Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings

Ondřej Dušek and **Filip Jurčíček**
Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, CZ-11800 Prague, Czech Republic
{odusek,jurcicek}@ufal.mff.cuni.cz

Abstract

我们提出了一种基于Seq-to-Seq
方法的自然语言生成器，可以训
练它从输入对话行为生成自然语
言字符串和深度语法依赖树，
将它直接比较两步生成
(分别为句子规划和表面实现阶段)
和联合的一步生成方法的性能

We present a natural language generator based on the sequence-to-sequence approach that can be trained to produce natural language strings as well as deep syntax dependency trees from input dialogue acts, and we use it to directly compare two-step generation with separate sentence planning and surface realization stages to a joint, one-step approach.

我们能够使用很少的训练数据成
功地训练两种设置。联合设置提
供了更好的性能，在基于ngram的
分数方面超过了最先进的水平，
同时提供了更多相关的输出

We were able to train both setups successfully using very little training data. The joint setup offers better performance, surpassing state-of-the-art with regards to n-gram-based scores while providing more relevant outputs.

1 Introduction

In spoken dialogue systems (SDS), the task of natural language generation (NLG) is to convert a meaning representation (MR) produced by the dialogue manager into one or more sentences in a natural language. It is traditionally divided into two subtasks: *sentence planning*, which decides on the overall sentence structure, and *surface realization*, determining the exact word forms and linearizing the structure into a string (Reiter and Dale, 2000). While some generators keep this division and use a two-step pipeline (Walker et al., 2001; Rieser et al., 2010; Dethlefs et al., 2013), others apply a joint model for both tasks (Wong and Mooney, 2007; Konstas and Lapata, 2013).

We present a new, conceptually simple NLG system for SDS that is able to operate in both modes: it either produces natural language strings or generates deep syntax dependency trees, which are subsequently processed by an external surface realizer (Dušek et al., 2015). This allows us to show a direct comparison of two-step generation, where sentence planning and surface realization are separated, with a joint, one-step approach.

Our generator is based on the sequence-to-sequence (seq2seq) generation technique (Cho et al., 2014; Sutskever et al., 2014), combined with beam search and an n-best list reranker to suppress irrelevant information in the outputs. Unlike most previous NLG systems for SDS (e.g., (Stent et al., 2004; Raux et al., 2005; Mairesse et al., 2010)), it is trainable from unaligned pairs of MR and sentences alone. We experiment with using much less training data than recent systems based on recurrent neural networks (RNN) (Wen et al., 2015b; Mei et al., 2015), and we find that our generator learns successfully to produce both strings and deep syntax trees on the BAGEL restaurant information dataset (Mairesse et al., 2010). It is able to surpass n-gram-based scores achieved previously by Dušek and Jurčíček (2015), offering a simpler setup and more relevant outputs.

We introduce the generation setting in Section 2 and describe our generator architecture in Section 3. Section 4 details our experiments, Section 5 analyzes the results. We summarize related work in Section 6 and offer conclusions in Section 7.

2 Generator Setting

The input to our generator are *dialogue acts* (DA) (Young et al., 2010) representing an action, such as *inform* or *request*, along with one or more attributes (*slots*) and their values. Our generator operates in two modes, producing either deep syntax trees (Dušek et al., 2012) or natural language strings (see Fig. 1). The first mode corresponds to the sentence planning NLG stage as it decides the syntactic shape of the output sentence; the resulting deep syntax tree involves content words (lemmas) and their syntactic form (formemes, purple in Fig. 1). The trees are linearized to strings using a

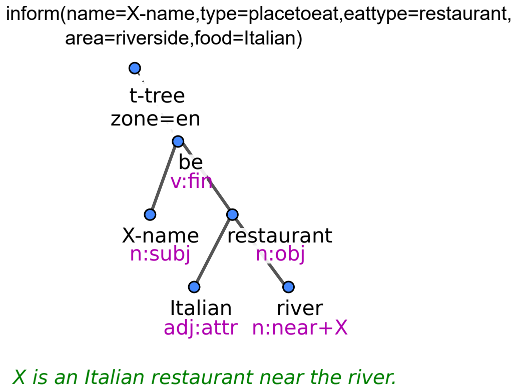
inform(name=X-name,type=placetoeat,eattype=restaurant,
area=riverside,food=Italian)
t-tree
zone=en
be
v:fin
X-name
n:subj
restaurant
n:obj
Italian
adj:attr
river
n:near+X
X is an Italian restaurant near the river.

Figure 1: Example DA (top) with the corresponding deep syntax tree (middle) and natural language string (bottom)

surface realizer from the TectoMT translation system (Dušek et al., 2015). The second generator mode joins sentence planning and surface realization into one step, producing natural language sentences directly.

Both modes offer their advantages: The two-step mode simplifies generation by abstracting away from complex surface syntax and morphology, which can be handled by a handcrafted, domain-independent module to ensure grammatical correctness at all times (Dušek and Jurčíček, 2015), and the joint mode does not need to model structure explicitly and avoids accumulating errors along the pipeline (Konstas and Lapata, 2013).

3 The Seq2seq Generation Model

Our generator is based on the seq2seq approach (Cho et al., 2014; Sutskever et al., 2014), a type of an encoder-decoder RNN architecture operating on variable-length sequences of tokens. We address the necessary conversion of input DA and output trees/sentences into sequences in Section 3.1 and then describe the main seq2seq component in Section 3.2. It is supplemented by a reranker, as explained in Section 3.3.

3.1 Sequence Representation of DA, Trees, and Sentences

We represent DA, deep syntax trees, and sentences as sequences of tokens to enable their usage in the sequence-based RNN components of our generator (see Sections 3.2 and 3.3). Each token is represented by its embedding – a vector of floating-point numbers (Bengio et al., 2003).

To form a sequence representation of a DA, we create a triple of the structure "DA type, slot, value" for each slot in the DA and concatenate the triples (see Fig. 3). The deep syntax tree output from the seq2seq generator is represented in a bracketed notation similar to the one used by Vinyals et al. (2015, see Fig. 2). The inputs to the reranker are always a sequence of tokens; structure is disregarded in trees, resulting in a list of lemma-formeme pairs (see Fig. 2).

3.2 Seq2seq Generator

Our seq2seq generator with attention (Bahdanau et al., 2015, see Fig. 3)[1] starts with the encoder stage, which uses an RNN to encode an input sequence $\mathbf{x} = \{x_1, \ldots, x_n\}$ into a sequence of encoder outputs and hidden states $\mathbf{h} = \{h_1, \ldots, h_n\}$, where $h_t = \text{lstm}(x_t, h_{t-1})$, a non-linear function represented by the long-short-term memory (LSTM) cell (Graves, 2013).

The decoder stage then uses the hidden states to generate a sequence $\mathbf{y} = \{y_1, \ldots, y_m\}$ with a second LSTM-based RNN. The probability of each output token is defined as:

$$p(y_t | y_1, \ldots, y_{t-1}, \mathbf{x}) = \text{softmax}((s_t \circ c_t) W_Y)$$

Here, s_t is the decoder state where $s_0 = h_n$ and $s_t = \text{lstm}((y_{t-1} \circ c_t) W_S, s_{t-1})$, i.e., the decoder is initialized by the last hidden state and uses the previous output token at each step. W_Y and W_S are learned linear projection matrices and "$\circ$" denotes concatenation. c_t is the *context vector* – a weighted sum of the encoder hidden states $c_t = \sum_{i=1}^{n} \alpha_{ti} h_i$, where α_{ti} corresponds to an *alignment model*, represented by a feed-forward network with a single $\tanh$ hidden layer.

On top of this basic seq2seq model, we implemented a simple beam search for decoding (Sutskever et al., 2014; Bahdanau et al., 2015). It proceeds left-to-right and keeps track of log probabilities of top n possible output sequences, expanding them one token at a time.

3.3 Reranker

To ensure that the output trees/strings correspond semantically to the input DA, we implemented a classifier to rerank the n-best beam search outputs and penalize those missing required information and/or adding irrelevant one. Similarly to Wen et al. (2015a), the classifier provides a binary decision for an output tree/string on the presence of all dialogue act types and slot-value combinations seen in the training data, producing a 1-hot vector.

[1] We use the implementation in the TensorFlow framework (Abadi et al., 2015).

(<root> <root> ((X-name n:subj) be v:fin ((Italian adj:attr) restaurant n:obj (river n:near+X))))
X-name n:subj be v:fin Italian adj:attr restaurant n:obj river n:near+X

Figure 2: Trees encoded as sequences for the seq2seq generator (top) and the reranker (bottom)

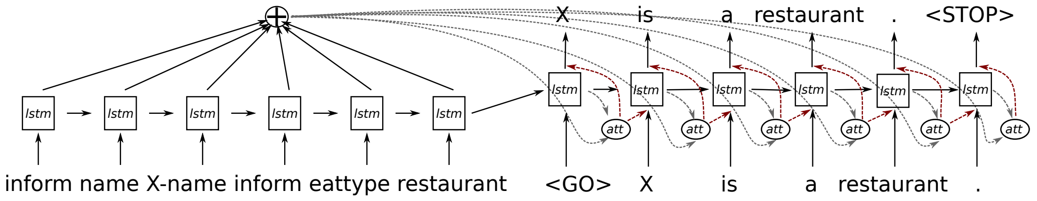

Figure 3: Seq2seq generator with attention

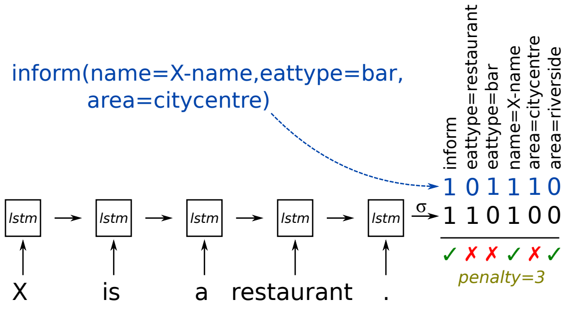

Figure 4: The reranker

The input DA is converted to a similar 1-hot vector and the reranking penalty of the sentence is the Hamming distance between the two vectors (see Fig. 4). Weighted penalties for all sentences are subtracted from their n-best list log probabilities.

We employ a similar architecture for the classifier as in our seq2seq generator encoder (see Section 3.2), with an RNN encoder operating on the output trees/strings and a single logistic layer for classification over the last encoder hidden state. Given an output sequence representing a string or a tree $\mathbf{y} = \{y_1, \dots, y_n\}$ (cf. Section 3.1), the encoder again produces a sequence of hidden states $\mathbf{h} = \{h_1, \dots, h_n\}$ where $h_t = \text{lstm}(y_t, h_{t-1})$. The output binary vector o is computed as:

$$o_i = \text{sigmoid}((h_n \cdot W_R + b)_i)$$

Here, W_R is a learned projection matrix and b is a corresponding bias term.

4 Experiments

We perform our experiments on the BAGEL data set of Mairesse et al. (2010), which contains 202 DA from the restaurant information domain with two natural language paraphrases each, describing restaurant locations, price ranges, food types etc. Some properties such as restaurant names or phone numbers are delexicalized (replaced with "X" symbols) to avoid data sparsity.[2]Unlike Mairesse et al. (2010), we do not use manually annotated alignment of slots and values in the input DA to target words and phrases and let the generator learn it from data, which simplifies training data preparation but makes our task harder. We lowercase the data and treat plural *-s* as separate tokens for generating into strings, and we apply automatic analysis from the Treex NLP toolkit (Popel and Žabokrtský, 2010) to obtain deep syntax trees for training tree-based generator setups.[3] Same as Mairesse et al. (2010), we apply 10-fold cross-validation, with 181 training DA and 21 testing DA. In addition, we reserve 10 DA from the training set for validation.[4]

To train our seq2seq generator, we use the Adam optimizer (Kingma and Ba, 2015) to minimize unweighted sequence cross-entropy.[5] We perform 10 runs with different random initialization of the network and up to 1,000 passes over the training data,[6] validating after each pass and selecting the parameters that yield the highest BLEU score on the validation set. Neither beam search nor the reranker are used for validation.

We use the Adam optimizer minimizing cross-entropy to train the reranker as well.[7] We perform a single run of up to 100 passes over the data, and we also validate after each pass and select the parameters giving minimal Hamming distance on both validation and training set.[8]

[2] We adopt the delexicalization scenario used by Mairesse et al. (2010) and Dušek and Jurčíček (2015).

[3] The input vocabulary size is around 45 (DA types, slots, and values added up) and output vocabulary sizes are around 170 for string generation and 180 for tree generation (45 formemes and 135 lemmas).

[4] We treat the two paraphrases for the same DA as separate instances in the training set but use them together as two references to measure BLEU and NIST scores (Papineni et al., 2002; Doddington, 2002) on the validation and test sets.

[5] Based on a few preliminary experiments, the learning rate is set to 0.001, embedding size 50, LSTM cell size 128, and batch size 20. Reranking penalty for decoding is 100.

[6] Training is terminated early if the top 10 so far achieved validation BLEU scores do not change for 100 passes.

[7] We use the same settings as with the seq2seq generator.

[8] The validation set is given 10 times more importance.

Setup	BLEU	NIST	ERR
Mairesse et al. (2010)*	∼67	-	0
Dušek and Jurčíček (2015)	59.89	5.231	30
Greedy with trees	55.29	5.144	20
+ Beam search (b. size 100)	58.59	5.293	28
+ Reranker (beam size 5)	60.77	5.487	24
(beam size 10)	60.93	5.510	25
(beam size 100)	60.44	5.514	19
Greedy into strings	52.54	5.052	37
+ Beam search (b. size 100)	55.84	5.228	32
+ Reranker (beam size 5)	61.18	5.507	27
(beam size 10)	62.40	5.614	21
(beam size 100)	62.76	5.669	19

Table 1: Results on the BAGEL data set

NIST, BLEU, and semantic errors in a sample of the output.

*Mairesse et al. (2010) use manual alignments in their work, so their result is not directly comparable to ours. The zero semantic error is implied by the manual alignments and the architecture of their system.

5 Results

The results of our experiments and a comparison to previous works on this dataset are shown in Table 1. We include BLEU and NIST scores and the number of semantic errors (incorrect, missing, and repeated information), which we assessed manually on a sample of 42 output sentences (outputs of two randomly selected cross-validation runs).

The outputs of direct string generation show that the models learn to produce fluent sentences in the domain style;[9] incoherent sentences are rare, but semantic errors are very frequent in the greedy search. Most errors involve confusion of semantically close items, e.g., *Italian* instead of *French* or *riverside area* instead of *city centre* (see Table 2); items occurring more frequently are preferred regardless of their relevance. The beam search brings a BLEU improvement but keeps most semantic errors in place. The reranker is able to reduce the number of semantic errors while increasing automatic scores considerably. Using a larger beam increases the effect of the reranker as expected, resulting in slightly improved outputs.

Models generating deep syntax trees are also able to learn the domain style, and they have virtually no problems producing valid trees.[10] The surface realizer works almost flawlessly on this limited domain (Dušek and Jurčíček, 2015), leaving the seq2seq generator as the major error source. The syntax-generating models tend to make different kinds of errors than the string-based models: Some outputs are valid trees but not entirely syntactically fluent; missing, incorrect, or repeated information is more frequent than a confusion of semantically similar items (see Table 2). Semantic error rates of greedy and beam-search decoding are lower than for string-based models, partly because confusion of two similar items counts as two errors. The beam search brings an increase in BLEU but also in the number of semantic errors. The reranker is able to reduce the number of errors and improve automatic scores slightly. A larger beam leads to a small BLEU decrease even though the sentences contain less errors; here, NIST reflects the situation more accurately.

A comparison of the two approaches goes in favor of the joint setup: Without the reranker, models generating trees produce less semantic errors and gain higher BLEU/NIST scores. However, with the reranker, the string-based model is able to reduce the number of semantic errors while producing outputs significantly better in terms of BLEU/NIST.[11] In addition, the joint setup does not need an external surface realizer. The best results of both setups surpass the best results on this dataset using training data without manual alignments (Dušek and Jurčíček, 2015) in both automatic metrics[12] and the number of semantic errors.

6 Related Work

While most recent NLG systems attempt to learn generation from data, the choice of a particular approach – pipeline or joint – is often arbitrary and depends on system architecture or particular generation domain. Works using the pipeline approach in SDS tend to focus on sentence planning, improving a handcrafted generator (Walker et al., 2001; Stent et al., 2004; Paiva and Evans, 2005) or using perceptron-guided A* search (Dušek and Jurčíček, 2015). Generators taking the joint approach employ various methods, e.g., factored language models (Mairesse et al., 2010), inverted parsing (Wong and Mooney, 2007; Konstas and Lapata, 2013), or a pipeline of discriminative classifiers (Angeli et al., 2010). Unlike most previous

[9] The average sentence length is around 13 tokens.

[10] The generated sequences are longer, but have a very rigid structure, i.e., less uncertainty per generation step. The average output length is around 36 tokens in the generated sequence or 9 tree nodes; surface realizer outputs have a similar length as the sentences produced in direct string generation.

[11] The difference is statistically significant at 99% level according to pairwise bootstrap resampling test (Koehn, 2004).

[12] The BLEU/NIST differences are statistically significant.

Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, area=citycentre, near=X-near, food=”Chinese takeaway”, food=Japanese)
Reference	X is a Chinese takeaway and Japanese restaurant in the city centre near X.
Greedy with trees	X is a restaurant offering chinese takeaway in the centre of town near X. [Japanese]
+ Beam search	X is a restaurant and japanese food and chinese takeaway.
+ Reranker	X is a restaurant serving japanese food in the centre of the city that offers chinese takeaway.
Greedy into strings	X is a restaurant offering italian and indian takeaway in the city centre area near X. [Japanese, Chinese]
+ Beam search	X is a restaurant that serves fusion chinese takeaway in the riverside area near X. [Japanese, citycentre]
+ Reranker	X is a japanese restaurant in the city centre near X providing chinese food. [takeaway]
Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, area=riverside, food=French)
Reference	X is a French restaurant on the riverside.
Greedy with trees	X is a restaurant providing french and continental and by the river.
+ Beam search	X is a restaurant that serves french takeaway. [riverside]
+ Reranker	X is a french restaurant in the riverside area.
Greedy into strings	X is a restaurant in the riverside that serves italian food. [French]
+ Beam search	X is a restaurant in the riverside that serves italian food. [French]
+ Reranker	X is a restaurant in the riverside area that serves french food.
Input DA	inform(name=X-name, type=placetoeat, eattype=restaurant, near=X-near, food=Continental, food=French)
Reference	X is a French and Continental restaurant near X.
Greedy with trees	X is a french restaurant that serves french food and near X. [Continental]
+ Beam search	X is a french restaurant that serves french food and near X. [Continental]
+ Reranker	X is a restaurant serving french and continental food near X.
Greedy into strings	X is a french and continental style restaurant near X.
+ Beam search	X is a french and continental style restaurant near X.
+ Reranker	X is a restaurant providing french and continental food, near X.

Table 2: Example outputs of different generator setups (beam size 100 is used). Errors are marked in color (missing, superfluous, repeated information, disfluency).

NLG systems, our generator is trainable from unaligned pairs of MR and sentences alone.

Recent RNN-based generators are most similar to our work. Wen et al. (2015a) combined two RNN with a convolutional network reranker; Wen et al. (2015b) later replaced basic sigmoid cells with an LSTM. Mei et al. (2015) present the only seq2seq-based NLG system known to us. We extend the previous works by generating deep syntax trees as well as strings and directly comparing pipeline and joint generation. In addition, we experiment with an order-of-magnitude smaller dataset than other RNN-based systems.

7 Conclusions and Future Work

We have presented a direct comparison of two-step generation via deep syntax trees with a direct generation into strings, both using the same NLG system based on the seq2seq approach. While both approaches offer decent performance, their outputs are quite different. The results show the direct approach as more favorable, with significantly higher n-gram based scores and a similar number of semantic errors in the output.

We also showed that our generator can learn to produce meaningful utterances using a much smaller amount of training data than what is typically used for RNN-based approaches. The resulting models had virtually no problems with producing fluent, coherent sentences or with generating valid structure of bracketed deep syntax trees. Our generator was able to surpass the best BLEU/NIST scores on the same dataset previously achieved by a perceptron-based generator of Dušek and Jurčíček (2015) while reducing the amount of irrelevant information on the output.

Our generator is released on GitHub at the following URL:

`https://github.com/UFAL-DSG/tgen`

We intend to apply it to other datasets for a broader comparison, and we plan further improvements, such as enhancing the reranker or including a bidirectional encoder (Bahdanau et al., 2015; Mei et al., 2015; Jean et al., 2015) and sequence level training (Ranzato et al., 2015).

Acknowledgments

This work was funded by the Ministry of Education, Youth and Sports of the Czech Republic under the grant agreement LK11221 and core research funding, SVV project 260 333, and GAUK grant 2058214 of Charles University in Prague. It used language resources stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071). We thank our colleagues and the anonymous reviewers for helpful comments.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. arXiv:1406.1078.

N. Dethlefs, H. Hastie, H. Cuayáhuitl, and O. Lemon. 2013. Conditional Random Fields for Responsive Surface Realisation using Global Features. In *Proceedings of ACL*, Sofia.

G. Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

O. Dušek and F. Jurčíček. 2015. Training a Natural Language Generator From Unaligned Data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 451–461, Beijing, China. Association for Computational Linguistics.

Ondřej Dušek, Zdeněk Žabokrtský, Martin Popel, Martin Majliš, Michal Novák, and David Mareček. 2012. Formemes in English-Czech Deep Syntactic MT. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 267–274, Montréal, Canada. Association for Computational Linguistics.

O. Dušek, L. Gomes, M. Novák, M. Popel, and R. Rosa. 2015. New Language Pairs in TectoMT. In *Proceedings of the 10th Workshop on Machine Translation*, pages 98–104, Lisbon, Portugal. Association for Computational Linguistics.

A. Graves. 2013. Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*, August.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

D. Kingma and J. Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*. arXiv:1412.6980.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

I. Konstas and M. Lapata. 2013. A Global Model for Concept-to-Text Generation. *Journal of Artificial Intelligence Research*, 48:305–346.

F. Mairesse, M. Gašić, F. Jurčíček, S. Keizer, B. Thomson, K. Yu, and S. Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561.

H. Mei, M. Bansal, and M. R. Walter. 2015. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. *arXiv:1509.00838 [cs]*, September.

D. S. Paiva and R. Evans. 2005. Empirically-based control of natural language generation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 58–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

M. Popel and Z. Žabokrtský. 2010. TectoMT: modular NLP framework. In *Proceedings of IceTAL, 7th International Conference on Natural Language Processing,*, pages 293–304, Reykjavík.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks. *arXiv:1511.06732 [cs]*, November.

A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*. Citeseer.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, studies in natural language processing edition.

V. Rieser, O. Lemon, and X. Liu. 2010. Optimising information presentation for spoken dialogue systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1009–1018.

A. Stent, R. Prasad, and M. Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 79–86.

I. Sutskever, O. Vinyals, and Q. VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112. arXiv:1409.3215.

O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015. Grammar as a Foreign Language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2755–2763.

M. A. Walker, O. Rambow, and M. Rogati. 2001. SPoT: a trainable sentence planner. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

T.-H. Wen, M. Gasic, D. Kim, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 275–284, Prague, Czech Republic, September. Association for Computational Linguistics.

T.-H. Wen, M. Gasic, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young. 2015b. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Y. W. Wong and R. J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, pages 172–179.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. 2010. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, April.