Deep text generation – Using hierarchical decomposition to mitigate the effect of rare data points

Nina Dethlefs and Alexander Turner

School of Engineering and Computer Science
University of Hull
Cottingham Road, HU7 6RX, UK
n.dethlefs@hull.ac.uk | alexander.turner@hull.ac.uk

Abstract. Deep learning has recently been adopted for the task of natural language generation (NLG) and shown remarkable results. However, learning can go awry when the input dataset is too small or not well balanced with regards to the examples it contains for various input sequences. This is relevant to naturally occurring datasets such as many that were not prepared for the task of natural language processing but scraped off the web and originally prepared for a different purpose. As a mitigation to the problem of unbalanced training data, we therefore propose to decompose a large natural language dataset into several subsets that "talk about" the same thing. We show that the decomposition helps to focus each learner's attention during training. Results from a proof-of-concept study show 73% times faster learning over a flat model and better results.

Keywords: artificial intelligence, natural language processing, deep learning

1 Introduction

Language data for almost all domains can be scraped off the web with relative ease and in large quantities. As more data becomes available, research is needed into methods that facilitate access to this data, including the generation of natural language text from non-linguistic data input. Many NLG techniques are available to generate high-quality outputs from annotated datasets [23, 13, 8, 6], aligned corpora [12, 1] or annotated databases [19, 11]. More work is also done recently on generating language from unaligned input data [5, 16], where it is the learner's task to work out the mapping between non-linguistic data points and sequences of words. Nonetheless, most state-of-the-art techniques in statistical NLG still do not readily transfer to language datasets that were scraped off the web and are unaligned and thus in some respects "messy".

The most promising method for such messy tasks is probably deep learning. Recurrent neural networks (RNNs) have made important advances in natural language processing, including NLG. Results have been particularly impressive for deep learning models that are trained from large datasets and that contain a well-balanced distribution of the linguistic target phenomena. For applications in NLG, the sequence-to-sequence encoder-decoder model by [24, 9] has been shown to successfully learn to map an input sequence $\mathbf{x}$ of symbolic inputs, such as any form of non-linguistic data, to an output sequence $\mathbf{y}$ of words expressing the data in natural language. Again however, this technique works best when a sufficiently large, and well-balanced, dataset is available. For many natural language processing tasks, datasets are smaller due to the time and effort involved in preparing them. In such cases, it can be challenging to learn good feature representations. This is the problem we focus on in this short paper.

We present an approach that decomposes a large NLG task into a set of subtasks, each of which can be represented by an individual neural network that focuses on a subset of the training data. The idea is to automatically identify partitions of the training data that focus on the same semantics. Intuitively, the words and syntactic phrases used to forecast "rain" might be different from those used to forecast "wind"—even though overlaps are possible. Our idea is that focusing on a subset of relevant training instances will help a deep learning agent perform better on infrequent data points that would otherwise "get lost" in the large input dataset in a flat learning setup. Experiments in the weather forecast domain show that our approach substantially increases performance on outputs that occur infrequently in the dataset. We find that the divide-and-conquer approach is also much faster to train and achieves higher than state-of-the-art performance in a comparison with previous work.

2 Background

The sequence-to-sequence model that underlies much work on deep learning for NLG was first presented by [20] and [4], who use an RNN Encoder-Decoder model to learn a mapping from an input sequence to an output sequence for machine translation. By training both models jointly, a mapping from the source to the target language is learnt. This model was first applied to NLG by [24], who use an LSTM to do sentence planning and surface realisation in an application to information-seeking dialogue, and show that their model outperforms other approaches to the same domain. [9] apply a similar model to generate output sequences alongside dependency trees, and show that additional benefits can be gained from the use of an attention mechanism during training.

The idea of learning a direct mapping from inputs to outputs without intermediate annotation is related to work on NLG from unaligned datasets. Two popular approaches to do this are the use of parallel corpora or annotated databases. [1] define the generation process as a sequence of hierarchically-organised local decisions, and [11] generate language from weighted hypergraphs.

A practical limitation of deep learning algorithms is that they rely on large datasets to learn good representations. This can be problematic in domains like NLG or other NLP tasks that rely on a paired set of inputs and outputs—which are not available in large quantities. Previous work on hierarchical reinforcement learning for NLG has shown that when using a divide-and-conquer approach to decompose a complex task into a hierarchical set of subtasks, it becomes feasible to solve the problem in a scalable way without significant loss in performance [7]. Similar results have been observed for the hierarchical decomposition of neural language models [15].

Algorithm 1 Finding a hierarchy.

```
1: function FINDSUBTASKS(forecast texts $f$, alignments $a$, weather events $e$) return $subtasks$
2:     subtasks = list []
3:     for each sentence $s$ in $f$ do
4:         Get alignments $a_s$ with weather events $e_s$
5:         event_combination = the types of all weather events expressed in $s$ as identified from alignments $a$
6:         if event_combination is not in subtasks then
7:             add event_combination to subtasks as an object with input and output examples
8:         end if
9:     end for
10:    for each element in subtasks do
11:        Create a separate subtask (neural net).
12:    end for
13: end function
```

3 Deep learning model

A neural network, such as a *multi-layer perceptron*, learns a hidden representation $\mathbf{h}$ of an input sequence $\mathbf{x} = (x_1, \ldots, x_N)$ by learning an increasingly abstract encoding of the inputs, and a mapping from $\mathbf{h}$ to either a single output (for classification tasks) or an output sequence $\mathbf{y} = (y_1, \ldots, y_M)$ (in a sequence-to-sequence learning task). The hidden representation $\mathbf{h}$ can be computed as $\mathbf{h} = f(x)$, where f is an activation function, such as sigmoid, tangent or relu. During training, the goal is to minimise the loss L between the input and output, e.g. using cross entropy.

In a *recurrent neural net*, we follow the same procedure, except that $\mathbf{h}$ is learnt as an increasingly abstract representation through recursive updates at each time step t: $\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$. As conventional RNNs are associated with the problem of vanishing or exploding gradients [3], we use an LSTM and follow the definition of [10].

4 Data and Learning Task

As we aim mainly for a proof-of-concept study in this paper, we will focus our experiments on a single domain: weather forecast generation. We use the WEATHERGOV dataset from [12], which contains 29,528 weather scenarios. It contains 12 different weather events, such as *temperature*, *skyCover*, etc. We use this existing dataset for ease of availability and also to allow for a comparison of our model's performance with previous work on the same dataset. Given that the dataset was originally collected from the web[1], we argue that our technique will be transferable to other such collected datasets. Future work will endeavour to demonstrate this.

Each of the weather events can be seen as a collection of lexical-syntactic constructions that describe the same semantic concepts. In other words, we can identify a subset of words and syntactic phrases that are reused in the datasets to describe a certain group

[1] `http://www.weather.gov/`

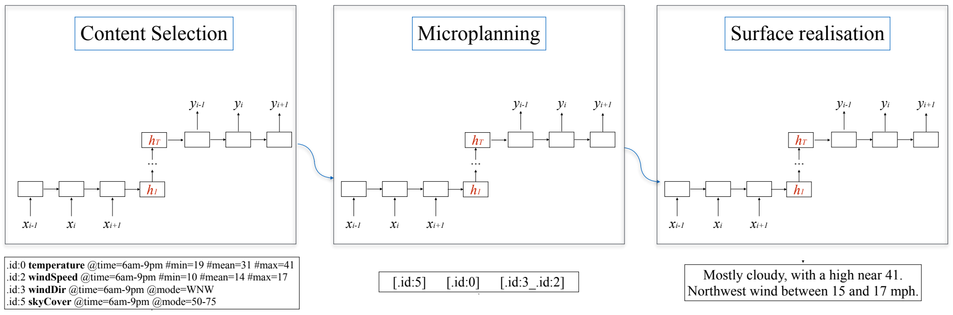
Content Selection
Microplanning
Surface realisation
y_{i-1}
y_i
y_{i+1}
h_T
h_1
x_{i-1}
x_i
x_{i+1}
.id:0 **temperature** @time=6am-9pm #min=19 #mean=31 #max=41
.id:2 **windSpeed** @time=6am-9pm #min=10 #mean=14 #max=17
.id:3 **windDir** @time=6am-9pm @mode=WNW
.id:5 **skyCover** @time=6am-9pm @mode=50-75
[.id:5] [.id:0] [.id:3_.id:2]
Mostly cloudy, with a high near 41.
Northwest wind between 15 and 17 mph.

Fig. 1. Illustration of the NLG task as a pipeline [18]: *content selection* decides which weather events to include in the forecast; *microplanning* produces an ordered list of these events, and *surface realisation* produces a string of words. Example representations are shown for each stage.

of weather events. However, not all weather events map neatly onto a single sentence. Instead, we find cases of sentences that express more than one weather record and we find weather records that get described across multiple sentences.

4.1 Hierarchical decomposition

Algorithm 1 shows the heuristics we used to identify a hierarchy of weather events from [12]'s WEATHERGOV dataset, which provides alignments between weather events and sentences. From these alignments, it is possible find out exactly which weather events are expressed in which sentence. To find a hierarchy, we loop over each sentence in the human forecasts, find the weather events that it expresses and create one generation subtask for each unique combination of aggregated weather events. This led to 22 subtasks overall, which can be seen as subtasks of a single large task "weather forecast".

Subtask generators can be seen as sentence generators that are associated with their own portion of the dataset including input examples (weather events) and output examples (word sequences). To generate a forecast, we obtain individual sentences from their respective generators and then concatenate them into a single text.

4.2 Natural language generation tasks

Figure 1 shows our NLG pipeline with the representations used at each stage. The outputs of each generation stage are fed as inputs to the next stage. This is important in order not to generate fragmented and isolated sequences of words but a coherent output text. The first stage *content selection* decides for each weather record whether it should be included in the final forecast or not. We learn a binary classification value: 1 if the record is used in the forecast and 0 if it is not used. An LSTM with 2 layers and 20 hidden units does not find this task very challenging and achieves an accuracy of 98% for WEATHERGOV after 100 training epochs. All events chosen for inclusion

System	BLEU-4	CORRECTNESS	FLUENCY
HUMAN	1.0	4.01	4.37
OURSYSTEM	0.67	3.91	3.96*
FLAT BASELINE	0.23	-	-
ANGELI ET AL. (2010)	0.52	4.22	4.12
KONSTAS ET AL. (2012)	0.34	4.03	3.92
MEI ET AL. (2016)	0.70	-	-

Table 1. Overview of objective and subjective results.* indicates statistically different from HUMAN using a 2-tailed Wilcoxon signed-rank test.

at the content selection stage are passed on to the *microplanning* module in the second stage. The aim of this stage is to rank and order the weather events that were selected for inclusion. Part of this problem is also to decide whether to present weather events individually in a single sentence or to aggregate several events into the same sentence. We treated the microplanning task as a sequence-to-sequence learning task, e.g. [.id:0, .id:2, .id:3, .id:5] is mapped to [.id:5 .id:0 .id:3_.id:2] in Figure 1, representing a new order of events and the fact that .id:3 and .id:2 should be aggregated into one sentence. We train an LSTM with 4 layers and 20 hidden units for 2000 epochs, and obtained an accuracy of 87% for WEATHERGOV. The *surface realisation* stage finally is also implemented as an LSTM with 4 layers and 50 hidden units. The input sequence corresponds to an ordered set of non-linguistic measurements $\mathbf{x} = (x_1, \ldots, x_N)$, and the output sequence is a sequence of words $\mathbf{y} = (y_1, \ldots, y_M)$. All models are trained with 32 batches over 2,000 epochs. To facilitate training, we use a `BOS` and an `EOS` symbol to denote the beginning and end of a sequence. All results use the same data split as previous work [1, 11]: 25,000 for training, 1,000 for validation and 3,528 for testing.

5 Experiments and Evaluation

Our experiments evaluate the final outputs generated by the generator illustrated in Figure 1. A future evaluation could investigate the errors made at individual generation stages and quantify their contribution to the overall objective and subjective assessment; see Table 1. In this paper, we focus on evaluating the overall output quality.

5.1 Objective Evaluation

Table 1 (left) shows results in terms of the BLEU modified precision score [17] measuring similarity with human examples. We compare against a human upper-bound and previous work on the same domain by [1], [11] and [14]. Our system reaches higher BLEU scores than the former two studies, 22% and 49%, respectively, but a slightly lower score than [14], 4.3%. Table 2 shows an example of one of our generated forecasts and a human equivalent. We chose to present a non-perfect example, as the type of error shown is representative for our system's outputs. The LSTM policy learns the correct word sequences and mappings from semantic inputs, but occasionally duplicates

	Generated output
HUMAN	A 30 percent chance of showers and thunderstorms after noon. Mostly cloudy, with a high near 69. South wind between 10 and 20 mph, with gusts as high as 30 mph.
WEATHERGOV	A 30 percent chance of showers and thunderstorms after noon. Mostly cloudy, Mostly cloudy, with a high near 69. With a south wind 10 to 20 mph, with gusts as high as 30 mph.

Table 2. Generated example outputs.

phrases. We were able to train our hierarchical model in 45 hours on a GPU (Tesla K40)—73% faster than we observed for a flat setup.

We also present results from a flat baseline model that learns from the whole training set without hierarchical decomposition. This model skips the steps shown in Figure 1 and maps (the redundant set of) weather records directly to words. As can be seen in Table 1, the results are not strong. We believe that this is due to some examples being too rare in the training data in the flat case for the LSTM to learn good representations for them. For example, while *temperature* occurs in over 90% of weather forecasts across both datasets, *snow* occurs in only 1%, thus "getting lost" in the training data in the flat setup. Given the low BLEU score (and manual inspection) we decided not to evaluate the flat model with human raters.

5.2 Subjective Evaluation

Table 1 (right) shows results from a human rating study. To allow for a comparison with previous work, we asked human raters the same questions as [1] and [11]: "Does the meaning conveyed by the text correspond to the database input?" to determine *semantic correctness* and "Is the text grammatical and overall understandable?" to determine *English fluency*. *Semantic correctness* is evaluated on a 1-5 scale with values mapping to "perfect", "near perfect", "minor errors", "major errors" and "completely wrong". *English fluency* also uses 5 values "flawless", "good", "non-native", "disfluent" and "gibberish". 43 human raters were recruited from Amazon Mechanical Turk (`www.mturk.com`) to rate altogether 800 weather forecasts (400 system-generated, 400 human) on a scale of 1-5. The human-authored forecasts were rated better than our system-generated ones for both categories. The difference is statistically significant at $p<0.01$ for *fluency*, according to a 2-tailed Wilcoxon ranked sum test. OURSYSTEM forecasts receive scores comparable to earlier work by [11]. [14] do not report a subjective evaluation, which is unfortunate for our comparison given the low correlation between BLEU scores and human quality assessment [2].

The difference for *semantic correctness* is not significant between OURSYSTEM and HUMAN. We believe that rounding of measurements played a role in the correctness ratings. The human forecasts might round a windSpeed of "max:78" to "up to 80 mph". This is frequent in the data and our system learnt to do this too. Some raters penalised this phenomenon more severely than others, but it occurred in both our human and system-generated data. We are surprised to see that Angeli's and Konstas' systems achieve better results than our human examples in the correctness case.

6 Conclusion and Future Work

We have presented a proof-of-concept study on using hierarchical decomposition for large NLG tasks into subsets of smaller tasks within a deep learning framework. The divide-and-conquer approach to learning can (a) lead to much faster learning (up to 73%), and (b) allow us to learn good generation policies for a dataset that is not ideally balanced with some examples occurring much more frequently than others. We believe that this work is relevant to language datasets that are scraped off the web. While language data is available in abundance on the web, directly scraped data is often less clean and structured as required for many state-of-the-art techniques in NLP, including deep learning. We observed that the latter can particularly struggle to learn good representations for infrequent data points in large training sets.

Future work needs to find ways to decompose datasets of examples automatically in a more principled way than based on heuristics, e.g. using genetic algorithms [21, 22]. Also, we have applied our technique to a vanilla LSTM model only and could explore more sophisticated deep learning models including e.g. such with attention mechanism. Also, we plan to test our approach on additional language datasets scraped off the web.

Acknowledgements

We acknowledge the VIPER high-performance computing facility of the University of Hull and its support team. We are also grateful for Nvidia's donation of a Titan X Pascal graphics card for our work on deep learning.

References

1. Angeli, G., Liang, P., Klein, D.: A Simple Domain-Independent Probabilistic Approach to Generation. In: Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP). Cambridge, Massachusetts (2010)
2. Belz, A., Gatt, A.: Intrinsic vs. extrinsic evaluation measures for referring expression generation. In: Proc. of the 46th Annual Meeting of the Association for Computational Linguistics (ACL). Columbus, OH, USA (2008)
3. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2), 157–166 (1994)
4. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: 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). Doha, Qatar (2014)
5. Cuayáhuitl, H., Dethlefs, N., Hastie, H., Liu, X.: Training a Statistical Surface Realiser from Automatic Slot Labelling. In: Proceedings of the IEEE Workshop on Spoken Language Technology (SLT). South Lake Tahoe, USA (2014)
6. Dethlefs, N., Cuayáhuitl, H.: Hierarchical Reinforcement Learning and Hidden Markov Models for Task-Oriented Natural Language Generation. In: Proc. of the 49th Annual Conference of the Association for Computational Linguistics (ACL-HLT). Short Papers. Portland, OR, USA (2011)
7. Dethlefs, N., Cuayáhuitl, H.: Hierarchical Reinforcement Learning for Situated Natural Language Generation. Natural Language Engineering 21, 391–435 (2015)

8. Dethlefs, N., Hastie, H., Cuayáhuitl, H., Lemon, O.: Conditional Random Fields for Responsive Surface Realisation. In: Proc. of the 51st Annual Meeting of the Association for Computational Linguistics (ACL). Sofia, Bulgaria (2013)
9. Dusek, O., Jurcicek, F.: Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings. In: Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL). Berlin, Germany (2016)
10. Graves, A.: Generating Sequences With Recurrent Neural Networks. CoRR abs/1308.0850 (2013), `http://arxiv.org/abs/1308.0850`
11. Konstas, I., Lapata, M.: Unsupervised Concept-to-Text Generation with Hypergraphs. In: Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL). Montreal, Canada (2012)
12. Liang, P., Jordan, M., Klein, D.: Learning Semantic Correspondences with Less Supervision. In: Proc. of the 47th Annual Meeting of the Association for Computational Linguistics (ACL). Singapore (2009)
13. Mairesse, F., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., Young, S.: Phrase-Based Statistical Language Generation Using Graphical Models and Active Learning. In: Proc. of the 48th Annual Meeting of the Association of Computational Linguistics (ACL). Uppsala, Sweden (2010)
14. Mei, H., Bansal, M., Walker, M.: What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In: Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL). San Diego, CA, USA (2016)
15. Morin, F., Bengio, Y.: Hierarchical probabilistic neural network language model. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 246–252 (2005)
16. Novikova, J., Rieser, V.: The aNALoGuE Challenge: Non Aligned Language GEneration. In: Proceedings of the 9th International Natural Language Generation Conference (INLG) (2016)
17. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: A Method for Automatic Evaluation of Machine Translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL). pp. 311–318. Association for Computational Linguistics (2001)
18. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press, New York, NY, USA (2000)
19. Snyder, B., Barzilay, R.: Database-Text Alignment via Structured Multilabel Classication. In: Proc. of 20th International Joint Conference on Artificial Intelligence (IJCAI). Hyderabad, India (2007)
20. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems (NIPS) 27. pp. 3104–3112 (2014)
21. Turner, A.P., Caves, L.S., Stepney, S., Tyrrell, A.M., Lones, M.A.: Artificial epigenetic networks: Automatic decomposition of dynamical control tasks using topological self-modification. IEEE Transactions on Neural Networks and Learning Systems (2016)
22. Turner, A.P., Lones, M.A., Fuente, L.A., Stepney, S., Caves, L.S., Tyrrell, A.M.: The artificial epigenetic network. In: Evolvable Systems (ICES), 2013 IEEE International Conference on. pp. 66–72. IEEE (2013)
23. Walker, M., Stent, A., Mairesse, F., Prasad, R.: Individual and Domain Adaptation in Sentence Planning for Dialogue. Journal of Artificial Intelligence Research 30(1), 413–456 (2007)
24. Wen, T.H., Gašić, M., Mrkšić, N., Su, P.H., Vandyke, D., Young, S.: 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) (2015)