Modeling Hip Hop Challenge-Response Lyrics as Machine Translation

Dekai WU and **Karteek ADDANKI** and **Markus SAERS**
Human Language Technology Center
Department of Computer Science
HKUST, Clear Water Bay, Hong Kong
{dekai,vskaddanki,masaers}@cs.ust.hk

Abstract

We cast the problem of hip hop lyric generation as a translation problem, automatically learn a machine translation system that accepts hip hop lyric challenges and improvises rhyming responses, and show that improving the training data by learning an unsupervised rhyme detection scheme further improves performance. Our approach using unsupervised induction of stochastic transduction grammars is the first to apply the learning algorithms of SMT to the woefully under-explored genre of lyrics in music. A novel feature of our model is that it is completely unsupervised and does not make use of any *a priori* linguistic or phonetic information. Unlike the handful of previous approaches to modeling lyrics, we choose the domain of hip hop lyrics which is particularly noisy and unstructured. In order to cope with the noisy nature of the data in this domain, we compare the effect of two data selection schemes on the quality of the responses generated, and show the superiority of selection via a dedicated rhyme scheme detector that is also acquired through unsupervised learning. We also propose two strategies to mitigate the effect of disfluencies in the data which are common in the domain of hip hop lyrics, on the performance of our model. Despite the particularly noisy and unstructured nature of the domain, our model produces fluent and rhyming responses compared to a standard phrase based SMT baseline in human evaluations.

1 Introduction

Despite being a form of language that has had the most impact across almost all human cultures, there have been very few attempts to apply computational linguistics to the genre of lyrics in music. As an attempt to motivate further research in this direction, we apply the learning algorithms of statistical machine translation (SMT) to address some of the modeling issues in song lyrics. We choose hip hop, a genre of music that emphasizes rapping, spoken or chanted rhyming lyrics against strong beats or simple melodies as a starting point for this investigation. Unlike poetry and some other genres of music, hip hop lyrics present a significant number of challenges for learning as it lacks well-defined structure in terms of rhyme scheme, meter, or overall meaning.

We argue that the rich tools offered by SMT for unsupervised learning of stochastic transduction grammars are ideal for representing structural relationships between lines of lyrics. We model the problem of improvising a rhyming response given any hip hop lyric challenge as "translating" from a challenge line to a rhyming response. We describe a new fully-automatically learned challenge-response system which uses a stochastic transduction grammar as the translation model for this task. Despite the highly unstructured and noisy domain, our model is completely unsupervised and uses no prior phonetic or linguistic knowledge whatsoever.

The very high degree of variation that is permitted in the meter of the lyrics, and large amounts of colloquial vocabulary and slang from the subculture employed make the domain of hip hop lyrics highly unstructured compared to more structured domains such as classical poetry on which statistical methods have been applied in the past. The variance in the permitted meter coupled with the broad range of unorthodox vocabulary used in hip

Sima'an, K., Forcada, M.L., Grasmick, D., Depraetere, H., Way, A. (eds.)
Proceedings of the XIV Machine Translation Summit (Nice, September 2–6, 2013), p. 109–116.

hop makes it difficult to use off-the-shelf NLP tools for doing phonological and/or morphological analysis. Differences in intonation and lack of robust transcription (Liberman, 2010) further compound the challenges involved in apply statistical learning algorithms to the domain of hip hop.

Given the noisy nature of the domain of hip hop lyrics, we employ a rhyme scheme detector acquired through unsupervised learning to select our training data. The rhyme scheme detector segments each verse of a hip hop song into stanzas and identifies the lines in each stanza that rhyme with each other. The rhyming lines are subsequently used to induce the stochastic transduction grammar. We demonstrate the superiority of our training data selection method by comparing the quality of the responses generated by the models trained on data selected with and without using the rhyme scheme detector.

Disfluencies and backing vocals[1] occur very frequently in the domain of hip hop lyrics, unlike conventional spoken and written language, which affect the performance of SMT models designed for translating well-formed sentences. We propose two strategies for alleviating the effect of disfluencies on the performance of our models and compare the performance of each strategy in human evaluations. Finally, in order to illustrate the challenges faced by traditional SMT tools, we contrast the performance of our model against a conventional, widely used phrase-based SMT model.

A brief terminological note: "stanza" and "verse" are frequently confused and sometimes conflated. Worse yet, their usage for song lyrics is often contradictory to that for poetry. To avoid ambiguity we consistently follow these technical definitions for segments in decreasing size of granularity:

verse a large unit of a song's lyrics. A song typically contains several verses interspersed with choruses. In the present work, we do not differentiate choruses from verses. In song lyrics, a verse is most commonly represented as a separate paragraph.

stanza a segment within a verse which has a meter and rhyme scheme. Stanzas often consist of 2, 3, or 4 lines, but stanzas of more lines are also common. Particularly in hip hop, a single verse often contains many stanzas with different rhyme schemes and meters.

line a segment within a stanza consisting of a single line. In poetry, strictly speaking this would be called a "verse", which however conflicts with the conventional use of "verse" in song lyrics.

Some of the previous work that applies SMT models and other statistical methods to similar unconventional problems are discussed in Section 2. Sections 3 and 4 contain our system description and experimental setup respectively. Results and conclusions are presented in Sections 5 and 6.

2 Related work

Although a handful of previous approaches applied SMT models and other statistical learning methods have been to unconventional domains in the past, ours is the first known work on the domain of hip hop lyrics. Most of the past work in this vein can be classified into two categories. In the first category some form of prior linguistic knowledge about the domain, such as pronunciation dictionaries Genzel *et al.* (2010) or phonological or morphological information is used to bootstrap the learning. While the second category uses unsupervised learning methods to identify word association probabilities, appropriate bias is provided by the inherent constraints in the domain such as a set number of words in a line (in Chinese couplets), or a set meter (in classical poetry).

In this work, we present a completely unsupervised model on a domain that inherently has very few such constraints. As previously mentioned, hip hop lyrics unlike poems (especially in classical poetry where, for example, an octave has exactly 10 syllables per line and 8 lines per stanza) do not require a set number of syllables in a line. Also, not all words in the lyrics are required to be a part of the lexicon. Finally, rhyming is frequently achieved via intonation and assonance making it hard to apply prior phonological constraints. A brief summary of the related work is presented below.

Jiang and Zhou (2008) trained a phrase-based SMT system to translate the first line of of a Chinese couplet or *duilian* into the second. Linguistic constraints were applied to the n best output of the SMT system to select the most suitable next line.

[1] Particularly the repetitive chants, exclamations, and interjections in hip hop "hype man" style backing vocals.

However in contrast to Chinese couplets, which adhere to strict rules requiring, for example, an identical number of characters in each line and one-to-one correspondence in their metrical length, the domain of hip hop lyrics is far more unstructured and there exists no clear constraint that would ensure fluent and rhyming responses to hip hop challenge lyrics.

An SMT system was used in conjunction with stress patterns and rhymes found in a pronunciation dictionary to produce translations of poems by Genzel *et al.* (2010). However, it was challenging to produce translations of full verses that satisfied all the constraints enforced by classical poetry. For example, the translations could not comply with the desired meter of the line although the rhyming constraints were satisfied. These results indicate the difficulty of producing quality output via automatic methods even for very structured domains.

A *et al.* (2009) proposed a model for automatically generating Tamil lyrics given a melody. The lyrics were represented as a sequence of labels using the *KNM* system where *K*, *N* and *M* represented the long vowels, short vowels and consonants respectively. They solved the sequence labeling problem of generating the lyrics given a melody using conditional random fields.

Others have attempted to identify word-to-word relationships, stress patterns (Greene *et al.*, 2010) and rhyming words (Reddy and Knight, 2011), mostly in the domain of poetry. Greene *et al.* (2010) used an FST to assign stress patterns to words given the meter of a line in Shakespeare's sonnets. The stress patterns were combined with a language model to generate poems. Sonderegger (2011) applied graph theory to identify the rhyming patterns and therefore inferred the pronunciation of words in old English poetry. However, their heuristic of clustering words with similar IPA endings resulted in large clusters of false positives such as bloom and numb. Reddy and Knight (2011) proposed a language-independent generative model for stanzas in poetry for discovering rhyme schemes in French and English poetry.

3 Challenge-response system

We describe our machine translation system that accepts hip hop lyric challenges (i.e., a line of a hip hop verse) and "tranlates" the challenge into a fluent response which rhymes with the challenge. We describe the rhyme scheme detector in Section 3.1 that is used to identify the rhyming lines in the verses which is used to select the training data for our system. Sections 3.2 and 3.3 discuss the details of our transduction grammar induction and decoding algorithm respectively.

3.1 Data selection: Rhyme scheme detection

Although our approach adapts an SMT system toward the problem of generating fluent and rhyming hip hop responses, it would be undesirable to train an SMT system on all the successive lines of the verses as described in Jiang and Zhou (2008) due to variance in hip hop rhyming patterns. For example, it is very common for a stanza to follow the **ABAB** rhyme scheme. Adding successive lines as the training instances to the SMT system would drive the transduction grammar towards learning incorrect rhyme correspondences. This problem is exacerbated by the fact that a verse (which is usually represented as a separate paragraph) may contain multiple stanzas of varying length and rhyme schemes. The large size of the verse makes it impossible to add all possible pairs of lines as training example as it would explode the size of the training data not counting the degree of noise such an approach would introduce.

In order to select training instances that are likely to rhyme, we employ a rhyme scheme detection model as mentioned in (Addanki and Wu, 2013). Lines belonging to the same stanza and marked as rhyming according to the rhyme scheme detection model are added to the training corpus of the SMT system. We believe that this data selection scheme will improve the rhyming associations learned during the transduction grammar induction thereby biasing the model towards producing fluent and rhyming output. We briefly describe the rhyme scheme detection model here.

We train a hidden Markov model (HMM) which generates a verse of hip hop lyrics to serve as our rhyme scheme detection module. Instead of partitioning the verses into stanzas according to some pre-specified rules, a number of hidden states corresponding to stanzas of varying length are used to automatically obtain a *soft-segmentation* of the verse. Each state in the HMM corresponds to a stanza with a particular rhyme scheme such as **AA**, **ABAB**, **AAAA** while the emissions correspond to

the final words in the stanza.

We restrict the maximum length of a stanza to be four as the total number of rhyme schemes given a stanza of length n (and hence the corresponding states in our HMM model) is the n^{th} Bell number (OEIS, 2013) and exhaustively considering the exponential number of partitions is prohibitively expensive and not very useful. Further, we only used states to represent stanzas whose rhyme schemes could not be partitioned into smaller schemes without losing a rhyme correspondence. For example, a rhyme scheme of length 3 **AAB** can be partitioned into a sequence of two smaller rhyme schemes **AA** and **B** without losing any rhyme correspondences. Hence, **AAB** is not represented as a state in our HMM. After applying these constraints our HMM model is fully connected with the following 9 states: **A**, **AA**, **ABA**, **AAA**, **ABAB**, **AABA**, **ABAA**, **BAAA**, **AAAA**.

We train the HMM using the EM algorithm (Devijver, 1985) on our corpus generated by taking the final word of each line in the hip hop lyrics. We then segment each verse into stanzas with their respective rhyme schemes according to the Viterbi parse of the model. We then select the lines from each stanza that rhyme with each other according to its rhyme scheme and add them as training instances for our transduction grammar based SMT model. Each selected pair generates two training instances: a challenge-response and a response-challenge pair as the source and target languages are identical.

3.2 Unsupervised learning: Stochastic transduction grammar induction

We choose to induce a token based inversion transduction grammar (ITG) model (Wu, 1997, 1995a,b) because of its expressiveness and the empirical evidence for its representational capacity across a wide spectrum of natural language tasks including textual entailment (Wu, 2006), mining parallel sentences (Wu and Fung, 2005) and machine translation (Zens and Ney, 2003; Haghighi *et al.*, 2009). We restrict the ITG to a bracketing ITG (BITG) and use it as the translation model for our SMT system as the focus of our model is to learn the token level correspondences in order to identify potential rhyming candidates. We chose an ITG as opposed to a monotonic finite-state transduction grammar model in order to be able to learn token level correspondences involving alignments which are not purely monotonic. We trade-off some of the initial fluency that segmental phrase-based models offer for the flexibility offered by the token based models.

We induce the bracketing ITG on the corpus generated from the previous stage to identify the word associations between rhyming lines. Expectation maximization (Dempster *et al.*, 1977) is used to estimate the model parameters for the bracketing grammar. As the corpora are fairly large, beam pruning is used to make the training faster. Further details of the transduction grammar induction can be found in (Saers *et al.*, 2012; Saers and Wu, 2011).

3.3 Decoding: Challenge-response algorithm

We translate the challenge into responses using our in-house ITG decoder Wu (1996); Wu and Wong (1998) for the task of decoding. The decoder builds the parse forest using a CKY-style parsing algorithm which is represented in an efficient hypergraph structure. The translation hypotheses are scored using the transduction grammar and the language model efficiently using cube pruning (Chiang, 2007)

In our decoding algorithm, we allow only straight rules as we want to produce responses with the same rhyming order as the challenge. Interleaved rhyming order is harder to evaluate without the larger context of the song and we do not address that problem in our current model. Singleton rules are penalized, as successive lines in a stanza are typically of similar length. Lastly, we add a penalty to *reflexive* translation rules that map the same surface form to itself such as $A \rightarrow$ ”yo”/”yo”. We observed that such rules have a high probability in our learned grammar due to presence of sentence pairs in our training data where both the input and output are identical strings as many stanzas in our data contain repeated chorus lines.

4 Experiments

We describe our data set, baseline phrase based SMT model and the comparative model used for gauging the efficacy of our data selection scheme, and propose two strategies to handle disfluencies in the training data. We also describe the evaluation schemes used for determining the performance of our rhyme scheme detector and comparing the per-

formance of different models on the task of improvising responses to hip hop lyric challenges.

4.1 Dataset

We downloaded the lyrics of approximately 52,000 hip hop songs (about 800Mb of raw HTML content). The data was cleaned by stripping HTML tags, metadata and normalizing for special characters and case differences. The entire corpus contained 22 million tokens with 260,000 verses and 2.7 million lines of hip hop lyrics. A subset of 85 lines was randomly chosen as a test set to provide the hip hop challenge lyrics to the systems. In order to train the rhyme scheme detector, we extracted the end-of-line words and words before all the commas[2] from each verse. We obtained a corpus containing 4.2 million tokens corresponding to potential rhyming candidates (with around 153,000 unique token types).

4.2 Phrase-based SMT baseline

In order to evaluate the performance of standard SMT alignment and search strategies on this novel "translation" task, we also trained a standard Moses baseline (Koehn *et al.*, 2007) and compared its performance to our transduction grammar based SMT system. A 4-gram language model which was trained on the entire training corpus using SRILM (Stolcke, 2002) was used in decoding both the baseline and our model. Both the baseline and our bracketing ITG model were used to decode a held out test set with a slightly higher language model weight which was empirically chosen using a small development set to produce fluent outputs. The best translation produced by both these SMT systems was used to evaluate their performance at the task of improvising fluent and rhyming responses given a challenge.

4.3 Rhyme scheme detection vs. adjacent lines

To gauge the efficacy of our training data selection scheme, we train both the baseline SMT system and our transduction grammar based model on the data selected using the rhyme scheme detector as well as all the adjacent lines in a verse.

Rhyme scheme detection Upon extracting a training corpus as described in Section 3.1, we obtained around 600,000 training instances. Due to high time and memory cost of inducing stochastic transduction grammars, we only added those lines that were adjacent *and* labeled as rhyming by the rhyme scheme detector as training instances resulting in a training corpus of size 200,000.

Adjacent lines Considering all adjacent lines in a verse resulted in a corpus of over 5 million training instances. In order to ensure fair comparison of models trained on both versions of training data, we randomly chose 200,000 training instances from the generated corpus. The training corpus thus generated shared around 15% of training instances with the corpus generated through our proposed data selection scheme.

4.4 Disfluency handling

We noticed a disturbingly high proportion of generated responses contained disfluencies with successive repetitions of words such as the and I in our error analysis of initial runs. Upon inspection we noticed that 10% of our training data contained similar disfluencies and backing vocal lines. To tackle this problem ,we propose and compare two alternative strategies: (1) *disf luency filtering* which is to filter out all lines from our training corpus which contained disfluencies and (2) *disfluency correction* which corrects the disfluencies in the training corpus (for example, the the the, in the training corpus is corrected to simply the). The baseline and transduction grammar based SMT systems were trained on both the filtered and corrected versions of the training corpus.

4.5 Evaluation

The performance of the SMT systems was evaluated on the task of generating a improvised fluent and rhyming response given a single line of a hip hop verse as a challenge. The output of all the systems on the test set was evaluated by three independent frequent hip hop listeners according on the criterion of fluency and the degree of rhyming. They were allowed to choose the tune to evaluate rhyming as the beats of the song were not used during training. Each evaluator was scored the response of each system on the criterion of fluency and rhyming as being *good*, *acceptable* or *bad*. The performance of our rhyme scheme detector was also evaluated on a random sample of 75 verses

[2]Shorter lines in hip hop stanzas are typically joined with a comma and represented as a single line of text.

from our training data as the rhyme scheme detection model is completely unsupervised. A gold standard rhyme scheme was assigned to each verse by human evaluators and precision, recall and f-score were computed.

5 Results

In this section, we present the results of human evaluation on the improvised responses of our SMT systems and demonstrate that using rhyme scheme detection produces better quality responses. We also describe the accuracy of our rhyme scheme detector and provide some examples of the responses generated by our model trained using both the data selection schemes versus the phrase-based SMT baseline. We also compare the effect of proposed disfluency handling strategies on our task.

5.1 Rhyme scheme detector accuracy

For the rhyme scheme detector used in the data selection phase, a precision of 35.81% and a recall of 57.25%, giving an f-score of 44.06%. Error analysis indicated mostly false positive errors which were a result of the model bias of HMMs toward minimizing the number of state transitions resulting in improper segmentation of stanzas, especially when there was a single line that did not rhyme with any other line.

5.2 Phrase-based SMT baseline performs poorly

Table 1 shows the average fraction of sentences rated *good* and *acceptable* for each model and the disfluency correction strategy used to train the model. The transduction grammar based models (BITG) produce a significantly higher percentage of *good* and *acceptable* rhyming responses compared to the phrase-based SMT (PBSMT) baseline for both disfluency filtering and disfluency correction strategies and on both corpus selection methods. BITG+RS and BITG correspond to models trained using rhyme scheme and adjacent line corpus selection methods respectively. Also, BITG outperforms PBSMT on the task of fluency when trained on the disfluency corrected version of the corpus. Although BITG falls behind PBSMT in generating sentences with a *good* fluency under the disfluency filtering strategy, the cumulative fraction of sentences that were labeled *good* or *acceptable* is larger compared to PBSMT. When the training corpus was generated using adjacent lines, the performance of PBSMT is much poorer compared to using rhyme scheme detection. These results indicate that conventional PBSMT approaches are not very effective on this novel task.

5.3 Rhyme scheme detection helps fluency

Results in Table 1 indicate that using the rhyme scheme detector for corpus generation helps produce significantly more fluent responses compared to using adjacent lines. A possible explanation for this could be that adding all adjacent lines as training instances introduces a lot of noise into the model which hurts the fluency of the responses generated. Also, the cumulative fraction of sentences that were labeled *good* or *acceptable* is larger when rhyme scheme detection was used to generate the training data (Although, the BITG model trained on the corpus generated using adjacent lines produces a higher percentage of rhyming responses that were rated *good*). Given the significantly higher rate of response fluency when using rhyme scheme detection, we argue that using rhyme scheme detector for corpus generation is beneficial. We believe that incorporating more accurate rhyme scheme detection algorithms for corpus selection would greatly improve the quality of the responses produced by the SMT systems.

5.4 Disfluency correction helps

The results in Table 1 indicate that the disfluency correction strategy outperforms the filtering strategy, on both fluency and rhyming criteria irrespective of the corpus selection method. With disfluency correction, 34.12% of the generated responses were rated *good* on fluency, while the disfluency filtering strategy produced only *good* responses 28.63% of the time using rhyme scheme detection for corpus selection. Similarly, disfluency correction produced a fluency of 21.18% while filtering produced 17.25% when using the corpus generated by using all the adjacent lines. These observations can be explained by the loss of useful word association information necessary for rhyming due to the much harsher pruning of training corpus enforced by disfluency filtering strategy. Future research is necessary to fully investigate the effect of alternative disfluency correction strategies on the performance of the challenge-response systems.

Table 1: Percentage of *good* and ≥*acceptable* (i.e., either good or acceptable) responses on fluency and rhyming criteria. PBSMT and BITG correspond to the phrase based and transduction grammar based models trained using corpus generated from all adjacent lines in a verse. PBSMT+RS and BITG+RS are models trained on rhyme scheme based corpus selection strategy.

model	*disfluency strategy*	*fluency (good)*	*fluency (≥acceptable)*	*rhyming (good)*	*rhyming (≥acceptable)*
PBSMT	filtering	4.3%	13.72%	3.53%	7.06%
PBSMT+RS	filtering	31.76%	43.91%	12.15%	21.17%
BITG	filtering	17.25%	46.27%	18.04%	33.33%
BITG+RS	filtering	28.63%	56.86%	14.90%	34.51%
PBSMT	correction	3.14%	4.70%	1.57%	4.31%
PBSMT+RS	correction	30.59%	43.53%	1.96%	9.02%
BITG	correction	21.18%	54.51%	**23.53%**	39.21%
BITG+RS	correction	**34.12%**	**60.39%**	20.00%	**42.74%**

Table 2: Examples of challenges and responses.

challenge	man i **die** to **see** em all thun i just don't **care**
BITG+RS	in the **sky** and **me** the in polla and the you **there**
BITG	and the ride the me the and white the i the **air**
PBSMT+RS	man i live to see em all i just don't care
challenge	did a twelve year **bid** in the streets and held it **down**
BITG+RS	to the girls here **kid** the and to the thought the **now**
BITG	to the p's here **did** the a the i was the the
PBSMT+RS	did a year in the streets and it down
challenge	oh i believe in **yesterday**
BITG+RS	can you see the **day**
BITG	now you see the **way**
PBSMT+RS	oh i believe in tomorrow
challenge	what would i **do**
BITG+RS	just me and **you**
BITG	and you and **you**
PBSMT+RS	what would you do
challenge	cause you ain't going **home** till the early **morn**
BITG+RS	and the you this **alone** i i gotta **on**
BITG	and i you my on the a home we
PBSMT+RS	cause you and your friends aint nothing but

5.5 Challenge-response examples

Table 2 shows some of the challenges and the corresponding responses of PBSMT+RS, BITG+RS and BITG model. While PBSMT+RS and BITG+RS models generate responses reflecting a high degree of fluency, the output of the BITG contains a lot of articles. It is interesting to note that BITG+RS produces responses comparable in fluency to PBSMT+RS despite being a token based transduction grammar. However, PBSMT models tend to produce responses that are too similar to the challenge compared to the BITG models which improvise responses that rhyme better (shown in boldface). In fact BITG models frequently produce responses that rhyme words not only at the end but also in the middle of challenges as our transduction grammar model captures structural associations more effectively than the phrase-based model.

6 Conclusions

We demonstrated that the problem of hip hop lyric generation can be cast into a translation problem and existing SMT learning algorithms can be applied to automatically learn a machine translation system that accepts hip hop lyric challenges and improvises rhyming responses. We compared the performance of our SMT model that uses a stochastic transduction grammar against the widely used phrase-based SMT model and demonstrated that conventional PBSMT algorithms fall short in tackling the noisy and highly unstructured domain of hip hop lyrics. We showed that the performance of our SMT model improves when the training data was generated using a rhyme scheme detector as opposed to adding all adjacent lines of a verse to the training corpus. Some task-specific problems resulting from disfluencies and backing vocals, which are characteristic to the domain of hip hop lyrics were identified. In our future work we plan to further investigate novel transduction grammar models that simultaneously learn rhyming words and alignments thereby eliminating the need for a dedicated rhyme scheme detector.

Acknowledgements

This material is based upon work supported in part by the Hong Kong Research Grants Council (RGC) research grants GRF620811, GRF621008, GRF612806; by 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.

References

Ananth Ramakrishnan A, Sankar Kuppan, and Sobha Lalitha Devi. "Automatic generation of Tamil lyrics for melodies." *Workshop on Computational Approaches to Linguistic Creativity (CALC-09)*, 40–46. 2009.

Karteek Addanki and Dekai Wu. "Unsupervised rhyme scheme identification in hip hop lyrics using hidden Markov models." *1st International Conference on Statistical Language and Speech Processing (SLSP 2013)*. Tarragona, Spain, 2013.

David Chiang. "Hierarchical phrase-based translation." *Computational Linguistics*, 33(2):201–228, 2007.

Arthur Pentland Dempster, Nan M. Laird, and Donald Bruce Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Pierre A Devijver. "Baum's forward-backward algorithm revisited." *Pattern Recognition Letters*, 3(6):369–373, 1985.

Dmitriy Genzel, Jakob Uszkoreit, and Franz Och. "Poetic statistical machine translation: Rhyme and meter." *Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, 158–166. Association for Computational Linguistics, 2010.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. "Automatic analysis of rhythmic poetry with applications to generation and translation." *Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, 524–533. Association for Computational Linguistics, 2010.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. "Better word alignments with supervised itg models." *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009)*, 923–931. Suntec, Singapore, August 2009.

Long Jiang and Ming Zhou. "Generating Chinese couplets using a statistical MT approach." *20th International Conference on Computational Linguistics (COLING 2008)*. 2008.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. "Moses: Open source toolkit for statistical machine translation." *45th Annual Meeting of the Association for Computational Linguistics (ACL 2007) Demo and Poster Sessions*, 177–180. Prague, Czech Republic, June 2007.

Mark Liberman. "Rap scholarship, rap meter, and the anthology of mondegreens." `http://languagelog.ldc.upenn.edu/nll/?p=2824`, December 2010. Accessed: 2013-06-30.

OEIS. "Bell or exponential numbers: ways of placing n labeled balls into n indistinguishable boxes." *The On-Line Encyclopedia of Integer Sequences*, `http://oeis.org/A000110`, 2013. Accessed: 2013-06-30.

Sravana Reddy and Kevin Knight. "Unsupervised discovery of rhyme schemes." *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, vol. 2, 77–82. Association for Computational Linguistics, 2011.

Markus Saers, Karteek Addanki, and Dekai Wu. "From finite-state to inversion transductions: Toward unsupervised bilingual grammar induction." *24th International Conference on Computational Linguistics (COLING 2012)*, 2325–2340. Mumbai, India, December 2012.

Markus Saers and Dekai Wu. "Reestimation of reified rules in semiring parsing and biparsing." *Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-5)*, 70–78. Portland, Oregon: Association for Computational Linguistics, June 2011.

Morgan Sonderegger. "Applications of graph theory to an English rhyming corpus." *Computer Speech & Language*, 25(3):655–678, 2011.

Andreas Stolcke. "SRILM – an extensible language modeling toolkit." *International Conference on Spoken Language Processing (INTERSPEECH 2002)*, 901–904. Denver, Colorado, September 2002.

Dekai Wu. "An algorithm for simultaneously bracketing parallel texts by aligning words." *33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, 244–251. Cambridge, Massachusetts, June 1995a.

Dekai Wu. "Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora." *14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, vol. 95, 1328–1335. 1995b.

Dekai Wu. "A Polynomial-time Algorithm for Statistical Machine Translation." *34th annual meeting on Association for Computational Linguistics (ACL-96)*, 152–158. Morristown, NJ, USA, 1996.

Dekai Wu. "Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora." *Computational Linguistics*, 23(3):377–403, 1997.

Dekai Wu. "Textual entailment recognition using inversion transduction grammars." Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d' Alché Buc (eds.), *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, vol. 3944 of *Lecture Notes in Computer Science*, 299–308. Berlin: Springer, 2006.

Dekai Wu and Pascale Fung. "Inversion transduction grammar constraints for mining parallel sentences from quasi-comparable corpora." *Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, 257–268. Springer, 2005.

Dekai Wu and Hongsing Wong. "Machine translation with a stochastic grammatical channel." *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL '98)*, 1408–1415. 1998.

Richard Zens and Hermann Ney. "A comparative study on reordering constraints in statistical machine translation." *41st Annual Meeting on Association for Computational Linguistics (ACL-03)*, 144–151. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003.