Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Learning to Rap Battle with Bilingual Recursive Neural Networks

Dekai Wu and **Karteek Addanki**
Human Language Technology Center
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{dekai—vskaddanki}@cs.ust.hk

Abstract

We describe an unconventional line of attack in our quest to teach machines how to rap battle by improvising hip hop lyrics on the fly, in which a novel recursive *bilingual* neural network, TRAAM, implicitly learns soft, context-dependent generalizations over the structural relationships between associated parts of challenge and response raps, while avoiding the exponential complexity costs that symbolic models would require. TRAAM learns feature vectors *simultaneously* using context from both the challenge and the response, such that challenge-response association patterns with similar structure tend to have similar vectors. Improvisation is modeled as a quasi-translation learning problem, where TRAAM is trained to improvise fluent and rhyming responses to challenge lyrics. The soft structural relationships learned by our TRAAM model are used to improve the probabilistic responses generated by our improvisational response component.

1 Introduction

Improvising rap responses in the face of freestyle challenges is a highly creative endeavor that demands numerous levels of artistry. The response should of course be somehow salient to the challenge lyrics, but a wide range of different types of salience are allowable. At the same time, the response lyrics should flow fluently, yet allowing for stylistic ungrammaticality, disfluencies such as stuttering, and slang constructs. Moreover, a response containing some metrical and/or syntactic structures that parallel those in the challenge are highly appreciated. On top of this, aesthetic responses should usually rhyme not only at the ends of the lines, but also frequently at different points in middle of lines, especially near key endpoints of metrical substructures.

Learning to improvise such rap responses presents an even more challenging AI problem. All of these preferences or biases are soft; none are hard and fast rules. More importantly, the choices influence and constrain each other; in other words, they are highly context dependent.

This context dependence poses, for traditional symbolic machine learning and language processing models, nasty combinatorial complexity issues in the hypothesis spaces, for both the response generation search and the model learning search. To tackle this, we propose instead to shift to a radically different approach, that allows soft sets of multiple similar hypotheses to be simultaneously entertained by representing them using a single compositional distributed vector. As with conventional neural networks, since similar vectors tend to represent similar association patterns between challenge and response parts, any single vector represents a neighborhood of similar hypotheses.

Conventional neural nets, however, do not support any way of encoding or learning vectors capable of capturing all of the structurally complex types of preferences and biases we face in this task. We therefore propose instead to use a recursive *bilingual* neural network, that can be seen as a generalized form of recursive auto-associative memory or RAAM [Pollack, 1990]. Our bilingual generalization is essential because RAAM only models a single language (just like symbolic language models do)whereas to capture association patterns between the challenge language and the response language, we need to model relationships between *two* languages.

Under our proposed approach, improvisation is modeled as a quasi-translation task in which any given challenge is "translated" into a response. This is translation, or transduction, in the mathematical sense, as in formal language theory; it is of course not translation in the linguistic sense.

2 Related work

Our TRAAM recursive bilingual neural network builds on different aspects of a spectrum of previous work. A large body of work exists on learning distributed representations for modeling recursive structure, but mostly in monolingual setting. Even in applications to machine translation or cross-lingual modeling, the typical practice has been to insert neural network scoring components while maintaining older modeling assumptions like bags-of-words/phrases, "shake'n'bake" translation that relies on heuristic based phrase extraction and model tuning using feature weights in contrast to our fully integrated model. We survey representative work across the spectrum.

2.1 Monolingual related work

RAAM approaches have been successfully used to model monolingual compositional structure given their flexibility and the ability to learn task-specific representations. Stolcke

and Wu [1992] present a Unification RAAM model which learns a distributed representation to perform unification on the distributed vector representations of hierarchical feature structures. Socher *et al.* [2011] use monolingual recursive autoencoders for sentiment prediction and report performance improvement; this was perhaps the first use of a RAAM style approach on a large scale NLP task, albeit monolingual.

Distributed vector representations have long been used for n-gram language modeling; these continuous-valued models exploit the generalization capabilities of neural networks, although there is no hidden contextual or hierarchical structure as in RAAM. Bengio *et al.* [2003] shows that a feed-forward neural network based language model can be trained efficiently and outperforms symbolic approaches. More recently, the probabilistic NNLMs of Mikolov *et al.* [2010], use simple recurrent neural networks (RNNs or SRNs) of Elman [1990], where hidden layer representations are fed back to the input to dynamically represent an aggregate of the immediate contextual history and achieve state-of-the-art performance.

Lee *et al.* [2009] use convolution neural networks to represent hierarchical tree structure using vector representations. Collobert and Weston [2008] use a convolution neural network layer quite effectively to learn vector representations for words which are then used in a host of NLP tasks such as POS tagging, chunking, and semantic role labeling.

2.2 Bilingual related work

The majority of work on learning bilingual distributed vector representations has not made use of recursive approaches or hidden contextual or compositional structure, as in the bilingual word embedding learning of Klementiev *et al.* [2012] or the bilingual phrase embedding learning of Gao *et al.* [2014]. Schwenk [2012] uses a non-recursive neural network to predict phrase translation probabilities in conventional phrase-based SMT systems.

Attempts have been made to generalize the distributed vector representations of monolingual n-gram language models, avoiding any hidden contextual or hierarchical structure. Working within the framework of n-gram translation models, Son *et al.* [2012] generalize left-to-right monolingual n-gram models to bilingual n-grams, and study bilingual variants of class-based n-grams. However, their model does not allow tackling the challenge of modeling cross-lingual constituent order, as our model does; instead it relies on the assumption that some other preprocessor has already managed to accurately re-order the words of the input sentence into exactly the order of words in the output sentence.

Similarly, generalizations of monolingual SRNs to the bilingual case have been studied. Zou *et al.* [2013] generalize the monolingual recurrent NNLM model of Mikolov *et al.* [2010] to learn bilingual word embeddings using word alignments, and demonstrate that the resulting embeddings outperform the baselines in word semantic similarity. Compared to our model, however, they only learn non-compositional features, with distributed vectors only representing biterminals (as opposed to biconstituents or bilingual subtrees), and so other mechanisms for combining biterminal scores still need to be used to handle hierarchical structure, as opposed to seamlessly being integrated into the distributed vector representation model. Devlin *et al.* [2014] obtain translation accuracy improvements by extending the probabilistic NNLMs of Bengio *et al.* [2003], which are used for the output language, by adding input language context features. Unlike our model, neither of these approaches symmetrically model the recursive structure of both input and the output languages.

A few applications of monolingual RAAM-style recursive autoencoders to bilingual tasks have also appeared. For cross-lingual document classification, Hermann and Blunsom [2014] use two separate monolingual fixed vector composition networks, one for each language. One provides the training signal for the other, and training is only on the embeddings. Li *et al.* [2013] use a monolingual recursive autoencoder for predicting reordering in an ITG model using the context vector generated using the recursive autoencoder. Only input language context is used, unlike our model which can use both input and output language contexts equally.

3 TRAAM

TRAAM is a bilingual generalization of the monolingual recursive autoassociative memory model of Pollack [1990] and serves as our bilingual recursive neural network used in improvising hip hop lyrics. TRAAM implicitly learns soft, context-dependent generalizations over the structural relationships between the corresponding parts of the challenges and responses in our training data and unlike symbolic models, our model does not incur the exponential cost of modeling sensitivity to context. As improvisation is modeled as a translation problem, our neural network model is a distributed representation of a transduction grammar and is ideally suited to be the translation model.

3.1 Why use transduction grammars?

Stochastic transduction grammars have been highly successful for translating from one human language to another, but mathematically, they simply model structural relationships between two sequential representation languages, and thus provide an excellent framework for modeling the relationship between challenge languages and response languages. An Inversion Transduction grammar or an ITG [Wu, 1997] is a subset of the syntax directed transduction grammars [Lewis and Stearns, 1968] and contains transduction rules that generate both input and output language subtrees and have been used successfully across a spectrum of NLP tasks. The subtrees are generated by recursively combining smaller bispans (chunks of aligned input and output segments) into larger bispans. The alignment between these subtrees is restricted to be only straight or inverted, thus permitting reordering in the input and output language tokens. Although ITGs can accommodate transduction grammar rules with different categories of nonterminals, ITGs with just a single undistinguished nonterminal (known as Bracketing Inversion Transduction Grammars or BITGs) have shown to have high coverage in cross-lingual tasks [Zens and Ney, 2003; Saers *et al.*, 2009; Addanki *et al.*, 2012]. Further, using BITGs enables us to model our input and output in the same natural language (English) without any linguistic assumptions. The left half of the

Figure 1, shows a BITG parse tree with the inverted alignments represented by a horizontal dash and each bispan is represented using a 4-tuple s, t, u, v which corresponds to the input segment with tokens $e_s, e_{s+1}, \ldots, e_t$ and the output segment with tokens $f_u, f_{u+1}, \ldots, f_v$. We show some examples of the transduction grammar rules learned from our data below.

$$A \rightarrow [AA], A \rightarrow \langle AA \rangle, A \rightarrow \text{singin/dancing}, A \rightarrow \text{angels/top}$$

In the context of our task, rap battling, we would like to generate responses that are fluent and rhyme with the challenge and not merely generation of coherent lyrics. For this purpose, we use ITGs so as to capture the correspondences between the challenge and the response lines using transduction rules. By correspondences, we refer to rhyming, alliteration and the context of the challenge-response pair. Our goal is to learn these correspondences in a completely unsupervised manner, without using any language specific resources. Simpler models such as WFSTs are restricted by their assumption of monotonicity of bilingual relationships and fail to capture interesting relationships captured by ITGs. For example, ITG constraints on our training data not only captured rhyming associations such as A stop/drop, A history/mysteries, A big/figs, but also context specific relations such as A big/pac (referring to the artists Notorious B.I.G and 2Pac). Data inspection revealed that a number of these associations were a result of non-monotonic alignments between the input and output languages.

While most previous approaches, even in more structured domains, employ an uninformed model to generate fluent output and make use of specialized knowledge resources to ensure output compatible with the constraints of the problem being modeled [Barbieri *et al.*, 2012; Jiang and Zhou, 2008], we attempt to extract as much task-specific knowledge as possible from our training data. Further, our model can be easily extended to employ task-specific resources such as rhyming dictionaries or heuristics such as phoneme matching.

3.2 Distributed representations are sensitive to context

Our TRAAM model learns the distributed representations that parallel the structural composition of a BITG. Figure 1 shows a biparse using a bracketing inversion transduction grammar (BITG) and the corresponding network of the TRAAM model. However, unlike the single undifferentiated nonterminal of a BITG, the feature vectors in effect represent soft categories of cross-lingual relations. Our model can integrate elegantly with the transduction grammar formalism enabling one to augment the symbolic models with distributed, soft, context-dependent generalizations over the input and output languages.

In our recursive neural network, each bispan s, t, u, v is represented using a feature vector $\mathbf{v}_{stuv}$ of dimension d and similar to BITGs, feature vectors of larger bispans are recursively generated from smaller bispans using a *compressor* network (represented by C in Figure 1). The compressor network takes two feature vectors of dimension d (corresponding to smaller bispans), along with a single bit corresponding to straight or inverted order and outputs a feature vector of

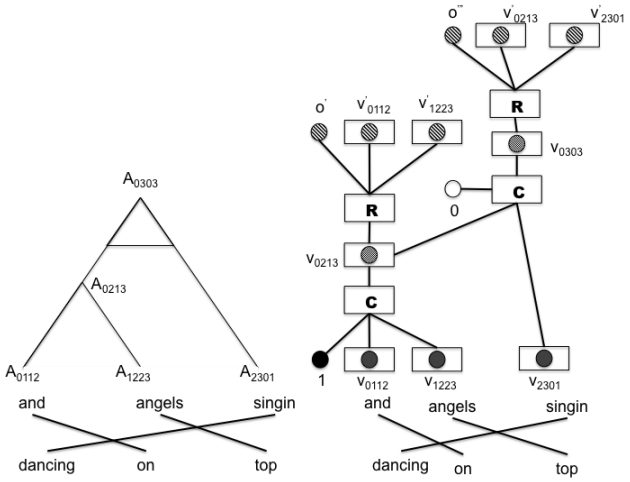

Figure 1: Symbolic model (left) and the corresponding TRAAM model (right)

dimension d (corresponding to the larger bispan). The reconstruction network (represented by R in Figure 1) provides the loss function which can be used to train the representations of the bitokens and weights of the networks as explained in the next section.

The loss function defined by the neural network is a continuous valued function whose output varies on the input feature vectors. However, unlike the BITG, which can only guarantee a fixed probability for all straight or inverted combinations, the neural network model can facilitate for more complex interactions such as nonlinear interactions using compressor and reconstructor networks. As the output feature vector is dependent on the bispan being dominated, the neural network model is more sensitive to context than a BITG model. For the BITG model to have the same context-dependent behavior as the neural model, the continuous valued function needs to be emulated using different nonterminal categories such that both straight and inverted combinations of each pair of nonterminals, can have a different value. This would cause the number of nonterminals and hence the number of model parameters to explode making reliable parameter estimation extremely hard.

3.3 Training algorithm

We assume a "translation lexicon" (corresponding to the associations of words between challenges and responses), as one of the inputs to train our model. We obtain this lexicon by training a BITG in a completely unsupervised fashion on our training data [Saers *et al.*, 2009]. The lexical rules in the grammar provide us with a translation lexicon. The transduction grammar also enable faster training, as exhaustively considering all possible alignments between the challenge and response makes it expensive to train our neural network.

Initialization

Each entry in the translation lexicon is associated with a pair of feature vectors, corresponding to the input and output language segments respectively. Using the terminology of trans-

duction grammars, we refer to the pair of input and output language segments in the translation lexicon a *biterminal*. The feature vector for the biterminal is a concatenation of the input and output language vectors. In the case, where a translation lexicon is unavailable, the model can be naively trained assuming that all the tokens in the input sentence translate to all the tokens in the output sentence. We assume that each of the vectors are of dimension $d/2$ and are initialized randomly such that their magnitude is 1. For all the segments $\mathbf{e}$ and $\mathbf{f}$ such that $\mathbf{e}/\mathbf{f}$ exists in the translation lexicon:

$$\mathbf{v_e} = [v_{\mathbf{e}_1}, \ldots, v_{\mathbf{e}_{d/2}}]$$

$$\mathbf{v_f} = [v_{\mathbf{f}_1}, \ldots, v_{\mathbf{f}_{d/2}}]$$

Given a sentence pair, we will need to initialize the chart such that for each bispan s, t, u, v a feature vector $\mathbf{v}_{stuv}$ of dimension d by concatenating the feature vectors $\mathbf{v}_{e_{st}}$ and $\mathbf{v}_{f_{uv}}$ corresponding to the input and output segments respectively.

$$\mathbf{v}_{stuv} = [\mathbf{v}_{e_{st}}; \mathbf{v}_{f_{uv}}]$$

where ; represents the concatenation operator for the feature vectors. The feature vectors for larger spans are generated recursively from the smaller spans using the compressor network, the details of which are described in the next section.

Recursion

Our model generates the feature vectors for larger bispans recursively from smaller bispans using both input and output language contexts simultaneously. However, similar to monolingual RAAM models we would also like to keep the dimensionality of the feature vectors fixed irrespective of the length of the bispan. Hence when two smaller bispans combine, we need to "compress" two feature vectors of dimension d corresponding to both the bispans, to one feature vector of dimension d. The compressor network serves this purpose and provides the necessary language bias by forcing the network to capture the generalizations while reducing the dimensions of the input vectors. As the bispans can combine in straight or inverted fashion, an additional bit of input corresponding to the permutation order is also fed into the compressor. We use a single layer with a nonlinear activation function ($\tanh$) similar to the monolingual recursive autoencoder [Socher *et al.*, 2011] as shown below.

$$\mathbf{v}_{stuv} = \frac{\tanh(W_1[1;\mathbf{v}_{sSuU};\mathbf{v}_{StUv}]+b_1)}{|\tanh(W_1[1;\mathbf{v}_{sSuU};\mathbf{v}_{StUv}]+b_1)|}$$

$$\mathbf{v}_{stuv} = \frac{\tanh(W_1[0;\mathbf{v}_{sSUv};\mathbf{v}_{StuU}]+b_1)}{|\tanh(W_1[0;\mathbf{v}_{sSUv};\mathbf{v}_{StuU}]+b_1)|}$$

The feature vector for the larger span is the output of the activation layer of the compressor network, whose inputs are the feature vectors of smaller bispans and the first bit of the input is set depending on whether they combine in straight or inverted order. The vectors are normalized at each stage to prevent the model from reaching a degenerate optimum of arbitrarily small vectors. The bispans are combined recursively in a bottom-up manner till the feature vector corresponding to the entire bisentence is obtained. In order to reduce the training time, we use the Viterbi biparse from a BITG to provide the tree structure for our training.

One of the goals of our model is to capture: (1) the context of the smaller bispans as efficiently as possible in the larger bispan, and (2) the reordering information i.e., the order in which the bispans combine recursively. While the compressor network can realize these goals, we need to train our model so that the learned parameters realize this objective as well as possible. Therefore, we use a reconstructor network which attempts to reconstruct the permutation order and the feature vectors of the children from the feature vector of a bispan. As our reconstructor network, we use a single layer network with a nonlinear activation function ($\tanh$) as shown below. Note that the equation below assumes that the original feature vectors were combined in a straight permutation order but the converse is also fairly straightforward.

$$[o'; \mathbf{v}'_{sSuU}; \mathbf{v}'_{StUv}] = \tanh(W_2\mathbf{v}_{stuv} + b_2)$$

$\mathbf{v}'_{sSuU}$ and $\mathbf{v}'_{StUv}$ are the reconstructed feature vectors corresponding to the respective bispans and o' is the reconstructed permutation order. The reconstructed feature vectors and permutation order are used to compute the loss function which ensures that our trained model achieves the objective stated above. Each internal node in the biparse tree reconstructs the output and the permutation order of its children using the reconstructor network (represented by R) as shown in the Figure 1.

The parameters of the model are the weight and bias matrix of W_1 and b_1 of the compressor network, and the weight and the bias matrix of the reconstructor W_2 and b_2 and the feature vectors of the input and output segments in the translation lexicon. These parameters are estimated by minimizing the loss function described in the next section.

Gradient computation

Our objective of training is such that the model captures the context and the permutation order of the children as efficiently as possible. We define the loss function as the sum of the normalized reconstruction error at all the internal nodes in the biparse tree. The loss function is defined as a linear combination (with the linear weighting factor α) of the L2 norm of the reconstruction error of the children and the cross-entropy loss of reconstructing the permutation order. We define the error at each internal node corresponding to the bispan s, t, u, v generated by the straight combination is shown below and the equation for the inverted combination follows similarly.

$$E_{stuv} = \frac{\alpha}{2}\|[v'_{sSuU}; v'_{StUv}] - [v_{sSuU}; v_{StUv}]\|^2 + (1-\alpha)\log(1+o')$$

The global objective function J is the sum of the error function at all internal nodes in the biparse trees, averaged over the total number of sentences T in the corpus. A regularization parameter λ is used on the norm of the model parameters θ, to avoid overfitting. As the bisegment embeddings are also a part of the model parameters, the optimization objective is similar to a moving target training objective and we use backpropagation with structure [Goller and Kuchler, 1996] to compute the gradients efficiently. L-BFGS algorithm is used in order to minimize the loss function.

$$J = \frac{1}{T}\Sigma_{stuv}E_{stuv} + \lambda\|\theta\|^2$$

3.4 Decoding algorithm

Our decoding algorithm is similar to decoding with an ITG model and makes use of the BITG we use in training. However, the major distinction is that each translation hypothesis computes an additional *feature* score which is the inverse of the reconstruction error of the feature vectors its children when fed through the compressor network. The *feature* score along with the transduction grammar and LM score is used to score each hypothesis using a weighted linear combination. The *feature* score enables us to generate a more fine grained score instead of a single value (as in case of grammar score for BITGs) for each straight and inverted combination of the hypotheses, enabling the parses preferred by our TRAAM model to rank higher. Without re-defining the entire decoding algorithm for ITGs, we just define the parts that are relevant in computing the *feature* score.

Initialization

For each input span f_{st}, we initialize the set of translation hypotheses using the grammar rules that match the input span i.e., all grammar rules $X \rightarrow \mathbf{e}/\mathbf{f}$ where $\mathbf{f} = f_{st}$. The feature vector v_{styz} of the translation hypothesis of the span f_{st} is initialized as the feature vector from our model where the bisegment matches $\mathbf{e}/\mathbf{f}$, y and z are the first and last $N-1$ tokens of $\mathbf{e}$, and N is the order of the language model used.

Recursion

The recursion step is similar to that of Wu [1996], with the additional step of computing the feature score, using the reconstruction error of each hypothesis combined. The grammar score, LM score and the feature score are combined using a weighted linear combination to give each hypothesis a score.

1. When two hypotheses h_{skyY} and h_{ktZz} corresponding to spans (s,k) and (k,t) combine we compute the feature score for their combination as follows:
   (a) If the combination is according to a straight rule then the feature score τ_{styz} corresponding to the combined hypothesis h_{styz}
      i. $v_{styz} = \text{compressor}\left([1; v_{skyY}; v_{ktZz}]\right)$
      ii. $[p'; v'_{skyY}; v'_{ktZz}] = \text{reconstructor}\left(v_{styz}\right)$
      iii. $\tau_{styz} = \dfrac{1}{\parallel [v'_{skyY}; v'_{ktZz}] - [v_{skyY}; v_{ktZz}] \parallel}^2$
   (b) Else then the feature score τ_{stZY} corresponding to the combined hypothesis h_{stZY}
      i. $v_{stZY} = \text{compressor}\left([0; v_{skyY}; v_{ktZz}]\right)$
      ii. $[p'; v'_{skyY}; v'_{ktZz}] = \text{reconstructor}\left(v_{stZY}\right)$
      iii. $\tau_{stZY} = \dfrac{1}{\parallel [v'_{skyY}; v'_{ktZz}] - [v_{skyY}; v_{ktZz}] \parallel}^2$

Reconstruction

Reconstruction is similar to the Viterbi decoding mentioned in Wu [1996] except for the inclusion of the feature scores in computing the score of each hypothesis.

4 Experiments

We used freely available user generated hip hop lyrics on the Internet to provide training data for our experiments. The processed corpus contained 22 million tokens with 260,000 verses and 2.7 million lines of hip hop lyrics. As human evaluation is expensive, a small subset of 85 lines was chosen as the test set to provide challenges for comparing the quality of responses generated by different systems. We followed the evaluation scheme proposed by Addanki and Wu [2014] as it achieved very encouraging inter-evaluator agreements despite the high degree of subjectivity of the evaluation task. The output of both the baseline and our model, was given to three independent frequent hip hop listeners familiar with freestyle rap battling for manual evaluation. 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 evaluator was asked to score the response of each system on the criterion of fluency and rhyming as being *good*, *acceptable* or *bad*.

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

model	*fluency (acceptable)*	*rhyming (acceptable)*
PBSMT	43.53%	9.02%
TRAAM	**60.39%**	**42.74%**

4.1 TRAAM model training

We use the TRAAM bilingual recursive neural network model discussed earlier along with a token based transduction grammar model trained on around 200,000 lines of challenge response pairs. The challenge response pairs were selected using a rhyme scheme detection module proposed in Addanki and Wu [2013]. We use the translation lexicon from the trained transduction grammar along with the biparses to train our neural network model. Both these models are then used to improvise the responses using a 4-gram language model which was trained on the entire training corpus using SRILM [Stolcke, 2002]. The weights of the feature scores were determined empirically observing the performance on a small subset of the test data. In order to evaluate the performance of an out-of-the-box phrase-based SMT (PBSMT) system toward this novel task of generating rhyming and fluent responses, a standard Moses baseline [Koehn *et al.*, 2007] was also trained in order to compare its performance with our TRAAM model.

5 Results

In this section, we present the results of human evaluation on the improvised responses of our model and demonstrate that our TRAAM model does provide better quality responses compared to a phrase based SMT baseline.

5.1 Phrase-based SMT baseline performs poorly

Table 1 shows the average fraction of sentences rated *good* and *acceptable* for each model. Compared to the phrase-based SMT (PBSMT) baseline, our TRAAM model produces significantly higher percentage of *good* and *acceptable* rhyming responses. Surprisingly, despite being a token based model, our model outperforms the segmental PBSMT model even on the criterion of fluency. These results indicate that

Table 2: Examples of challenges and responses generated.

challenge	picture on the **quota** its time to **roll**
TRAAM	ya for a **quarter** i mind and **soul**
PBSMT	on the odor time to roll
challenge	thug deep in my **soul** that got me **bugged**
TRAAM	love you on the **control** the **drugs**
PBSMT	thug in my soul that got me
challenge	nights of 51 jipped be **light** on this **cash**
TRAAM	in the concrete mics you **right** in the **ass**
PBSMT	and be light on this **ass**
challenge	what would i **do**
TRAAM	just me and **you**
PBSMT	what would you do
challenge	we get rid of the **child**
TRAAM	and the number and a **wild**
PBSMT	we get of the

our TRAAM bilingual neural network model captures enough context to generate fluent responses, significantly augmenting the performance of a token based model.

5.2 Challenge-response examples

Table 2 shows some of the challenges and the corresponding responses of our model and the PBSMT baseline. Our model produces responses comparable in fluency to the segmental PBSMT model despite being a token based model. While PBSMT model produces responses which are too similar to the challenge, our model improvises responses which rhyme better (shown in boldface). As seen from examples, our model frequently produces responses that rhyme words not only at the end, but also in the middle of challenges, indicating that it captures structural associations more effectively than the phrase-based model.

6 Conclusion

Teaching machines to rap battle is a quest that encapsulates numerous interacting levels of improvisational artistry in a complex, structured AI learning challenge. We have described an unconventional line of attack in which a recursive *bilingual* neural network, TRAAM, sidesteps the exponentially complex hypothesis space needed by existing suitable symbolic learning models for both the improvisational response generation search and the model learning search, by instead using compositional distributed vector representations in which a single vector implicitly represents an entire neighborhood of multiple similar association patterns between corresponding structural aspects of challenges and responses. The fact that challenge-response association patterns that are structurally similar tend to have similar vectors allows training to learn soft, context-dependent generalizations over all kinds of structural challenge-response associations patterns, from concrete to abstract patterns, and from short to long patterns.

Our approach is unlike conventional approaches to poetry in being completely unsupervised, making zero use of any linguistic or phonetic features in spite of an extremely unstructured and noisy domain. Modeling improvisation as a quasi-translation learning problem means that for any challenge, the machine must learn on its own what kinds of improvised responses would be fluent, salient, rhyming, and of similar metrical and syntactic structure. The distributed feature vectors that encode challenge-response association patterns are learned *simultaneously* by our TRAAM network, using context from both the challenge and the response. The soft structural relationships learned are used to improve the probabilistic responses generated by our improvisational response component, as judged by human rap listeners.

Acknowledgements

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under BOLT contract nos. HR0011-12-C-0014 and HR0011-12-C-0016, and GALE contract nos. HR0011-06-C-0022 and HR0011-06-C-0023; by the European Union under the FP7 grant agreement no. 287658; and by the Hong Kong Research Grants Council (RGC) research grants GRF620811, GRF621008, GRF612806, FSGRF13EG28, and FSGRF14EG35. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the EU, or RGC.

References

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

[Addanki and Wu, 2014] Karteek Addanki and Dekai Wu. Evaluating improvised hip hop lyrics - challenges and observations. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[Addanki *et al.*, 2012] Karteek Addanki, Chi-Kiu Lo, Markus Saers, and Dekai Wu. LTG vs. ITG coverage of cross-lingual verb frame alternations. In *16th Annual Conference of the European Association for Machine Translation (EAMT-2012)*, Trento, Italy, May 2012.

[Barbieri *et al.*, 2012] Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. In *20th European Conference on Artificial Intelligence, (ECAI 2012)*, pages 115–120, 2012.

[Bengio *et al.*, 2003] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning.

In *25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[Devlin *et al.*, 2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[Elman, 1990] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[Gao *et al.*, 2014] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, 2014.

[Goller and Kuchler, 1996] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.

[Hermann and Blunsom, 2014] Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. In *52nd Annual Meeting of the Association for Computational Linguistics*, volume abs/1404.4641, 2014.

[Jiang and Zhou, 2008] Long Jiang and Ming Zhou. Generating Chinese couplets using a statistical MT approach. In *22nd International Conference on Computational Linguistics (COLING 2008)*, 2008.

[Klementiev *et al.*, 2012] Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. Inducing crosslingual distributed representations of words. In *24th International Conference on Computational Linguistics (COLING 2012)*. Citeseer, 2012.

[Koehn *et al.*, 2007] 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 *Interactive Poster and Demonstration Sessions of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pages 177–180, Prague, Czech Republic, June 2007.

[Lee *et al.*, 2009] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[Lewis and Stearns, 1968] Philip M. Lewis and Richard E. Stearns. Syntax-directed transduction. *Journal of the Association for Computing Machinery*, 15(3):465–488, 1968.

[Li *et al.*, 2013] Peng Li, Yang Liu, and Maosong Sun. Recursive autoencoders for itg-based translation. In *EMNLP*, pages 567–577, 2013.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[Pollack, 1990] Jordan B Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105, 1990.

[Saers *et al.*, 2009] Markus Saers, Joakim Nivre, and Dekai Wu. Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm. In *11th International Conference on Parsing Technologies (IWPT'09)*, pages 29–32, Paris, France, October 2009.

[Schwenk, 2012] Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *COLING 2012: Posters*, pages 1071–1080. Citeseer, 2012.

[Socher *et al.*, 2011] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[Son *et al.*, 2012] Le Hai Son, Alexandre Allauzen, and François Yvon. Continuous space translation models with neural networks. In *2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 39–48. Association for Computational Linguistics, 2012.

[Stolcke and Wu, 1992] Andreas Stolcke and Dekai Wu. Tree matching with recursive distributed representations. In *AAAI 1992 Workshop on Integrating Neural and Symbolic Processes —The Cognitive Dimension*, 1992.

[Stolcke, 2002] Andreas Stolcke. SRILM – an extensible language modeling toolkit. In *7th International Conference on Spoken Language Processing (ICSLP2002 - INTERSPEECH 2002)*, pages 901–904, Denver, Colorado, September 2002.

[Wu, 1996] Dekai Wu. A polynomial-time algorithm for statistical machine translation. In *34th Annual Meeting of the Association for Computational Linguistics (ACL96)*, pages 152–158, Morristown, NJ, USA, 1996.

[Wu, 1997] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

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

[Zou *et al.*, 2013] Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398, 2013.