Unsupervised Rhyme Scheme Identification in Hip Hop Lyrics Using Hidden Markov Models

Karteek Addanki and Dekai Wu

Human Language Technology Center
Dept. of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
{vskaddanki,dekai}@cs.ust.hk

Abstract. We attack a woefully under-explored language genre—lyrics in music—introducing a novel hidden Markov model based method for completely unsupervised identifica-tion of rhyme schemes in hip hop lyrics, which to the best of our knowledge, is the first such effort. Unlike previous approaches that use supervised or semi-supervised approaches for the task of rhyme scheme identification, our model does not assume any prior phonetic or labeling information whatsoever. Also, unlike previous work on rhyme scheme identification, we attack the difficult task of hip hop lyrics in which the data is more highly unstructured and noisy. A novel feature of our approach comes from the fact that we do not manually segment the verses in lyrics according to any pre-specified rhyme scheme, but instead use a number of hidden states of varying rhyme scheme lengths to automatically impose a *soft segmentation*. In spite of the level of difficulty of the challenge, we nevertheless were able to obtain a surprisingly high precision of 35.81% and recall of 57.25% on the task of identifying the rhyming words, giving a total f-score of 44.06%. These encouraging results were obtained in the face of highly noisy data, lack of clear stanza segmentation, and a very wide variety of rhyme schemes used in hip hop.

1 Introduction

Among the many genres of language that have been studied in computational linguistics and spoken language processing, there has been an inexplicable dearth of work on lyrics in music, despite the major impact that this form of language has across almost all human cultures. We aim to spur research addressing this gap, by bringing the statistical methods of language technologies to bear upon modeling issues in song lyrics. An ideal starting point for this investigation is hip hop, a genre of music that emphasizes rapping, spoken or chanted rhyming lyrics against strong beats or simple melodies. This complex domain presents a fertile range of challenges for learning since there is typically no obvious structure in terms of rhyme scheme, meter, or overall meaning.

As a first challenge in learning hip hop lyrics, we attack the problem of rhyme scheme identification. Rhyming is central to many genres of music, and is taken

A.-H. Dediu et al. (Eds.): SLSP 2013, LNAI 7978, pp. 39–50, 2013.

to an extreme level of importance in hip hop. Although statistical methods have been applied to identify rhyming patterns in poetry, most of the approaches either restricted themselves to a narrow and structured domain such as sonnets. Some of the approaches incorporated morphological and phonological features that are language specific while some used supervised learning methods using labeled data.

The domain of hip hop lyrics is very *unstructured* when compared to classical poetry which makes it hard for the model to make any kind of assumptions about the data. We use to the term *unstructured* to describe degree of permitted variation in the meter of the lyrics and the large number of colloquial words and slangs that are common in hip hop subculture. The variance in the permitted meter makes it hard to make any assumptions about the stress patterns of verses in order to identify the rhyming words. The broad range of terms used in hip hop make it hard to use of off-the-shelf NLP tools for doing phonological and/or morphological analysis. Problems discussed above are exacerbated by differences in intonation of the same word and lack of robust transcription.[1]

However, hip hop lyrics *do* contain enough information to identify words that rhyme as rhyming is one of the characteristic features of hip hop lyrics. This high degree of rhyming enable us to learn the rhyme schemes in a completely unsupervised manner without making any language dependent assumptions. Identification of rhyme schemes also provides useful information about the overall structure of the songs and the variation of rhyme schemes within a single song. Further, accurate identification of rhyming words will enable compilation of large scale rhyming dictionaries which are currently manually compiled [1]

In this paper, we introduce a novel method for completely unsupervised identification of rhyme schemes in hip hop lyrics that utilizes an HMM model, which to the best of our knowledge is first such effort. We discuss some of the previous work that applies statistical methods to the problems similar to rhyme scheme detection in various domains in section 2. In section 3 we discuss our algorithm and provide motivation for our choice of the HMM structure. Sections 4 and 5 talk about our experimental setup and results respectively. Conclusions and future research directions are proposed in section 6.

2 Related Work

Most of the work in the past involved automatic analysis of poetry to discover word to word relationships, stress patterns [2] and rhyming words [3]. Results were combined with language models to generate new poems [2,4,5].

The stress pattens of words in English rhythmic poetries were analyzed by [2]. Their task was to assign stress patterns to words where the meter of each line is known. A meter is the beat that is heard when the poem is read aloud. For example, Iambic is a common meter where the words sound like da-DUM da-DUM da-DUM. The words were labeled S for light tones and S' for stressed tones, e.g. beLIEVE would be labeled $S\ S'$. Finite state transducer was applied to all

[1] `http://languagelog.ldc.upenn.edu/nll/?p=2824`

the words in Shakespeare's sonnets to assign probable stress patterns. Combining stress patterns with language models for fluency, a poem was generated.

Apart from stress patterns, the positions of the words in a line of Chinese poems were also suitable to learn word associations. Researchers have used statistical machine translation (SMT) to generate lines in a Chinese couplet, duilian [5]. Given a single line of a couplet, an n-best list of "translations" that were potential candidates for the next line were generated. Subsequently, each of these lines were analyzed and eliminated according by applying linguistic constrains to obtain the most suitable option.

SMT algorithms were used in conjunction with stress patterns and rhymes found in a pronunciation dictionary to produce translations of poems [4]. Although many constraints were applied in translating full verses of poems, in practice it was difficult to satisfy all the constraints. For example, some of the translations contained rhyming words but did not give the same meter to the line.

Graph theory had been applied to analyze the rhyming and therefore inferred the pronunciation of words in old poetry for an English rhyming corpus [6]. Training data was obtained from poetries where the pronunciation of the words could found in a pronunciation dictionary. The pronunciations were represented by the International Phonetic Alphabet(IPA) symbols. A word was assumed to rhyme with another when the IPA symbols ended similarly. The rhyming words were organized into rhyme graphs where the nodes were words and edges were rhymes. However, this method gave large clusters of words that had the same IPA endings but did not fully rhyme, such as 'bloom' and 'numb'.

Automatic lyric generation given melodies was also investigated [7]. The training data for the model included melodies and the associated lyrics, where the lyrics were represented using the *KNM* system, where *K*, *N* and *M* represented the long vowel, short vowels and consonants respectively. The trained model was then used to label any input melody with the *KNM* system. Subsequently, words were chosen to fit the *KNM* system of the lyrics and constraints were applied to make sure that the lyrics were meaningful.

Most of the past work in this vein can be classified into two categories. The first category uses some form of prior linguistic knowledge about the domain, such as pronunciation dictionaries or phonological or morphological information. The second category uses unsupervised learning methods to identify word association probabilities but enforces harsher constraints warranted by the domain, such as a set number of words in a line, or a set meter. Our current work differs in the sense that we present a completely unsupervised model on a domain that inherently has very few such constraints. For example, not all words in the lyrics are a part of the lexicon. Hip hop does not require a set number of syllables in a line, unlike poems (especially in classical poetry where, for example, an octave has exactly 10 syllables per line and 8 lines per stanza). Also, surprising and unlikely rhymes in hip hop are frequently achieved via intonation and assonance, which makes it hard to apply prior phonological constraints.

Our current work was inspired by the work done on unsupervised identification of rhyming words in poems [3]. To learn the probabilities of rhyming words, a set of chosen hidden rhyming patterns were applied by labeling the last word of each line with a rhyme label, such as **A** in a **ABAB** rhyme scheme. However, unlike the original model our model cannot afford to make any assumptions about the meter of the lyrics and the length of each rhyming scheme. Hence we extend the model to accommodate the more unstructured domain of hip hop lyrics. Further details about our extension are provided in the following section.

3 Detection of Rhyming Words

A collection of hip hop lyrics typically contains *repetition of rhyming pairs*. For example, the word *y'all* rhymes with *real* across different songs by different artists. This can be partly attributed to the scarcity of the rhymes and to the similarity in the pronunciation caused by similarity in the music "beat" patterns.

In this section, we will describe an unsupervised algorithm that harnesses this repetition, based on a verse generation model. This algorithm shares some similarity to the one proposed in [3]. But their model was proposed for English poetry which is much more structured than hip hop lyrics. In order to account for the wide range of variations in the domain of hip hop lyrics, we generalize their model to account for the varying lyric lengths and possible sequences of rhyming patterns in hip hop lyrics. We also use the exhaustive set of all possible rhyme schemes (up to length 4) instead of manually selecting rhyme schemes. The states are selected so that it is possible to generate all the rhyme schemes and can be represented using a minimum number of states.

In this section, we will describe our model to identify the rhyming schemes and the experimental pipeline that was used to obtain and process the lyrics. A brief description of the choice of the rhyme schemes and the underlying motivation is also provided.

3.1 Generative Model of a Verse

We assume that our data is a set of independent and identically distributed verses. We propose the following generative model for a verse given a HMM of rhyming schemes. The rhyming schemes could be of varying length and the choice of the appropriate rhyming schemes is discussed in the subsequent subsections.

1. We have a fully connected HMM with $i+1$ states, where each state represents a different rhyme scheme such as **AA**, **ABAB** *etc.* The output of each state has variable length and transitions are possible from one state to all other states as shown in Figure 1. The start state is not shown for the sake of brevity.
2. For each $l \in [1, n]$, pick a word sequence $x_{1...T}$, choosing the last words and the words before commas in each line as follows[2] :

[2] Manual inspection of the data revealed that words before commas also rhyme in addition to words at the end of the line.

(a) Choose a rhyme scheme r corresponding to the state i, from the start state with a probability of π_i using the transition probability of the HMM.
(b) If, according to r, the i^{th} line does not rhyme with any previous line in the verse, pick as word x_i from a vocabulary of line-end words with probability $P(x_i)$
(c) 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 $\prod_{j<i:r_i=r_j} P(x_i|x_j)$.
(d) Choose a rhyme scheme r corresponding to the state j with a probability of a_{ij} using the transition probability of the HMM.
(e) Goto step (b).

The probability of a verse $P(x)$ is obtained by summing over all state sequences $\mathcal{S}$ that generate the x as shown in 1, where o_t is the output of state s_t, S is a sequence of states $s_1, \ldots, s_t$ and π_{s_t} denotes the probability that s_t will be the first state in the HMM.

$$P(x) = \sum_{S \in \mathcal{S}} P(x)P(x|S) = \sum_{S \in \mathcal{S}} (\prod_{t \in [1,|S|]} p(o_t|s_t)a_{s_t s_{t+1}} + \pi_{s_1}) \tag{1}$$

The probability of generating a sequence of tokens o_t of length n given a state s_t corresponding to a rhyme scheme r (similar to the emission probabilities in traditional HMMs) is given by 2. $I_{i,r}$ is the indicator variable for whether line i rhymes with at least one previous line under r.

$$P(o_t|s_t) = P(x|s_t) = \prod_{i=1}^{n} (1 - I_{i,r})P(x_i) + I_{i,r} \prod_{j<i:r_i=r_j} P(x_i|x_j) \tag{2}$$

3.2 Learning

We denote our data by $\mathcal{X}$, a set of verses. Each verse x is represented as a sequence of the line-end tokens and words before commas, $x_1, \ldots, x_l$. We also have a fully connected HMM, with a finite number of states, that is capable of producing all the partitions of up to length 4 (details are given the next subsection) via different path sequences. If each verse in the data is generated independently the log likelihood of the data is $\sum_{x \in \mathcal{X}} \log P(x)$. We apply expectation maximization learning algorithm to maximize this over all possible rhyme sequences produced by the HMM. While doing so, we re-estimate the latent variables θ, which represents a *pairwise rhyming strength*, and π_i,a_{ij}, the initial and transition probabilities for each of the states in the HMM. 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.

The formulation described above is slightly different from the traditional parameter learning problem of HMMs in the sense that states do not have a multinomial distribution in terms of the emitted variables (or alphabets in a

FSA formulation). They do however, share the distribution of *pairwise rhyming strength* parameter, θ that needs to be updated instead of the emission probabilities in the maximization step. The $\theta_{v,w}$ reflect the co-occurrence of v and w which in turn indicates the probability that they might be rhymes.

Initialize: θ,π and a uniformly (giving θ the same positive value for all word pairs that co-occur in a verse)

Expectation Step: Compute $P(r|x_{u,\ldots,v}) = P(s_i|x_{u,\ldots,v})$, where r is the rhyme scheme of the state s_i. $P(s_i|x_{u,\ldots,v}) = \alpha_i(u,\ldots,v)\beta_i(u,\ldots,v)$ which indicates the probability of being in the state i given the tokens in the sequence. The likelihood of generating the sequence $x_{u,\ldots,v}$ given a rhyme scheme r (or a state s_i) is

$$P(x|r) = \prod_{i=1}^{n}(1 - I_{i,r})P(x_i) + I_{i,r} \prod_{j<i:r_i=r_j} \frac{\theta_{x_i,x_j}}{\sum_w \theta_{w,x_j}} \tag{3}$$

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

Maximization Step: Update θ, a and π, where the state s_t has a rhyming scheme r

$$\theta_{v,w} = \sum_{r,x:v \text{ rhymes with } w} P(r|x) = \sum_{r,x:v \text{ rhymes with } w} \gamma_{s_t}(x) \tag{4}$$

$$\gamma_x(s_t) = \frac{\alpha_x(s_t)\beta_x(s_t)}{P(x)} \tag{5}$$

where $\alpha_x(s_t)$ and $\beta_x(s_t)$ are the forward and backward probabilities for the state s_t outputting x. The update rules for the transition probabilities are similar to that of a usual HMM formulation where $u-w$ is the rhyme length of state i.

$$\xi_{u,v}(i,j) = \frac{\alpha_{w,u}(i)a_{ij}P(x_{u,\ldots,v}|j)}{P(x)} \tag{6}$$

$$a_{ij} = \frac{\sum_{\forall u,\forall v} \xi_{u,v}(i,j)}{\sum_{\forall u,\forall v} \gamma_{u,v}(i,j)} \tag{7}$$

where $\gamma_{u,v}(i,j)$ is the same as the one mentioned in 5.

After convergence: Label each verse x with the best rhyme scheme, $\arg\max_{S\in\mathcal{S}} P(S|x)$

3.3 Choosing the HMM States

As mentioned earlier, the total number of rhyming patterns given a sequence of length n is n^{th} Bell number[3] which is also equal to the total number of partitions of size n. It is evident that the number of states grow exponentially

[3] `http://oeis.org/A000110`

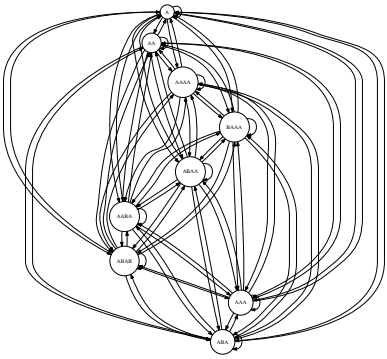

Fig. 1. Fully connected Hidden Markov Model

as n becomes larger and exhaustively considering all the rhyme schemes for a given verse could explode the size and therefore the computational complexity of the training algorithm. In addition to that, considering most of these rhyming schemes is not helpful for two reasons: (1) rhyme schemes that have rhyming words seperated over long distances (such as **ABCDA**) are meaningless as the listeners cannot keep track of more than previous four lines, and (2) it becomes very hard to find rhyming word group that contain more than five or six words that provide relevant meaning to the lyrics. Keeping these reasons in mind, we have decided to place an upper bound of four on the length of the rhyme schemes. This helps keep the computation tractable while providing enough flexibility and co-occurrence counts.

The total number of partitions for all the sequences of length 4 are 23. However, we do not need to exhaustively include all of them as most of the sequences can be generated from sequences of smaller length. For example, a rhyme scheme of length 3 **AAB** can be generated by a sequence of rhyme schemes of length 2 and 1, **AA** and **B** respectively. Therefore, for $n = 1$ to 4, we only choose those sequences that cannot be generated from smaller sequences.

After the application of above constraints, we finally end up with the following 9 states **A**, **AA**, **ABA**, **AAA**, **ABAB**, **AABA**, **ABAA**, **BAAA**, **AAAA** which makes the HMM significantly compact compared to exhaustively considering all possible rhyme schemes. We hope that the transition probabilities will help achieve the necessary bias towards states that are not explicitly represented in the HMM.

4 Experiments

4.1 Dataset

We exploited the vast amount of user generated hip hop lyrics available online. We implemented a web crawler to scrape websites that offer hip hop lyrics.

We downloaded the lyrics of about 52,000 hip hop songs (about 800MB of raw html content). The data was cleaned by stripping HTML tags, various metadata (e.g., the artist, song, lines corresponding to a chorus, beats, etc.) Verses were identified using simple heuristics and marked up. We also normalized for special characters and case differences, and filtered out any malformed HTML tags left in the data. We then extracted the end-of-line words and words before all the commas from each verse. We obtained a corpus containing 260,000 verses with 4.2 Million tokens (with around 153,000 unique tokens).

4.2 Evaluation

The performance of our model was evaluated on the task of labeling a given verse with the rhyme schemes. As there are currently no annotated gold standard corpora for the evaluation of performance on such a task we performed manual evaluation. As our model is completely unsupervised, we chose a random sample of 75 sentences from our training data as our test set. Two native English speakers were asked to annotate the verse with a gold standard rhyme scheme. They were asked to segment any rhyme schemes of length more than four appropriately. Precision and recall were aggregated for the Viterbi parse [8] of each verse against this gold standard and the f-score was calculated.

5 Results

In this section, we present the results of our experiment. Firstly, we report the number of types and tokens in our corpus and make an empirical comparison of the frequency of the types to that of a Zipfian distribution to establish that our corpus is comparable with other naturally occurring natural language data [9]. In the next subsection, we present some examples of the identified rhyming pairs and discuss some errors that the model is prone to making. Finally, we discuss precision and recall scores on a randomly selected subset of the training data.

5.1 Cumulative Probability Distribution

The occurrences of each of the 153673 words in all the songs were counted, given by the cumulative frequency. The words were organized into bins where each bin represented a cumulative frequency interval of 100, i.e. the bin ranked 1 will have words that appeared between 0-100 times. The total number of words in each bin was counted to get the probability distribution over the range of frequencies. Figure 2 showed a graphical illustrations of the cumulative probability distribution of each bin against rank of the bin.

Results show that the cumulative probability of the words and the ranks follow a distribution similar to the the Zipfian distribution to a certain extend. The law states that many words appear only a few times in the language and a few words appear many times, which can be roughly approximated by $1/Rank$ where the most occurred word occurred approximately twice as much as the

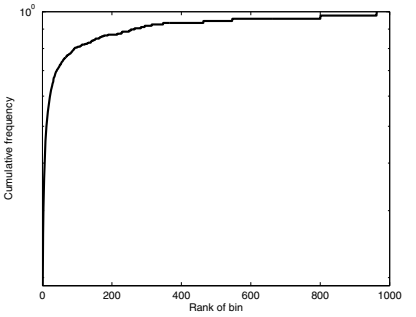

Fig. 2. Frequency density of word occurrence in the corpus

second and three times as much as the third. This relationship is said to hold for any corpus in natural language utterance. The similarity suggests that the data used here is a representative sample of naturally occurring data and our method can be applied to other hip hop lyrics corpora with similar distribution and expect similar performance.

5.2 Rhyme Scheme Variations

In this section, we briefly describe some observations regarding the occurence of rhyme schemes as learned by our model. As is the model bias with HMMs, the transitional probabilities are higher when the transitioning state covers a longer span. Rhyme schemes typically begin with states **ABAB**, **AABA** and **ABAA**. In fact, there is a strong bias against beginning with states that correspond to smaller rhyme schemes, which is a weakness. We have observed that this strong bias towards states with large rhyme schemes sometimes results in improper segmentation i.e., the model chooses to assign a longer rhyme scheme to certain lines in the verse where using a short rhyme scheme would improve the overall score. Somewhat surprisingly, we noticed that the transition probability from the state **AAAA** to the state **AA** is almost 1. Upon data inspection, we observed that this was due to the presence of chorus lines in the data which contained successive repeated lines thereby biasing the model towards transitioning to the state **AA**. The presence of chorus lines also explained the very high transition probabilities for transitions from states **ABAB**, **AABA** and **ABAA** to **AA**.

5.3 Preprocessing Errors Cause False Positive Rhyme Pairs

In order to determine whether or not our model identifies rhyming words correctly, we manually inspected the rhyming pairs with highest pairwise rhyming strength. From Table 1, we can notice that the model does identify rhyming word pairs fairly accurately. The model also identifies rhyming words that are not in English such as “sonorisateur – l’heure”. Most of the pairs with high scores were

Table 1. Samples from some of the top rhyming pairs

examples of rhyming pairs		log probability
sonorisateur	l'heure	0
seazin	weezin	0
rieces	telekineses	0
on	longjohn	-0.000608862
in	homosapien	-0.0018521
convenant	parent	-1.05447
convenant	it	-1.09564
convenant	transparent	-1.14791
convenant	terminated	-45.0261
convenant	loot	-48.0058
convenant	incarcerated	-102.884

fairly accurate although we found some pairs which were not really rhymes but were semantically related such as "*steppenwolf – wild*", "*voldemort – horcruxes*" and "*eenie-meenie – minie-mo*". This is due to the fact that these tokens occurred very infrequently and were mislabeled as rhymes because there was no other way to explain the data given our model. We also show an example of the top pairwise rhyming strengths of the word "*covenant*" with other words in Table 1. The rhyming words "*parent*" and "*transparent*" are identified correctly and the pairwise rhyming strength is low for others. The rhyming pair "*it*" is incorrect but has a high probability. Upon inspecting the data, we noticed that this was due to the presence of a comma after "*it*" in the lyric which caused it to be considered as a likely rhyming candidate. Such preprocessing errors were a major source of errors in rhyme pair identification and most of the false positives contained stop words before a comma. We believe better preprocessing heuristics will help alleviate the problem and improve the accuracy of our model.

5.4 Measurement of Labeling Accuracy

We obtain a precision of 35.81% and a recall of 57.25% giving an f-score of 44.06%. We find these results to be very encouraging in the face of highly noisy data, lack of clear stanza segmentation, and a very wide variety of rhyme schemes used in hip hop. Table 2 shows an example verse labeled with the viterbi parse using the model after the 10 iterations of EM. The stanzas "*sicker—remenants—liquor*" and "*radiation—head–ahead—instead*" are labeled correctly, while the stanza "*are—these—leave—tv*" has "*these*" incorrectly labeled as rhyming with "*leave*" and "*tv*". The model is prone towards committing such false positive errors especially for tokens that do not belong to the rhyme scheme which also explains the lower precision and higher recall. This is due to the inherent bias of HMM models towards minimizing the number of transitions. Hence, the token is incorrectly added to the rhyme scheme instead of creating a separate state for the single token. A possible solution could be to introduce an appropriate bias in

Table 2. Viterbi labeling of the example

The clock ticks, —	**A**
— your name off the list	**B**
Life moves quick so we gotta move quicker	**A**
The world's devilish, —	**B**
— makin me sicker	**A**
I smell the remanents, —	**B**
— of drugs sex and liqour	**A**
The kids are, —	**B**
— immune to these	**A**
Cos they leave, —	**A**
— they get a fix on they new tv	**A**
Parents dont need to talk to their kids no more	**B**
They feed em videos, —	**A**
— show em the internet	**A**
Log on to the porn site, —	**A**
— here we go	**B**
Television, —	**A**
— the drug of a nation	**A**
Breeding ignorance, —	**A**
— feeding radiation	**B**
Frenzhal hit the nail on the head	**A**
Wish we were sailing ahead	**A**
But we're shipwrecked and failing instead	**A**
But the tests have changed,	**B**
— we're all passin	**A**
So long as your a expert in arson	**A**
I'm askin, —	**A**
— which ways up,	**B**
cos my compass has gone	**A**
You know what? I'm a let my beatin heart be my guide through the eye of the storm	**A**

the initial transition probabilities to counteract the model bias of choosing longer rhyme schemes. Also, noisy transcription of lyrics such as incorrectly placing the last token "*storm*" on a separate verse line, affect the performance of the model.

6 Conclusion and Future Work

We presented a novel unsupervised algorithm to identify the rhyme schemes in hip hop lyrics. We presented the learning algorithm, discussed the nature of the corpus and presented precision and recall against a manually annotated gold standard. These results are very encouraging given that our task is highly unstructured compared to similar tasks performed on other domains . Also, no clear segmentation of rhyme schemes makes our task more challenging. In the

future, we would like to experiment with different rhyme scheme selections and weighting schemes for the transition probabilities. Also, the rhyme pairs learnt in the model can be used in tandem with language models to generate fluent and rhyming hip hop lyrics.

Acknowledgements. This material is based upon work supported in part by the Hong Kong Research Grants Council (RGC) research grants GRF620811, GRF621008, FSGRF13EG28, and GRF612806; the Defense Advanced Research Projects Agency (DARPA) under BOLT contract no. HR0011-12-C-0016, and GALE contract nos. HR0011-06-C-0022 and HR0011-06-C-0023; and by the European Union under the FP7 grant agreement no. 287658. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the RGC, EU, or DARPA. We would also like to thank Belsy Yuen and Subrotho Mondal Kumar for their help in human evaluation and data analysis.

References

1. Mitchell, K.: Hip-hop rhyming dictionary. Alfred Publishing Company, Incorporated (2003)
2. Greene, E., Bodrumlu, T., Knight, K.: Automatic analysis of rhythmic poetry with applications to generation and translation. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 524–533. Association for Computational Linguistics (2010)
3. Reddy, S., Knight, K.: Unsupervised discovery of rhyme schemes. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers, vol. 2, pp. 77–82. Association for Computational Linguistics (2011)
4. Genzel, D., Uszkoreit, J., Och, F.: Poetic statistical machine translation: rhyme and meter. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 158–166. Association for Computational Linguistics (2010)
5. Jiang, L., Zhou, M.: Generating chinese couplets using a statistical mt approach. In: Proceedings of the 22nd International Conference on Computational Linguistics, COLING 2008, vol. 1, pp. 377–384. Association for Computational Linguistics, Stroudsburg (2008)
6. Sonderegger, M.: Applications of graph theory to an english rhyming corpus. Computer Speech & Language 25(3), 655–678 (2011)
7. Ramakrishnan A, A., Kuppan, S., Devi, S.L.: Automatic generation of tamil lyrics for melodies. In: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity, pp. 40–46. Association for Computational Linguistics (2009)
8. Forney Jr., G.: The viterbi algorithm. Proceedings of the IEEE 61(3), 268–278 (1973)
9. Manning, C., Schütze, H.: Foundations of statistical natural language processing. MIT press (1999)