Unsupervised Discovery of Rhyme Schemes

Sravana Reddy
Department of Computer Science
The University of Chicago
Chicago, IL 60637
sravana@cs.uchicago.edu

Kevin Knight
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
knight@isi.edu

Abstract

This paper describes an unsupervised, language-independent model for finding rhyme schemes in poetry, using no prior knowledge about rhyme or pronunciation.

1 Introduction

Rhyming stanzas of poetry are characterized by rhyme schemes, patterns that specify how the lines in the stanza rhyme with one another. The question we raise in this paper is: *can we infer the rhyme scheme of a stanza given no information about pronunciations or rhyming relations among words?*

Background A rhyme scheme is represented as a string corresponding to the sequence of lines that comprise the stanza, in which rhyming lines are denoted by the same letter. For example, the limerick's rhyme scheme is *aabba*, indicating that the $1^{st}, 2^{nd}$, and 5^{th} lines rhyme, as do the the 3^{rd} and 4^{th}.

Motivation Automatic rhyme scheme annotation would benefit several research areas, including:

- *Machine Translation of Poetry* There has been a growing interest in translation under constraints of rhyme and meter, which requires training on a large amount of annotated poetry data in various languages.
- *'Culturomics'* The field of digital humanities is growing, with a focus on statistics to track cultural and literary trends (partially spurred by projects like the Google Books Ngrams[1]). Rhyming corpora could be extremely useful for large-scale statistical analyses of poetic texts.
- *Historical Linguistics/Study of Dialects* Rhymes of a word in poetry of a given time period or dialect region provide clues about its pronunciation in that time or dialect, a fact that is often taken advantage of by linguists (Wyld, 1923). One could automate this task given enough annotated data.

An obvious approach to finding rhyme schemes is to use word pronunciations and a definition of rhyme, in which case the problem is fairly easy. However, we favor an unsupervised solution that utilizes no external knowledge for several reasons.

- Pronunciation dictionaries are simply not available for many languages. When dictionaries are available, they do not include all possible words, or account for different dialects.
- The definition of rhyme varies across poetic traditions and languages, and may include slant rhymes like *gate/mat*, 'sight rhymes' like *word/sword*, assonance/consonance like *shore/alone*, *leaves/lance*, etc.
- Pronunciations and spelling conventions change over time. Words that rhymed historically may not anymore, like *prove* and *love* – or *proued* and *beloued*.

2 Related Work

There have been a number of recent papers on the automated annotation, analysis, or translation of po-

[1] http://ngrams.googlelabs.com/

etry. Greene et al. (2010) use a finite state transducer to infer the syllable-stress assignments in lines of poetry under metrical constraints. Genzel et al. (2010) incorporate constraints on meter and rhyme (where the stress and rhyming information is derived from a pronunciation dictionary) into a machine translation system. Jiang and Zhou (2008) develop a system to generate the second line of a Chinese couplet given the first. A few researchers have also explored the problem of poetry generation under some constraints (Manurung et al., 2000; Netzer et al., 2009; Ramakrishnan et al., 2009). There has also been some work on computational approaches to characterizing rhymes (Byrd and Chodorow, 1985) and global properties of the rhyme network (Sonderegger, 2011) in English. To the best of our knowledge, there has been no language-independent computational work on finding rhyme schemes.

3 Finding Stanza Rhyme Schemes

A collection of rhyming poetry inevitably contains *repetition of rhyming pairs*. For example, the word *trees* will often rhyme with *breeze* across different stanzas, even those with different rhyme schemes and written by different authors. This is partly due to sparsity of rhymes – many words that have no rhymes at all, and many others have only a handful, forcing poets to reuse rhyming pairs.

In this section, we describe an unsupervised algorithm to infer rhyme schemes that harnesses this repetition, based on a model of stanza generation.

3.1 Generative Model of a Stanza

1. Pick a rhyme scheme r of length n with probability $P(r)$.
2. For each $i \in [1, n]$, pick a word sequence, choosing the *last*[2] word x_i as follows:

   (a) If, according to r, the i^{th} line does not rhyme with any previous line in the stanza, pick a word x_i from a vocabulary of line-end words with probability $P(x_i)$.

   (b) If the i^{th} line rhymes with some previous line(s) j according to r, choose a word x_i that rhymes with the last words of all such lines with probability $\prod_{j<i:r_i=r_j} P(x_i|x_j)$.

The probability of a stanza x of length n is given by Eq. 1. $I_{i,r}$ is the indicator variable for whether line i rhymes with at least one previous line under r.

$$P(x) = \sum_{r \in R} P(r)P(x|r) =$$

$$\sum_{r \in R} P(r) \prod_{i=1}^{n} (1 - I_{i,r})P(x_i) + I_{i,r} \prod_{j<i:r_i=r_j} P(x_i|x_j) \quad (1)$$

3.2 Learning

We denote our data by X, a set of stanzas. Each stanza x is represented as a sequence of its line-end words, $x_i, \ldots x_{len(x)}$. We are also given a large set R of all possible rhyme schemes.[3]

If each stanza in the data is generated independently (an assumption we relax in §4), the log-likelihood of the data is $\sum_{x \in X} \log P(x)$. We would like to maximize this over all possible rhyme scheme assignments, under the latent variables θ, which represents *pairwise rhyme strength*, and ρ, the distribution of rhyme schemes. $\theta_{v,w}$ is defined for all words v and w as a non-negative real value indicating how strongly the words v and w rhyme, and ρ_r is $P(r)$.

The expectation maximization (EM) learning algorithm for this formulation is described below. The intuition behind the algorithm is this: after one iteration, $\theta_{v,w} = 0$ for all v and w that never occur together in a stanza. If v and w co-occur in more than one stanza, $\theta_{v,w}$ has a high pseudo-count, reflecting the fact that they are likely to be rhymes.

Initialize: ρ and θ uniformly (giving θ the same positive value for all word pairs).

Expectation Step: Compute $P(r|x) = P(x|r)\rho_r / \sum_{q \in R} P(x|q)\rho_q$, where

$$P(x|r) = \prod_{i=1}^{n} (1 - I_{i,r})P(x_i) + I_{i,r} \prod_{j<i:r_i=r_j} \theta_{x_i,x_j} / \sum_{w} \theta_{w,x_i} \quad (2)$$

[2] A rhyme may span more than one word in a line – for example, *laureate... / Tory at... / are ye at* (Byron, 1824), but this is uncommon. An extension of our model could include a latent variable that selects the entire rhyming portion of a line.

[3] While the number of rhyme schemes of length n is technically the number of partitions of an n- element set (the Bell number), only a subset of these are typically used.

$P(x_i)$ is simply the relative frequency of the word x_i in the data.

Maximization Step: Update θ and ρ:

$$\theta_{v,w} = \sum_{r,x:v \text{ rhymes with } w} P(r|x) \quad (3)$$

$$\rho_r = \sum_{x \in X} P(r|x) / \sum_{q \in R, x \in X} P(q|x) \quad (4)$$

After Convergence: Label each stanza x with the best rhyme scheme, $\arg\max_{r \in R} P(r|x)$.

3.3 Data

We test the algorithm on rhyming poetry in English and French. The English data is an edited version of the public-domain portion of the corpus used by Sonderegger (2011), and consists of just under 12000 stanzas spanning a range of poets and dates from the 15^{th} to 20^{th} centuries. The French data is from the ARTFL project (Morrissey, 2011), and contains about 3000 stanzas. All poems in the data are manually annotated with rhyme schemes.

The set R is taken to be all the rhyme schemes from the gold standard annotations of both corpora, numbering 462 schemes in total, with an average of 6.5 schemes per stanza length. There are 27.12 candidate rhyme schemes on an average for each English stanza, and 33.81 for each French stanza.

3.4 Results

We measure the **accuracy** of the discovered rhyme schemes relative to the gold standard. We also evaluate for each word token x_i, the set of words in $\{x_{i+1}, x_{i+2}, \ldots\}$ that are found to rhyme with x_i by measuring **precision and recall**. This is to account for partial correctness – if *abcb* is found instead of *abab*, for example, we would like to credit the algorithm for knowing that the 2^{nd} and 4^{th} lines rhyme.

Table 1 shows the results of the algorithm for the entire corpus in each language, as well as for a few sub-corpora from different time periods.

3.5 Orthographic Similarity Bias

So far, we have relied on the repetition of rhymes, and have made no assumptions about word pronunciations. Therefore, the algorithm's performance is strongly correlated[4] with the predictability of rhyming words. For writing systems where the written form of a word approximates its pronunciation, we have some additional information about rhyming: for example, English words ending with similar characters are most probably rhymes. We do not want to assume *too much* in the interest of language-independence – following from our earlier point in §1 about the nebulous definition of rhyme – but it is safe to say that rhyming words involve some orthographic similarity (though this does not hold for writing systems like Chinese). We therefore initialize θ at the start of EM with a simple similarity measure: (Eq. 5). The addition of $\epsilon = 0.001$ ensures that words with no letters in common, like *new* and *you*, are not eliminated as rhymes.

$$\theta_{v,w} = \frac{\text{\# letters common to } v \text{ \& } w}{\min(len(v), len(w))} + \epsilon \quad (5)$$

This simple modification produces results that outperform the naïve baselines for most of the data by a considerable margin, as detailed in Table 2.

3.6 Using Pronunciation, Rhyming Definition

How does our algorithm compare to a standard system where rhyme schemes are determined by predefined rules of rhyming and dictionary pronunciations? We use the accepted definition of rhyme in English: two words rhyme if their final stressed vowels and all following phonemes are identical. For every pair of English words v, w, we let $\theta_{v,w} = 1 + \epsilon$ if the CELEX (Baayen et al., 1995) pronunciations of v and w rhyme, and $\theta_{v,w} = 0 + \epsilon$ if not (with $\epsilon = 0.001$). If either v or w is not present in CELEX, we set $\theta_{v,w}$ to a random value in $[0, 1]$. We then find the best rhyme scheme for each stanza, using Eq. 2 with uniformly initialized ρ.

Figure 1 shows that the accuracy of this system is generally much lower than that of our model for the sub-corpora from before 1750. Performance is comparable for the 1750-1850 data, after which we get better accuracies using the rhyming definition than with our model. This is clearly a reflection of language change; older poetry differs more significantly in pronunciation and lexical usage from con-

[4] For the five English sub-corpora, $R^2 = 0.946$ for the negative correlation of accuracy with entropy of rhyming word pairs.

Table 1: Rhyme scheme accuracy and F-Score (computed from average precision and recall over all lines) using our algorithm for *independent stanzas*, with *uniform initialization* of θ. Rows labeled 'All' refer to training and evaluation on all the data in the language. Other rows refer to training and evaluating on a particular sub-corpus only. Bold indicates that we outperform the naïve baseline, where most common scheme of the appropriate length from the gold standard of the entire corpus is assigned to every stanza, and italics that we outperform the 'less naïve' baseline, where we assign the most common scheme of the appropriate length from the gold standard of the given sub-corpus.

	Sub-corpus (time-period)	*Sub-corpus overview*			*Accuracy (%)*			*F-Score*		
		# of stanzas	*Total # of lines*	*# of line-end words*	*EM induction*	*Naïve baseline*	*Less naïve baseline*	*EM induction*	*Naïve baseline*	*Less naïve*
En	All	11613	93030	13807	***62.15***	56.76	60.24	***0.79***	0.74	0.77
	1450-1550	197	1250	782	17.77	53.30	97.46	0.41	0.73	0.98
	1550-1650	3786	35485	7826	**67.17**	62.28	74.72	**0.82**	0.78	0.85
	1650-1750	2198	20110	4447	***87.58***	58.42	82.98	***0.94***	0.68	0.91
	1750-1850	2555	20598	5188	31.00	69.16	74.52	0.65	0.83	0.87
	1850-1950	2877	15587	4382	***50.92***	37.43	49.70	***0.81***	0.55	0.68
Fr	All	2814	26543	10781	**40.29**	39.66	64.46	**0.58**	0.57	0.80
	1450-1550	1478	14126	7122	28.21	58.66	77.67	0.59	0.83	0.89
	1550-1650	1336	12417	5724	**52.84**	18.64	61.23	**0.70**	0.28	0.75

temporary dictionaries, and therefore, benefits more from a model that assumes no pronunciation knowledge. (While we may get better results on older data using dictionaries that are historically accurate, these are not easily available, and require a great deal of effort and linguistic knowledge to create.)

Initializing θ as specified above and then running EM produces some improvement compared to orthographic similarity (Table 2).

4 Accounting for Stanza Dependencies

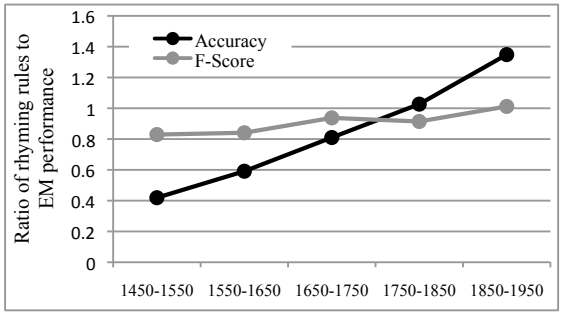
So far, we have treated stanzas as being independent of each other. In reality, stanzas in a poem are usually generated using the same or similar rhyme schemes. Furthermore, some rhyme schemes span multiple stanzas – for example, the Italian form *terza rima* has the scheme *aba bcb cdc...* (the 1^{st} and 3^{rd} lines rhyme with the 2^{nd} line of the previous stanza).

4.1 Generative Model

We model stanza generation within a poem as a Markov process, where each stanza is conditioned on the previous one. To generate a poem y consisting of m stanzas, for each $k \in [1, m]$, generate a stanza x^k of length n^k as described below:

1. If $k = 1$, pick a rhyme scheme r^k of length n^k with probability $P(r^k)$, and generate the stanza as in the previous section.

Figure 1: Comparison of EM with a definition-based system



(a) Accuracy and F-Score ratios of the rhyming-definition-based system over that of our model with orthographic similarity. The former is more accurate than EM for post-1850 data (ratio > 1), but is outperformed by our model for older poetry (ratio < 1), largely due to pronunciation changes like the Great Vowel Shift that alter rhyming relations.

	Found by EM	Found by definitions
1450-1550	*left/craft*, *shone/done*	*edify/lie*, *adieu/hue*
1550-1650	*appeareth/weareth*, *speaking/breaking*, *proue/moue*, *doe/two*	*obtain/vain*, *amend/depend*, *breed/heed*, *prefers/hers*
1650-1750	*most/cost*, *presage/rage*, *join'd/mind*	*see/family*, *blade/shade*, *noted/quoted*
1750-1850	*desponds/wounds*, *o'er/shore*, *it/basket*	*gore/shore*, *ice/vice*, *head/tread*, *too/blew*
1850-1950	*off/love*, *lover/half-over*, *again/rain*	*old/enfold*, *within/win*, *be/immortality*

(b) Some examples of rhymes in English found by EM but not the definition-based system (due to divergence from the contemporary dictionary or rhyming definition), and vice-versa (due to inadequate repetition).

Table 2: Performance of EM with θ initialized by *orthographic similarity* (§3.5), *pronunciation-based rhyming definitions* (§3.6), and the *HMM for stanza dependencies* (§4). Bold and italics indicate that we outperform the naïve baselines shown in Table 1.

	Sub-corpus (time-period)	*Accuracy (%)*				*F-Score*			
		HMM stanzas	*Rhyming definition init.*	*Orthographic initialization*	*Uniform initialization*	*HMM stanzas*	*Rhyming defn. init.*	*Ortho. init.*	*Uniform init.*
En	All	***72.48***	***64.18***	***63.08***	***62.15***	***0.88***	***0.84***	***0.83***	***0.79***
	1450-1550	**74.31**	**75.63**	**69.04**	17.77	**0.86**	**0.86**	**0.82**	0.41
	1550-1650	***79.17***	**69.76**	**71.98**	**67.17**	***0.90***	***0.86***	***0.88***	**0.82**
	1650-1750	***91.23***	***91.95***	***89.54***	***87.58***	***0.97***	***0.97***	***0.96***	***0.94***
	1750-1850	49.11	42.74	33.62	31.00	0.82	0.77	0.70	0.65
	1850-1950	***58.95***	***57.18***	***54.05***	***50.92***	***0.90***	***0.89***	***0.84***	***0.81***
Fr	All	**56.47**	-	**48.90**	**40.29**	***0.81***	-	**0.75**	**0.58**
	1450-1550	**61.28**	-	35.25	28.21	**0.86**	-	0.71	0.59
	1550-1650	***67.96***	-	***63.40***	**52.84**	***0.79***	-	***0.77***	**0.70**

2. If $k > 1$, pick a scheme r^k of length n^k with probability $P(r^k|r^{k-1})$. If no rhymes in r^k are shared with the previous stanza's rhyme scheme, r^{k-1}, generate the stanza as before. If r^k shares rhymes with r^{k-1}, generate the stanza as a continuation of x^{k-1}. For example, if x^{k-1} = [*dreams, lay, streams*], and r^{k-1} and r^k = *aba* and *bcb*, the stanza x^k should be generated so that x_1^k and x_3^k rhyme with *lay*.

4.2 Learning

This model for a poem can be formalized as an *autoregressive HMM*, an hidden Markov model where each observation is conditioned on the previous observation as well as the latent state. An observation at a time step k is the stanza x^k, and the latent state at that time step is the rhyme scheme r^k. This model is parametrized by θ and ρ, where $\rho_{r,q} = P(r|q)$ for all schemes r and q. θ is initialized with orthographic similarity. The learning algorithm follows from EM for HMMs and our earlier algorithm.

Expectation Step: Estimate $P(r|x)$ for each stanza in the poem using the forward-backward algorithm. The 'emission probability' $P(x|r)$ for the first stanza is same as in §3, and for subsequent stanzas $x^k, k > 1$ is given by:

$$P(x^k|x^{k-1}, r^k) = \prod_{i=1}^{n^k}(1 - I_{i,r^k})P(x_i^k) + I_{i,r^k} \prod_{j<i:r_i^k=r_j^k} P(x_i^k|x_j^k) \prod_{j:r_i^k=r_j^{k-1}} P(x_i^k|x_j^{k-1}) \quad (6)$$

Maximization Step: Update ρ and θ analogously to HMM transition and emission probabilities.

4.3 Results

As Table 2 shows, there is considerable improvement over models that assume independent stanzas. The most gains are found in French, which contains many instances of 'linked' stanzas like the *terza rima*, as well as English data containing long poems made of several stanzas with the same scheme.

5 Future Work

Some possible extensions of our work include automatically generating the set of possible rhyme schemes R, and incorporating partial supervision into our algorithm as well as better ways of using and adapting pronunciation information when available. We would also like to test our method on a range of languages and texts.

To return to the motivations, one could use the discovered annotations for machine translation of poetry, or to computationally reconstruct pronunciations, which is useful for historical linguistics as well as other applications involving out-of-vocabulary words.

Acknowledgments

We would like to thank Morgan Sonderegger for providing most of the annotated English data in the rhyming corpus and for helpful discussion, and the anonymous reviewers for their suggestions.

References

R. H. Baayen, R. Piepenbrock, and L. Gulikers. 1995. *The CELEX Lexical Database (CD-ROM)*. Linguistic Data Consortium.

Roy J. Byrd and Martin S. Chodorow. 1985. Using an online dictionary to find rhyming words and pronunciations for unknown words. In *Proceedings of ACL*.

Lord Byron. 1824. Don Juan.

Dmitriy Genzel, Jakob Uszkoreit, and Franz Och. 2010. "Poetic" statistical machine translation: Rhyme and meter. In *Proceedings of EMNLP*.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of EMNLP*.

Long Jiang and Ming Zhou. 2008. Generating Chinese couplets using a statistical MT approach. In *Proceedings of COLING*.

Hisar Maruli Manurung, Graeme Ritchie, and Henry Thompson. 2000. Towards a computational model of poetry generation. In *Proceedings of AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

Robert Morrissey. 2011. ARTFL : American research on the treasury of the French language. http://artfl-project.uchicago.edu/content/artfl-frantext.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku : Generating Haiku with word associations norms. In *Proceedings of the NAACL workshop on Computational Approaches to Linguistic Creativity*.

Ananth Ramakrishnan, Sankar Kuppan, and Sobha Lalitha Devi. 2009. Automatic generation of Tamil lyrics for melodies. In *Proceedings of the NAACL workshop on Computational Approaches to Linguistic Creativity*.

Morgan Sonderegger. 2011. Applications of graph theory to an English rhyming corpus. *Computer Speech and Language*, 25:655–678.

Henry Wyld. 1923. *Studies in English rhymes from Surrey to Pope*. J Murray, London.