Bootstrapping Generators from Noisy Data

Citation for published version:

Digital Object Identifier (DOI):
10.18653/v1/N18-1137

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Abstract

A core step in statistical data-to-text generation concerns learning correspondences between structured data representations (e.g., facts in a database) and associated texts. In this paper we aim to bootstrap generators from large scale datasets where the data (e.g., DBPedia facts) and related texts (e.g., Wikipedia abstracts) are loosely aligned. We tackle this challenging task by introducing a special-purpose content selection mechanism.\footnote{Our code and data are available at https://github.com/EdinburghNLP/wikigen.} We use multi-instance learning to automatically discover correspondences between data and text pairs and show how these can be used to enhance the content signal while training an encoder-decoder architecture. Experimental results demonstrate that models trained with content-specific objectives improve upon a vanilla encoder-decoder which solely relies on soft attention.

1 Introduction

A core step in statistical data-to-text generation concerns learning correspondences between structured data representations (e.g., facts in a database) and paired texts (Barzilay and Lapata, 2005; Kim and Mooney, 2010; Liang et al., 2009). These correspondences describe how data representations are expressed in natural language (content realisation) but also indicate which subset of the data is verbalised in the text (content selection).

Although content selection is traditionally performed by domain experts, recent advances in generation using neural networks (Bahdanau et al., 2015; Ranzato et al., 2016) have led to the use of large scale datasets containing loosely related data and text pairs. A prime example are online data sources like DBPedia (Auer et al., 2007) and Wikipedia and their associated texts which are often independently edited. Another example are sports databases and related textual resources. Wiseman et al. (2017) recently define a generation task relating statistics of basketball games with commentaries and a blog written by fans.

In this paper, we focus on short text generation from such loosely aligned data-text resources. We work with the biographical subset of the DBPedia and Wikipedia resources where the data corresponds to DBPedia facts and texts are Wikipedia abstracts about people. Figure 1 shows an example for the film-maker Robert Flaherty, the Wikipedia infobox, and the corresponding abstract. We wish to bootstrap a data-to-text generator that learns to verbalise properties about an entity from a loosely related example text. Given the set of properties in Figure (1a) and the related text in Figure (1b), we want to learn verbalisations for those properties that are mentioned in the text and produce a short description like the one in Figure (1c).

In common with previous work (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017) our model draws on insights from neural machine translation (Bahdanau et al., 2015; Sutskever et al., 2014) using an encoder-decoder architecture as its backbone. Lebret et al. (2016) introduce the task of generating biographies from Wikipedia data, however they focus on single sentence generation. We generalize the task to multi-sentence text, and highlight the limitations of the standard attention mechanism which is often used as a proxy for content selection. When exposed to sub-sequences that do not correspond to any facts in the input, the soft attention mechanism will still try to justify the sequence and somehow distribute the attention weights over the input representation (Ghader and Monz, 2017). The decoder will still memorise high frequency sub-sequences in spite of these not being supported by any facts in the input.

We propose to alleviate these shortcom-
Previous attempts to exploit loosely aligned data and text corpora have mostly focused on extracting verbalisation spans for data units. Most approaches work in two stages: initially, data units are aligned with sentences from related corpora using some heuristics and subsequently extra content is discarded in order to retain only text spans verbalising the data. Belz and Kow (2010) obtain verbalisation spans using a measure of strength of association between data units and words, Walter et al. (2013) extract textual patterns from paths in dependency trees while Mrabet et al. (2016) rely on crowd-sourcing. Perez-Beltrachini and Gardent (2016) learn shared representations for data units and sentences reduced to subject-predicate-object triples with the aim of extracting verbalisations for knowledge base properties. Our work takes a step further, we not only induce data-to-text alignments but also learn generators that produce short texts verbalising a set of facts.

Our work is closest to recent neural network models which learn generators from independently edited data and text resources. Most previous work (Lebret et al., 2016; Chisholm et al., 2017; Sha et al., 2017; Liu et al., 2017) targets the generation of single sentence biographies from Wikipedia infoboxes, while Wiseman et al. (2017) generate game summary documents from a database of basketball games where the input is always the same set of table fields. In contrast, in our scenario, the input data varies from one entity (e.g., athlete) to another (e.g., scientist) and properties might be present or not due to data incompleteness. Moreover, our generator is enhanced with a content selection mechanism based on multi-instance learning. MIL-based techniques have been previously applied to a variety of problems including image retrieval (Maron and Ratan, 1998; Zhang et al., 2002), object detection (Carbonetto et al., 2008; Cour et al., 2011), text classification (Andrews and Hofmann, 2004), image captioning (Wu et al., 2015; Karpathy and Fei-Fei, 2015), paraphrase detection (Xu et al., 2014), and information extraction (Hoffmann et al., 2011). The application of MIL to content selection is novel to our knowledge.

We show how to incorporate content selection into encoder-decoder architectures following training regimes based on multi-task learning and reinforcement learning. Multi-task learning aims to improve a main task by incorporating joint learning of one or more related auxiliary tasks. It has been applied with success to a variety of sequence-prediction tasks focus-
We consider loosely coupled data and text pairs where the data component is a set \(\mathcal{P}\) of property-values \(\{p_1 : v_1, \ldots, p_{|\mathcal{P}|} : v_{|\mathcal{P}|}\}\) and the related text \(\mathcal{T}\) is a sequence of sentences \((s_1, \ldots, s_{|\mathcal{T}|})\). We define a mention span \(\tau\) as a (possibly discontinuous) subsequence of \(\mathcal{T}\) containing one or several words that verbalise one or more property-value from \(\mathcal{P}\). For instance, in Figure 1, the mention span “married to Frances H. Flaherty” verbalises the property-value \(\{\text{Spouse}(s) : \text{Frances Johnson Hubbard}\}\).

In traditional supervised data to text generation tasks, data units (e.g., \(p_i : v_j\) in our particular setting) are either covered by some mention span \(\tau_j\) or do not have any mention span at all in \(\mathcal{T}\). The latter is a case of content selection where the generator will learn which properties to ignore when generating text from such data. In this work, we consider text components which are independently edited, and will unavoidably contain unaligned spans, i.e., text segments which do not correspond to any property-value in \(\mathcal{P}\). The phrase “from 1914” in the text in Figure (1b) is such an example. Similarly, the last sentence, talks about Frances’ awards and nominations and this information is not supported by the properties either.

Our model checks content in both directions; it identifies which properties have a corresponding text span (data selection) and also foregrounds (un)aligned text spans (text selection). This knowledge is then used to discourage the generator from producing text not supported by facts in the property set \(\mathcal{P}\). We view a property set \(\mathcal{P}\) and its loosely coupled text \(\mathcal{T}\) as a coarse level, imperfect alignment. From this alignment signal, we want to discover a set of finer grained alignments indicating which mention spans in \(\mathcal{T}\) align to which properties in \(\mathcal{P}\). For each pair \((\mathcal{P}, \mathcal{T})\), we learn an alignment set \(\mathcal{A}(\mathcal{P}, \mathcal{T})\) which contains property-value word pairs. For example, for the properties spouse and died in Figure 1, we would like to derive the alignments in Table 1.

We formulate the task of discovering finer-grained word alignments as a multi-instance learning problem (Keeler and Rumelhart, 1992). We assume that words from the text are positive labels for some property-values but we do not know which ones. For each data-text pair \((\mathcal{P}, \mathcal{T})\), we derive \(|\mathcal{T}|\) pairs of the form \((\mathcal{P}, s)\) where \(|\mathcal{T}|\) is the number of sentences in \(\mathcal{T}\). We encode property sets \(\mathcal{P}\) and sentences \(s\) into a common multi-modal \(h\)-dimensional embedding space. While doing this, we discover finer-grained alignments between words and property-values. The intuition is that by learning a high similarity score for a property set \(\mathcal{P}\) and sentence pair \(s\), we will also learn the contribution of individual elements (i.e., words and property-values) to the overall similarity score. We will then use this individual contribution as a measure of word and property-value alignment. More concretely, we assume the pair is aligned (or unaligned) if this individual score is above (or below) a given threshold. Across examples like the one shown in Figure (1a-b), we expect the model to learn an alignment between the text span “married to Frances H. Flaherty” and the property-value \(\{\text{spouse} : \text{Frances Johnson Hubbard}\}\).

In what follows we describe how we encode \((\mathcal{P}, s)\) pairs and define the similarity function.

**Property Set Encoder** As there is no fixed order among the property-value pairs \(p : v\) in \(\mathcal{P}\), we individually encode each one of them. Furthermore, both properties \(p\) and values \(v\) may consist of short phrases. For instance, the property cause of death and value cerebral thrombosis in Figure 1. We

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spouse</td>
<td>Frances Johnson Hubbard</td>
</tr>
<tr>
<td>to</td>
<td>spouse : Frances Johnson Hubbard</td>
</tr>
<tr>
<td>Frances</td>
<td>spouse : Frances Johnson Hubbard</td>
</tr>
<tr>
<td>Flaherty</td>
<td>spouse : Frances Johnson Hubbard</td>
</tr>
<tr>
<td>death</td>
<td>died : july 23, 1951</td>
</tr>
<tr>
<td>in</td>
<td>died : july 23, 1951</td>
</tr>
<tr>
<td>1951</td>
<td>died : july 23, 1951</td>
</tr>
</tbody>
</table>

Table 1: Example of word-property alignments for the Wikipedia abstract and facts in Figure 1.
therefore consider property-value pairs as concatenated sequences \( pv \) and use a bidirectional Long Short-Term Memory Network (LSTM; Hochreiter and Schmidhuber, 1997) network for their encoding. Note that the same network is used for all pairs. Each property-value pair is encoded into a vector representation:

\[
p_i = \text{biLSTM}_{\text{denc}}(pv_i)
\]  

which is the output of the recurrent network at the final time step. We use addition to combine the forward and backward outputs and generate encoding \( \{p_1, \ldots, p_{|P|}\} \) for \( P \).

**Sentence Encoder** We also use a biLSTM to obtain a representation for the sentence \( s = w_1, \ldots, w_{|s|} \). Each word \( w_t \) is represented by the output of the forward and backward networks at time step \( t \). A word at position \( t \) is represented by the concatenation of the forward and backward outputs of the networks at time step \( t \):

\[
w_t = \text{biLSTM}_{\text{senc}}(w_t)
\]  

and each sentence is encoded as a sequence of vectors \( \{w_1, \ldots, w_{|s|}\} \).

**Alignment Objective** Our learning objective seeks to maximise the similarity score between property set \( P \) and a sentence \( s \) (Karpathy and Fei-Fei, 2015). This similarity score is in turn defined on top of the similarity scores among property-values in \( P \) and words in \( s \). Equation (3) defines this similarity function using the dot product. The function seeks to align each word to the best scoring property-value:

\[
S_{ps} = \sum_{i=1}^{|s|} \max_{p_i \in \{1, \ldots, |P|\}} p_i \cdot w_t
\]  

Equation (4) defines our objective which encourages related properties \( P \) and sentences \( s \) to have higher similarity than other \( P' \neq P \) and \( s' \neq s \):

\[
L_{\text{CA}} = \max(0, S_{ps} - S_{p's} + 1) + \max(0, S_{p's} - S_{ps} + 1)
\]  

### 4 Generator Training

In this section we describe the base generation architecture and explain two alternative ways of using the alignments to guide the training of the model. One approach follows multi-task training where the generator learns to output a sequence of words but also to predict alignment labels for each word. The second approach relies on reinforcement learning for adjusting the probability distribution of word sequences learnt by a standard word prediction training algorithm.

#### 4.1 Encoder-Decoder Base Generator

We follow a standard attention based encoder-decoder architecture for our generator (Bahdanau et al., 2015; Luong et al., 2015). Given a set of properties \( X \) as input, the model learns to predict an output word sequence \( Y \) which is a verbalisation of (part of) the input. More precisely, the generation of sequence \( Y \) is conditioned on input \( X \):

\[
P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|y_{1:t-1}, X)
\]

The encoder module constitutes an intermediate representation of the input. For this, we use the property-set encoder described in Section 3 which outputs vector representations \( \{p_1, \ldots, p_{|X|}\} \) for a set of property-value pairs. The decoder uses an LSTM and a soft attention mechanism (Luong et al., 2015) to generate one word \( y_t \) at a time conditioned on the previous output words and a context vector \( c_t \) dynamically created:

\[
P(y_{t+1}|y_{1:t}, X) = \text{softmax}(g(h_t, c_t))
\]  

where \( g(\cdot) \) is a neural network with one hidden layer parameterised by \( W_o \in \mathbb{R}^{|V| \times d}, |V| \) is the output vocabulary size and \( d \) the hidden unit dimension, over \( h_t \) and \( c_t \) composed as follows:

\[
g(h_t, c_t) = W_o \tanh(W_c [c_t; h_t])
\]  

where \( W_c \in \mathbb{R}^{d \times 2d} \). \( h_t \) is the hidden state of the LSTM decoder which summarises \( y_{1:t} \):

\[
h_t = \text{LSTM}(y_t, h_{t-1})
\]  

The dynamic context vector \( c_t \) is the weighted sum of the hidden states of the input property set (Equation (9)); and the weights \( \alpha_{it} \) are determined by a dot product attention mechanism:

\[
\alpha_{it} = \frac{\exp(h_t \cdot p_i)}{\sum_d \exp(h_t \cdot p_d)}
\]  

We initialise the decoder with the averaged sum of the encoded input representations (Vinyals et al., 2016). The model is trained to optimize negative log likelihood:

\[
L_{\text{nLL}} = -\sum_{t=1}^{|Y|} \log P(y_t|y_{1:t-1}, X)
\]
We extend this architecture to multi-sentence texts in a way similar to Wiseman et al. (2017). We view the abstract as a single sequence, i.e., all sentences are concatenated. When training, we cut the abstracts in blocks of equal size and perform forward backward iterations for each block (this includes the back-propagation through the encoder). From one block iteration to the next, we initialise the decoder with the last state of the previous block. The block size is a hyperparameter tuned experimentally on the development set.

4.2 Predicting Alignment Labels

The generation of the output sequence is conditioned on the previous words and the input. However, when certain sequences are very common, the language modelling conditional probability will prevail over the input conditioning. For instance, the phrase *from 1914* in our running example is very common in contexts that talk about periods of marriage or club membership, and as a result, the language model will output this phrase often, even in cases where there are no supporting facts in the input. The intuition behind multi-task training (Caruana, 1993) is that it will smooth the probabilities of frequent sequences when trying to simultaneously predict alignment labels.

Using the set of alignments obtained by our content selection model, we associate each word in the training data with a binary label $a_t$ indicating whether it aligns with some property in the input set. Our auxiliary task is to predict $a_t$ given the sequence of previously predicted words and input $X$:

$$P(a_{t+1}|y_{1:t},X) = \text{sigmoid}(g'(h_t,c_t))$$  \hspace{1cm} (12)

$$g'(h_t,c_t) = v_a \cdot \tanh(W_e[c_t,h_t])$$  \hspace{1cm} (13)


where $v_a \in \mathbb{R}^d$ and the other operands are as defined in Equation (7). We optimise the following auxiliary objective function:

$$L_{\text{align}} = - \sum_{t=1}^{|Y|} \log P(a_t|y_{1:t-1},X)$$  \hspace{1cm} (14)

and the combined multi-task objective is the weighted sum of both word prediction and alignment prediction losses:

$$L_{\text{MTL}} = \lambda L_{\text{NLL}} + (1 - \lambda) L_{\text{align}}$$  \hspace{1cm} (15)

where $\lambda$ controls how much model training will focus on each task. As we will explain in Section 5, we can anneal this value during training in favour of one objective or the other.

4.3 Reinforcement Learning Training

Although the multi-task approach aims to smooth the target distribution, the training process is still driven by the imperfect target text. In other words, at each time step $t$ the algorithm feeds the previous word $w_{t-1}$ of the target text and evaluates the prediction against the target $w_t$.

Alternatively, we propose a training approach based on reinforcement learning (Williams 1992) which allows us to define an objective function that does not fully rely on the target text but rather on a revised version of it. In our case, the set of alignments obtained by our content selection model provides a revision for the target text. The advantages of reinforcement learning are twofold: (a) it allows to exploit additional task-specific knowledge (Zhang and Lapata, 2017) during training, and (b) enables the exploration of other word sequences through sampling. Our setting differs from previous applications of RL (Ranzato et al., 2016; Zhang and Lapata, 2017) in that the reward function is not computed on the target text but rather on its alignments with the input.

The encoder-decoder model is viewed as an agent whose action space is defined by the set of words in the target vocabulary. At each time step, the encoder-decoder takes action $\hat{y}_t$ with policy $P_{\pi}(\hat{y}_t|\hat{y}_{1:t-1},X)$ defined by the probability in Equation (6). The agent terminates when it emits the End Of Sequence (EOS) token, at which point the sequence of all actions taken yields the output sequence $\hat{y} = (\hat{y}_1, \cdots, \hat{y}_{|\hat{Y}|})$. This sequence in our task is a short text describing the properties of a given entity. After producing the sequence of actions $\hat{Y}$, the agent receives a reward $r(\hat{Y})$ and the policy is updated according to this reward.

**Reward Function** We define the reward function $r(\hat{Y})$ on the alignment set $A(X,Y)$. If the output action sequence $\hat{y}$ is precise with respect to the set of alignments $A(X,Y)$, the agent will receive a high reward. Concretely, we define $r(\hat{Y})$ as follows:

$$r(\hat{Y}) = \gamma^{\text{pr}} r^{\text{pr}}(\hat{Y})$$  \hspace{1cm} (16)

where $\gamma^{\text{pr}}$ adjusts the reward value $r^{\text{pr}}$ which is the unigram precision of the predicted sequence $\hat{Y}$ and the set of words in $A(X,Y)$.

**Training Algorithm** We use the REINFORCE algorithm (Williams, 1992) to learn an agent that maximises the reward function. As this is a gradient descent method, the training loss of a sequence
is defined as the negative expected reward:

$$L_{RL} = -\mathbb{E}_{(\hat{y}_1, \ldots, \hat{y}_T)} \sim P_{\pi}(\cdot|X) [r(\hat{y}_1, \ldots, \hat{y}_T)]$$

where $P_{\pi}$ is the agent’s policy, i.e., the word distribution produced by the encoder-decoder model (Equation (6)) and $r(\cdot)$ is the reward function as defined in Equation (16). The gradient of $L_{RL}$ is given by:

$$\nabla L_{RL} \approx \sum_{t=1}^{T} \nabla \log P_{\pi}(\hat{y}_t | \hat{y}_{t-1}, X) [r(\hat{y}_1, \ldots, \hat{y}_T) - b_t]$$

where $b_t$ is a baseline linear regression model used to reduce the variance of the gradients during training. $b_t$ predicts the future reward and is trained by minimizing mean squared error. The input to this predictor is the agent hidden state $h_t$, however we do not back-propagate the error to $h_t$. We refer the interested reader to Williams (1992) and Ranzato et al. (2016) for more details.

**Document Level Curriculum Learning** Rather than starting from a state given by a random policy, we initialise the agent with a policy learnt by pre-training with the negative log-likelihood objective (Ranzato et al., 2016; Zhang and Lapata, 2017). The reinforcement learning objective is applied gradually in combination with the log-likelihood objective on each target block subsequence. Recall from Section 4.1 that our document is segmented into blocks of equal size during training which we denote as $\text{MAX BLOCK}$. When training begins, only the last $\tilde{O}$ tokens are predicted by the agent while for the first $\text{MAX BLOCK} - \tilde{O}$ we still use the negative log-likelihood objective. The number of tokens $\tilde{O}$ predicted by the agent is incremented by $\tilde{O}$ units every 2 epochs. We set $\tilde{O} = 3$ and the training ends when $\text{MAX BLOCK} - \tilde{O} = 0$. Since we evaluate the model’s predictions at the block level, the reward function is also evaluated at the block level.

5 Experimental Setup

**Data** We evaluated our model on a dataset collated from WikIBio (Lebret et al., 2016), a corpus of 728,321 biography articles (their first paragraph) and their infoboxes sampled from the English Wikipedia. We adapted the original dataset in three ways. Firstly, we make use of the entire abstract rather than first sentence. Secondly, we reduced the dataset to examples with a rich set of properties and multi-sentential text. We eliminated examples with less than six property-value pairs and abstracts consisting of one sentence. We also placed a minimum restriction of 23 words in the length of the abstract. We considered abstracts up to a maximum of 12 sentences and property sets with a maximum of 50 property-value pairs. Finally, we associated each abstract with the set of DBPedia properties $p : v$ corresponding to the abstract’s main entity. As entity classification is available in DBPedia for most entities, we concatenate class information $c$ (whenever available) with the property value, i.e., $p : vc$. In Figure 1, the property value $\text{spouse} : \text{Frances H. Flaherty}$ is extended with class information from the DBPedia ontology to $\text{spouse} : \text{Frances H. Flaherty Person}$.

**Pre-processing** Numeric date formats were converted to a surface form with month names. Numerical expressions were delexicalised using different tokens created with the property name and position of the delexicalised token on the value sequence. For instance, given the property-value for birth date in Figure (1a), the first sentence in the abstract (Figure (1b)) becomes “ Robert Joseph Flaherty, (February $DLX_{birth\ date}=2, DLX_{birth\ date}=4 - July \ldots$ ”. Years and numbers in the text not found in the values of the property set were replaced with tokens YEAR and NUMERIC.\(^2\) In a second phase, when creating the input and output vocabularies, $\mathcal{V}^i$ and $\mathcal{V}^o$ respectively, we delexicalised words $w$ which were absent from the output vocabulary but were attested in the input vocabulary. Again, we created tokens based on the property name and the position of the word in the value sequence. Words not in $\mathcal{V}^o$ or $\mathcal{V}^o$ were replaced with the symbol UNK. Vocabulary sizes were limited to $|\mathcal{V}^i| = 50k$ and $|\mathcal{V}^o| = 50k$ for the alignment model and $|\mathcal{V}^o| = 20k$ for the generator. We discarded examples where the text contained more than three UNKs (for the content aligner) and five UNKs (for the generator); or more than two UNKs in the property-value (for generation). Finally, we added the empty relation to the property sets.

Table 2 summarises the dataset statistics for the generator. We report the number of abstracts in the dataset (size), the average number of sentences and tokens in the abstracts, and the average number of properties and sentence length in tokens.
Table 2: Dataset statistics.

<table>
<thead>
<tr>
<th>generation</th>
<th>train</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>165,324</td>
<td>23,399</td>
<td>23,162</td>
</tr>
<tr>
<td>sentences</td>
<td>3.51±1.99</td>
<td>3.46±1.94</td>
<td>3.22±1.72</td>
</tr>
<tr>
<td>tokens</td>
<td>74.13±43.72</td>
<td>72.85±42.54</td>
<td>66.81±38.16</td>
</tr>
<tr>
<td>properties</td>
<td>14.97±8.82</td>
<td>14.96±8.85</td>
<td>21.6±9.97</td>
</tr>
<tr>
<td>sent.len</td>
<td>21.06±8.87</td>
<td>21.03±8.85</td>
<td>20.77±8.74</td>
</tr>
</tbody>
</table>

For the content aligner (cf. Section 3), each sentence constitutes a training instance, and as a result the sizes of the train and development sets are 796,446 and 153,096, respectively.

### Training Configuration

We adjusted all models’ hyperparameters according to their performance on the development set. The encoders for both content selection and generation models were initialised with GloVe (Pennington et al., 2014) pre-trained vectors. The input and hidden unit dimension was set to 200 for content selection and 100 for generation. In all models, we used encoder biLSTMs and decoder LSTM (regularised with a dropout rate of 0.3 (Zaremba et al., 2014)) with one layer. Content selection and generation models (base encoder-decoder and MTL) were trained for 20 epochs with the ADAM optimiser (Kingma and Ba, 2014) using a learning rate of 0.001. The reinforcement learning model was initialised with the base encoder-decoder model and trained for 35 additional epochs with stochastic gradient descent and a fixed learning rate of 0.001. Block sizes were set to 40 (base), 60 (MTL) and 50 (RL). Weights for the MTL objective were also tuned experimentally; we set $\lambda = 0.1$ for the first four epochs (training focuses on alignment prediction) and switched to $\lambda = 0.9$ for the remaining epochs.

### Content Alignment

We optimized content alignment on the development set against manual alignments. Specifically, two annotators aligned 132 sentences to their infoboxes. We used the Yawat annotation tool (Germann, 2008) and followed the alignment guidelines (and evaluation metrics) used in Cohn et al. (2008). The inter-annotator agreement using macro-averaged f-score was 0.72 (we treated one annotator as the reference and the other one as hypothetical system output).

Alignment sets were extracted from the model’s output (cf. Section 3) by optimizing the threshold $\text{avg}(\text{sim}) + a \times \text{std}(\text{sim})$ where $\text{sim}$ denotes the similarity between the set of property values and words, and $a$ is empirically set to 0.75; $\text{avg}$ and $\text{std}$ are the mean and standard deviation of $\text{sim}$ scores across the development set. Each word was aligned to a property-value if their similarity exceeded a threshold of 0.22. Our best content alignment model (Content-Aligner) obtained an f-score of 0.36 on the development set.

We also compared our Content-Aligner against a baseline based on pre-trained word embeddings (EmbeddingsBL). For each pair $(s, \mathcal{P})$, we computed the dot product between words in $s$ and properties in $\mathcal{P}$ (properties were represented by the the averaged sum of their words’ vectors). Words were aligned to property-values if their similarity exceeded a threshold of 0.4. EmbeddingsBL obtained an f-score of 0.057 against the manual alignments. Finally, we compared the performance of the Content-Aligner at the level of property set $\mathcal{P}$ and sentence $s$ similarity by comparing the average ranking position of correct pairs among 14 distractors, namely rank@15. The Content-Aligner obtained a rank of 1.31, while the EmbeddingsBL model had a rank of 7.99 (lower is better).

### 6 Results

We compared the performance of an encoder-decoder model trained with the standard negative log-likelihood method (ED), against a model trained with multi-task learning (EDMTL) and reinforcement learning (EDRL). We also included a template baseline system (Templ) in our evaluation experiments.

The template generator used hand-written rules to realise property-value pairs. As an approximation for content selection, we obtained the 50 more frequent property names from the training set and manually defined content ordering rules with the following criteria. We ordered personal life properties (e.g., `birth_date` or `occupation`) based on their most common order of mention in the Wikipedia abstracts. Profession dependent properties (e.g., `position` or `genre`), were assigned an equal ordering but posterior to the personal properties. We manually lexicalised properties into single sentence templates to be concatenated to produce the final text. The template for the property `position` and example verbalisation for the property-value `position: defender` of the entity `zanetti` are “[NAME] played as [POSITION].” and “Zanetti played as defender.” respectively.

### Automatic Evaluation

Table 3 shows the results of automatic evaluation using BLEU-4.
put multi-sentence text. Using the first sentence in the Wikipedia abstract as reference, we obtained a score of 37.29% (ED), 38.42% (ED$_{MTL}$) and 38.1% (ED$_{RL}$) which compare favourably with their best performing model (34.7%±0.36).

**Human-Based Evaluation** We further examined differences among systems in a human-based evaluation study. Using AMT, we elicited 3 judgments for the same 200 infobox-abstract pairs we used in the abstract revision study. We compared the output of the templates, the three neural generators and also included one of the human edited abstracts as a gold standard (reference). For each test case, we showed crowdworkers the Wikipedia infobox and five short texts in random order. The annotators were asked to rank each of the texts according to the following criteria: (1) Is the text faithful to the content of the table? and (2) Is the text overall comprehensible and fluent? Ties were allowed only when texts were identical strings. Table 5 presents examples of the texts (and properties) crowdworkers saw.

Table 4 shows, proportionally, how often crowdworkers ranked each system, first, second, and so on. Unsurprisingly, the human authored gold text is considered best (and ranked first 47% of the time). ED$_{MTL}$ is mostly ranked second and third best, followed closely by ED$_{RL}$. The vanilla encoder-decoder system ED is mostly forth and Templ is fifth. As shown in the last column of the table (Rank), the ranking of ED$_{MTL}$ is overall slightly better than ED$_{RL}$. We further converted the ranks to ratings on a scale of 1 to 5 (assigning ratings 5...1 to rank placements 1...5). This allowed us to perform Analysis of Variance (ANOVA) which revealed a reliable effect of system type. Post-hoc Tukey tests showed that all systems were significantly worse than RevAbs and significantly better than Templ (p < 0.05). ED$_{MTL}$ is not significantly better than ED$_{RL}$ but is significantly (p < 0.05) different from ED.

**Discussion** The texts generated by ED$_{RL}$ are shorter compared to the other two neural systems...
which might affect BLEU-4 scores and also the ratings provided by the annotators. As shown in Table 5 (entity dorsey burnette), ED\(_{MTL}\) drops information pertaining to dates or chooses to just verbalise birth place information. In some cases, this is preferable to hallucinating incorrect facts; however, in other cases outputs with more information are rated more favourably. Overall, ED\(_{MTL}\) seems to be more detail oriented and faithful to the facts included in the infobox (see dorsey burnette, aaron moores, or kirill moryganov). The template system manages in some specific configurations to verbalise appropriate facts (indrani bose), however, it often fails to verbalise infrequent properties (aaron moores) or focuses on properties which are very frequent in the knowledge base but are rarely found in the abstracts (kirill moryganov).

7 Conclusions

In this paper we focused on the task of bootstrapping generators from large-scale datasets consisting of DBpedia facts and related Wikipedia biog-
References


