used to compute the learning rate (default='optimal').

The data is split into a training set and testing set using 10-fold cross validation. Since the dataset was shrunk, cross validation was used to get a better estimate on how the data would perform had it been trained on the large dataset.

[4]https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.hamming.html

[5]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html

[6]https://pypi.org/project/editdistance/0.3.1/

[7]https://stackoverflow.com/a/42882629/15236519

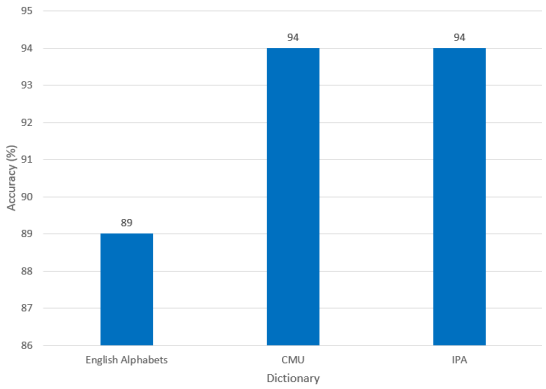

Figure 6: Graph showing the accuracy of detecting rhyming words using three different dictionaries. The CMU and IPA dictionary have high accuracies (and the same), thus both pronunciation dictionaries will be used to detect rhyming pairs.

6 Experimental Results

In order to determine which method, Method A or Method B, and which feature combination best detects rhyming words, the accuracy, log loss, and feature weights, will be evaluated.

Which (pronunciation) dictionary should be used to detect rhyme?

Figure 6 shows the accuracy when using english alphabets, CMU pronunciations, and IPA pronunciations. During the initial experiments, the english alphabets were used to determine whether two words rhymed or not. This was tested on a very small dataset of 200 rhyming pairs and 200 non-rhyming pairs. Using alphabets, the model had an accuracy of about 89%. The model was then trained to use the CMU dictionary and IPA dictionary. Both dictionaries had an accuracy score of 94%.

Using alphabets resulted in a lower accuracy because word pairs such as 'love' and 'move' are detected as rhyming. They have common alphabets, but are pronounced differently, therefore they are neither perfect rhymes, nor slant rhymes. The CMU dictionary and IPA dictionary use pronunciation instead, and therefore result in higher accuracies.The CMU dictionary and IPA dictionary contain many different words, thus a combination of these will increase the dataset size of words that contain phoneme translations. For these reason, the experiment was further conducted using a combination of CMU and IPA pronunciations.

What method should be used when retrieving phonemes?

Two methods can be used to retrieve phonemes. Method A, which uses the entire pronunciation, and Method B, which only uses part of the pronunciation, as explained in subsection 5.2. Figure 8 displays the average accuracy when using Method A and Method B on various feature combinations. It can be seen that for all combinations, Method B has a higher accuracy (about 2%). Thus based on these results, it can be indicated that Method B, using part of the phoneme translation, should be used to detect rhyming words.

Word pair	Output		Edit Distance	Hamming Distance	Jaccard Similarity	Longest Common Substring	Vowel and Consonant Weights
zeus, excuse	rhyming	Method A	0.714	0.714	0.214	0.286	0.314
		Method B	0.0	0.0	1.0	1.0	0.9
washed, stodgy	non-rhyming	Method A	0.8	0.8	0.2	0.2	0.36
		Method B	0.667	0.667	0.33	0.33	0.467

Figure 7: Similarity/Difference between word pairs when using Method A and Method B

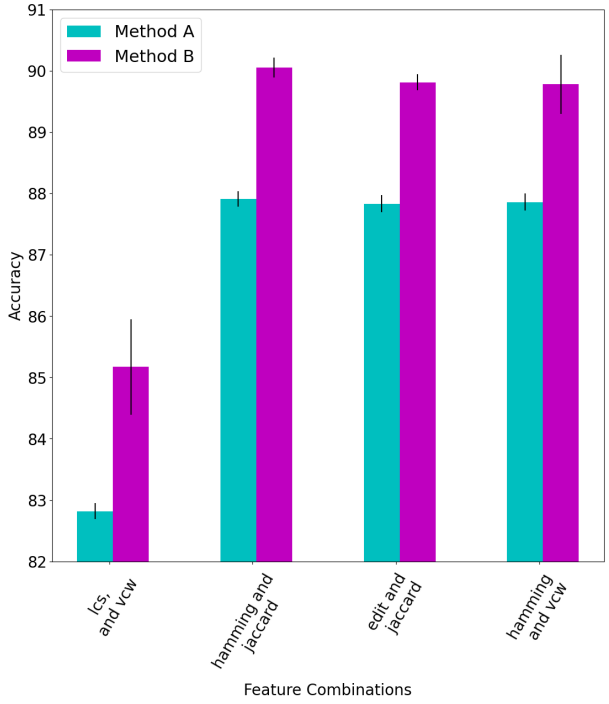

Figure 8: Graph comparing the average accuracy when using Method A and Method B. Method B has a higher accuracy for all feature combinations, thus based on these results, Method B should be used to retrieve phonemes.

After taking a deeper look at the results, the following observations were made. Method B can better detect rhyming pairs, since the word is split at the last stressed syllable. Thus, rhyming pairs will have a smaller distance (or larger similarity) when the score is normalized by the length of the longest part phoneme translation in the pair. An example of this is displayed in Figure 7. However, Method A (full phoneme translation) produces results that do not drastically vary from those of Method B. This is because the full phoneme translation can better identify words that do not rhyme, as displayed in Figure 7.

What combination(s) of features accurately detect rhyming pairs?

Figure 9 shows the learned feature weights, log loss and average accuracy of different feature combinations when using part of the phoneme translation. The lowest accuracy,

Edit Distance	Hamming Distance	Jaccard Similarity	Longest Common Substring	Vowel and Consonant Weights	Log loss	Accuracy	std
-0.135	-7.408	9.541	0.000	-1.037	0.25	89.88	0.17
-6.602			3.348	4.570	0.33	84.96	0.22
	-7.163	9.026			0.25	90.05	0.16
	-16.119			0.000	0.25	89.78	0.48
		16.288	0.000		0.26	89.73	0.11

Figure 9: Feature weights, log loss, and average accuracy for different feature combinations when using part of the phoneme translation. Features that were not used to train the model do not contain feature weights. Based on these results, it appears that the most important features are Hamming Distance and Jaccard Similarity.

84.96, was obtained when using a combination of edit distance, longest common substring, and vowel and consonant weights. This combination also has the highest log loss.

It is also observed that the combinations that include jaccard similarity and/or hamming distance have a much higher accuracy. However, the weights of jaccard similarity and hamming distance are always a lot more important than the others in the combination. When hamming distance is used with vowel and consonant weights, the latter has a learned feature weight of zero. The same for jaccard similarity and longest common substring. When all features are used, longest common substring has a learned weight of zero. Thus, it can be indicated that longest common substring is the least important feature.

Based on these results, it appears that jaccard similarity and hamming distance are the most important features for rhyme detection. Taking a closer look at the results, it was noticed that the hamming distance identifies more correct perfect rhymes. This is because it does an index match.

Are more rhyming or non-rhyming pairs detected?

The dataset contains two classes: rhyming pairs, which are the end rhymes, and non-rhyming pairs, which may be other types of rhymes, such as beginning rhymes, or no rhymes. Figure 10 shows the accuracy of two imbalanced datasets, one where majority of the word pairs are rhyming, and the other where majority of the word pairs are non-rhyming. These accuracies are obtained when using part of the phoneme translation.

From the figure, it can be seen that the dataset that the dataset that contains more rhyming pairs has a slightly higher accuracy than the one that contains more non-rhyming pairs. However, the accuracies differ by $\pm$ 0.2. Thus it can be indicated that both rhyming and non-rhyming pairs are detected when using part of the phoneme translation. This means that the model has a high accuracy (about 89%) of distinguishing an end rhyme from all other types of rhyme.

7 Discussion

This research determines the best way of splitting the phoneme translation and what feature combination gives an accurate way of determining whether two words rhyme. In

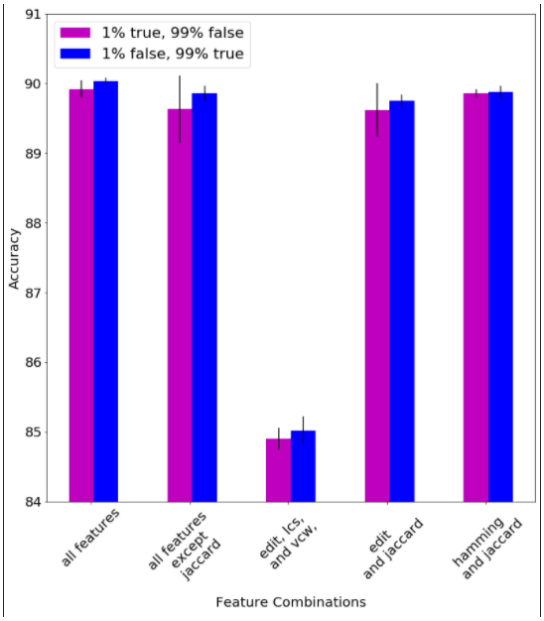

Figure 10: Graph comparing the accuracies of imbalanced datasets using part of the phoneme translation. It can be seen that the dataset containing more true values has a higher accuracy. However, the difference in accuracies is $\pm$ 0.2. It can be indicated that both rhyming and non-rhyming pairs are detected, but the rhyming class has a slightly higher accuracy.

section 6, the results were evaluated based on their accuracy, cross entropy, and feature weights. Accuracy and cross entropy are negatively correlated, as seen in Figure 9. While the accuracy looks at the number of correctly predicted data points, cross entropy looks at the predicted probability of data points. A higher accuracy results in a better performing model, while a lower log loss when predicted outputs are closer to their class label. Although there was no major difference, the model performed better when it was evaluated based on part of the phoneme. Furthermore, the features, hamming distance and jaccard similarity gave better results and had higher learned weights when compared to edit distance, longest common substring, and vowel and constant weights.

Using part of the pronunciation is a method that was inspired by two previous works[12] [9]. In both papers, they split the word at the last vowel. As they discussed, this approach detects words such as drummer and weaker as rhyming. But these words are not relevant rhymes due to their lack of stress. End rhymes are identified based on the last stressed syllable onward[5]. Method B takes this into account and splits the word at the last stressed syllable. It essentially only compares the parts of the words that are used to determine whether the two words rhyme. For this reason, method B is able to better detect rhyming words.

Furthermore, it was not a surprise that the longest common substring was the least important feature. In previous works [10], this feature was used to detect rhyming words. However, it was not able to accurately do so, as described in Section 2. This is because some rhyming pairs do not have identical pronunciations from the last stressed syllable onward.

Slant rhymes, for example, have varying identical pronunciations, thus the longest common substring is almost always very short.

The feature that is most striking is the vowel and consonant weights. This is the only metric that accounts for words that have the same vowel with different stresses. Many slant rhymes do not have the same stress, and yet they are rhyming. Thus, the vowel and consonant weights metric should be able to also identify slant rhymes. Kesarwani used this feature to train the rhyme detection model, which resulted in an accuracy of 96.51%. However, this model was trained on a small dataset of 50 rhyming poems. Thus making it incomparable to our model which was trained on a large dataset of word pairs containing both rhyming and non-rhyming words. The reason for vowel and consonant weights being a low importance feature remains to be an open question. For future experiments, the ratio of perfect to slant rhymes should be taken into account to determine whether this feature does indeed work better for slant rhymes.

Another question that remains unclear is why the combination of jaccard similarity and hamming distance has a high accuracy. Figure 11 shows the feature scatter plot for hamming distance and jaccard similarity when using part of the phoneme translation. When looking at this plot, there is no clear distinction between both classes.The non-rhyming pairs are saturated near hamming distance = 1.0 and jaccard similarity = 0.0. However, the rhyming pairs have hamming distances and jaccard similarities from 0 to 1. This could be because slant rhymes do not have identical phonemes, and some even have different stressed vowels. Jaccard similarity and hamming distance do not take different stress markings into account.

From this plot, it seems as if non-rhyming pairs are better identified than rhyming pairs. This contradicts the results when using imbalanced datasets, as seen in Figure 10. Additional experiments using different datasets can be done to further investigate why these two features are the most important.

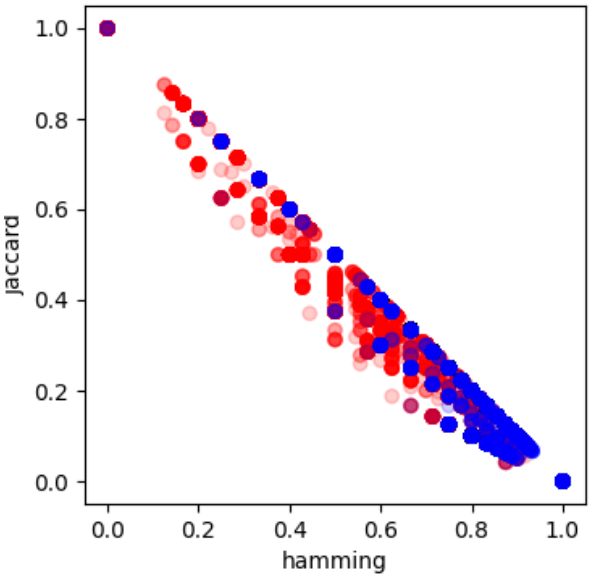

Figure 11: Feature scatter plot for hamming distance and jaccard similarity when using part of the phoneme translation. The red labels represent rhyming pairs and the blue labels represent non-rhyming pairs.

8 Conclusions and Future Work

The aim of this study is to determine the most accurate way of detecting whether two English words rhyme. To do this, a combination of two pronunciation dictionaries were used, CMU and IPA. Furthermore, rhyming pairs were detected by comparing two methods for retrieving phoneme translations: using the full phoneme translation and part of the phoneme translation. Combinations of five different features (edit distance, hamming distance, jaccard similarity, longest common substring, and vowel and consonant weights) were experimented with to determine which combination gives the best results. After examining the accuracy, log loss, and learned feature weights, it can be concluded that the combination of jaccard similarity and hamming distance on part of the phoneme translation can more accurately detect rhyming words. The model identifies both rhyming pairs and non-rhyming pairs.

Using a combination of jaccard similarity or hamming distance with another other metric on half of the phoneme translation have an accuracy of about 89%, with a log loss of 0.25. However, the learning weights of jaccard similarity and hamming distance are always more important than the others. Thus the best combination is jaccard similarity and hamming distance using part of the phoneme translation, which has a 90.05% accuracy. The reason for this combination giving the best accuracy still remains to be an open question as there is no clear separation between the two classes.

In the future, this model can be extended to classify end rhymes as perfect rhymes and slant rhymes. This can be done by experimenting with the feature combinations to see which ones give better results for each type of rhyme. This experiment will also clarify whether vowel and consonant weights can detect slant rhymes. It may also clarify whether the unclear separation between the classes when using jaccard similarity and hamming distance is indeed because of the slant rhymes. Furthermore, to aid poem generation, the model should be able to give a list of rhyming words for every word given to it. This can be done with a very large dataset and grouping the words based on their pronunciations.

9 Responsible Research

This research was conducted as part of the course CSE3000 at Delft University of Technology. It was performed without funding and there was no conflict of personal interest. The source code along with the parameters, libraries and packages used can be found on the GitHub repository[8], thus allowing for reproducibility. The data was collected from an external

[8]https://github.com/simran0413/RhymeDetection

source. Some rhyming pairs were removed and a justified explanation has been provided in appendix A. The datasets will be kept confidential as per the license granted. Nonetheless, the steps taken to collect the data and clean the dataset are described in methodology and appendix A. Since probability played a role in the experimental setup and the data from the dataset was taken at random, some results may come out slightly different when the experiment is reproduced. Cross validation was used to obtain the results. These results had a very small standard deviation, therefore, repeating the experiment will give results that will not be significantly different from the ones obtained in this research. However, the general conclusion drawn from the results will remain the same.

References

[1] Longest common substring problem, Apr 2021.

[2] Bennet Bergman. Slant rhyme, May 2017.

[3] Steven Bird. Source code for nltk.corpus.reader.cmudict, 2014.

[4] Kathryne Bradesca. Why are rhythm & rhyme important in poems?, Jan 2019.

[5] Benji Davies and Alexis Deacon. Quality children's literature at the heart of all learning.

[6] Wikimedia Foundation. International phonetic alphabet, Jun 2021.

[7] Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating topical poetry. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Nov 2016.

[8] Michael Hickey. The reason for rhyme, Jan 2007.

[9] Hussein Hirjee and Daniel Brown. Using automated rhyme detection to characterize rhyming style in rap music. *Empirical Musicology Review*, 5(4):121–145, 2010.

[10] Nils Hulzebosch, Mostafa Dehghani, and Sander van Splunter. Deeplyricist: Automatic generation of rap lyrics using sequence-to-sequence learning.

[11] Vaibhav Kesarwani. Automatic poetry classification using natural language processing. pages 27–42, 2018.

[12] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

[13] Kevin Lenzo. The cmu pronuouncing dictionary, 2007.

[14] Jason Lineberger. Take online courses. earn college credit. research schools, degrees & careers.

[15] MasterClass. Perfect vs. imperfect rhymes: Definition, uses, and differences - 2021, Nov 2020.

[16] Tomasz P. Szynalski. The sounds of english and the international phonetic alphabet, 2004.

[17] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, Nov 2018.

A Data collection and cleaning

Since the classifier uses a supervised learning method, a large dataset of rhyming words and non-rhyming words was required. Unfortunately, the dictionaries of rhyming words that were made available were very small (100 pairs), thus a self made dictionary was used. Using the webscraper built by Tom Viering and Arman Nasari Jafari , the website www.rhymer.com2 was scraped for end rhymes. To obtain a relatively large and diverse dataset, one syllable and two syllable end rhymes were retrieved. 78,475 words were collected, each having a list of one syllable and two syllable end rhymes.

However, not all of these rhyming pairs had phoneme translations in the CMU Dictionary or IPA American English Dictionary. Thus, the rhyming pairs that did not have phoneme translations were removed from the dataset. This resulted in 44,660 words, with lists of rhyming words. These pairs were removed so that when the data is split into a training set and test set, there is a fair 80:20 ratio of word pairs that have phonemes and can be used by the model. Moreover, keeping these words in the dataset would have negatively affected the performance of the classifier (due to unknown imbalanced datasets).

In order to make a dictionary of non-rhyming pairs, the rhyming pairs dictionary was used, and the rhyming words lists were swapped around. Each word was mapped to two lists of rhyming words that belonged to other words. These non-rhyming words were filtered to remove any words that rhyme with the main word.

The dataset of 44,660 words with large lists of rhyming words made computation very expensive. Thus, a smaller dataset was used to train the model. The first 25,000 words made up the True category while the last 25,000 made up the False category. Each of these words had a list of about 5 rhyming or non-rhyming words. The reason for using the last 25,000 words for the false category comes from previous experiments[12]. In these experiments, they discovered that having false words that were not part of the rhyming category can improve the performance of the model. To sum up, the smaller dataset contains a total of 248,905 rhyming and non-rhyming pairs.

B All results

Edit Distance	Hamming Distance	Jaccard Similarity	Longest Common Substring	Vowel and Constant Weights	Average Accuracy	std (training score)	Coeffients	Log loss	Test Accuracy (1 percent true)	std (Test Accuracy (1 percent true))	Test Accuracy (1 percent false)	std(Test Accuracy (1 percent false))
✔	✔	✔	✔	✔	89.88	0.1700	[-0.135, -7.408, 9.541, 0.0, -1.037]	0.25	89.92	0.12	90.03	0.05
✔	✔	✔	✔		89.80	0.0800	[0.0, -7.165, 9.180, 0.0]	0.25				
✔	✔	✔		✔	89.92	0.1300	[-0.291, -7.403, 9.331, -1.02]	0.25	90.00	0.15	89.91	0.13
✔	✔		✔	✔	89.49	0.5100	[0.0, -15.493, 0.162, 0.0]	0.25	89.63	0.48	89.86	0.11
✔		✔	✔	✔	89.78	0.3400	[-1.413, 15.370, 0.0, -0.72]	0.25			89.83	0.35
	✔	✔	✔	✔	89.98	0.1600	[-7.387, 9.714, 0.0, -1.058]	0.25	89.86	0.11	89.90	0.11
	✔		✔	✔	89.20	0.4500	[-15.043, 0.715, 0.0]	0.25				
✔			✔	✔	84.96	0.2200	[-6.602, 3.348, 4.570]	0.33	84.90	0.16	85.02	0.20
			✔	✔	85.17	0.7800	[9.157, 5.701]	0.34				
✔	✔	✔			89.73	0.2900	[-0.218, -6.859, 9.176]	0.25				
	✔	✔			90.05	0.1600	[-7.163, 9.026]	0.25	89.86	0.06	89.88	0.09
✔		✔			89.81	0.1300	[-1.116, 14.979]	0.25	89.62	0.38	89.75	0.09
✔	✔				89.72	0.4600	[-0.313, -15.930]	0.25	90.03	0.09	89.63	0.46
				✔	83.27	0.1900	[11.671]	0.41				
			✔		82.74	0.1400	[12.423]	0.35				
✔			✔		84.26	0.2000	[-8.143, 4.473]	0.35				
	✔			✔	89.78	0.4800	[-16.119, 0.0]	0.25	90.01	0.15	89.94	0.44
	✔				89.97	0.3700	[-16.277]	0.25				
✔					84.10	0.1600	[-11.687]	0.35				
		✔			89.73	0.0900	[16.233]	0.25				
		✔	✔		89.73	0.1100	[16.288, 0.0]	0.26	89.83	0.09	89.74	0.08
	✔	✔		✔	89.91	0.1000	[-7.158, 9.899, 1.000]	0.25				

Figure 12: Detailed results of all the feature combinations performed when part of the phoneme translation was used. Results include the average test score, standard deviation of test score, coefficient weights, log loss, test scores of two imbalanced sets with their standard deviations

Edit Distance	Hamming Distance	Jaccard Similarity	Longest Common Substring	Vowel and Constant Weights	Average Accuracy	std (accuracy)	Coeffients	Log Loss	Test Accuracy (one percent true)	std (Test Accuracy (one percent true))	Test Accuracy (one percent false)	std (Test Accuracy (one percent false))
✔	✔	✔	✔	✔	88.07	0.1800	[-1.751, -7.509, 11.190, 0.0, 0.0]	0.27	87.62	0.12	88.73	0.16
✔	✔	✔	✔		88.22	0.1700	[-1.697, -7.702, 11.110, 0.0]	0.27				
✔	✔	✔		✔	88.08	0.1100	[-2.08, -7.375, 11.044, 0.0]	0.27				
✔	✔		✔	✔	88.21	0.2100	[-2.394, -16.032, 0.0, 0.493]	0.28	86.85	0.17		
✔		✔	✔	✔	88.11	0.1500	[-3.413, 16.925, 0.0, 0.0]	0.28	87.64	0.12		
	✔	✔	✔	✔	87.83	0.1500	[-8.189, 11.903, 0.0, 0.0]	0.27	88.18	0.08	87.61	0.15
	✔		✔	✔	87.29	0.3400	[-8.962, 3.026, 3.133]	0.32				
✔			✔	✔	85.16	0.3500	[-6.460, 3.115, 5.207]	0.36	87.43	0.22	84.32	0.29
			✔	✔	82.82	0.1300	[6.276, 6.765]	0.41	83.57	0.17	78.14	0.27
✔	✔	✔			87.93	0.1500	[-3.701, -6.116, 5.763]	0.29				
	✔	✔			87.91	0.1300	[-7.281, 6.816]	0.29	88.21	0.09	87.63	0.11
✔		✔			87.83	0.1400	[-5.080, 8.725]	0.31	87.63	0.10	88.57	0.11
✔	✔				87.71	0.1100	[-4.851, -8.800]	0.31	87.51	0.12	89.65	0.20
				✔	80.28	0.1800	[9.185]	0.46				
			✔		79.13	0.1000	[9.151]	0.47				
✔			✔		83.32	0.1100	[-7.704, 4.288]	0.39	85.70	0.16	80.92	0.11
	✔			✔	87.86	0.1400	[-9.889, 3.466]	0.32	88.08	0.11	88.52	0.11
	✔				87.59	0.3200	[-11.435]	0.32				
✔					83.51	0.1300	[-9.579]	0.40				
		✔			87.57	0.0900	[11.412]	0.32				
		✔	✔		87.10	0.1800	[10.082, 3.651]	0.32	88.39	0.11	87.70	0.11

Figure 13: Detailed results of all the feature combinations performed when the entire phoneme translation was used. Results include the average test score, standard deviation of test score, coefficient weights, log loss, test scores of two imbalanced sets with their standard deviations