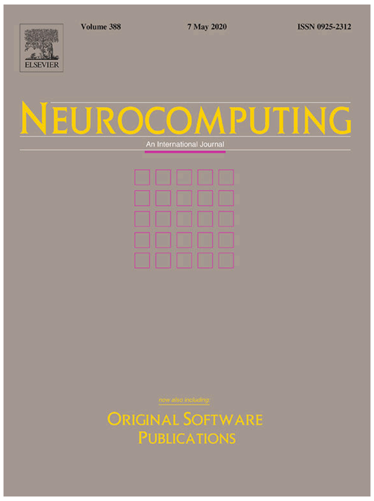

A Divide-and-Conquer Approach to Neural Natural Language Generation from Structured Data

Nina Dethlefs, Annika Schoene, Heriberto Cuayáhuitl

PII:	S0925-2312(20)31995-0
DOI:	https://doi.org/10.1016/j.neucom.2020.12.083
Reference:	NEUCOM 23223
To appear in:	*Neurocomputing*
Received Date:	24 September 2019
Accepted Date:	14 December 2020

A Divide-and-Conquer Approach to Neural Natural Language Generation from Structured Data

Nina Dethlefs[a,*], Annika Schoene[a], Heriberto Cuayáhuitl[b]

[a]*University of Hull, Department of Computer Science and Technology, Big Data Analytics Research Group, Cottingham Road, Hull HU6 7RX, United Kingdom*
[b]*University of Lincoln, School of Computer Science, Lincoln Centre for Autonomous Systems (L-CAS), Brayford Pool, Lincoln LN6 7TS, United Kingdom*

Abstract

Current approaches that generate text from linked data for complex real-world domains can face problems including rich and sparse vocabularies as well as learning from examples of long varied sequences. In this article, we propose a novel divide-and-conquer approach that automatically induces a hierarchy of "generation spaces" from a dataset of semantic concepts and texts. Generation spaces are based on a notion of similarity of partial knowledge graphs that represent the domain and feed into a hierarchy of sequence-to-sequence or memory-to-sequence learners for concept-to-text generation. An advantage of our approach is that learning models are exposed to the most relevant examples during training which can avoid bias towards majority samples. We evaluate our approach on two common benchmark datasets and compare our hierarchical approach against a flat learning setup. We also conduct a comparison between sequence-to-sequence and memory-to-sequence learning models. Experiments show that our hierarchical approach overcomes issues of data sparsity and learns robust lexico-syntactic patterns, consistently outperforming flat baselines and previous work by up to 30%. We also find that while memory-to-sequence models can outperform sequence-to-sequence models in some cases, the latter are generally more stable in their performance and represent a safer overall choice.

Keywords: Neural networks, artificial intelligence, natural language processing

*Corresponding author
Email address: `n.dethlefs@hull.ac.uk` (Nina Dethlefs)

1. Introduction

Current approaches for text generation from complex real-world data, such as Wikipedia or other rich domains, often face the problem of large and varied vocabularies, long sequences and unbalanced representations of input-output pairs. While state-of-the-art natural language generation (NLG) architectures, such as sequence-to-sequence models, have made good improvements over the last years [1, 2, 3, 4, 5], the problem of generating long sequences of text and maintaining coherence, grammatical and semantic correctness remains a challenge.

Recent approaches to text generation from Wikipedia have made use of the fact that semantically similar examples often share similar lexico-syntactic realisations. For example, [6] generate text from a semantically conditioned language model, and [7] group their input semantically rather than sequentially. While both of these models report improved performance over baselines that do not exploit semantic context, they still only achieve modest BLEU scores in the range of 22% and 34% respectively, which in many cases does not lead to a coherent output text.

In this article, we propose a novel divide-and-conquer approach for natural language generation that induces a hierarchy of generation spaces (i.e. input-output pairs, effectively) automatically from an unlabelled corpus of examples. Generation spaces are based on a notion of similarity of partial knowledge graphs with entity and relation embeddings and can be identified with off-the-shelf clustering algorithms. Subsequently, we train specialised generators for individual generation spaces, so as to reduce data sparsity, improve the coherence of outputs and reduce the amount of errors in generated texts. To maintain prior knowledge of the domain and context, we share a set of common weights across the hierarchy of generators that are pre-trained and passed from parents to children. See Figure 1 for an illustration. Our approach is different to other hierarchical approaches such as [8, 9] who work with hierarchical hidden representations to capture domain context and dialogue history but still train a single model for the full domain.

For the implementation of our generators, we experiment with both sequence-to-sequence (Seq2Seq) and memory-to-sequence (Mem2Seq) models, mostly to allow a

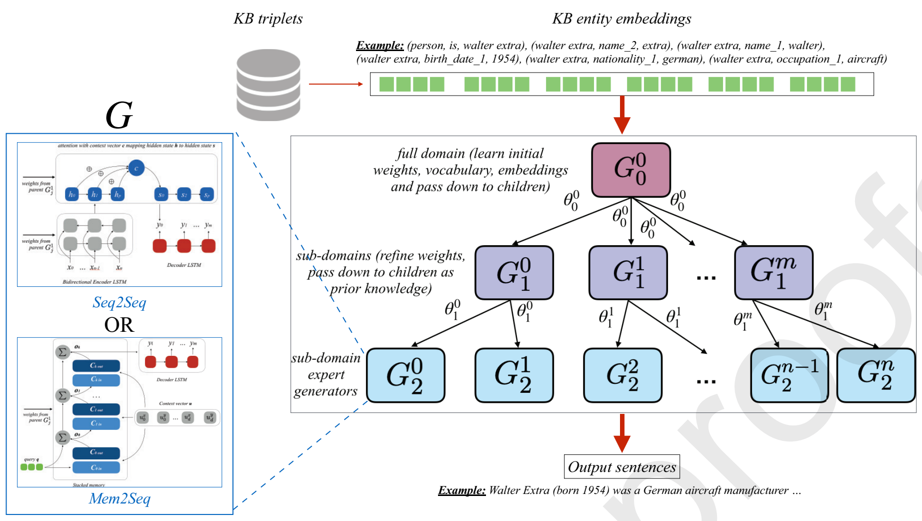
KB triplets
KB entity embeddings
Example: (person, is, walter extra), (walter extra, name_2, extra), (walter extra, name_1, walter), (walter extra, birth_date_1, 1954), (walter extra, nationality_1, german), (walter extra, occupation_1, aircraft)
G
full domain (learn initial weights, vocabulary, embeddings and pass down to children)
sub-domains (refine weights, pass down to children as prior knowledge)
sub-domain expert generators
Seq2Seq
OR
Mem2Seq
Output sentences
Example: Walter Extra (born 1954) was a German aircraft manufacturer ...

Figure 1: Hierarchy of generators $G^0_0 \ldots G^n_2$, where parent G^0_0 represents the full generation domain, $G^0_1 \ldots G^m_1$ represent sub-domains and are trained to capture shared pre-trained weights as prior knowledge, and $G^0_2 \ldots G^n_2$ are sub-domain "expert" generators. Generators in the hierarchy are modelled as either Seq2Seq or Mem2Seq networks as shown, please see Section 3.3 for details. Agents in a lower level of the hierarchy use pre-trained weights from the upper level and cannot start training until the above level has beed trained. In other words, training is done in stages: Stage 1 trains model G^0_0, Stage 2 trains models G^i_1, and Stage 3 trains models G^i_2. During testing, only the models derived from Stage 3 (bottom level) are used.

comparison of alternative attention mechanisms. The general idea of a divide-and-conquer approach is transferable to other models too, e.g. [10].

We present experiments using two existing benchmarked datasets, WEBNLG of the recent WebNLG challenge for generating text from RDF triplets [11], and WIKI-BIO, a large dataset of Wikipedia info boxes and biography texts [6]. We find across datasets and metrics that the hierarchical models consistently outperform the flat models, including competitive baselines. The hierarchical models overcome problems of incoherent long sequences and learn a robust set of lexico-syntactic patterns specific to their sub-domain. We also find that while both Seq2Seq and Mem2Seq models achieve good performance in a divide-and-conquer setup, the former is more consistent and shows less variation in output quality.

The article is structured as follows. We review related work on neural NLG and NLG from structured data in Section 2. We then introduce our learning model in Section 3 including mechanisms for knowledge graph representation, hierarchy induction and alternative attention mechanisms. Section 4 describes our experimental setup, metrics and baselines, and Section 5 discusses results from objective and human evaluations. We conclude and discuss future research in Section 6.

2. Related Work

2.1. Neural Approaches to Natural Language Generation

Our general approach is related to work on sequence-to-sequence models [12] that encode a sequence of input symbols and decode them onto a separate sequence of output symbols, such as pioneering work on machine translation [13, 14]. Natural language generation systems have mostly mapped a semantic input sequence, e.g. semantic slots to be filled in a task-oriented dialogue setting, onto a sequence of words, experimenting e.g. with semantic control gates [1], linguistically inspired input representations [2, 15, 16] or graph-based generation inputs [17, 18] in order to gain semantic control over generated sequences. Other related work has explored sequence-to-sequence architectures for dialogue generation [4, 19, 3] and programming code generation [20, 21, 22], amongst others.

An active area of research has also been the role of linguistic information in encoder-decoder architectures to improve generated output quality. [3] apply an LSTM encoder-decoder to the same restaurant domain data as [1] and show that additional performance can be gained through the use of dependency trees as an intermediate representation. [15] uses an LSTM encoder-decoder for cross-domain NLG, showing that using linguistically-rich abstract meaning representations (AMRs) [23] as inputs to a generator can aid domain transfer. While [15]'s work was based on human annotation of AMR graphs, [24] are able to extract an effective AMR parser from unlabelled data which they utilise to produce annotated data for an encoder-decoder generator. Other work that explores abstract meaning representations to gain semantic control includes

[17] who generate language from encoded graph networks and [18] who explore graph encoding itself.

Other approaches to neural natural language generation include [2] who jointly model content selection and surface realisation for weather report generation and Robocup summaries; [25] generate weather reports and focus on learning models for rare data points. [4] generate outputs for a task-oriented spoken dialogue system and show that using an ensemble of different encoders — LSTM, bidirectional LSTM and CNN — outperforms a standard LSTM encoder-decoder with attention. [19] propose to tackle the problem of semantic inaccuracies in encoder-decoder models through a sequence of pre-processing operations on semantic fields. Most recently, there has also been an increase in approaches that apply transformer networks to NLG [10, 26, 27].

An alternative model for neural text generation is based on variational autoencoders (VAEs). Those models are mostly applied to unlabelled data and can consequently offer less semantic control over the outputs in some cases. An advantage of these architectures is that they mostly operate on unlabelled data, thus cutting down on development costs whenever high-quality data is available. [28] present a VAE that combines convolutional and deconvolutional techniques with a recurrent output layer to overcome problems of existing purely recurrent VAEs that lack coherence in long output sequences. Working on the problem of semantically variable outputs, [29] show that VAEs trained from systematically noisy input data can learn to produce semantically-relevant outputs by treating semantic fields as a corrupted version of the desired representation.

Finally, recent work has (re-)explored natural language templates in the context of neural output generation. [30] train a neural hidden Semi-Markov Model that associates semantic fields with phrases in an attempt to impose more structure on the generation process and overcome the lack of transparency and semantic control found in purely neural encoder-decoder models.

2.2. Natural Language Generation from Large Structured Knowledge Bases

Recent approaches to generating text from linked data are mostly based on recurrent neural network architectures that encode a set of semantic fields and decode them to a text that describes them. [6] train an encoder-decoder LSTM generator to map a sequence of input semantic fields onto a sequence of output words describing them. The authors introduce a set of mechanisms that help address the linguistic variety in the texts, e.g. the fact that the biography of a cricket player is likely to contain different semantic fields than the biography of a senior politician. They show that copy actions can effectively deal with words that apply mostly to subsets of biographies and that word embeddings are more useful when trained separately for each individual concept. Related research has faced similar issues relating to the size and sparsity of vocabulary and variety in semantic fields. [7] similarly aim to take advantage of semantic "sub-spaces" in their data when training an open domain NLG system from DBPedia data. They encode input semantics into a vector of triplets that can then get decoded to word sequences, especially representing input sequences in semantic groupings rather than sequentially. Closely related to this work, [31] address the same task using a different dataset extracted from Wikipedia. The authors employ an encoder-decoder LSTM architecture but at the same time use autencoding to reverse-encode outputs back to inputs in an attempt to constrain generation to include only the required input semantics. Results outperform the same n-gram baseline as reported by [6]. It is regrettable that no direct comparison on the same dataset is presented.

Whereas the above approaches aim to improve the semantic encoding for generators, [32] focus on a different sub-aspect of sequence classification, the prediction of rare words. They introduce a novel pointer sentinel architecture that can decide to predict a word or reproduce one that is salient in the current linguistic context. The authors evaluate their approach on a new Wikipedia dataset, this being a challenging domain given the large and sparse vocabulary.

Finally, the recent WebNLG challenge[1] made a step towards more comparability in the generation of text from linked data. The challenge received 8 system submissions in 2017, three of which were rule- or template-based, one used supervised learning and the remaining four relied on neural approaches. The winning system UMELBOURNE used an encoder-decoder with attention and entity delexicalisation [11].

Common issues discussed in all of the above approaches include the size and spar-

[1]http://webnlg.loria.fr/pages/results.html

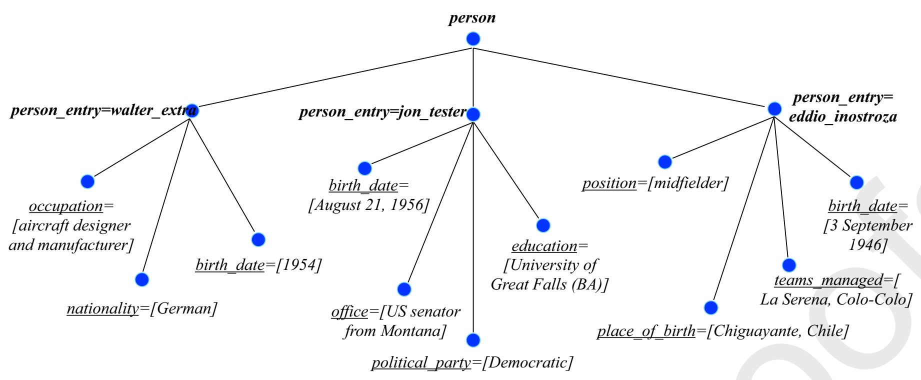

Figure 2: Illustration of a knowledge graph with top-level concept *person*, e.g. for the biography domain. Individual entities are associated with different combinations of semantic fields, leading to different biography outputs.

sity of vocabulary and variety in semantic fields when generating from linked data. This is often exacerbated when using real-world data, such as scraped off the web. Our work is closely related to the above approaches, e.g. [6] and [7], in their idea to exploit the fact that some portions of data are more similar in their lexico-syntactic patterns than others and that these patterns can be correlated with semantic fields to train semantically specialised models. In contrast to previous work that has learnt a single generation policy, we propose to work with an automatically induced hierarchy of generators, which learn specialised generation policies. Divide-and-conquer approaches have previously shown promise in other domains [9, 33, 34] but remain relatively unexplored for natural natural language generation.

3. Learning Model

3.1. Knowledge graph representation

We define the inputs to our generator as a set of (partial) knowledge graphs consisting of triplets (e^a, r, e^b) where e^a is an entity to the left, e^b is an entity to the right and r is a relation between e^a and e^b, e.g. $(molina, occupation, actor)$. We further embed knowledge graph entities and relations into a d-dimensional vector space using

Word2Vec [35] and padding so that e.g. an embedded entity e is represented as a vector $\mathbf{e} \in \mathbb{R}^d$ [36]. The aim is to enhance the generalisability of our models over using raw input graphs, following studies on knowledge graph embeddings by [37, 38, 39, 40], amongst others. Note though that we are not using full-fledged knowledge graph embeddings [37, 38] in this article, we merely aim to embed individual entities and relations in order to generalise across individual facts in our knowledge base. Vectors are subsequently concatenated to form inputs to the generation models. The final training set of knowledge graph triplets is then defined as $\mathbf{x} = (e_0^a, r_0, e_0^b), \ldots, (e_m^a, r_m, e_m^b)$ for a dataset of m samples, where we concatenate individual vectors of entities and relations to serve as input to generation.

3.2. Hierarchical Generation Spaces

Figure 2 shows an example of a partial knowledge graph representing entities from the Wikipedia domain, where nodes represent fields taken from a Wikipedia info box describing a person. We specify a high-level parent concept *person* as a root node whose children are individual Wikipedia entries. Each entry has a subset of semantic fields relevant to the person it describes, where common fields are *birth_date* and *death_date* (if relevant) as well as *occupation*. Other semantic fields are only relevant for particular subsets of people. For example, *position*, *teams_managed* and *clubs* seem particularly relevant to sports personalities, while *office*, *constituency* and *political_party* seem relevant to politicians.

The intuition is that combinations of semantic fields as shown are closely related to the linguistic choices and lexico-syntactic structures used to express them. In other words, we make use of the observation that biographies of sports people share a vocabulary and a set of syntactic constructions, so that a football player's biography reads more similarly to a cricket player's biography than a senior politician's. Earlier work has captured these "clusters" of input-output pairs by learning separate embeddings for semantic fields [6] or by encoding semantic information according to semantic groups [7]. Our approach is to explore the idea of automatically inducing *generation spaces* from data, i.e. clusters of data points (triplets) from knowledge graphs, so as to be able to train generation models for individual sub-spaces (of input-output pairs) of the complete generation domain.

To this end, we define a hierarchy H of natural language generators, where each generator is defined as a 3-tuple:

$$G_i^j, = \langle \mathbf{x}_i^j, \mathbf{y}_i^j, \theta_i^j \rangle, \qquad (1)$$

where $\mathbf{x}_i^j$ is a set of input knowledge graphs $\mathbf{x}_0 \dots \mathbf{x}_n$, $\mathbf{y}_i^j$ is a set of output sequences of words and θ_i^j is a set of weights specific to G_i^j. Indices i and j uniquely identify the model in the hierarchy of generators $H = \{G_0^0 \dots G_p^q\}$, see Figure 1 for an illustration. Note that i and j are used only for identification in the hierarchy and are independent of the time step during learning. While index i denotes a layer in the hierarchy, index j denotes a model within the current layer.

We use a joint embedded vocabulary across our hierarchy of generators and a shared set of initial weights, trained from the whole dataset, to capture the general domain genre and to allow for model transfer across generation spaces. To do this, parent model G_0^0 is trained for the full domain with the primary purpose being to learn a set of initial embeddings and weights that can be passed down to the children as prior knowledge. Models $G_1^0 \dots G_1^m$ have a similar purpose in that they refine weights, and models $G_2^0 \dots G_2^n$ represent sub-domain generators that are used to generate the final output texts from the input graphs. For the actual output generation part in Section 3.3, we make use of our "expert generators" $G_2^0 \dots G_2^n$ only.

To identify sub-spaces of input data that lend themselves as separate generators for a sub-domain (i.e. *generation space*), we use K-Means++ clustering based on the set of partial knowledge graphs identified in Section 3.1, where K can be a different number of clusters for level G_1^m and level G_2^n of the hierarchy, respectively, see experiments in Section 4. Algorithm 1 shows pseudocode for identifying clusters of similar data points that should be combined into a generator. Our algorithm starts with a random initialisation of weights for the parent agent G_0^0. The parent is then trained on a full domain dataset to obtain initial weight distributions that capture the domain on the whole and that can be passed down to child agents. Once training is completed, we apply K-Means++ clustering to the semantic input set (knowledge graph triplets) $\mathbf{x}$ to identify K clusters that behave in semantically similar ways. The number of clusters K

Algorithm 1 Inducing a hierarchy of natural language generators

```
1: function FINDGENERATORS(set of all knowledge graphs triplets $\mathbf{x}$, corresponding output texts $\mathbf{y}$)
2:     $\mathbf{x} =$ numerical encoding of $\mathbf{x}$
3:     $G_0^0 = \langle \mathbf{x}, \mathbf{y}, \theta \rangle$
4:     $G_1^K$ = list []
5:     $G_2^L$ = list []
6:
7:     Train $G_0^0$ to obtain weights $\theta$
8:     $\mathcal{K} \leftarrow$ decompose $\mathbf{x}$ into $K$ clusters
9:     for each cluster $k$ in $\mathcal{K}$ do:
10:        append $\langle \mathbf{x}_1^k, \mathbf{y}_1^k, \theta \rangle$ to $G_1^k$
11:        train $G_1^k$ to obtain $\theta_1^k$ from prior weights $\theta$
12:    end for
13:    for each generator in $G_1^k$ do:
14:        $\mathcal{L} \leftarrow$ decompose $\mathbf{x}_1^k$ into $L$ clusters
15:        for each cluster $l$ in $\mathcal{L}$ do:
16:            append $\langle \mathbf{x}_2^l, \mathbf{y}_2^l, \theta_1^k \rangle$ to $G_2^l$
17:            train $G_2^l$ to obtain $\theta_2^l$ from prior weights $\theta_1^k$
18:        end for
19:    end for
20: end function
```

is established empirically through trial and error, see Section 4 for details. Once a set of clusters has been found, the sub-set of $\mathbf{x}$ belonging to these clusters is paired with their expected outputs $\mathbf{y}$, and each of the new child agents is trained in turn. These generators make up the second layer of our hierarchy, $G_1^0 \dots G_1^m$. Their semantic inputs are in turn used to find further sub-clusters that will, once trained, form the final layer of the hierarchy $G_2^0 \dots G_2^n$.

During generation, we identify the best generator for a new semantic input graph by assigning the input to an existing cluster, i.e. a generator G_i^j, by calculating its Euclidean distance with the cluster.

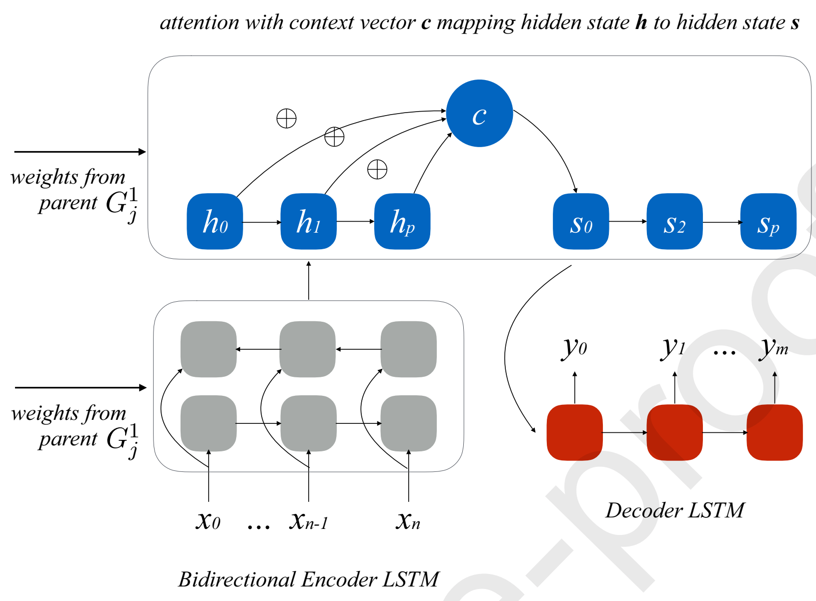
attention with context vector c mapping hidden state h to hidden state s
c
weights from parent G_j^1
h_0
h_1
h_p
s_0
s_2
s_p
y_0
y_1
...
y_m
weights from parent G_j^1
x_0
...
x_{n-1}
x_n
Decoder LSTM
Bidirectional Encoder LSTM

Figure 3: Architecture of the H-Seq2Seq model including a bi-LSTM encoder and LSTM decoder with attention mechanism as in [42].

3.3. Output generation

To implement our generators, we experiment with two alternative learning models, **H-Seq2Seq** which comprises Seq2Seq learning models [1], and **H-Mem2Seq** which comprises Mem2Seq models [41].

H-Seq2Seq. A Seq2Seq model conditions an output sequence of words on a sequence of inputs, i.e. knowledge graph triplets in our case. This is done by learning an increasingly abstract representation of the input captured as the hidden state $\mathbf{h}$ which is found through updates to a non-linear activation function $f(\mathbf{x}_t, \mathbf{h_{t-1}})$ at timestep t. The goal is to minimise the loss between expected and generated outputs ($\mathbf{y}$, $\hat{\mathbf{y}}$, respectively) by for example using categorical cross entropy:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_i y_i \log \hat{y}_i. \tag{2}$$

We opted for an LSTM solution to our recurrent neural network which computes $\mathbf{h}$ under consideration of control gates that manage the loss and retention of information to the current cell state, see [43]. We follow previous work that has demonstrated the benefits of using a bidirectional setup to encode inputs [44]. We will work with an abbreviated notation of the update functions for the forward and backward step:

$$\begin{aligned}\overrightarrow{\mathbf{h}}_t &= LSTM(\overrightarrow{\mathbf{W}}_{xh}\mathbf{x}_t + \overrightarrow{\mathbf{W}}_{hh}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_h),\\ \overleftarrow{\mathbf{h}}_t &= LSTM(\overleftarrow{\mathbf{W}}_{xh}\mathbf{x}_t + \overleftarrow{\mathbf{W}}_{hh}\overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{b}}_h),\end{aligned} \tag{3}$$

where $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias term. We can then compute our final output sequence $\mathbf{y}_t$ as:

$$\mathbf{y}_t = \mathbf{W}_{\overrightarrow{h}y}\overrightarrow{h}_t + \mathbf{W}_{\overleftarrow{h}y}\overleftarrow{\mathbf{h}}_t + \mathbf{b}_y. \tag{4}$$

Finally, we integrate an attention mechanism into our Seq2Seq model that computes each word y_t from the decoder's hidden state $\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_t, \mathbf{c}_t)$, where $\mathbf{c}_t$ is a weighted combination of input states $\mathbf{h}_0 \dots \mathbf{h}_{t-1}$ [42]. See Figure 3 for an illustration.

H-Mem2Seq. The Mem2Seq model combines the notion of a Seq2Seq generator with the idea of a memory network that computes an output sequence $\mathbf{y}$ based on a set of memories that can capture more of the preceding context than is typically available in a Seq2Seq model. For example, in dialogue generation a Mem2Seq model may predict an utterance over an entire dialogue history of user and system turns rather than just the input semantics [45, 41]. For our application to NLG, we represent context $\mathbf{u}$ as a concatenation of the d immediately preceding pairs $\langle \mathbf{x}, \mathbf{y} \rangle$, i.e. knowledge graph triplets $\{u_0^x, \dots, u_d^x\}$ and generated output sequences $\{u_0^y, \dots, u_d^y\}$.

The encoder network is defined as a standard memory network with adjacent weight tying, which uses $\mathbf{u}$ as an input and represents memories as embedded matrices $\mathbf{C}^0, \dots,$ $\mathbf{C}^K$ that map tokens in $\mathbf{u}$ to continuous vectors. The query term $\mathbf{q}$ is also embedded and corresponds in our setting to a single knowledge graph, e.g. $\mathbf{x}_t$ as defined in the previous section. We can then compute a probability vector $\mathbf{p}_i^k$ over memories $\mathbf{C}_i^k$ from $\mathbf{q}^k$:

$$\mathbf{p}_i^k = \text{Softmax}((\mathbf{q}^k)^T \mathbf{C}_i^k), \tag{5}$$

where i is the current memory index and k indicates the current memory hop. The output memory $\mathbf{o}$ is computed as the weighted sum over the transformed inputs $\mathbf{C}_i^K$:

$$\mathbf{o}^k = \sum_i \mathbf{p}_i^k \mathbf{C}_i^K. \tag{6}$$

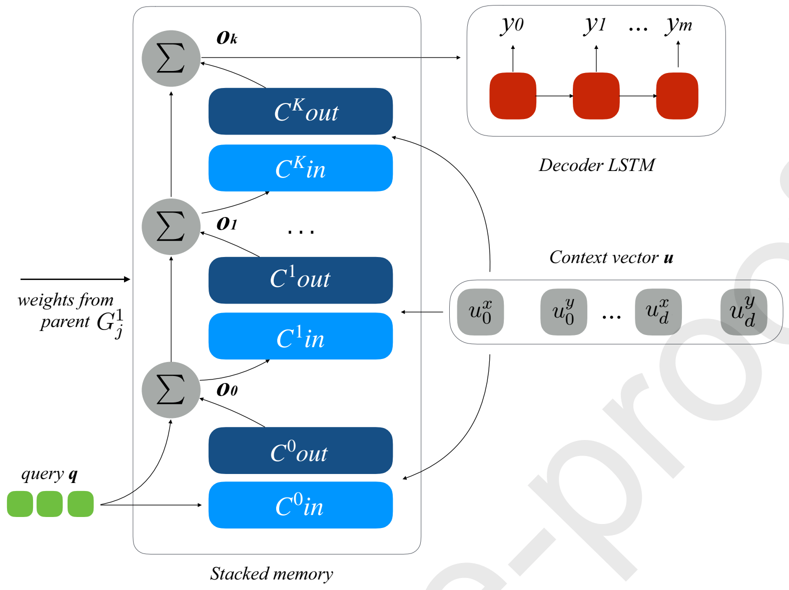

Figure 4: Architecture of the H-Mem2Seq model including a stacked memory network encoder as in [45] and LSTM decoder.

The query $\mathbf{q}$ is updated incrementally through $q^{k+1} = q^k + o^k$. After k memory hops this will produce output vector $\mathbf{o}^k$ which will serve as an input sequence to the decoder.

The decoder computes an output sequence $\mathbf{y}$ as described for the Seq2Seq model above except that updates to hidden state $\mathbf{h}_t$ are conditioned on $\mathbf{C}$ rather than an attention vector $\mathbf{s}_t$ as above:

$$\mathbf{h}_t = LSTM(\mathbf{C}^0(\hat{y}_{t-1}, \mathbf{h}_{t-1})). \tag{7}$$

A probability distribution over words can then be obtained for output sequence $\mathbf{y}$ using greedy or beam search. We follow the implementation of [41] for our Mem2Seq model except that we replace their standard GRU model in the decoder for an LSTM as in the Seq2Seq model above. See Figure 4 for an illustration.

4. Experiments

We present our datasets, baselines and details on experimental setup and hyper-parameters.

	WIKIBIO	WEBNLG
Instances	728,321	25,298
Training split	80%	80%
Validation split	10%	–
Test split	10%	20%
Vocabulary	400k	6,547
Fields	74,689	373

Table 1: Overview of dataset properties.

4.1. Datasets and Data Preparation

We use the following datasets that contain linked data and textual descriptions of the data:

- WEBNLG[2] – maps RDF triplets to text where inputs are of varying complexity containing between one and seven triplets for generation. Outputs are lexico-syntactically varied and include microplanning operations such as sentence aggregation or discourse relations. See [46] for further detail.
- WIKIBIO[3] – maps Wikipedia info boxes to biography paragraphs. This dataset is noisy and sparse as info boxes are not restricted in the fields they can contain, therefore often not leading to a correspondence between info boxes and text. Only about a third of the output text is represented in the info box. See [6] for further detail.

Table 1 shows a comparison of the datasets in terms of their size, vocabulary, number of semantic field types as well as the split used for training, testing and validation. We follow the same split as used in previous work. As can be seen the WIKIBIO dataset is a much larger collection of semantic inputs and texts with sparse vocabulary. WEBNLG is smaller but still challenging due to the smaller number of training examples.

[2]`https://webnlg-challenge.loria.fr/download/`
[3]`https://github.com/DavidGrangier/wikipedia-biography-dataset`

In terms of data preparation of the semantic inputs, the WEBNLG data is represented in the form of RDF triplets so can be directly transferred to our knowledge graph notation from Section 3.1. The WIKIBIO data comes in the form of a Wikipedia info box, matching a semantic field, e.g. *occupation* to a value, e.g. *actor*. We transfer this notation to knowledge graph triplets (e^a, r, e^b) as before, where e^a is the person whose biography is being generated, r is a semantic relation and e^b is its value. For both datasets we follow previous work [1, 11] and delexicalise e^a and e^b during generation to reduce the sparsity associated with specific names, places, dates etc. These values are copied back into a generated output before presentation to a user, e.g., directly from a semantic input representation.

4.2. Experimental Setup and Baselines

We compare the following setups:

- **H-Seq2Seq** is a hierarchy of Seq2Seq models described in Section 3.3. We use 4 layers, 256 hidden units, dropout of 0.2, learning rate of 0.0001, Adam optimisation and train for 2,000 epochs with early stopping (patience=6).
- **H-Mem2Seq** is a hierarchy of Mem2Seq models described in Section 3.3. We use 3 hops, 256 hidden units, dropout of 0.2, learning rate of 0.0001, Adam optimisation and 2,000 training epochs with early stopping (patience=6).
- **Seq2Seq** is a flat Seq2Seq bi-LSTM model with attention, using the same model setup as above. This model corresponds to generating from G_0^0 directly under the Seq2Seq option.
- **Mem2Seq** is a flat Mem2Seq model with an LSTM model for decoding, using the same model setup as above. This model corresponds to generating from G_0^0 directly under the Mem2Seq option.
- **Human** is the human reference data for each dataset. We present this baseline as an upper-bound for the performance of other models.

In addition we compare with two published dataset-specific baselines from previous work that has used the same data:

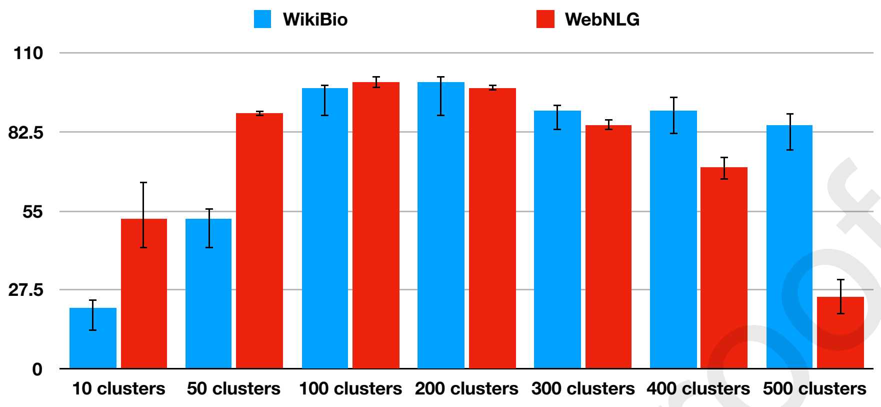

Figure 5: Performance comparison of using different numbers of clusters for training for WIKIBIO (blue) and WEBNLG (red) datasets to determine the best number of clusters for the bottom of our hierarchy (layer G_2^n). Here, the top performance achieved across both datasets is marked as 100%. Performance was measured on the test set.

- **Lebret-2016** use a Seq2Seq LSTM with attention, a conditional neural language model and copy actions to address sparsity [6] for WIKIBIO. We discussed this model in more detail in Section 2.
- **MELBOURNE**[4] is the winning system of the WEBNLG challenge, i.e. using the same test set that we will be using. It is based on a standard encoder-decoder architecture with Bahdanau attention [42], but specfically enriches input sequences before learning. In particular, this involves appending DBpedia types to entities in the input, whenever available, and basing delexicalisation on an n-gram search procedure to ensure that the most accurate delexicalised input sequence is used.

To determine a good number of generators for our hierarchy, we experimented with numbers $[3, 5, 10]$ for level G_n^1 and with $[10, 50, 100, 200, 300, 400, 500]$ clusters for level G_m^2 (generation) of our hierarchy. Based on these experiments, all final experiments will use 3 clusters for G_n^1 and 100 clusters for G_m^2 for both datasets. Figure 5 illustrates a comparison of different numbers of clusters for G_m^2. The top performance achieved across both datasets (in terms of BLEU score) is used as the upper-bound in

[4]`http://webnlg.loria.fr/pages/melbourne_report.pdf`

this case and other configurations are presented in relation to the upper bound.

5. Results

Model	BLEU	ERR	AVE LEN	INF	NAT
H-SEQ2SEQ	51.78 ± 0.17	4.90 ± 1.38	**24.78** ± 4.16	**4.0** (4) ± 1.04	3.27 (3) ± 1.15
SEQ2SEQ	7.24 ± 0.15	**3.33** ± 1.64	14.78 ± 4.45	–	–
H-MEM2SEQ	**65.59** ± 33.80	5.70 ± 2.10	23.25 ± 12.31	3.97 (4) ± 0.98	**3.55** (4) ± 1.05
MEM2SEQ	6.50 ± 12.02	4.94 ± 1.88	12.05 ± 6.30	–	–
LEBRET-16	34.7 ± 0.36	–	–	–	–
HUMAN	–	0.44 ± 0.17	24.96 ± 5.60	4.51 (5) ± 0.68	4.53 (5) ± 0.74

Table 2: Objective and subjective results for WIKIBIO dataset. Numbers are averages with standard deviation. For subjective metrics INF and NAT, median ratings are shown in parentheses.

Model	BLEU	ERR	AVE LEN	INF	NAT
H-SEQ2SEQ	**48.71** ± 0.19	5.40 ± 1.18	**10.38** ± 1.28	**3.57** (4) ± 1.05	**3.97** (4) ± 1.21
SEQ2SEQ	6.12 ± 0.12	5.45 ± 1.24	6.90 ± 1.65	–	–
H-MEM2SEQ	10.23 ± 9.78	**1.69** ± 0.66	5.87 ± 2.51	3.32 (4) ± 1.22	3.54 (4) ± 1.53
MEM2SEQ	4.79 ± 5.92	13.20 ± 4.30	7.48 ± 1.98	–	–
MELBOURNE	45.13 ± –	–	–	–	–
HUMAN	–	3.39 ± 1.70	10.50 ± 1.26	4.16 (4) ± 0.79	4.10 (4) ± 1.06

Table 3: Objective and subjective results for WEBNLG dataset. Numbers are averages with standard deviation. For subjective metrics INF and NAT, median ratings are shown in parentheses.

5.1. Objective Evaluation

Table 2 shows objective results for the WIKIBIO data and Table 3 shows objective results for the WEBNLG dataset, both in terms of BLEU-4, the average length of generated sequences (AVE LEN) and the semantic error in outputs (ERR). The semantic error was computed from the number of semantic slots that are either missing or falsely inserted into the generated output in comparison to the slots found in the input:

$$\text{ERR} = \frac{\text{additional slots} + \text{missing slots}}{\text{total number semantic slots}} \tag{8}$$

BLEU. We can see that in terms of BLEU metrics, the hierarchical versions of both models, **H-Seq2Seq** and **H-Mem2Seq**, clearly outperform their flat counterparts for both datasets by a substantial margin, where **Seq2Seq** or **Mem2Seq** never make it over 10% for either dataset. This is in line with bi-LSTM baseline results reported for the WEBNLG challenge [11]. The most note-worthy difference between the **H-Seq2Seq** and **H-Mem2Seq** models is perhaps the large variability in output quality for **H-Mem2Seq**. While **H-Seq2Seq** performs well for both datasets, 51.78 for WIKIBIO and 48.71 WEBNLG, the **H-Mem2Seq** achieves as much as 65.59 for WIKIBIO but only 10.23 for WEBNLG. The variance in output quality for **H-Mem2Seq** is also clearly visible in the standard deviations for both datasets, ranging between 5.92 to 33.80. This means that while **H-Mem2Seq** achieves near-human BLEU scores in some of our hierarchical sub-domains, it fails to learn any reasonable prediction model for other sub-domains. **H-Seq2Seq** is much more stable in all cases with standard deviations between 0.12 and 0.19, despite never getting close to the human upper-bound for either dataset. For WEBNLG, we find that the second best system overall is MELBOURNE from the WebNLG challenge, achieving an overall BLEU score of 45.13 just after **H-Seq2Seq**. This shows that clean and rich input representations also have a substantial weight towards good outputs as was MELBOURNE's focus.

ERR. In terms of the semantic error (ERR), we find the errors relatively consistently between 3% and 6% for all except **H-Mem2Seq** (1.69%) and **Mem2Seq** (13.20) models for WEBNLG. Interestingly, semantic error rates are not necessarily higher with lower BLEU scores. Manual inspection revealed that in some cases, a model will learn to reproduce a sequence of the semantic input tokens, thus achieving a low error, but not learn to produce an intelligible lexico-syntactic output. It is further worth noting that human data does not achieve an error of 0 either, according to Equation 8. This is likely due to humans in a number of cases paraphrasing semantic content or adding information that is not represented in the input but that they may know through some other means, e.g. general world knowledge. See [6] for a similar observation.

The utterance is ...	rating (1-5)
human-like	5
nearly human-like	4
somewhat unnatural	3
very unnatural	2
completely wrong	1
naturalness	

The utterance contains ...	rating (1-5)
substantial information conveyed	5
some information conveyed	4
states the obvious	3
wrong information conveyed	2
no information conveyed	1
informativeness	

Table 4: On the left: definitions for varying scales of *naturalness*. On the right: definitions for scales of *informativeness*.

AVE LEN. In terms of the average length of generated outputs, **H-Seq2Seq** achieves an average of 24.78, getting closest to the human average of 24.96, followed by **H-Mem2Seq** (23.25) for WikiBio, though notice the high standard deviation again (12.31) for the latter model. For WebNlg, **H-Seq2Seq** achieves an average of 10.38, closest to the human average of 10.50, while **H-Mem2Seq** generates an average length of only 5.87. The latter is likely related to observations relating to other metrics such as the model just learning to list the correct semantic slots without much lexico-syntactic structure and the low BLEU scores achieved.

5.2. Subjective Evaluation

To evaluate the subjective quality of our generated outputs, we conducted a human rating study on Amazon Mechanical Turk (AMT) comparing our systems against each other in terms of subjective metrics on output quality. We focus on the subset of systems that achieved a BLEU score of at least 10% as well as the human upper-bounds. We restrict our subjective evaluation to the better systems based on the rationale that systems below 10% in BLEU scores are clearly sub-par and not sufficiently competitive to assess further. For our evaluation, we recruited 62 human judges from AMT and asked them to assess the *naturalness* (NAT) and *informativeness* (INF) of the generated outputs on a Likert scale of 1-5, where 5 is the best and 1 is the worst. Table 4 shows textual descriptions for the categories that were available to human raters, naturalness is shown on the left and informativeness is shown on the right.

	Semantic input: (Ted Zhanovich Ntirubuza, $birth\_date\_1$, 23 Jan 1995), (TZN, $occupation\_1$, footballer), (TZN, $current\_club\_1$, FC Solaris Moscow)
H-Seq2Seq	Ted Zhanovich Ntirubuza born 23 June 1995 in Voronezh, Russia, is a Russian footballer who currently plays for FC Solyaris Moscow.
H-Mem2Seq	Ted Zhanovich Ntirubuza (born 23 June 1995 in Voronezh, Russia) is a Russian footballer who plays as a defender for FC Solyaris Moscow.
Seq2Seq	Ted Zhanovich Ntirubuza (FC Solyaris Moscow) born 23 June 1995 is a Russian football is a Russian football.
Mem2Seq	Ted Zhanovich Ntirubuza born 1995 is Russian football.
Human	Ted Zhanovich Ntirubuza (born 23 June 1995) is a Russian football player who plays for FC Solyaris Moscow.

Table 5: Examples of generated outputs for WIKIBIO.

We collected overall 4,458 human ratings for a subset of generated outputs that were chosen randomly from the test sets of our two best performing systems per dataset (according to objective metrics) as well as our human (gold standard) upper-bound. Results are shown in Tables 2 and 3 as before in terms of average, median and standard deviation. Results mostly show the **H-Seq2Seq** models being ranked slightly higher than the **H-Mem2Seq** models even though the standard deviations are comparable for both systems. All systems are ranked lower than the human upper-bound, even though the latter is also not rated perfectly by AMT raters.

5.3. Error Analysis of Outputs

Tables 5 and 6 show example outputs for WIKIBIO and WEBNLG, respectively. Noteworthy phenomena are highlighted. For example, in some cases we can see additional information being inserted in an output that is not represented in the input. This is relatively frequent for the human data (see e.g. human baseline in Table 6) but can also be observed for systems (see e.g. **H-Mem2Seq** in Table 5). When systems

	Semantic input: (Bionico, *country*, Mexico), (Bionico, *course*, dessert), (Bionico, *region*, Guadalajara), (Bionico, *ingredient*, granola), (Mexico, *leader*, Enrique Pena Nieto
H-Seq2Seq	Enrique Pena Nieto is the leader of Mexico where Bionico is a food.
H-Mem2Seq	The granola-based dessert Bionico is a food found in Guadalajara Mexico.
Seq2Seq	Bionico is a food found in Jalisco Mexico in Mexico in in in.
Mem2Seq	Enrique Pena Nieto is located in Mexico is the leader.
Human	Enrique Pena Nieto is the leader of Mexico where the dish Bionico can be found in the Jalisco region.

Table 6: Examples of generated outputs for WEBNLG.

insert extra information, we find that this information is not in all cases correct and can correspond to "default" information that is frequent in the data overall rather than specifically relevant to the current data point. For example, for the **H-Seq2Seq** example in Table 5, it is possible that Ted Zhanovich Ntirubuza is a defender but since we do not know this from the semantic input graph, it is also conceivable that the dataset just contains a high number of defenders and the model has learnt to insert this slot when communicating about footballers. This has been observed for flat learning setups too, see e.g. [6]. Apart from this, we find that the non-hierarchical models mostly successfully generate the first part of an utterance but can lead to ungrammaticalities later in an output, particularly repeating words or semantic slots (see **Seq2Seq** example in Table 6) or ordering semantic phrases in a way that would seem unnatural to a human and slightly ungrammatical (see **Mem2Seq** example in Table 6).

5.4. Computational Comparison

To shed further light on the performance of our models, Table 7 provides a comparison in terms of the number of parameters and the time taken to execute a single epoch for our flat and hierarchical models. The computational results (in this section) were obtained with a MacBook Pro (2.7 GHz Intel Core i5) and 8 GB in RAM.

As we can see, the **Mem2Seq** and **H-Mem2Seq** models work with consistently fewer parameters across datasets, leading to faster computation. For example, the **Seq2Seq** model requires 332.4% more parameters than the **Mem2Seq**, while **H-Seq2Seq** requires 67% more parameters than the corresponding **H-Mem2Seq** model. The time differences are substantial in both cases as well.

At the same time, we can observe an advantage – both in terms of model size and computational efficiency – in our hierarchical models over their flat baselines. **H-Mem2Seq** requires 12.69M parameters (on average across datasets), 4.60% of the flat **Mem2Seq** model, and 2.03 seconds to execute a single in epoch (again across datasets) in comparison to **Mem2Seq**'s 78.05 hours, a reduction of over 99%. Similarly, **H-Seq2Seq** needs 38.16M parameters on average and 3.65 seconds, 4.5% of **Seq2Seq**'s parameters and less than 1% of its time. This in conjunction with the objective and subjective results presented above further supports the argument of a divide-and-conquer approach to natural language generation in large and noisy domains.

As outlined in Section 3.2, before training our model/s we need to find a set of clusters to structure the data space and induce a hierarchy of generation agents. This step is performed once and takes about 220 minutes (3 hours and 40 minutes) to complete. During testing, we need to allocate each new test instance to a cluster before being able to generate an output. This step takes on average 49 seconds to complete (per example) due to the cost of Euclidean distance calculations. In future we want to explore ways to shorten this time and make cluster allocation more efficient. For the time being, however, we consider that the advantages of our method outweigh the drawbacks, especially in a text generation domain, where speed and responsiveness are arguably less crucial than e.g. in a highly interactive scenario.

6. Conclusion and Future Work

We have presented a novel approach to neural NLG from knowledge graphs that applies a divide-and-conquer approach to automatically induce a hierarchy of genera-

	Model	Ave. parameters (in million)	Min	Max	Ave. tpe	Min	Max
WikiBio	H-Seq2Seq	74.53 ± 52.23	9.78	210.58	7.07 sec ± 120.2 sec	1.73 ms	6.43 min
	Seq2Seq	1,642.30	–	–	10.13 days	–	–
	H-Mem2Seq	24.23 ± 21.48	2.59	181.80	4.064 sec ± 27.14 sec	1.083 ms	4.01 min
	Mem2Seq	494.081	–	–	6.52 days	–	–
WebNLG	H-Seq2Seq	1.79 ± 416,393	1.37	4.19	0.23 sec ± 0.28 sec	13.18 ms	2.01
	Seq2Seq	62.46	–	–	30.65 sec	–	–
	H-Mem2Seq	1.15 ± 426,953	638,304	3.20	1.08 ms ± 2.59 ms	0.079 ms	22.76 ms
	Mem2Seq	18.79	–	–	1.97 sec	–	–

Table 7: Computational comparison of flat and hierarchical models in terms of the numbers of parameters Ave. parameters (in million) (average, minimal and maximal observed) and the time takes to train a single epoch Ave. time per epoch (tpe) (average, minimal and maximal observed). For flat models single numbers are reported.

tors from a dataset based on the similarity of their embedded inputs and K-means++ clustering. Each generator is trained as an "expert" for a sub-set of domain input-output pairs with shared knowledge propagated from parents to children. We presented experiments with two variants of our hierarchy, **H-Seq2Seq** and **H-Mem2Seq**, based on sequence-to-sequence and memory-to-sequence models respectively. We find that while **H-Mem2Seq** models can outperform **H-Seq2Seq** in some cases, the latter are more stable and reliable in their performance across datasets and evaluation metrics, and therefore make a safer choice in most cases. Overall our experiments show that hierarchical generators consistently outperform their flat counterparts by BLEU scores of up to 30%, including competitive baselines and previous work.

Future work can address the following:

- In this article we have applied the same setup to all learning models in our hierarchy but it is possible that different learning models and/or hyperparameter configurations are optimal for different parts of the "generation space". Future work can investigate combinations of different architectures and hyper-parameter configurations within the same hierarchy to further tailor towards individual requirements of data sub-portions and find an optimal learning setup.

- We expect that the general idea of partioning a dataset according to semantic and lexical similarity and thus presenting maximally relevant examples to a learning model during training is transferable and beneficial in other domains, such as caption generation or dialogue. Future work can establish this empirically.
- Finally, we have focused on a comparison of sequence-to-sequence and memory-to-sequence architectures in this articles as two of the most representatives forms of attention currently used. Future work can extend this comparison to transformer architectures [5] and observe any resulting features.

Acknowledgements

We gratefully acknowledge funding from Diffbot to carry out this project. We also acknowledge the VIPER high-performance computing facility of the University of Hull and its support team, and thanks to Andy Hancock for helping us with access to the DGX-1 for most of our experiments.

References

[1] T.-H. Wen, M. Gašić, N. Mrkšić, P.-H. Su, D. Vandyke, S. Young, 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.

[2] H. Mei, M. Bansal, M. Walter, What to talk about and how? selective generation using lstms with coarse-to-fine alignment, in: Proceedings of the 16th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL), 2016.

[3] O. Dusek, F. Jurcicek, 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.

[4] J. Juraska, P. Karagiannis, K. Bowden, M. Walker, A deep ensemble model with slot alignment for sequence-to-sequence natural language generation, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, 2018, pp. 152–162.
URL `http://aclweb.org/anthology/N18-1014`

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5998–6008.

[6] R. Lebret, D. Grangier, M. Auli, Neural text generation from structured data with application to the biography domain, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 1203–1213.

[7] P. Vougiouklis, H. Elsahar, L.-A. Kaffee, C. Gravier, F. Laforest, J. Hare, Elena, Neural Wikipedian: Generating Textual Summaries from Knowledge Base Triples, Journal of Web Semantics 52-53 (2018) 1–15.

[8] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, J. Pineau, Building end-to-end dialogue systems using generative hierarchical neural network models, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16), 2016.
URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11957`

[9] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, J.-Y. Nie, A hierarchical recurrent encoder-decoder for generative context-aware query suggestion, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, 2015, pp. 553–562.

[10] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, H.-W. Hon, Unified language model pre-training for natural language understanding

and generation, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 13063–13075.
URL http://papers.nips.cc/paper/9464-unified-language-model-pre-training-for-natural-language-understanding-pdf

[11] C. Gardent, A. Shimorina, S. Narayana, L. Perez-Beltrachini, The WebNLG Challenge: Generating Text from RDF Data, in: Proceedings of the International Natural Language Generation Conference (INLG), Santiago de Compostella, Spain, 2017.

[12] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems (NIPS) 27, 2014, pp. 3104–3112.

[13] I. Sutskever, J. Martens, G. Hinton, Generating text with recurrent neural networks, in: L. Getoor, T. Scheffer (Eds.), Proceedings of the 28th International Conference on Machine Learning (ICML-11), ICML '11, ACM, 2011, pp. 1017–1024.

[14] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, S. Khudanpur, Recurrent neural network based language model, in: INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010, 2010, pp. 1045–1048.

[15] N. Dethlefs, Domain Transfer for Deep Natural Language Generation from Abstract Meaning Representations, IEEE Computational Intelligence Magazine: Special Issue on Natural Language Generation with Computational Intelligence.

[16] I. Konstas, S. Iyer, M. Yatskar, Y. Choi, L. Zettlemoyer, Neural amr: Sequence-to-sequence models for parsing and generation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2017, pp. 146–157.

doi:10.18653/v1/P17-1014.
URL http://aclweb.org/anthology/P17-1014

[17] D. Beck, G. Haffari, T. Cohn, Graph-to-sequence learning using gated graph neural networks, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2018, pp. 273–283.
URL http://aclweb.org/anthology/P18-1026

[18] L. Song, Y. Zhang, Z. Wang, D. Gildea, A graph-to-sequence model for amr-to-text generation, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2018, pp. 1616–1626.
URL http://aclweb.org/anthology/P18-1150

[19] F. Nie, J. Wang, J.-G. Yao, R. Pan, C.-Y. Lin, Operations guided neural networks for high fidelity data-to-text generation, CoRR abs/1809.02735.
URL http://arxiv.org/abs/1809.02735

[20] S. Iyer, I. Konstas, A. Cheung, L. Zettlemoyer, Summarizing source code using a neural attention model, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 2073–2083.

[21] P. Yin, G. Neubig, A syntactic neural model for general-purpose code generation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 440–450.

[22] W. Ling, P. Blunsom, E. Grefenstette, K. M. Hermann, T. Kočiský, F. Wang, A. Senior, Latent predictor networks for code generation, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 599–609.

[23] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, N. Schneider, Abstract Meaning Representation for Sembanking, in: Proc. of the Linguistic Annotation Workshop (LAW), Sofia, Bulgaria, 2013.

[24] I. Konstas, S. Iyer, M. Yatskar, Y. Choi, L. Zettlemoyer, Neural amr: Sequence-to-sequence models for parsing and generation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2017, pp. 146–157. doi:10.18653/v1/P17-1014.
URL http://www.aclweb.org/anthology/P17-1014

[25] N. Dethlefs, A. Turner, Deep text generation - Using hierarchical decomposition to mitigate the effect of rare data points, in: Proceedings of Language, Data and Knowledge (LDK), Galway, Ireland, 2017.

[26] J. Chatterjee, N. Dethlefs, A Dual Transformer Model for Intelligent Decision Support for Maintenance of Wind Turbines, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), Glasgow, Scotland, 2020.

[27] S. Gehrmann, F. Dai, H. Elder, A. Rush, End-to-end content and plan selection for data-to-text generation, in: Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, 2018, pp. 46–56.

[28] S. Semeniuta, A. Severyn, E. Barth, A hybrid convolutional variational autoencoder for text generation, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2017, pp. 627–637.
URL http://aclweb.org/anthology/D17-1066

[29] M. Freitag, S. Roy, Unsupervised natural language generation with denoising autoencoders, CoRR abs/1804.07899.

[30] S. Wiseman, S. M. Shieber, A. M. Rush, Learning neural templates for text generation, CoRR abs/1808.10122.
URL http://arxiv.org/abs/1808.10122

[31] A. Chisholm, W. Radford, B. Hachey, Learning to generate one-sentence biographies from wikidata, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 633–642.

[32] S. Merity, C. Xiong, J. Bradbury, R. Socher, Pointer sentinel mixture models, in: arXiv:1609.07843, CoRR, 2016.

[33] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, J. B. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, 2016, pp. 3682–3690.

[34] H. Cuayáhuitl, D. Lee, S. Ryu, Y. Cho, S. Choi, S. Indurthi, S. Yu, H. Choi, I. Hwang, J. Kim, Ensemble-Based Deep Reinforcement Learning for Chatbots, Neurocomputing in press.

[35] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, CoRR abs/1301.3781.
URL http://arxiv.org/abs/1301.3781

[36] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: Theory and practice, IEEE Trans. Knowl. Data Eng. 15 (3) (2003) 515–528. doi:10.1109/TKDE.2003.1198387.
URL https://doi.org/10.1109/TKDE.2003.1198387

[37] H. Xiao, M. Huang, X. Zhu, Transg : A generative model for knowledge graph embedding, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 2316–2325.

[38] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Lake Tahoe, Nevada, 2013, pp. 2787–2795.

[39] A. Bordes, J. Weston, R. Collobert, Y. Bengio, Learning structured embeddings of knowledge bases, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI'11, AAAI Press, San Francisco, California, 2011, pp. 301–306.

[40] A. Bordes, N. Usunier, S. Chopra, J. Weston, Large-scale simple question answering with memory networks, in: arXiv:1506.02075, CoRR, 2015.

[41] A. Madotto, C.-S. Wu, P. Fung, Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2018, pp. 1468–1478. URL `http://aclweb.org/anthology/P18-1136`

[42] D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, in: Proc. of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 2015.

[43] A. Graves, Generating Sequences With Recurrent Neural Networks, CoRR abs/1308.0850.
URL `http://arxiv.org/abs/1308.0850`

[44] M. Schuster, K. Paliwal, Bidirectional recurrent neural networks, Trans. Sig. Proc. 45 (11) (1997) 2673–2681.

[45] S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, End-to-end memory networks, in: NIPS, 2015.

[46] C. Gardent, A. Shimorina, S. Narayan, L. Perez-Beltrachini, Creating training corpora for micro-planners, in: Proceedings of the 55th Annual Meeting of the

Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017.