Natural Language Generation through Character-Based RNNs with Finite-State Prior Knowledge

Raghav Goyal*
Twenty Billion Neurons
Berlin, Germany
raghav.goyal@twentybn.com

Marc Dymetman
Xerox Research Centre Europe
Grenoble, France
marc.dymetman@xerox.com

Eric Gaussier
LIG, Uni. Grenoble Alpes
Grenoble, France
eric.gaussier@imag.fr

Abstract

Recently Wen et al. (2015) have proposed a Recurrent Neural Network (RNN) approach to the generation of utterances from dialog acts, and shown that although their model requires less effort to develop than a rule-based system, it is able to improve certain aspects of the utterances, in particular their naturalness. However their system employs generation at the word-level, which requires one to pre-process the data by substituting named entities with placeholders. This pre-processing prevents the model from handling some contextual effects and from managing multiple occurrences of the same attribute.

Our approach uses a character-level model, which unlike the word-level model makes it possible to learn to "copy" information from the dialog act to the target without having to pre-process the input. In order to avoid generating non-words and inventing information not present in the input, we propose a method for incorporating prior knowledge into the RNN in the form of a weighted finite-state automaton over character sequences. Automatic and human evaluations show improved performance over baselines on several evaluation criteria.

1 Introduction

Rule-based Natural Language Generation systems (Reiter and Dale, 2000) have been quite successful but they suffer from some limitations. They require extensive human effort and tend to produce fixed, repetitive outputs, which do not closely match human-like utterances. For this reason, there has been much interest recently in developing NLG systems which are, at least partially, able to learn from human produced data (Langkilde and Knight, 1998; Belz, 2008).

In the last couple of years, Neural Network (NN) based approaches have gained enormous popularity within statistical NLP generally, with applications to Machine Translation (Sutskever et al., 2014), Conversation Modelling (Vinyals and Le, 2015) and Parsing (Tai et al., 2015), to cite only a few. In particular, architectures based on Recurrent Neural Network (RNN) such as LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014) have been succesfully used in Language Modelling tasks due to their ability to model sequential information with long-range dependencies.

An RNN-based approach to NLG has been recently proposed by Wen et al. (2015) in the context of a dialog system, where the input semantic representation is a Dialog Act (DA). The decoder uses words as the units of generation. This word-based model requires pre-processing the original data by substituting named entities with placeholders, a process called *de-lexicalisation*. This is necessary because a standard word-level RNN is not able to "copy" input entities into the target, but has to learn each correspondence individually, which it can only do with a lot of data.

Such a de-lexicalization approach has the advantage of reducing data sparsity, but it also suffers from various shortcomings: (i) it requires some reliable mechanism for named-entity recognition, (ii) it requires a post-processing "re-lexicalization" step, where the placeholders are replaced by the original named entities, (iii) it is unable to account for subtle morphological or lexical effects that a specific

* Work performed during Raghav Goyal's internship at XRCE in 2016.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1083–1092, Osaka, Japan, December 11-17 2016.

named entity may have on its context,[1] and (iv) it does not address the problem of multi-slots of the same type, as for instance when two restaurants are mentioned in the same input.

In this work, we propose to use a **character-level** model which does not suffer from the same sparsity issues as a word-level model, and therefore does not require de-lexicalisation. We show that this architecture, coupled with a bidirectional encoding and an attention mechanism (Bahdanau et al., 2014), is able to "copy" information from the dialog act into the target realization, and to produce reasonable results.

However, this model has two main defects. The first is that it can produce non valid words, the second that it can invent named entities not present in the dialogue act. In order to improve this, we propose to constrain the generation of characters through a certain weighted finite-state automaton that incorporates **prior knowledge**: (i) about well-formed strings of characters and (ii) about the fact that named entities in the realization originate from character strings present in the input.

The following points summarize the key contributions of this paper:

- We handle the NLG problem through an attention-based character-to-character model, in particular we encode the input semantics as a string of characters. By enabling copying at the character level, this prevents us from having to de-lexicalize some aspects of the input, as Wen et al. (2015) are obliged to do.

- In order to improve the quality of the generated utterances (avoiding the generation of non-words or the hallucination of named entities), we exploit *a priori* knowledge in the form of a weighted finite-state automaton that constrains the generated strings of characters to either conform to a predefined vocabulary of words, or to originate in portions of the semantic input. This automaton is integrated within the RNN by employing a generic "background-adaptor" mechanism, a technique recently proposed in (Dymetman and Xiao, 2016), which we explain briefly in a self-contained way.

We start in section 2 by defining the network architecture we apply to all our models and by explaining the background-adaptor technique. We proceed to give details of our different models in Section 3 and introduce a finite-state background over characters. In Section 4 we describe the experiments, with details about the dataset, implementation and evaluation; we also give examples illustrating differences between the models. In Section 5 we discuss related work and finally conclude in Section 6 with some perspectives.

2 Proposed Approach

2.1 The architecture

2.1.1 Recurrent Neural Network (RNN)

Our *encoder-decoder* RNN is based on LSTMs (Hochreiter and Schmidhuber, 1997), which have been shown to be effective in particular in Machine Translation (Sutskever et al., 2014). Our approach is close to that of (Bahdanau et al., 2014), who relax the constraint of having a limited, fixed, length vector representation of the source sentence by using the encoder to produce *bi-directional* embeddings of words of this sentence along with an *attention* mechanism; this mechanism dynamically weighs the source embeddings at each time step, thus enabling the network to "attend" to some specific parts of the (still accessible) input during generation.

In our case, the input (source) representation of the information to be realised by NLG is in the form of a dialog act, as in (Wen et al., 2015), for example:

Dialog Act: *inform(name='phoenix hotel'; area='civic center'; accepts_credit_cards='yes')*
Realization: *the phoenix hotel is near the civic center and accepts credit card -s .*

[1] For example, number or gender (in some languages) agreements between a restaurant name and the remainder of a sentence (French example: *le ritz (la belle époque) est situé (est située)* ...); or spurious repetitions of articles in hotel names (*the HOTEL_NAME is ...* → *the the renaissance is ...*).

While Wen et al. (2015) handle the dialog act as a binary vector encoding different slot-value pairs (after delexicalization), we instead directly encode it as a sequence of tokens, either at the word or at the character level.

2.1.2 Background-Adaptor RNNs

In certain of our experiments, we exploit a variant of recurrent networks which we will call "Background-Adaptor RNNs", which is based on a recent proposal by Dymetman and Xiao (2016) for incorporating prior knowledge in recurrent networks to help training in the presence of limited data.[2] We now briefly explain this technique, which is applied here for the first time to the problem of NLG and to character-based models.

Abstracting away from details, a standard RNN model defines a conditional probability distribution $p_\theta(x_{t+1}|x_1, x_2, ..., x_t; C)$, where C is the context of the model (for us, this is the input dialog act), where $x_1, x_2, ..., x_t$ are the already generated tokens, and where x_{t+1} is the token being generated; the matrix parameters of the models are denoted by θ. The background-adaptor technique extends this as follows:

$$p_\theta(x_{t+1}|x_1, x_2, ..., x_t; C) \propto \underbrace{a_\theta(x_{t+1}|x_1, x_2, ..., x_t; C)}_{\text{adaptor}} . \underbrace{b(x_{t+1}|x_1, x_2, ..., x_t; C)}_{\text{background}} . \quad (1)$$

The difference is that now p_θ is defined as a combined process, obtained by multiplying an "adaptor" a_θ, which is a standard RNN, with a "background" b, which is an arbitrary, *a priori* defined conditional language model; it is given externally and is fixed during training of the combined process. This process is simply obtained by the product of the adaptor and the background, normalized over the different possible vocabulary symbols x_{t+1} to obtain a probability distribution (as indicated by the proportionality symbol). Overall, the process is still only parametrized by θ, and training with it only requires a small modification of the log-loss for taking into account the background factor.

To provide some intuition, let us note that when the background process is a uniform distribution over the vocabulary, we are back to the usual RNN. The other extreme is when the background process exactly corresponds to the actual observed data, in which case the adaptor only has to learn to produce a (close to) uniform distribution (an easy task). The interesting cases are intermediary situations, where the background process incorporates some prior information (for example a generic language model trained on a large corpus), which the adaptor can leverage in order to more easily adapt to the training data (for example a small in-domain corpus). In our application, the background process will provide some hard or soft constraints that the generated symbols have to conform to, presented in the form of finite-state automata.

3 The Models

Word-based model (WORD)

Our first, word-based, model uses the encoder-decoder architecture with the attention mechanism described above. As mentioned earlier, for a word-based model the data has to be de-lexicalised first[3] i.e. the named entities, such as restaurant names, addresses, telephone numbers, etc., have to be replaced by place-holders such as REST_NAME, TEL_NUMBER, and the like. These entities are then re-lexicalized at the end of the decoding to form the final utterance.

[2]The log-linear RNNs introduced in (Dymetman and Xiao, 2016) go beyond background-adaptor RNNs as described here: we only exploit the part of log-linear RNNs concerned with the distinction between the "background" and the "adaptor", not the aspects having to do with log-linear features. The use of a background for supporting RNN training is also implicit in the semantic parsing paper (Xiao et al., 2016).

[3]To emphasize this point, let us note that, in standard word-based seq2seq RNNs, the input and output vocabularies are totally disjoint. In order to map the input word "Ritz" into the output word "Ritz", the RNN has to learn the mapping from scratch based on training data, and cannot rely on any a priori knowledge about the correspondence; if "Ritz" is rare (or nonexistent) in the training data, the mapping cannot be learnt reliably (or at all). By de-lexicalizing "Ritz" into the generic "HOTEL", both in the input and output, the training is much simplified: the RNN only has to learn to map "HOTEL" to "HOTEL", a much more frequent observation.

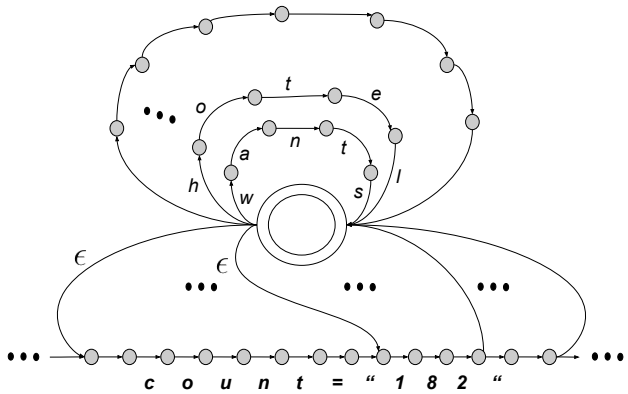

Figure 1: The Background FSA which shows the inclusion of target vocabulary words such as "hotel" and "wants" (top of the figure). Also, it is capable of accepting any substring from the dialog act such as the number "182" (bottom of the figure). The large central state is both initial and final.

Differently from (Wen et al., 2015), who use a one-hot encoding for the input DA, and also add a special gate mechanism to better control the consumption of slots, here we simply treat the DA as a sequence of word tokens.

Character-based model (C)

Our second model is a character-based model in which both the input and the output are character strings. Such models have been used in NMT (Neural MT) to tackle the problem of rare words (Ling et al., 2015; Luong and Manning, 2016). One advantage is that they work over a small vocabulary ($\sim$ 50 symbols), which they have each observed many times; in consequence, they have the ability to learn to map a character onto itself, if the context requires it. This copy mechanism is useful for carrying material from the original unprocessed input to the target, and, perhaps counter-intuitively, is not present in usual word-based RNNs, which have to learn each mapping from scratch (but see fn. 3, as well as the related work section 5 below about augmenting word-level RNNs with a copy ability).

Character-based model(s) with a Finite State Background (C-NWFSA and C-WFSA)

Our last class of models use a background-adaptor approach where the character-based model (adaptor) gets help from a finite-state automaton (FSA) which provides some prior knowledge (background).

The background FSA, in its non-weighted version (C-NWFSA) is illustrated in Figure 1. It accepts: (1) **All common words from the target vocabulary**: these words are all the words (around 500) that may appear in realizations, excluding the named-entities, independently of the DA input; (2) **All substrings from the source dialog act**: the FSA has the freedom to transition to and from any character position in the input DA; this acts as a prior, conditional on the input, which allows the adaptor to copy substrings, in particular named entities, from any part of the DA.

The idea behind this background automaton is that although the character-level model has the capacity to exploit dialog acts without de-lexicalisation (in contrast to the word-level models), nothing prevents it from generating on the one hand non-words, and on the other hand strings of characters that have no evidence in the DA. Through the "intersective" technique of equation (1), the background automaton b constrains the combined process p_θ to only produce common words or substrings of the DA. While the RNN adaptor a_θ works similarly similarly to model (C), the overall training loss depends on b also.

The FSA that we just described is the *unweighted* automaton (C-NWFSA), and its effect is strictly to accept or reject strings of characters. We also consider a *weighted* version (C-WFSA), with the same topology, but where the probabilistic weights on the arcs are parametrized (in a very elementary way) through a single parameter $\alpha \in [0, 1]$, as we now explain.

The total mass of transitions from the start state to one of the "common words" at the top of the figure is α, and the remaining mass $(1 - \alpha)$ is used for (epsilon) transitions to the character representation of

the DA at the bottom of the figure. Between the common words the mass is then distributed uniformly, and for edges from the start state to the DA, the mass is also distributed uniformly onto all the positions of the DA, that is, inversely proportionally to the length of the DA. Once on a position in the DA, the probability of escaping the DA back to the initial state is set to $1 - \alpha$, that of continuing in the DA to α. Overall a higher value of α indicates on the one hand that it is more difficult to "escape" from using common words once using common words, and also that once one is inside the DA, it is more difficult to escape from it back to the common words, which then encourages longer stretches of the DA to be produced once inside it. The way in which we fix α is detailed in section 4.

4 Experiments

4.1 Dataset

The datasets used for our experiments are those made available by Wen et al. (2015), in two domains, *hotel* and *restaurant*. Each set consists of DAs along with their natural language realisations. There are eight different DA types that indicate the communicative intent such as *inform*, *reject*, *confirm* etc. Each DA is a combination of slot-value pairs of the information to be conveyed, with a total of 13 different possible slots such as *name, pricerange, address* etc.

Each domain consists of roughly 5K samples, which we split in the ratio 8:1:1 into training, development and test.[4]

4.2 Implementation

The implementation is done using the Python libraries: Theano (Theano Development Team, 2016) and Lasagne (Dieleman et al., 2015). We implemented the attention mechanism (Bahdanau et al., 2014) by modifying the LSTM class of the Lasagne library. Also, the probability of the combined process c is calculated by element-wise multiplication of individual processes a_θ and b followed by normalization.

The FSA over characters is handled through PyFST[5], a python wrapper for OpenFST (Allauzen et al., 2007). We use this tool to perform the operations of ϵ-removal and determinization (otherwise delicate to program directly), in that order, from an initial non-deterministic weighted or unweighted FSA (which depends in part on the DA input). Finally, the automaton obtained is exported in matrix form to facilitate integration with Theano.

The models are all trained with the same configuration of hyperparameters: the forward and backward RNNs of the bidirectional encoder have 300 hidden units each, similarly the decoder RNN has 300 hidden units. The number of hidden units used in the single layer perceptron for calculating the attention weights is 100. For training, we use SGD together with Adam (Kingma and Ba, 2014) updates, with an initial learning rate of 0.001. A small minibatch of size 20 is chosen due to GPU memory constraints when storing FSA matrices for each sample contained in the minibatch. After the training procedure, beam search is used to sample utterances from the obtained conditional language model. A beam of length 5 is used to obtain the top 5 realisations and the one with highest probability is selected as the prediction.

The α parameter of the weighted FSA is fit to the data by performing a search over the list of values $[0.99, 0.95, 0.9, 0.8, 0.7, ..., 0.1]$. Log-likelihood of the training dataset is used as the optimization criterion, calculated as the sum of log probabilities of all target realizations present in the training set. The maximum likelihood is obtained for $\alpha = 0.9$, both for Hotel and for Restaurant, and this α is used in all our experiments.[6]

Also, the average number of states present in the deterministic weighted FSAs of the training samples is approximately 70K and 120K for the hotel and restaurant domain respectively.

[4]Note that we had to produce our own split, the data provided along with (Wen et al., 2015) does not specify their split, which prevents comparison with their results.

[5]https://github.com/vchahun/pyfst .

[6]As we already mentioned, our approach to fitting the automaton to the data with a single parameter α is simplistic. We could clearly use more parameters, and fit them through some EM procedure.

Model	BLEU	
	Hotel	Restaurant
WORD	0.4495	0.4322
C	0.4200**	0.3699**
C-NWFSA	0.4109**	0.3971**
C-WFSA	**0.4655**	**0.4381**

Table 1: BLEU scores of the models computed on the test set. Statistical significance is calculated using paired bootstrap resampling (Koehn, 2004) $^{**}p < 0.01$.

	Model	Adequacy		Fluency		
		Precision	Recall	No non-words	Non-redundant	Naturalness
Hotel	WORD	0.952*	**0.844**	0.989	0.941*	1.841*
	C	0.844**	0.633**	0.911**	0.974	1.674**
	C-NWFSA	0.844**	0.615**	0.974*	0.974	1.756**
	C-WFSA	**0.978**	0.815	**0.996**	**0.978**	**1.926**
Restaurant	WORD	0.956	**0.793**	0.994	0.976	1.908
	C	0.846**	0.530**	0.926**	**0.988**	1.787**
	C-NWFSA	0.820**	0.609**	0.959**	0.935**	1.731**
	C-WFSA	**0.973**	0.778	**0.997**	0.982	**1.932**

Table 2: Human evaluation of top realisation of the models. Statistical significance is computed through a pairwise difference one-tailed Student's t-test between the model with maximum score against the others. $^{*}p < 0.05$, $^{**}p < 0.01$.

4.3 Evaluation

4.3.1 Automatic Evaluation

Automatic evaluation results are shown in Table 1, using BLEU-4 (Papineni et al., 2002).[7] The results are shown for the four models (WORD, C, C-NWFSA, C-WFSA) described earlier, evaluated over our test sets for Hotel (538 realisations) and Restaurant (520 realisations).

4.3.2 Manual Evaluation

The automatic metrics do not always correlate with human judgement, so we also perform manual evaluation, based on the following adequacy (information conveyed) and fluency (linguistic quality) scales:

1. **Adequacy:**
   - Precision (i.e. "Correctness") [1/0]: all information in the DA is present in the generated utterance (1=yes, 0=no).
   - Recall (i.e. "Completeness") [1/0]: all information in the utterance is present in the DA (1=yes, 0=no).
2. **Fluency:**
   - No non-words [1/0]: all tokens in the utterance are actual words (1=yes, 0=no).
   - Non-redundant [1/0]: there is no repeated information in the utterance (1=yes, 0=no).
   - Natural or "good english" [2/1/0]: the utterance is "good" english (grammatical and natural) (2=good,1=acceptable,0=not acceptable). This is the main linguistic quality measure.

The evaluation was conducted on the full test set, both for Hotel and Restaurant, and is reported in Table 2.

[7] We used the mteval-v13a.pl script with default options from http://www.itl.nist.gov/iad/mig/tests/mt/2009/.

	Selected samples from the Hotel domain
1	inform(name='the inn san francisco';address='943 s van ness ave';phone='4156410188') [C]: the address of the inn san francisco is 943 s van ness ave . their phone number is 4156410188 . [C-NWFSA]: the inn san francisco 's phone number is 4156410188 [C-WFSA]: the address of the inn san francisco is 943 s van ness ave . the phone number is 4156410188 . [WORD]: **the the** inn san francisco 's address is 943 s van ness ave and the phone number is 4156410188 .
2	inform(name='hotel des arts';price_range='moderate') [C]: the hotel des **artea** hotel is in the moderate price range . [C-NWFSA]: the hotel des arts is in the moderate price range . [C-WFSA]: hotel des arts is in the moderate price range . [WORD]: hotel des arts is moderate -ly priced .
	Selected samples from the Restaurant domain
3	inform(name='yummy yummy';price_range=cheap;good_for_meal=dinner) [C]: i have found a restaurant called **mimmmmey** is good for dinner . [C-NWFSA]: **mmy** much is a good restaurant is good for dinner . [C-WFSA]: yummy yummy is cheap and good for dinner . [WORD]: yummy yummy is a cheap dinner restaurant with a cheap price range .
4	inform(name='straits restaurant';price_range=expensive;food=singaporean;good_for_meal=dinner) [C]: straits restaurant is an expensive restaurant that serves singaporean food . [C-NWFSA]: straits restaurant is expensive and serves singaporean food for dinner . [C-WFSA]: straits restaurant is expensive and is good for dinner . [WORD]: straits restaurant is an expensive restaurant that serves singaporean food and is good for dinner .

Table 3: Example realisations of the models. The most probable realization from a beam of length 5 is shown in each case.

4.3.3 Discussion

In terms of BLEU scores, we observe that the character-based model with a weighted finite-state background (C-WFSA) is significantly better than the other character-based models, and slightly better than the (WORD) model.

In terms of manual evaluation, the results are a bit more contrasted. Overall, (C-WFSA) is significantly better than the other two character-based models, apart from the case of "non-redundancy", where it behaves in a quite similar way; this is however not very surprising, because nothing in the FSA background that we presented specifically controls for non-redundancy (the non-repetition of semantic material). (C-WFSA)

All the models are quite deficient in Recall, but (WORD) is a bit better there; again, the FSA background does not control for Recall. On the other dimensions than Recall, (C-WFSA) is slightly better than (WORD), but not always significantly; it is especially good in overall linguistic quality (Naturalness) and in Precision (which is something that the background more directly controls for); interestingly, the unweighted model (C-NWFSA) is significantly worse on precision than the weighted version.

4.4 Illustrations

Examples We show in Table 3 a few examples from the different models. Example (1) shows a case where (C) and (C-WFSA) are both correct, (C-NWFSA) is deficient in recall, and (WORD) after re-lexicalization produces a sequence of two 'the'. Example (2) is a case where (C) invents a named entity. In example (3), (C) also invents a named entity and (C-NWFSA) incompletely copies one. In Example (4), (C) and (C-WFSA) are both deficient in recall, but the other models are good.

Attention Heatmap Because of its independent interest, we also show in Figure 2 an "attention heatmap", here in the case of model (C), for the following example:

```
[DIALOG-ACT]: inform(name='noe 7s nest bed and breakfast';address='1257 guerrero st')
[REFERENCE]: * the address is 1257 guerrero st for noe 7s nest bed and breakfast. #
[REALISATION]: noe 7s nest bed and breakfast is located at 1257 guerrero st .
```

The input DA is written along the columns of the array and the corresponding target realisation is written along the rows. An x^{th} row in the array represents the weights given by the attention mechanism over the character embeddings of the source dialog act at the point where it is generating the y^{th} character of the realisation.

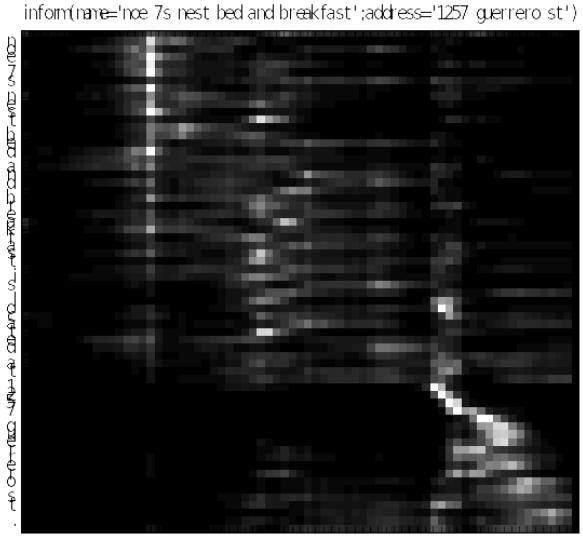

Figure 2: Attention heatmap for a selected example. The x-axis and y-axis denote the input DA and realisation respectively. Each pixel shows the attention weight (white largest) of the x^{th} source character at the point where the y^{th} target character is produced.

We observe a certain "2-block" structure of the attention, with the attention over the hotel slot value when generating the hotel name, over the address slot value when generating its address. The focus of attention is specially strong when generating the four digits 1,2,5,7; one should note that the (C) model has to learn to copy these digits one by one from the input to the target. By contrast, there is very little attention specifically paid to the initial part of the DA "i n f o r m (n a m e", because the little specific information contained there is easy to convey in a more distributed way over the whole input encoding.

5 Related Work

In the field of Neural Machine Translation (NMT), the problem of translating "rare words" such as named entities has recently attracted a fair amount of attention. The main approaches have been ones that either try to augment RNNs with some form of word-copying, or else have, similar to us, some character-level aspects.

Luong et al. (2014) preprocess the data and replace each unknown word in the target sentence by a placeholder token also containing a positional pointer to the corresponding word in the source sentence. Through these pointers, they learn an explicit mapping for copying unknown words from the source to the target. This is a bit similar to the de-lexicalization process of (Wen et al., 2015) in NLG, although the positional aspect might allow to handle several values associated with the same slot type.

Ling et al. (2015) tackle the problem of unseen words by proposing the use of a character-level model instead of a word-level model. As in our case, the limited vocabulary (namely, the different characters) is small enough that the model can learn to copy characters in certain contexts. This model is different from a strict character-level model as it introduces a hierarchy for forming words from characters instead of regarding a sentence as a flat string of characters. Luong and Manning (2016) also propose an hybrid word-character model to handle the rare word problem. For encoding the source sentence, they use a character-level model for rare words and a word-level model for frequent words. Then, for generating a target sentence, they use a word level model to get a first realisation of the target which contains <unk> for rare words and in a second step use a character-level model to generate a realisation of the rare word using contextual information.

Ling et al. (2016), in the context not of NMT this time, but of code generation, introduce a multiple predictor framework, where they can handle jointly the generation of generic code tokens and that of specific tokens for variable names and the like. They choose a character-level softmax for generating generic target tokens along with a pointer network (Vinyals et al., 2015) for copying specific tokens.

We are not aware of any prior work making use of models for *a priori* constraining the string of characters generated by an RNN, as we are doing here.

6 Conclusion

In this work we have proposed a character-level generator which is able to "copy" information from the source dialog act to the target utterance, and which uses original data without requiring pre-processing. By incorporating prior knowledge in the form of a finite-state automaton, exploiting a notion of "background-augmented" RNN, we discourage the character-level model from generating non-existing words or information for which there is no evidence in the input. Overall, the weighted version of the automaton performs much better than the version without prior knowledge, better than the non-weighted version of the automaton, and slightly better than the word-based version that requires de- and re- lexicalisation.

The main areas where our weighted automaton does not perform well (along with all other models, to different extents) are those of "Recall" (expressing all information present in the DA) and "Non-redundancy" (not expressing the same content twice). These are the same areas in which the original model of (Wen et al., 2015) introduced specific machinery, both in terms of a technique for controlling the "consumption" of slots, and of the use of a reranker on top of the operation of the RNN. As a perspective, we could also easily use a reranker, but as a continuation of our overall approach we would preferably incorporate the corresponding constraints as *a priori* knowledge, for instance by intersecting the current automaton with one that (i) forced certain substrings of the DA to appear in the target (improving recall), and (ii) prevented certain substrings to appear twice (improving non-redundancy).[8]

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. Openfst: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Anja Belz. 2008. Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models. *Natural Language Engineering*, 14(04):431–455.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave. 2015. Lasagne: First release., August.

Marc Dymetman and Chunyang Xiao. 2016. Log-Linear RNNs: Towards Recurrent Neural Networks with Flexible Prior Knowledge. *arXiv: 1607.02467*, pages 1–22.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. Citeseer.

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*, ACL '98, pages 704–710, Stroudsburg, PA, USA. Association for Computational Linguistics.

[8] One important potential advantage of the background approach over reranking is that it operates more locally and can be used in situations where a reranker would need to filter among a myriad of not-so-good candidates; additionally, the background is exploited by the adaptor when learning the parameters θ. We note also that the background approach does not require b to be realized by an automaton, but can be realized by any predefined process of the form described in (1).

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based Neural Machine Translation. *ICLR'16*, pages 1–11.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Association for Computational Linguistics (ACL)*, Berlin, Germany, August.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the Rare Word Problem in Neural Machine Translation. *Arxiv*, pages 1–11.

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*, pages 311–318. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. *Neural Information Processing Systems 2015*, pages 1–9.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings Association For Computational Linguistics*, Berlin, August.