FREESTYLE: A Challenge-Response System for Hip Hop Lyrics via Unsupervised Induction of Stochastic Transduction Grammars

Dekai WU, Karteek ADDANKI, Markus SAERS

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

Abstract

We attack an inexplicably under-explored language genre of spoken language—lyrics in music—via completely unsupervised induction of an SMT-style stochastic transduction grammar for hip hop lyrics, yielding a fully-automatically learned challenge-response system that produces rhyming lyrics given an input. Unlike previous efforts, we choose the domain of hip hop lyrics, which is particularly unstructured and noisy. A novel feature of our approach is that it is completely unsupervised and requires no a priori linguistic or phonetic knowledge. In spite of the level of difficulty of the challenge, the model nevertheless produces fluent output as judged by human evaluators, and performs significantly better than widely used phrase-based SMT models upon the same task.

Index Terms: statistical machine translation, lyrics translation, stochastic transduction grammars, unsupervised grammar induction

1. Introduction

Among the many genres of language that have been studied in computational linguistics and spoken language processing, there has been a 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 modern statistical 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.

We argue that statistical machine translation's rich tools for unsupervised learning of stochastic transduction grammars are ideal for representing structural relationships between lines of lyrics. We describe FREESTYLE, a new fully-automatically learned challenge-response system that learns to improvise a rhyming response given any input challenge line—with no prior phonetic or linguistic knowledge whatsoever. Instead, the model induces an SMT-style stochastic transduction grammar that "translates" from a challenge line to a rhyming response. The method is completely unsupervised, despite the highly unstructured and noisy domain.

The domain of hip hop lyrics is particularly unstructured when compared to classical poetry, a domain on which statistical methods have been applied in the past. Hip hop lyrics are unstructured in the sense that a very high degree of variation is permitted in the meter of the lyrics, and large amounts of colloquial vocabulary and slang from the subculture are employed. 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 used when generating output. The broad range of unorthodox vocabulary used in hip hop make it difficult to use off-the-shelf NLP tools for doing phonological and/or morphological analysis. These problems are further exacerbated by differences in intonation of the same word and lack of robust transcription [1].

To gauge the performance of traditional SMT approaches on this novel domain and task, we also contrast the performance of our model against a conventional, widely used phrase-based SMT model. We briefly describe a novel unsupervised rhyme scheme detection algorithm which generates the training data for our model from a corpus of hip hop lyrics. We also highlight some of the issues that result from disfluencies and chorus lyrics in the data which are characteristic to hip hop lyrics.

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.

In Section 2, we discuss some of the previous work that applies statistical methods to similar problems. We then describe our system and the experimental setup in Sections 3 and 4. Results and conclusions are presented in Sections 5 and 6.

2. Related work

Ours is the first known work on this domain, although similar work has been done in the past on more structured domains such as poetry. Flat phrase-based SMT has been used to model the two lines of a Chinese couplet or *duilian* [2]. In that work, an SMT system was trained to translate the first line of the couplet into the second. An n best list was generated for each test input, and then linguistic constraints were applied to select the most suitable next line. 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 good responses to challenge lines.

Other previous works have attempted to identify word-to-word relationships, stress patterns [3] and rhyming words [4], mostly in the domain of poetry. In some cases, the learned results were combined with language models to generate new poems [3, 5, 2]. Most of this vein of past work can be classified into two categories. The first category uses some form of prior linguistic knowledge about the domain, such as pronunciation dictionaries [5] 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 new 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, making it hard to apply prior phonological constraints. We present a brief summary of related work below.

The stress pattens of words in English rhythmic poetries were analyzed by [3]. 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, the common *iambic* meter follows a *da-DUM da-DUM da-DUM* pattern. An FST was applied to all the words in Shakespeare's sonnets to assign probable stress patterns. Combining stress patterns with a language model for fluency, a poem was generated.

In [5], SMT was used in conjunction with stress patterns and rhymes found in a pronunciation dictionary to produce translations of poems. Although many constraints were applied in translating full verses of poems, it was challenging to satisfy all the constraints. For example, some words rhymed but could not comply with the desired meter of the line. This illustrates the extent of the challenge faced by automatic methods to produce quality output even for very structured domains.

Graph theory was applied to analyze the rhyming and therefore inferred the pronunciation of words in old poetry for an English rhyming corpus by [6]. The pronunciations are represented by the International Phonetic Alphabet (IPA) symbols. Two words were assumed to rhyme when their 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 have the same IPA endings but do not fully rhyme, such as bloom and numb. Their method also introduced partitions on the graphs to obtain better results.

Automatic lyric generation given melodies was also investigated by [7]. The training data included melodies and accompanied lyrics, where the lyrics were represented using the KNM system, where K, N and M represented the long vowels, short vowels and consonants respectively. The trained model was then used to label any input melody with the KNM system. Secondly, words were chosen to fit the KNM system of the lyrics and constraints were applied for enforcing fluency.

3. System description

In this section, we describe our model for learning to generate a response to any given challenge (i.e., a line of a hip hop verse) that is fluent and rhymes with the challenge. We describe how the training data is selected using unsupervised rhyme scheme detection in Section 3.1. The unsupervised induction of our transduction grammar is described in 3.2 and Section 3.3 discusses our decoding algorithm.

3.1. Data selection: Rhyme scheme detection

Our approach adapts an SMT approach toward the problem of generating fluent and rhyming lyrics. However, unlike the approach used by [2], we could not train an SMT system on all the successive lines of the verses in the corpus, given the noisy nature of our data and differences in the rhyme sequences amongst the lyrics. In order to select pairs of lines from our corpus that are likely to rhyme, we employed a rhyme scheme detection model for hip hop lyrics. Through this data selection step, we improve the likelihood that the associations learned during our transduction grammar induction will be more biased towards producing fluent and rhyming output as opposed to blindly training on all successive lines. We briefly describe the rhyme scheme detection model here; for more details see [8].

We use a completely unsupervised HMM model to identify the lines of a verse that rhyme with each other. A novel feature of our approach comes from the fact that we do not manually partition verses into stanzas according to some pre-specified rhyme scheme but instead use a number of hidden states of varying length to automatically impose a *soft segmentation*.

We assume a generative model that stochastically emits a sequence of verses. Each verse is composed of many stanzas each with its own rhyme scheme such as **AA**, **ABAB**, **AAAA**. The total number of rhyme schemes given a stanza of length n is the n^{th} Bell number[9] and exhaustively considering the exponential number of partitions is prohibitively expensive and not very useful. For example, the rhyme scheme **ABCDA** seems very unlikely to be a part of any hip hop verse. This is because long distance rhymes are meaningless as listeners cannot keep track of more than a few previous lines. Hence, we restricted our maximum rhyme scheme length to be four. In order to keep the size of the HMM tractable we only used states for those rhyme schemes that could not be generated via transitioning through other states. 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. 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 select the training data for our transduction grammar based SMT model in the next stage by first labeling the corpus with rhyme schemes according to the Viterbi parse. We select adjacent lines that rhyme with each other (according to the rhyme scheme label) in the stanza and add them as parallel sentences to the training corpus. As the source and target languages are identical, each selected pair generates two training instances: a challenge-response and a response-challenge pair.

3.2. Unsupervised learning: Stochastic transduction grammar induction

We learn a stochastic inversion transduction grammar [10, 11, 12] that attempts to transduce any given challenge into a rhyming and fluent response. We choose the framework of inversion transduction grammars given the significant amount of empirical evidence for their representational capacity across a spectrum of natural language tasks such as textual entailment [13], mining parallel sentences [14] and machine translation [15, 16]. Restricting to bracketing ITGs encourages the learning to focus on token level correspondences for the task of identifying potential rhyming candidates. We chose an ITG model as opposed to a finite-state transduction grammar in order to be able to identify potential rhyming candidates in the middle of stanzas as multiple rhyming lines are frequently separated by commas instead of newlines. To obtain these types of flexibility, we trade off some of the initial fluency offered by segmental models.

We induce the bracketing ITG on the corpus generated from the previous stage to identify the word associations between rhyming lines. Expectation maximization [17] is used to estimate the model parameters for the bracketing grammar, using the algorithm for ITG training from [18]. 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 [19, 20, 21, 22, 23].

3.3. Decoding: Challenge-response algorithm

We use our in-house ITG decoder for the task of decoding [24, 25]. The decoder uses a CKY-style parsing algorithm [26] with cube pruning [27]. The decoder builds an efficient hypergraph structure which is then scored using the induced grammar.

In our decoding algorithm, we restrict the reordering to only be monotonic as we want to produce output that follows the same rhyming order of 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. We also penalize singletons to produce responses of similar length as successive lines in a stanza are of similar length. Finally, we add a penalty to *reflexive* translation rules that map the same surface form to itself such as $A \rightarrow$ ”yo”/”yo”. We obtain these rules with a high probability due to the presence of sentence pairs where both the input and output are identical strings. This is because many stanzas in our data contain repeated chorus lines which are labeled to be rhyming (and rightly so) in the rhyme scheme detection step.

4. Experiments

In the following subsections, we describe our data set, baseline model and our evaluation scheme for gauging the performance of the models.

4.1. Dataset

We exploited the vast amount of user generated hip hop lyrics available online. For our experiments, we used the lyrics of approximately 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 then extracted the end-of-line words and words before all the commas from each verse. We obtained a corpus containing 260,000 stanzas with 4.2 million tokens (with around 153,000 unique token types). We also normalized for special characters and case differences, and filtered out any malformed HTML tags left in the data.

4.2. Phrase-based SMT baseline

We labeled the stanzas in our corpus with rhyme schemes and extracted a training corpus as described in Section 3.1. We obtained a training corpus of about size 200,000 and trained our bracketing inversion transduction grammar model. In order to evaluate the performance of standard SMT alignment and search strategies on this task we also trained a standard Moses baseline [28] on the same data and compared the performance of our model. A 4-gram language model was trained on the training corpus using SRILM [29] for the purposes of decoding. 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 output of both these models was used to evaluate the performance of these models on evaluating the fluency and the degree of rhyming of the responses generated.

4.3. Evaluation

We evaluated the performance of rhyme scheme detection stage of our system on the task of labeling a given verse with the rhyme schemes. As our model is completely unsupervised, we chose a random sample of 75 verses from our training data as our test set. Two native English speakers were asked to annotate the stanza with a gold standard rhyme scheme. Precision and recall were aggregated for the viterbi parse of each verse against this gold standard and f-score was calculated.

The performance of our bracketing ITG model was evaluated on the task of generating a subsequent line given a single line of a hip hop verse. We evaluated on a random sample of 85 lines from the held-out test set data. The output of both the systems on the test set was given to three independent frequent hip hop listeners. They were asked to evaluate the system outputs according to fluency and the degree of rhyming. They were free to choose the tune to make the lyrics rhyme as the beats of the song were not used in the training data. Each was asked to score system response on the criterion of fluency and rhyming as being *good*, *acceptable* or *bad*.

5. Results

In this section, we present the results of our system on the tasks of rhyme scheme labeling and generation of fluent, rhyming responses given a challenge. We provide some examples of the responses generated by our approach versus the phrase-based SMT baseline. We also note the effect of disfluencies and backing vocals on our output and compare the performance of two alternative strategies to alleviate the problem.

5.1. Data selection accuracy

On the data selection phase utilizing rhyme scheme labeling, we obtained a precision of 35.81% and a recall of 57.25%, giving an f-score of 44.06%. Our model was prone to committing false positive errors especially for tokens that do not belong to a stanza which explains the lower precision and higher recall. The bias of HMM models towards minimizing the number of transitions resulted in improper segmentation of stanzas, especially when there was a single line that did not rhyme with any other line. Despite the false positives, the model serves as a useful step in the generation of training corpus as opposed to adding all possible line combinations in a verse to the training data.

Table 1: Percentage of *good* and ≥*acceptable* (i.e., either good or acceptable) responses on fluency and rhyming criteria.

model	*disfluency strategy*	*fluency (good)*	*fluency (≥acceptable)*	*rhyming (good)*	*rhyming (≥acceptable)*
Pbsmt	filtering	31.76%	43.91%	12.15%	21.17%
Freestyle	filtering	28.63%	56.86%	14.90%	34.51%
Pbsmt	correction	30.59%	43.53%	1.96%	9.02%
Freestyle	correction	**34.12%**	**60.39%**	**20.00%**	**42.74%**

5.2. Disfluency filtering vs. disfluency correction

Error analysis of our initial runs showed a disturbingly high proportion of responses generated by our system that contained disfluencies with successive repetitions of words such as the and I. Upon inspection of data we noticed that the training lyrics actually did contain such disfluencies and backing vocal lines,[1] amounting to 10% of our training data. We therefore compared two alternative strategies to tackle this problem. The first strategy involved filtering out all lines from our training corpus which contained such disfluencies. In the second strategy, we implemented a disfluency detection and correction algorithm (for example, the the the, which frequently occurred in the training corpus, was corrected to simply the). We trained both the phrase based SMT system (Pbsmt) and our model (Freestyle) on both the corrected and the filtered versions of the corpora.

The results in Table 1 indicate that the disfluency correction strategy outperforms the filtering strategy, on both fluency and rhyming criteria. With disfluency correction, 34.12% of the generated responses were rated *good* on average, while the disfluency filtering strategy produced only *good* responses 28.63% of the time. This result is not surprising, as harshly pruning the training corpus causes useful word association information necessary for rhyming to be lost. It would thus be interesting in future research to investigate the effect of alternative disfluency correction strategies on the performance of our system.

5.3. Challenge-response results

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. Freestyle produces a significantly higher percentage of *good* and *acceptable* rhyming responses compared to the standard baseline for both disfluency filtering and disfluency correction strategies. Also, Freestyle outperforms Pbsmt on the task of fluency when trained on the corrected version of the corpus. (Although Freestyle 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.) These results indicate that conventional Pbsmt approaches are not very effective on this novel task. We also notice that with an appropriately chosen data selection scheme, transduction model, and search heuristics, we can achieve significantly better results.

Table 2: Examples of challenges and responses.

challenge	man i **die** to **see** em all thun i just don't **care**
Freestyle	in the **sky** and **me** the in polla and the you **there**
Pbsmt	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**
Freestyle	to the girls here **kid** the and to the thought the **now**
Pbsmt	did a year in the streets and it down
challenge	oh i believe in **yesterday**
Freestyle	can you see the **day**
Pbsmt	oh i believe in tomorrow
challenge	what would i **do**
Freestyle	just me and **you**
Pbsmt	what would you do
challenge	cause you ain't going **home** till the early **morn**
Freestyle	and the you this **alone** i i gotta **on**
Pbsmt	cause you and your friends aint nothing but

Table 2 shows some of the challenges and the corresponding responses of Pbsmt and Freestyle. Both generate responses reflecting a high degree of fluency, even though Freestyle does not use a segmental phrase-based transduction grammar. However, Pbsmt tends to produce responses that are too similar to the challenge. Moreover, Freestyle produces responses that indeed rhyme better (shown in boldface). In fact, Freestyle tries to rhyme words not only at the end but also in middle of the lines, as our transduction grammar model captures structural associations more effectively than the phrase-based model.

6. Conclusions

We have presented the first known fully-automatically learned challenge-response system to produce rhyming hip hop lyrics using no prior phonetic or linguistic knowledge, relying instead on completely unsupervised induction of an SMT-style stochastic transduction grammar. We compared the performance of our system 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 also identified some of the problems resulting from disfluencies and backing vocals, which are characteristic to the domain of song lyrics. We plan follow-on investigation into training data selection, disfluency handling, search heuristics, and novel transduction models to further improve performance.

7. Acknowledgements

This material is based upon work supported in part by the Hong Kong Research Grants Council (RGC) research grants GRF620811, GRF612806, DAG03/04.EG09, RGC6256/00E, and RGC6083/99E; 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.

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

8. References

[1] Mark Liberman. (2010, December) Rap scholarship, rap meter, and the anthology of mondegreens. [Online]. Available: http://languagelog.ldc.upenn.edu/nll/?p=2824

[2] Long Jiang and Ming Zhou, "Generating Chinese couplets using a statistical MT approach," in *20th International Conference on Computational Linguistics (COLING-08)*, 2008.

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

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

[5] Dmitriy Genzel, Jakob Uszkoreit, and Franz Och, "Poetic statistical machine translation: rhyme and meter," in *2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 158–166.

[6] Morgan Sonderegger, "Applications of graph theory to an English rhyming corpus," *Computer Speech & Language*, vol. 25, no. 3, pp. 655–678, 2011.

[7] Ananth Ramakrishnan A, Sankar Kuppan, and Sobha Lalitha Devi, "Automatic generation of Tamil lyrics for melodies," in *Workshop on Computational Approaches to Linguistic Creativity*, 2009, pp. 40–46.

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

[9] On-Line Encyclopedia of Integer Sequences. Bell or exponential numbers: ways of placing n labeled balls into n indistinguishable boxes. [Online]. Available: http://oeis.org/A000110

[10] Dekai Wu, "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora," *Computational Linguistics*, vol. 23, no. 3, pp. 377–403, 1997.

[11] ——, "An algorithm for simultaneously bracketing parallel texts by aligning words," in *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, Massachusetts, June 1995, pp. 244–251. [Online]. Available: http://www.aclweb.org/anthology/P95-1033

[12] ——, "Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora," in *IJCAI*, vol. 95, 1995, pp. 1328–1335.

[13] ——, "Textual entailment recognition using inversion transduction grammars," in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, ser. Lecture Notes in Computer Science, Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d' Alché Buc, Eds. Springer Berlin Heidelberg, 2006, vol. 3944, pp. 299–308. [Online]. Available: http://dx.doi.org/10.1007/11736790_17

[14] Dekai Wu and Pascale Fung, "Inversion transduction grammar constraints for mining parallel sentences from quasi-comparable corpora," in *Natural Language Processing–IJCNLP 2005*. Springer, 2005, pp. 257–268.

[15] Richard Zens and Hermann Ney, "A comparative study on reordering constraints in statistical machine translation," in *41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 144–151. [Online]. Available: http://dx.doi.org/10.3115/1075096.1075115

[16] Aria Haghighi, John Blitzer, John DeNero, and Dan Klein, "Better word alignments with supervised itg models," in *Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, August 2009, pp. 923–931. [Online]. Available: http://www.aclweb.org/anthology/P/P09/P09-1104

[17] 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)*, vol. 39, no. 1, pp. 1–38, 1977.

[18] Dekai Wu, "Trainable coarse bilingual grammars for parallel text bracketing," in *Third Annual Workshop on Very Large Corpora*, Cambridge, Massachusetts, Jun 1995, pp. 69–81.

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

[20] Markus Saers, Joakim Nivre, and Dekai Wu, "Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm," in *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, Paris, France, October 2009, pp. 29–32.

[21] ——, "Word alignment with stochastic bracketing linear inversion transduction grammar," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, California, June 2010, pp. 341–344.

[22] Markus Saers and Dekai Wu, "Principled induction of phrasal bilexica," in *Proceedings of the 15th International Conference of the European Association for Machine Translation*, Leuven, Belgium, May 2011, pp. 313–320.

[23] Markus Saers, Joakim Nivre, and Dekai Wu, "A systematic comparison between inversion transduction grammar and linear transduction grammar for word alignment," in *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, Beijing, China, August 2010, pp. 10–18.

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

[25] Dekai Wu and Wong Hongsing, "Machine translation with a stochastic grammatical channel," in *Proceedings of the 17th international conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1998, pp. 1408–1415.

[26] John Cocke, *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences, New York University, 1969.

[27] David Chiang, "Hierarchical phrase-based translation," *Computational Linguistics*, vol. 33, no. 2, pp. 201–228, 2007. [Online]. Available: http://aclweb.org/anthology-new/J/J07/J07-2003.pdf

[28] 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," in *45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic, June 2007, pp. 177–180.

[29] Andreas Stolcke, "SRILM – an extensible language modeling toolkit," in *International Conference on Spoken Language Processing*, Denver, Colorado, September 2002, pp. 901–904.