Comparing Bayesian and LSTM Networks in Natural Language Generation

Panagiotis Karagiannis *
Department of Computer Science
University of California Santa Cruz
Santa Cruz, CA 95064
pkaragia@ucsc.edu

Juraj Juraska †
Department of Computer Science
University of California Santa Cruz
Santa Cruz, CA 95064
jjuraska@ucsc.edu

Abstract

In this paper we focus on the comparison of two very different methods to perform natural language generation. We use a graphical and a deep learning approach in order to produce utterances given meaning representations in the restaurant domain. The first part of this paper presents a fully data-driven generation method that performs the language generation task as a search over the most likely sequence of surface realization phrases according to Factored Language Models. The second part of this paper focuses on a statistical generator based on Long Short-Term Memory neural networks (LSTMs). This model learns from unaligned data and jointly performs the sentence planning and the surface realization. We report our results after experimenting on the BAGEL dataset produced by 42 untrained Amazon Mechanical Turkers.

1 Introduction

With the continuous efforts to develop general artificial intelligence, there has recently been a substantial amount of research done in the fields of natural language processing (NLP) and generation (NLG). It gave rise to popular digital personal assistants in smartphones, such as Siri or Cortana, as well as separate devices with the sole purpose of offering the services of such personal assistants through interactive communication, such as Amazon Echo (powered by the Alexa assistant) or Google Home. The capabilities of these conversational agents are still fairly limited and lacking in various aspects, ranging from distinguishing the sentiment of human utterances to producing utterances with human-like coherence and naturalness.

In our work, we focus on the NLG module in task-oriented dialogue systems (see Fig. 1), in particular on generation of textual utterances from structured *meaning representations* (MRs). An MR describes a single dialogue act in the form of a list of pieces of information that need to be conveyed to the other party in the dialogue (typically a human user). Each piece of information is represented by a slot-value pair, where the *slot* identifies the type of information and the *value* is the corresponding content (see Table 1). *Dialogue acts* (DAs) can be of different types, depending on the intent of the dialogue manager component. They can range from simple ones, such as a *goodbye* DA with no slots at all, to complex ones, such as an *inform* DA containing multiple slots with various types of values.

The objective of the language generation phase of a spoken dialogue system is to produce a syntactically correct utterance from a given MR, which is the outcome of the previous phase (dialogue management). Aside from being correct, the utterance should express all the information

*https://www.soe.ucsc.edu/people/pkaragia
†https://www.soe.ucsc.edu/people/jjuraska

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Table 1: An example of a meaning representation (MR) and a corresponding utterance.

MR	*inform*(name[**The Golden Curry**], food[**Japanese**], priceRange[**moderate**], kidsFriendly[**yes**], near[**The Bakers**])
Utterance	Located near **The Bakers**, **kid-friendly** restaurant, **The Golden Curry**, offers **Japanese** cuisine with a **moderate** price range.

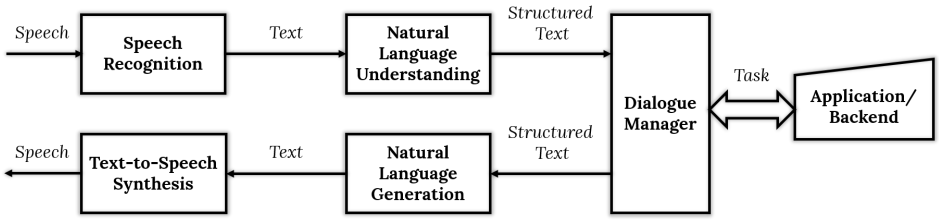

Figure 1: Architecture of a spoken dialogue system.

contained in the MR, and the formulation should be as natural as possible. The process of assembling an utterance from an MR is often performed in two stages: *sentence planning*, which determines the structure of the utterance (order in which the information is presented, number of sentences, etc.), and *surface realization*, in which the actual words are plugged in and the utterance is given a linear form.

In our work, we compare the performance of a graphical model to a neural model in the context of NLG using a small, but semantically aligned (structured), training dataset. We use the BAGEL dataset (**?**) to evaluate both types of models. We implement the former based on **?**, and we use an existing solution for the latter (**?**).

2 Related Work

Researchers have taken various statistical approaches to solving the language generation problem over the years. One of the first notable efforts was done by **?** who augmented a handcrafted generator with statistical reranking. The generator produces candidate utterances based on hard-coded grammar rules, and then an n-gram language model, trained on a corpus of data, reranks the candidates and selects the best among them. **?** took the idea one step further by incorporating a trainable sentence planner, which was later improved by **?** by making it better adaptable to different domains. Another line of research is focused on learning the parameters of the model through optimizing an objective function. An example of such a method is using reinforcement learning to train a language generation policy, such that the expected reward is maximized (**?**). Nevertheless, all of the above heavily rely on handcrafting parts of the model, which results in a decreased flexibility and portability to different domains, as well as limited variability of the utterances.

In order to avoid the inconvenience of developing grammars or rules, and maintaining them, new approaches emerged attempting to train the language model as a whole on a corpus of data. The earlier works among them used semantically aligned data to train the language models (**??**). The alignment provides valuable information during training, but requires significant additional effort in properly annotating the entire dataset. Therefore, it is feasible to be performed on small datasets only, and preferably by annotators with appropriate linguistic knowledge. On the other hand, the extra information enables the model to learn faster from a smaller dataset, which may be an important factor in certain domains where data is hard to obtain.

More recently, methods were developed that can do without any annotations in the data and often use an *end-to-end* approach to training, performing the sentence planning and the surface realization simultaneously (**?**). Later systems, trained on unaligned data successfully utilized recurrent neural networks (RNNs) paired with an encoder-decoder architecture design (**???**), or other concepts, such as imitation learning (**?**). These generation models, however, typically require greater amount of data to be able to learn, due to the lack of semantic alignment.

In our work, we take a closer look at the language generator utilizing dynamic Bayesian networks (DBNs) (**?**), trained on aligned data, and compare it with an end-to-end RNN-based language model trained on unaligned data, previously shown to achieve competitive results (**?**). The purpose of the comparison is to assess the importance of data alignment in training data-driven language models.

3 Graphical Model Approach

In this section, we describe the DBN language model that learns to determine the most probable utterance given an MR from semantically aligned data. We use a *stack-based* semantic representation (see Fig. 2) to constrain the sequence of semantic concepts to be searched. As in the BAGEL model (**?**), we distinguish stacks as *mandatory* and *intermediate* (or *filler*). Mandatory stacks are derived from the input MR, whereas intermediate stacks contain additional information to help increase the naturalness of the utterance. Therefore, given a set of mandatory semantic stacks, the model's task is to find the most likely sequence of realization phrases, which ultimately forms the output utterance.

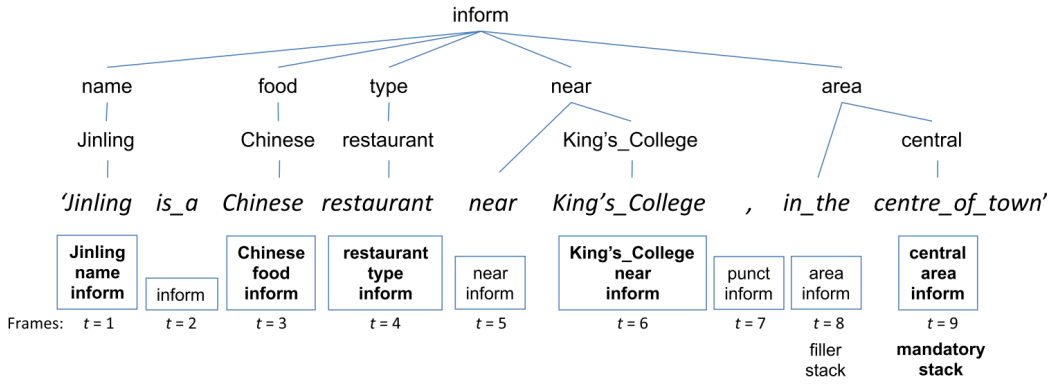

Figure 2: An example of the stack-based semantic representation of an utterance.

Let the *ordered* set of mandatory and intermediate stacks be S_m and S_i, respectively. In order to constrain our search space, we make the simplifying assumption that an intermediate stack can exist between two consecutive mandatory stacks if it satisfies one of two conditions: the top concept (such as `name` or `area`) of the intermediate stack has to match the top concept of either of the adjacent mandatory stacks, or, if the intermediate stack contains no top concept, then its bottom concept (such as `inform`) has to match either of the bottom concepts of the mandatory stacks. Following the notation introduced by **?**, let $Seq(S_m)$ be the set of sequences of stacks that can be obtained by inserting intermediate stacks $s \in S_i$ in S_m (ordered arbitrarily) and satisfy the aforementioned conditions. Also, let the sequence of realization phrases be $\mathbf{R} = (r_1, \ldots, r_n)$. Then we need to find:

$$\mathbf{R}^* = \arg\max_{\mathbf{R}} P(\mathbf{R}|S_m) \tag{1}$$

$$\arg\max_{\mathbf{R}} \sum_{\mathbf{S} \in Seq(S_m)} P(\mathbf{R}, \mathbf{S}|S_m) \tag{2}$$

$$\arg\max_{\mathbf{R}} \sum_{\mathbf{S} \in Seq(S_m)} P(\mathbf{R}|\mathbf{S}, S_m) P(\mathbf{S}|S_m) \tag{3}$$

$$\arg\max_{\mathbf{R}} \sum_{\mathbf{S} \in Seq(S_m)} P(\mathbf{R}|\mathbf{S}) P(\mathbf{S}|S_m) \tag{4}$$

Finding this sum, the complexity of our model would explode, since $Seq(S_m)$ grows exponentially with respect to the size of S_m. Nevertheless, we can make the assumption that the above sum is dominated by the most likely stack $\mathbf{S}^*$ given S_m. This assumption is experimentally justified

according to **?**. Therefore, our previous formula is reduced to:

$$\mathbf{R}^* = \arg\max_{\mathbf{R}} P(\mathbf{R}|\mathbf{S}^*)P(\mathbf{S}^*|S_m) \tag{5}$$

such that

$$\mathbf{S}^* = \arg\max_{\mathbf{S}\in Seq(S_m)} P(\mathbf{S}|S_m) \tag{6}$$

In order to solve equations 5 and 6, we follow a similar factoring approach to that proposed by **?**. We assume that the current stack only depends on the two previous stacks already inferred (see Fig. 3), whereas the current realization phrase depends only on the previous one, as well as on the previous, current and next stacks (see Fig. 4). Mathematically, it amounts to the following equations:

$$P(\mathbf{S}|S_m) = \begin{cases} \prod_{t=1}^{T} P(s_t|s_{t-1}, s_{t-2}) & \text{if } \mathbf{S} \in Seq(S_m) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$P(\mathbf{R}|\mathbf{S}^*) = \prod_{t=1}^{T} P(r_t|r_{t-1}, s_t^*, s_{t-1}^*, s_{t+1}^*) \tag{8}$$

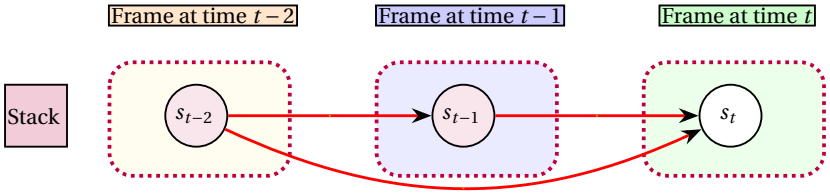

Figure 3: Dynamic Bayesian network used for stack sequence inference. Shading indicates the node is observed in the context of inferring the conditional probability of s_t.

Naturally, we first find $\mathbf{S}^*$ using equation 7 and then treat $\mathbf{S}^*$ as observed in order to find $\mathbf{R}^*$ using equation 8.

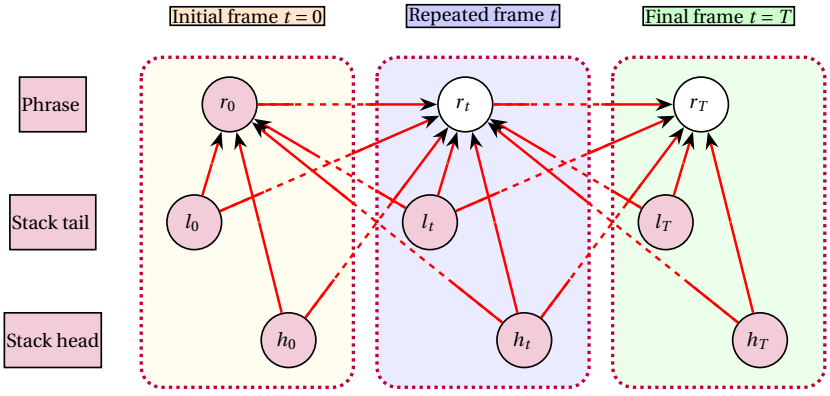

Figure 4: Dynamic Bayesian network for the realization phrase inference. Stack head and tail are utilized in the back-off method to help the model generalize. Shading indicates the node is observed in the context of inferring the conditional probability of r_t.

3.1 Generalizing

The above model does not easily generalize to unseen semantic stacks, since if a specific stack-phrase configuration is not observed during training, then equations 7 and 8 give probabilities of

Table 2: BLEU score for models using various phrase back-off levels, when the training set is 70% of the BAGEL dataset.

Back-off level	0	1	2	3	4
BLEU score	0.44	0.47	0.63	0.63	0.63

0. In order to account for this fact we introduce *back-off* schemes which get activated only if any of the terms in equations 7 and 8 becomes equal to 0. In order to implement the back-off schemes we follow the approach suggested in **?** and make each realization phrase dependent on *understack* configurations, namely the *tail* and the *head* of the stack. More precisely, given any semantic stack s represented as in Fig. 2, we define its *head* h to be the top entry of the stack and its *tail* to be the stack l obtained by removing the head from s. So for example, in time step $t = 4$ of Fig. 2, the head h_4 of the stack is `restaurant` and the tail l_4 of the stack is `inform(type)`.

The back-off methods are summarized by the diagrams in Fig. 5. Intuitively, they work as a "safety net" that prevents the model from getting a probability of 0 for a given stack-phrase configuration, which would result in an empty output utterance. Therefore, following these alternative "paths" while performing inference, we manage to limit the negative effect of a small training dataset and generalize more accurately to unseen contexts. A non-zero probability is guaranteed to be encountered using the deepest level of back-off. Backing off can naturally lead to less informative or slightly incorrect utterances, but, in general, such utterances are preferred to no output at all.

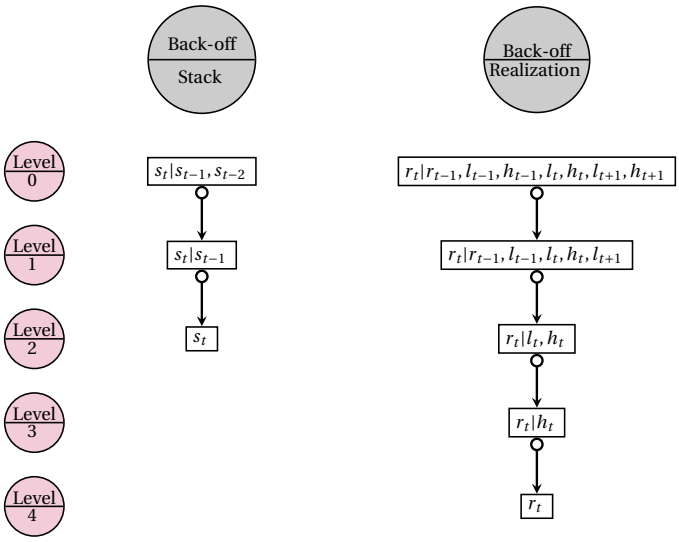

Figure 5: Back-off schemes for stack and phrase inference.

As our early experiments (using fixed ordering of the mandatory stacks) showed, following the back-off schemes suggested in Fig. 5, the full back-off model (i.e. level 4) manages to achieve a significantly higher score than models utilizing no or a shallow back-off level (see Table 2). Nevertheless, using this scheme also results in the back-off levels 2, 3 and 4 yielding identical results and a fairly large gap between level 1 and level 2.

Therefore, in order to create a "smoother" and more comprehensive back-off scheme, where each level performs gradually better and captures more complex understack relationships, we develop an extended phrase back-off scheme inspired by **?**. The alternative back-off paths are summarized in Fig. 6, where the nodes are explored in a depth-first fashion until a non-zero probability is discovered for a given stack-phrase configuration.

Finally, we should mention that a consequence of this model is that shorter sentences are likely to be favored. Nevertheless, according to **?** the learned probabilities are skewed enough to allow the existence of intermediate stacks in $\mathbf{S}^*$.

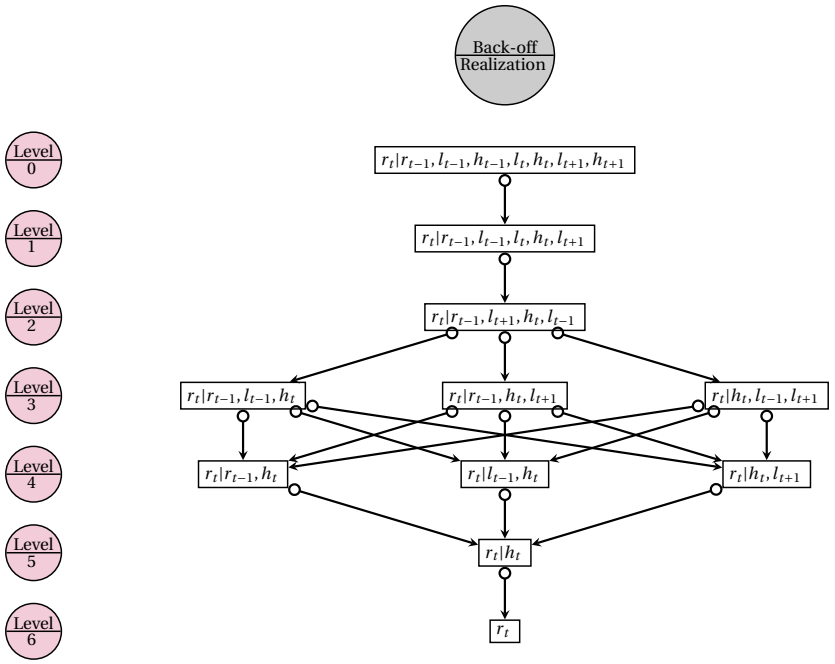

Figure 6: Extended back-off scheme for phrase inference.

4 Deep Learning Approach

The recurrent neural network (RNN) is a natural extension of the feed-forward neural network to be used with sequential input. In an RNN, the output at each time step directly affects the output at the next time step and indirectly at all future time steps. Therefore, given a sequence of inputs $(w_1, \ldots, w_n)$, the RNN computes a corresponding sequence $(u_1, \ldots, u_n)$ from the following pair of equations:

$$\mathbf{h_t} = \sigma(\mathbf{W^h h_{t-1}} + \mathbf{W^w w_t})$$
$$\mathbf{u_t} = \mathbf{W^u h_t}$$

where the $\mathbf{W}$ matrices represent parameters of the model to be learned.

Nevertheless, as we see from the formulation of RNNs, it is unclear how to map a sequence onto a sequence of a different size when the alignment of inputs and outputs is not known in advance. A common approach is discussed in **?**, where an RNN is used to map the input sequence to a vector of fixed dimensionality, and in turn, that vector is mapped to the output sequence using another RNN. However, this approach suffers from the well-known problem of the vanishing gradient in RNNs when trying to learn long dependencies (**?**). In order to tackle these problems, we focus on Long Short-Term Memory (LSTM) neural networks, which are better at learning dependencies over many time steps (**?**).

As the second model for our project we use a state-of-the-art method for NLG. We model this problem using LSTM networks in order to generate utterances that contain the natural variations of the human language. Following the approach suggested by **?**, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. The model employs two LSTM networks (see Fig. 7): the *encoder* LSTM transforms the input sequence to a single fixed-length *context vector*, which forms an intermediate representation of the input. The *decoder* LSTM then infers the target sequence from the intermediate vector by peeking at it at every time step (using it thus as additional input information). In general, there exist several different flavors of LSTM networks, nevertheless, in this paper we explore a neural network suggested by **?**, which is governed by the following equations:

$$\mathbf{i_t} = \sigma(\mathbf{W_i w_t} + \mathbf{U_i h_{t-1}}) \tag{9}$$

$$\mathbf{f_t} = \sigma(\mathbf{W_f w_t} + \mathbf{U_f h_{t-1}}) \tag{10}$$

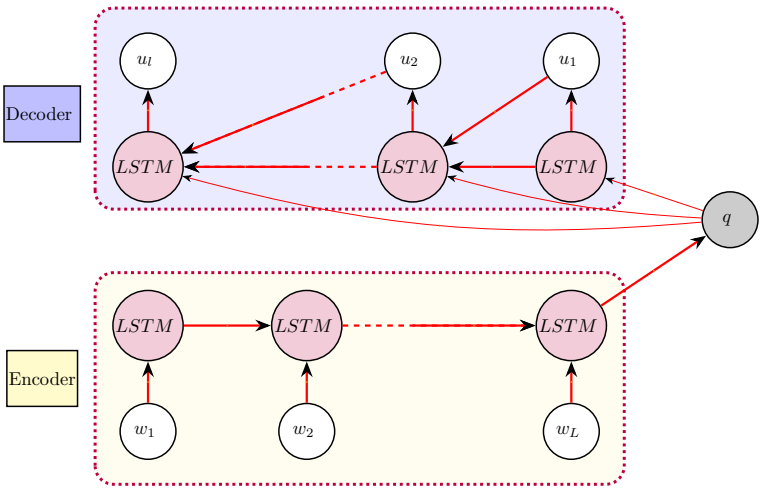

Figure 7: Standard encoder-decoder architecture. Input sequence $(w_1 \dots w_L)$ is mapped to a context vector q. Using q we produce the output sequence $(u_1 \dots u_l)$.

$$\mathbf{o_t} = \sigma(\mathbf{W_o w_t} + \mathbf{U_o h_{t-1}}) \tag{11}$$

$$\widetilde{\mathbf{c}}_\mathbf{t} = tanh(\mathbf{W_c w_t} + \mathbf{U_c h_{t-1}}) \tag{12}$$

$$\mathbf{r_t} = \sigma(\mathbf{W_r w_t} + \mathbf{U_r h_{t-1}}) \tag{13}$$

$$\mathbf{d_t} = \mathbf{r_t} \odot \mathbf{d_{t-1}} \tag{14}$$

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \widetilde{\mathbf{c}}_\mathbf{t} + tanh(\mathbf{W_d d_t}) \tag{15}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot tanh(\mathbf{c_t}) \tag{16}$$

where $\mathbf{W}$ and $\mathbf{U}$ are the parameters to be learned, $\mathbf{d_t}$ is a one-hot representation of the DA, and $\mathbf{i_t}, \mathbf{f_t}, \mathbf{o_t}, \mathbf{r_t} \in [0,1]^n$ constitute the input, forget, output and read gates, respectively. Notice, that the equations that distinguish this model from standard LSTM flavors are (17)-(15). More specifically, the newly introduced control cell (17-14) affects the generation process by manipulating the DA features, in order to make the produced utterance represent the intended meaning.

The aforementioned structure consists only of a single LSTM cell, nevertheless, the proposed architecture can be easily extended to a deep neural network by stacking multiple LSTM cells on top of each other. As explained in **?**, in order to allow hidden layers to affect the read gate, we simply have to update equation 17 to:

$$\mathbf{r_t} = \sigma(\mathbf{W_r w_t} + \sum_l \alpha_l \mathbf{U_r^l h_{t-1}^l}) \tag{17}$$

where l is the level of the LSTM layer and α_l is a layer-wise constant.

The goal of the LSTM is to compute the conditional probability $P(u_1, \dots u_l | w_1 \dots w_L)$, where $(w_1 \dots w_L)$ is an MR encoded as a sequence of embedding vectors, and $(u_1, \dots, u_l)$ is a sequence of one-hot vectors representing an utterance. Notice, that in contrast to the vanilla RNN architecture presented at the beginning of this section, the length of the input and output sequence need not be the same.

For the development of this model, we modify the existing codebase provided by **?** in order for the implementation to accept MRs with a single reference utterance in the training phase.

Table 3: Example of a dialogue act along with the produced utterance by our model.

MR	inform(name="De Luca Cucina and Bar", type="placetoeat", eattype="restaurant", near="Iceland Foods Plc", area="riverside", food="Italian")
Delexicalized MR	inform(name="X1", type="placetoeat", eattype="restaurant", near="X2", area="riverside",food="Italian")
Annotated reference	[name+X]X []is a [eattype+restaurant]restaurant [near]near [near+X]X [area]on the [area+riverside]riverside []that [food]serves [food+Italian]Italian [food]cuisine.
Produced utterance	De Luca Cucina and Bar is a restaurant at the side of the river near Iceland Foods Plc that serves Italian food.

5 Data

The dataset we use in this project was produced as a part of the development of the BAGEL model (**?**). The MRs were generated by simulating 20,000 dialogues between a statistical dialogue system and a user simulator. The *delexicalization* of the MRs (replacing of values by tokens, or placeholders) yielded 202 distinct MRs. The authors then had annotators produce utterances matching the generated MRs, using the Amazon Mechanical Turk service. For each MR two different paraphrases were collected. Finally, the annotators aligned the utterances with the MRs by matching the words and phrases in an utterance with individual slot-value pairs in the corresponding MR.

The DAs are in the restaurant domain and, according to the authors, should contain two types of acts: `inform` and `reject`. Having analyzed the dataset, however, we did not observe any DAs of the `reject` type. The MRs representing the DAs contain 8 different slot types, namely: `name`, `food`, `near`, `pricerange`, `postcode`, `phone`, `address`, and `area`, where `food`, `pricerange`, and `area` take on enumerable values, while the remaining slots may take arbitrary numerical or nominal values.

Each sample in the BAGEL dataset consists of three parts: an MR, a delexicalized MR, and an annotated delexicalized reference utterance (see Table 3) . For the purposes of testing the LSTM-based model, we need to remove the semantic alignment information from the data, as the model does not take advantage of it, and expects raw utterances as references in training. That requires performing a relexicalization to substitute original values for the tokens in the utterances. There is one additional technical detail concerning the LSTM model implementation that we use: in order to be able to use the BAGEL dataset as input to it, we need to convert the dataset to the JSON format and account for the fact that not each MR in the training set has two utterances associated with it. Since we cannot manually create utterances for all of the MRs in BAGEL with only a single corresponding reference utterance, we simply duplicate the original utterance for each such MR.

5.1 Metrics

We present our results using common automatic evaluation metrics in both models. The most popular metrics include BLEU (**?**) and METEOR (**?**), originally used in machine translation research, but successfully applied in other NLP fields as well. The BLEU score indicates the amount of n-gram co-occurrence in a pair of utterances (or text documents in general). The most common variant is BLEU-4 which considers n-grams of the length of up to 4. On the other hand, to compute the METEOR metric, the two utterances are aligned first, matched by tokens. Based on the alignment, the harmonic mean of precision and recall is then calculated as the score. After experimenting with both metrics, we observe that both capture the same qualitative information when comparing the reference utterances with the model-produced utterances. Therefore, we only present BLEU scores in our evaluation.

6 Evaluation

First of all, we performed experiments to demonstrate the impact of the back-off scheme introduced in Section 3. We can observe two essential phenomena in Fig. 8. With no phrase back-off (i.e. level 0), fewer utterances tend to be produced as we increase the level of stack back-off. On

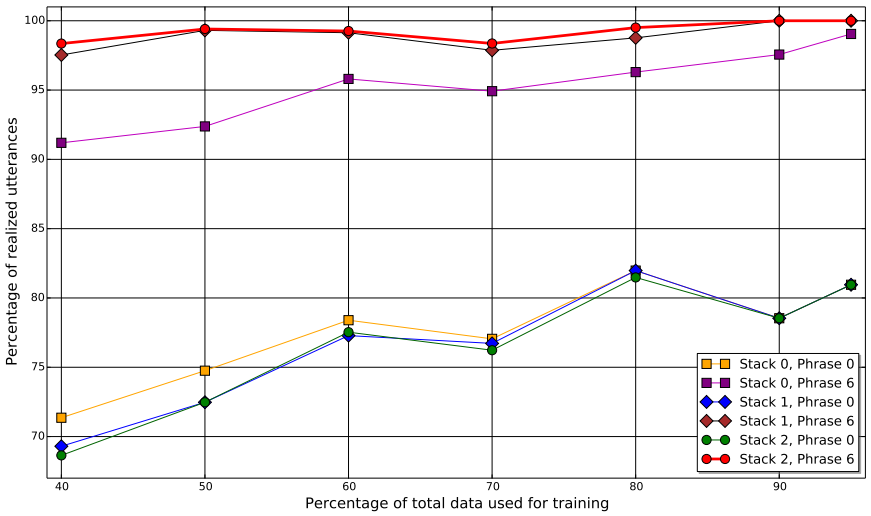

Figure 8: Success realization success rate of models with different combinations of levels of stack and phrase back-off. The experiments were performed on different training-test split, as reflected by the x-axis. The results are averaged over 10 folds using cross-validation.

Table 4: BLEU scores for the DBN model using full stack and phrase back-offs, and the LSTM model, across various splits of the dataset. The results are averaged over 5 folds using cross-validation.

Training proportion	40%	50%	60%	70%	80%	90%	95%
DBN model	0.639	0.637	0.640	0.640	0.644	0.649	0.646
LSTM model	0.304	0.271	0.245	0.249	0.231	0.136	0.190

the other hand, when the full phrase back-off (i.e. level 6) is enabled, then we see that the surface realization becomes increasingly successful with the increasing depth of stack back-off. These may seem contradictory at first glance. However, the former is the result of the fact that the predicted stack sequences using back-off are more likely to have not been present in the training set. This not only is not an issue when using the phrase back-off, but it also gives the phrase inference an opportunity to explore more candidates beyond the training set from which the model learned.

Let us also point out that using full back-off for both the stack and the phrase inference enables the model to produce an utterance for every single test MR, if the model is trained on 90% or 95% of the dataset (and tested on the remainder). If the proportion of the training data is lower, the success rate remains above 98% nonetheless, which is rather impressive considering that the lowest proportion, i.e. 40% of the dataset, corresponds to mere 161 samples.

Moving on to the main set of results of our work, Table 4 shows the BLEU scores we obtained with each of the two models while training them on various proportions of the BAGEL dataset. The first thing we observe is that the scores of the DBN model across different splits only differ insignificantly. This is to be attributed to the fact that when the DBN model manages to produce an utterance, it tends to be a very good utterance. However, as we can see in Fig. 8, the model does not always succeed in the surface realization.

On the other hand, the scores we obtained using the LSTM model have a decreasing tendency with the increasing proportion of the training data. This result is slightly surprising, and we believe it is due to the way the BLEU metric works. The LSTM model produces poor utterances regardless of the data split, only the BLEU metric assigns a lower score if the output utterance is longer and, hence, contains more of the irrelevant content.

The BLEU score of the utterances generated by the LSTM model barely reaches 0.3 in the best scenario, which is significantly lower than the score of 0.731 that the authors reported on their own

dataset (**?**). There are several possible reasons that could contribute to their model performing poorly in our experiments, however, the most likely culprit is the small size of the BAGEL dataset. Since the DBN model utilizes the aligned structure of the utterances, it needs less data in order to make accurate predictions. Hence, the sophisticated end-to-end LSTM model is outperformed by the more straightforward Bayesian approach.

7 Conclusion and Future Work

In this paper, we compare two very different models for performing NLG. We first present BAGEL, which is a language generator model that utilizes dynamic Bayesian networks in order to produce natural and informative utterances. This NLG system learns content ordering, lexical selection, aggregation and realization directly from data, and it requires no handcrafting beyond the semantic stack annotation of the training data. The second NLG model that this paper presents is based on semantically conditioned LSTM structures that can learn from unaligned data. After experimentation, we observe that the former model outperforms the latter in terms of BLEU score. We attribute this performance difference to the fact that the neural model, not capable of taking advantage of semantic alignment, requires substantially more data in order to be properly trained. As we see in our experiments, the small size of the dataset does not affect the Bayesian model, which exploits the semantic annotation of the data and, hence, requires fewer data instances to learn the desired probabilities.

Possible future work includes a more thorough investigation of the significance of the semantic alignment's impact on the model's performance. We could observe the performance of our models after introducing certain variability in the amount of alignment between the meaning representations and the corresponding utterances.

References

Simard Bengio and Frasconi. 1994. Learning long term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5(2):157-166* .

Merrienboer Cho, Bougares Gulcehre, and Bengio Schwenk. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491* .

Schmidhuber Hochreiter. 1997. Long short-term memory. *Neural Computation* .

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)* 48:305–346.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 1101–1112.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 704–710.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Stroudsburg, PA, USA, StatMT '07, pages 228–231. http://dl.acm.org/citation.cfm?id=1626355.1626389.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve 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*. Association for Computational Linguistics, pages 1552–1561.

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computational Linguistics* .

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Verena Rieser and Oliver Lemon. 2010. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Empirical methods in natural language generation*, Springer, pages 105–120.

Amanda Stent, Rashmi Prasad, and Marilyn 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*. Association for Computational Linguistics, page 79.

Marilyn A Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research* 30:413–456.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* .