The Rare Word Issue in Natural Language Generation: A Character-Based Solution
Giovanni Bonetta, Marco Roberti, Rossella Cancelliere, Patrick Gallinari

To cite this version:
Giovanni Bonetta, Marco Roberti, Rossella Cancelliere, Patrick Gallinari. The Rare Word Issue in Natural Language Generation: A Character-Based Solution. Informatics, MDPI, 2021, 8 (1), pp.20. 10.3390/informatics8010020. hal-03184301

HAL Id: hal-03184301
https://hal.sorbonne-universite.fr/hal-03184301
Submitted on 29 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
The Rare Word Issue in Natural Language Generation: A Character-Based Solution

Giovanni Bonetta 1, Marco Roberti 1, Rossella Cancelliere 1,* and Patrick Gallinari 2,3

Abstract: In this paper, we analyze the problem of generating fluent English utterances from tabular data, focusing on the development of a sequence-to-sequence neural model which shows two major features: the ability to read and generate character-wise, and the ability to switch between generating and copying characters from the input: an essential feature when inputs contain rare words like proper names, telephone numbers, or foreign words. Working with characters instead of words is a challenge that can bring problems such as increasing the difficulty of the training phase and a bigger error probability during inference. Nevertheless, our work shows that these issues can be solved and efforts are repaid by the creation of a fully end-to-end system, whose inputs and outputs are not constrained to be part of a predefined vocabulary, like in word-based models. Furthermore, our copying technique is integrated with an innovative shift mechanism, which enhances the ability to produce outputs directly from inputs. We assess performance on the E2E dataset, the benchmark used for the E2E NLG challenge, and on a modified version of it, created to highlight the rare word copying capabilities of our model. The results demonstrate clear improvements over the baseline and promising performance compared to recent techniques in the literature.

Keywords: data-to-text generation; deep learning; sequence-to-sequence models; natural language processing

1. Introduction

Natural Language Generation (NLG) is the research domain that focuses on automatically generating narratives and reports in fluent, well-structured, and rich natural language text, in order to describe, summarize, or explain input data [1,2]. It includes, among others, the fields of machine translation, automatic summarization, simplification of complex texts, generation of paraphrases, and data journalism.

In the last decade, a paradigm shift occurred: neural networks and deep learning-based methods have increasingly been used as building blocks in NLG algorithms, to obtain completely end-to-end outcomes, i.e., outputs generated without non-neural pre-processing or post-processing [3,4]. As deep learning is data-driven by definition, and it is typically used in an end-to-end mode, the availability of big data makes it possible to obtain new systems which shift from symbolic to data-driven methods, and from modular to comprehensive design. This shift has the major benefit of obtaining architectures that are intrinsically more general and directly applicable to very different domains [5].

This paper involves the application of deep Recurrent Neural Networks (RNNs) in particular to Data-to-Text (DTT) generation, a subfield of computational linguistics and natural language generation which aims at transcribing structured data into natural language descriptions [6]. RNN architectures are among the most used architectures for text processing because the input can be treated sequentially by reading inputs token
by token while updating network recurrent states. Some of the most interesting DTT applications are soccer and weather reports, summaries of patient information in clinical contexts, and robo-journalism.

Neural methods for DTT have shown the ability to produce fluent texts conditioned on input data in several domains [7,8], without relying on heavy manual work from field experts. They successfully combined ideas from machine translation, such as encoder–decoder architectures [9] and attention mechanisms [10,11], and from extractive summarization, such as the copy mechanism [12,13], often explicitly modeling the key-value structure of the input table [14–19]. However, a common issue to deal with is that most of the commonly used models share the word-by-word representation of data in both input sequences and generated utterances; such schemes cannot be effective without a special, non-neural delexicalization phase that handles rare or unknown words, such as proper names, telephone numbers, or foreign words [20].

Various recent studies tried to solve this problem: Gu et al. [12] present CopyNet, a word-based technique that can integrate output generation with a copying mechanism that can choose portions of the input sequence and include them in the final sentence. Similarly, in the word-based Pointer-Generator Network [13], a soft switch determines whether the next output token is generated or copied from the input, re-using the attention distribution. Such techniques, albeit conceived for words, can be adapted to the character copying task, leading to more robust and effective models.

One of the very first attempts to model natural language via a character-level mechanism is described by Sutskever et al. [21]. According to this paper, a simple variant of the vanilla recurrent neural network can generate well-formed sentences after being trained based on sequences of characters. A representative case of character-based systems is Goyal et al. [22], but this model incorporates prior knowledge in the form of a finite-state automaton to prevent “the generation of non-words and the hallucination of named entities”.

The character-based approach developed in our work is characterized by the ability to switch between generating a character and copying it when required. This makes it possible to naturally handle rare words using the copy mechanism for carrying data from the original raw input directly to output. This feature is very hardly obtained by word-based architectures, which have to learn every single word-to-word self-mapping from scratch. Moreover, during the training phase, we also exploit an innovative shift mechanism that helps the model focusing on the word to be copied, character by character.

Our technique allows us to deal with a significantly smaller vocabulary, reduced to ∼100 symbols, i.e., the printable ASCII characters, obtaining the desirable side effect of reducing the computational effort, something that remains a bottleneck of deep learning systems; furthermore, a more general, domain-free technique is obtained this way.

In Section 2, we briefly review the background and present our model. Section 3 describes the datasets and the metrics chosen to evaluate performance, together with some implementation details. Section 4 is devoted to the achieved results, subsequently discussed in Section 5. Finally, in Section 6, we draw some conclusions and outline future work.

2. Materials and Methods

Figure 1 shows the encoder–decoder architecture with an attention mechanism introduced by Bahdanau et al. [10]. It builds upon the original encoder–decoder model, introduced by Cho et al. [9], and based on two RNNs. One RNN encodes a sequence of symbols into a representation vector (encoder RNN, left part of Figure 1), and the other one decodes the representation into another sequence of symbols (decoder RNN, right part of Figure 1). The sequence of symbols can be a phrase, part of a sentence, a sentence, or a longer linguistic unit [23].
Figure 1. Encoder–decoder with attention model.

Usually, the encoder is made of one or more stacked bi-directional RNNs (represented by double arrows on the left side of Figure 1) while the decoder is made of single-directional RNNs. Both are formed by Gated Recurrent Units (GRUs), the RNNs introduced by Cho et al. [9], to alleviate the well-known vanishing (or exploding) gradient problem. In fact, when dealing with longer texts, RNNs could run into several gradient issues, as the gradient is calculated through a long chain of recurrence.

Bahdanau et al. [10] successively added the now standard attention mechanism to this model, shown in the upper side of Figure 1. As can be seen, the encoder outputs a list of annotations $h_j$, one for each input embedding $x_j$; the decoder reads a context vector

$$\chi_i = \sum_{j=1}^{T_x} a_{ij} h_j,$$

built from these $T_x$ annotations.

The sum weights $a_{ij}$ are computed via the attention mechanism:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

where $e_{ij}$ is an alignment model, and $e_{ij} = \text{att}(s_{i-1}, h_j)$, which evaluates the matching between the $j$th input and the output produced in the $i$th time instant. It is based on the decoder’s previous output $s_{i-1}$ and on the $j$th encoder’s annotation, and is parameterized as a single hidden layer feed-forward neural network, jointly trained with the whole model. We can also reformulate $a_{ij}$ as

$$a_{ij} = [\text{softmax}(e_i)]_j, \quad j = 1, \ldots, T_x$$

where $e_i$ is the vector whose elements are the alignment model $e_{ij}$, which evaluates, in a given temporal instant $i$, the matching with every character $j$ of the input sequence.

In our work, we build upon this model, including the major feature of a character-based copy mechanism. This improvement results in a model that never produces non-words or hallucinations.

2.1. Copy Mechanism

The copy mechanism included in our system is inspired by the pointer-generator network [13], a word-based model that hybridizes Bahdanau et al. [10]’s model and a pointer network [24]. This combination allows for both word generation from a fixed
vocabulary and words copying via directly pointing to an input position, taking advantage of a soft switch between those two operating modes. To adapt this framework to our character-based model, we define two important probability distributions: the character distribution $\Pi_{\text{char}}$ and the attention distribution $\Pi_{\text{att}}$. $\Pi_{\text{char}}$ represents the probability of sampling a given token at time $i$, and is computed as

$$\Pi_{\text{char}}^i = \text{softmax}(o_i), \quad o_i = V \cdot [\chi_i, s_i] + b \tag{4}$$

where $o_i$ contains the unnormalized log probabilities over the alphabet at time $i$, computed from the decoder’s output state $s_i$ and the context vector $\chi_i$; $V$ and $b$ are parameters learned during the training phase. That is, $o_i$ is just the result of a linear projection from $[\chi_i, s_i]$ to a vector with the same dimension as the vocabulary, and each vector’s element is the GRU cell’s estimate of how likely is to pick the corresponding character as the next output token.

$\Pi_{\text{att}}$ is the distribution created by the attention mechanism over the input tokens: we define it as coinciding with $a_{ij}$, introduced in Equation (2):

$$\Pi_{\text{att}}^i = a_{ij} \tag{5}$$

$\Pi_{\text{att}}^i$ can be represented in a graphical form, known as the attention matrix: this meaningful diagram makes it easier to understand how the model is handling a certain input sequence. Some examples of attention matrices are shown in Section 5.

In Bahdanau et al. [10]’s model, the next output token is sampled just from the vocabulary distribution $\Pi_{\text{char}}^i$, while $\Pi_{\text{att}}^i$ is only used to weight the input annotations and create the context vector $\chi_i$.

We built upon this model by adding the capability to learn when to sample from $\Pi_{\text{char}}^i$ for generating a text item, or from $\Pi_{\text{att}}^i$, for copying the actual input item. This is made possible by the definition of a generation probability $p^{t}_{\text{gen}} \in [0, 1]$ computed as:

$$p^{t}_{\text{gen}} = \sigma(W_y \cdot y_{i-1} + W_s \cdot s_i + W_p \cdot p^{t-1}_{\text{gen}} + W_c \cdot \chi_i), \quad \text{where } y_{i-1} \text{ is the last output character’s embedding, } s_i \text{ is the current decoder cell state, } p^{t-1}_{\text{gen}} \text{ is the last computed value of } p^{t}_{\text{gen}}, \text{ and } \chi_i \text{ is the current context vector. } W_y, W_s, W_c, \text{ and } W_p \text{ are parameters learned during training and } \sigma \text{ is the sigmoid function.} \tag{6}$$

The resulting final probability distribution is the sum of $\Pi_{\text{char}}^i$ and $\Pi_{\text{att}}^i$, weighted by $p^{t}_{\text{gen}}$:

$$\Pi^i(c) = p^{t}_{\text{gen}} \cdot \Pi_{\text{char}}^i(c) + (1-p^{t}_{\text{gen}}) \sum_{j|y_j=c} \Pi_{\text{att}}^j(c), \tag{7}$$

where $c$ is a specific character and the $j$ index scrolls through all the occurrences of $c$ in the input sequence. $p^{t}_{\text{gen}}$ acts as a soft switch because when it is close to 1, $\Pi^i(c) \simeq \Pi_{\text{char}}^i(c)$: our model generates the output as if it was a standard encoder–decoder with attention. Conversely, when $p^{t}_{\text{gen}}$ is close to 0, $\Pi^i(c) \simeq \sum_{j|y_j=c} \Pi_{\text{att}}^j(c)$, and a copying step is carried out.

The key difference from the word-based pointer-generator network [13] is that in our model, $p^{t-1}_{\text{gen}}$ contributes to the determination of $p^{t}_{\text{gen}}$ (see equation). We introduced this dependency because in a character-based model, it is desirable that, at inference time, the switch maintains the same value—predictably near 1 or 0—for a fair number of time steps, to correctly complete the word; knowledge of the last choice helps this behavior. Conversely, in a word-based model, a single copying time step, when required, is typically enough.

We also propose a new formulation of $\Pi^i(c)$ which helps the model to learn how to copy characters:

$$\Pi^i(c) = p^{t}_{\text{gen}} \cdot \Pi_{\text{char}}^i(c) + (1-p^{t}_{\text{gen}}) \sum_{j|y_j=c} \Pi_{\text{att}}^j(c) \tag{8}$$

Here, $\Pi_{\text{att}}^j = \Pi_{\text{att}}^{j-1}$, i.e., the attention distribution shifted one step on the right with respect to the considered input position $j$. This is done because for the model, it is easier
to learn how to center the attention distribution on just a few characters; for instance, the space (’ ’) and the opening square bracket (’[’), which appear as word separators in the record strings, are frequently the prefix of a word that has to be copied. Once this is achieved, the attention distribution is translated one step on the right, over the first letter to copy. Figure 2 shows the convenience of this approach.

Figure 2. An example of the shift trick.

3. Experimental Setup

3.1. Datasets

We assess our model’s performance using two datasets: E2E, which is well-known and frequently used in the literature [25–30], and E2E-NY, which we specifically created to highlight the copying capabilities of our model.

The E2E dataset [31] was used as a benchmark for the E2E Challenge, organized by the Heriot-Watt University in 2017. It is a crowdsourced collection with 42,061/4672/4693 instances for train/dev/test, respectively. It is conceived for training data-driven, end-to-end NLG models in the restaurant domain. Table 1 shows a typical E2E data sample.

E2E has 8.1 reference sentences on average, for every Meaning Representation (MR); each MR is in turn composed of a list of key-value pairs. The ontology consists of eight attributes of different types.

Table 1. An E2E data instance. The meaning representation appears in the dataset once for each reference sentence.

<table>
<thead>
<tr>
<th>Meaning Representation</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>name[The Wrestlers],</td>
<td>Indian food meets coffee shop at The Wrestlers located in the city centre near Raja Indian Cuisine. This shop is family friendly and priced at less than 20 pounds.</td>
</tr>
<tr>
<td>eatType[coffe shop],</td>
<td>Near Raja Indian Cuisine, The Wrestlers provides the atmosphere of a coffee shop with Indian food. At less than 20 pounds, it provides a family friendly setting for its customers right in the city centre.</td>
</tr>
<tr>
<td>food[Indian]</td>
<td>The Wrestlers is a coffee shop providing Indian food in the less than L20 price range. It is located in the city centre. It is near Raja Indian Cuisine.</td>
</tr>
<tr>
<td>priceRange[less than L20]</td>
<td></td>
</tr>
<tr>
<td>area[city centre]</td>
<td></td>
</tr>
<tr>
<td>familyFriendly[yes]</td>
<td></td>
</tr>
<tr>
<td>near[Raja Indian Cuisine]</td>
<td></td>
</tr>
</tbody>
</table>

During our experiments, we noticed that the values contained in the E2E dataset are a little naive in terms of variability. In other words, a slot like name, which could virtually include a very broad range of possible values, contains only 19 different restaurant names. Moreover, the test set always contains values which are also present in the training set.
Consequently, we created E2E-NY, an augmented version of the E2E dataset, replacing all the values in the name slot with New York restaurant names, as contained in the Entree dataset presented in Burke et al. [32]. Consequently, values that belong to the training set are not found in the development set or in the test set, ensuring the absence of generation bias in the copy mechanism.

All the data were converted to ASCII prior to use, so that the model’s vocabulary is independent of the input.

3.2. Metrics

Performance was assessed according to five different metrics [6]: BLEU, NIST, METEOR, ROUGE_L, and CIDEr. These are the same metrics used in the E2E Challenge. Here is a brief description of each one:

- **BLEU** [33]: It is a precision-based metric that computes the n-gram overlap between the reference and the hypothesis. In particular, BLUE is the ratio of the number of overlapping n-grams over the total number of n-grams in the hypothesis;
- **NIST** [34]: It is a variant of BLEU which gives more credit to rare n-gram and less credit to common ones;
- **METEOR** [35]: It tries to overcome the fact that BLEU does not take recall into account and it only allows exact n-gram matching. Hence, METEOR uses the F-measure and a relaxed matching criteria;
- **ROUGE_L** [36]: It is based on a variation of the F-measure where the precision and recall are computed using the length of the longest common subsequence between hypothesis and reference;
- **CIDEr** [37]: It weighs each hypothesis’ n-gram based on its frequency in the reference set and in the entire corpus. The underlying idea is that frequent dataset’s n-grams are less likely to be informative/relevant.

3.3. Baseline and Competitors

In order to show that our proposed Encoder–Decoder model with Attention, Copy, and Shift (hereafter, ED+ACS) represents an effective improvement with respect to previous work, we compare it to the following models:

- **ED+A**: a character-based Encoder–Decoder model with Attention [10]; a standard baseline in the literature [13,22,38];
- **Qader et al.** [25]’s model: a word-based encoder–decoder with attention;
- **Puzikov and Gurevych** [29]’s model: a non-neural system that was ranked as the best template-based model in the E2E Challenge [39];
- **Dusek and Jurcicek** [40]’s model: the strong word-based baseline of the E2E challenge [39]. Its pipeline consists of a delexicalizer, a neural encoder–decoder system which outputs a syntax tree using beam search with re-ranking, a surface realizer, and a relexicalizer.

3.4. Implementation Details

We developed our system using the PyTorch framework, release 0.4 (https://pytorch.org/, accessed on 22 March 2021).

The training was performed via Teacher Forcing [41], aiming at maximizing the log-likelihood between output utterances and the target ones. We optimized this loss using the Adam [42] algorithm with learning rate cosine annealing [43].

As RNNs are typically subject to the exploding gradient problem, gradient norm clipping [44] was applied. Three-fold cross-validation was used to find the optimal hyperparameters and training settings values, as shown in Table 2. In this configuration, EDA+ACS has 5,940,960 trainable parameters.
Table 2. Model hyperparameters and training settings used in our experiments.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>32</td>
</tr>
<tr>
<td>GRU hidden size</td>
<td>300</td>
</tr>
<tr>
<td>No. of recurrent layers</td>
<td>3</td>
</tr>
<tr>
<td>Attention size</td>
<td>128</td>
</tr>
<tr>
<td>Learning rate</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>$\beta_1; \beta_2$</td>
<td>0.9; 0.999</td>
</tr>
<tr>
<td>$T_{\text{max}}; \eta_{\text{min}}$ (annealing)</td>
<td>50,000; 0</td>
</tr>
<tr>
<td>Max gradient norm</td>
<td>5</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td>No. of epochs</td>
<td>32</td>
</tr>
</tbody>
</table>

We use the evaluation script (https://github.com/tuetschek/e2e-metrics, accessed on 22 March 2021) provided by the E2E NLG Challenge organizers, which compares the models’ output utterances with target ones. We provide results on E2E and E2ENY test sets. Code and data are available at https://github.com/marco-roberti/char-dtt-rareword (accessed on 22 March 2021).

Training and inference were performed on 24GB NVIDIA GPUs (TITAN RTX and Quadro P6000). The training time was in the order of magnitude of $\sim 10$ h, depending on the hardware. Generation at inference occurred in real time, i.e., roughly 2 minutes for the 4693 test instances.

4. Results

Table 3 compares the performance from baselines and our model on the E2E dataset (results come from the proposed methods’ original papers, with the exception of ED+A, which we implemented).

In order to assess the statistical significance of our results, we trained and tested our models five times and validated them via Student’s $t$-test with a significance level of 99.5%. This showed that ED+ACS was consistently better than ED+A by at least 5%. Our method also performed better than Puzikov and Gurevych [29] BLEU, NIST, and CIDEr, while the values for METEOR and ROUGE_L were statistically equal. This fact can be explained by the fact that METEOR and ROUGE_L are based on the F-measure, which takes into account both the precision and the recall, and high recall is easily achieved by template-based systems as the work of Puzikov and Gurevych [29]. Statistical equality also applies with respect to the metrics of Qader et al. [25].

Dusek and Jurčícek [40] obtained better results for three metrics out of five, i.e., NIST, ROUGE_L, and CIDEr; however, this is not a neural end-to-end method, and needs the additional handcrafted pre- and post-processing phases described in Section 3.3.

Differences among the various models were even more interesting when the E2E-NY dataset was used for comparison, as shown in Table 4. The Student’s $t$-test results at a significance level of 99.5% demonstrated that:

Table 3. Performance comparison on the E2E test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>NIST</th>
<th>METEOR</th>
<th>ROUGE_L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED+A</td>
<td>0.5704</td>
<td>7.8060</td>
<td>0.3895</td>
<td>0.6283</td>
<td>1.5877</td>
</tr>
<tr>
<td>Qader et al. [25]</td>
<td>0.655</td>
<td>—</td>
<td>0.450</td>
<td>0.673</td>
<td>—</td>
</tr>
<tr>
<td>Puzikov and Gurevych [29]</td>
<td>0.5657</td>
<td>7.4544</td>
<td>0.4529</td>
<td>0.6614</td>
<td>1.8206</td>
</tr>
<tr>
<td>Dusek and Jurčícek [40]</td>
<td>0.6593</td>
<td>8.6094</td>
<td>0.4483</td>
<td>0.6850</td>
<td>2.2338</td>
</tr>
<tr>
<td>ED+ACS (our model)</td>
<td>0.6400</td>
<td>8.3467</td>
<td>0.4463</td>
<td>0.6680</td>
<td>2.0264</td>
</tr>
</tbody>
</table>
1. ED+ACS was always better than ED+A by more than 17% according to all metrics, achieving as much as a 190% improvement on the CIDEr value;

2. ED+ACS obtained better results than those presented by Qader et al. [25] and Puzikov and Gurevych [29] with respect to all metrics, except METEOR in the latter method;

3. ED+ACS demonstrated statistically equivalent results to those presented by Dusek and Jurcícek [40].

Table 4. Performance comparison on the E2E-NY test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>NIST</th>
<th>METEOR</th>
<th>ROUGE_L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED+A</td>
<td>0.5151</td>
<td>7.1702</td>
<td>0.3617</td>
<td>0.5608</td>
<td>0.8505</td>
</tr>
<tr>
<td>Qader et al. [25]</td>
<td>0.5092</td>
<td>7.2798</td>
<td>0.3756</td>
<td>0.5413</td>
<td>0.8768</td>
</tr>
<tr>
<td>Puzikov and Gurevych [29]</td>
<td>0.5606</td>
<td>7.7671</td>
<td>0.4535</td>
<td>0.6608</td>
<td>2.3787</td>
</tr>
<tr>
<td>Dusek and Jurcícek [40]</td>
<td>0.6517</td>
<td>8.8043</td>
<td>0.4421</td>
<td>0.6749</td>
<td>2.7136</td>
</tr>
<tr>
<td>ED+ACS (our model)</td>
<td>0.6482</td>
<td>8.6563</td>
<td>0.4521</td>
<td>0.6770</td>
<td>2.7346</td>
</tr>
</tbody>
</table>

Notice that the scores of the competing models did not drop below a certain threshold because, even if new names were not correctly reproduced, values occurring in other fields of the generated sentences were generally still correct. We hypothesize that their performances would be even worse on datasets containing unseen values in other fields as well (e.g., food, near).

5. Discussion

Table 5 shows three examples of output sentences generated when the same MR (from the E2E-NY test set) was presented to all models.

The first example consists of a short MR including three key-value pairs. Our model is not the only one that successfully verbalized the whole information; however, ED+ACS tried to express it in a lexically richer way than the models of Puzikov and Gurevych [29] and Dusek and Jurcícek [40]. This came at the cost of the repetition of the word “pub”. In contrast, both ED+A and Qader et al. [25]’s model failed to express the pub’s name, falling back to similar training names.

In the second example, we noted that the ED+A and Qader et al. [25]’s model were not able to reproduce the restaurant name “Brother’s Bar-B-Q” in the output, while all other models did so. In addition, in Dusek and Jurcícek [40]’s model, the price range was completely skipped. Only Puzikov and Gurevych [29]’ model and our model generated a complete and credible sentence relying on all key-value pairs.

The third table row shows an example in which our model generated poorly with respect to Dusek and Jurcícek [40] and Puzikov and Gurevych [29]’s models. It repeated “city centre” twice, and did not consider the “family friendly” value. The issue of repeating part of sentences is very common among neural recurrent models, and is also present in the ED+A output. Again, both ED+A and Qader et al. [25]’s model failed to reproduce the name.

In summary, ED+A and Qader et al. [25]’s approaches did not seem to be able to deal with the rare words in the E2E-NY test set. They continuously failed in realizing the name field, which contains values that were not seen in the training phase. In practice, both models could “borrow” values from other fields, use similar values from the training set, or simply avoid any reference to the restaurant names. In all cases, this results in weird, or even syntactically incorrect, sentences. The template-based model proposed by Puzikov and Gurevych [29] always leads to the realization of the whole input by design. However, its sentences always follow the same structure, resulting in more boring and less human-like sentences. The qualitative analysis confirmed what the metrics suggest: the performances of ED+ACS and Dusek and Jurcícek [40]’s were substantially equivalent from the point of view of generation capabilities. Nevertheless, we stress the fact that ED+ACS, differently from Dusek and Jurcícek [40]’s model, is a fully end-to-end system.
that does not need any pre-processing or post-processing phase, and that it is inherently more general thanks to its character-based design.

Table 5. Meaning representations, references, and model outputs on various E2E-NY test set examples.

<table>
<thead>
<tr>
<th>MR</th>
<th>Model</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>name[Zoe], eatType[pub], near[The Rice Boat]</td>
<td>Reference</td>
<td>Near The Rice Boat you can visit pub called Zoe.</td>
</tr>
<tr>
<td></td>
<td>ED+A</td>
<td>Zona Rosa is a pub near The Rice Boat.</td>
</tr>
<tr>
<td></td>
<td>Qader et al. [25]</td>
<td>Near The Rice Boat is a pub called the The Black Sheep which is a great place to eat.</td>
</tr>
<tr>
<td></td>
<td>Puzikov and Gurevych [29]</td>
<td>Zoe is a pub located near The Rice Boat.</td>
</tr>
<tr>
<td></td>
<td>Dusek and Jurčícek [40]</td>
<td>Zoe is a pub near The Rice Boat.</td>
</tr>
<tr>
<td></td>
<td>ED+ACS (our model)</td>
<td>Zoe is a pub that provides pub food items and is located near The Rice Boat.</td>
</tr>
<tr>
<td>name[Brother’s Bar-B-Q], eatType[pub], food[French], priceRange[moderate], customer rating[1 out of 5], area[riverside], familyFriendly[yes], near[Crowne Plaza Hotel]</td>
<td>Reference</td>
<td>A family friendly pub, Brother’s Bar-B-Q, has a moderately expensive menu that is located next to Crowne Plaza Hotel on the river.</td>
</tr>
<tr>
<td></td>
<td>ED+A</td>
<td>Brother Jimmy’s BBQ is a pub located in the riverside area. It serves French food in the moderate price range.</td>
</tr>
<tr>
<td></td>
<td>Qader et al. [25]</td>
<td>The French pub called Burger King is located near Crown Plaza Hotel in the riverside area. It is kid friendly and has a moderate price range.</td>
</tr>
<tr>
<td></td>
<td>Puzikov and Gurevych [29]</td>
<td>Brother’s Bar-B-Q is a family-friendly pub which serves French food in the moderate price range. It has a customer rating of 1 out of 5 and is located in the riverside area, near Crowne Plaza Hotel.</td>
</tr>
<tr>
<td></td>
<td>Dusek and Jurčícek [40]</td>
<td>Brother’s Bar-B-Q is a pub near Crowne Plaza Hotel in the riverside area. It serves French food and has a customer rating of 1 out of 5 and is kids friendly. Brother’s Bar-B-Q is a pub located near the Crowne Plaza Hotel in the riverside area. It is kids friendly and serves French food with a moderate price range and a customer rating of 1 out of 5.</td>
</tr>
<tr>
<td></td>
<td>ED+ACS (our model)</td>
<td>ED+ACS (our model)</td>
</tr>
<tr>
<td>name[Afghan Kebab House], eatType[restaurant], food[Italian], priceRange[high], area[city centre], familyFriendly[no], near[Raja Indian Cuisine]</td>
<td>Reference</td>
<td>Located in the city centre near Raja Indian Cuisine is an adult restaurant called Afghan Kebab House. This high priced restaurant serves Italian food.</td>
</tr>
<tr>
<td></td>
<td>ED+A</td>
<td>Antico Caffe is not a family-friendly Italian restaurant which is not family friendly.</td>
</tr>
<tr>
<td></td>
<td>Qader et al. [25]</td>
<td>The restaurant near Raja Indian Cuisine in the city centre is called The Leopard, which is not child friendly and serves Italian food but is highly priced.</td>
</tr>
<tr>
<td></td>
<td>Puzikov and Gurevych [29]</td>
<td>Afghan Kebab House is a restaurant which serves Italian food in the high price range. It is located in the city centre area, near Raja Indian Cuisine. It is not family friendly.</td>
</tr>
<tr>
<td></td>
<td>Dusek and Jurčícek [40]</td>
<td>Afghan Kebab House is an Italian restaurant in the city centre near Raja Indian Cuisine. It is not children friendly and has a high price range.</td>
</tr>
<tr>
<td></td>
<td>ED+ACS (our model)</td>
<td>Afghan Kebab House is a high priced Italian restaurant located in the city centre near Raja Indian Cuisine in the city centre.</td>
</tr>
</tbody>
</table>

We can also inspect the attention distribution for every time step in the decoding phase. In the attention matrices, every i\textsuperscript{th} row is the attention distribution that was used to output the i\textsuperscript{th} token, i.e., the $\Pi_{att}$ distribution described in Section 2.1. Figure 3a shows the attention matrix generated by ED+A when presented with the input “name[Thesaurus], customer rating[1 out of 5], area[city centre]”. Each row is similar. This means that the context vector $\chi_i$ (see Equation (1)) is always almost the same: the decoder sees similar input information at every time step. The dark column in the “Thesaurus” word position indicates that the restaurant’s name is the less weighted part of the input, another indication that the ED+A model is just skipping this field’s value. The decoder is prevented from the realization of the “name” field because the context is “unaware” of those data.
Figure 3. Attention matrix of the two models when presented with the input “name[Thesaurus],
customer rating[1 out of 5], area[city centre]”. In the x axes, there is the input sequence. In the y
axes, there is the output sequence. Every i-th row is the attention distribution that was used to output
the i-th token. Therefore, the value in position \((i, j)\) is the “focus” value over the \(j\)-th character when
the \(i\)-th letter was given in the output. Light dots mean higher attention than dark ones. (a) ED+A
model. (b) Encoder–Decoder model with Attention, Copy, and Shift (ED+ACS). The column on the
right shows the \(p_{gen}^t\) values for every time step, black being near 0 and white near 1.

Figure 3b shows the attention matrix generated by the ED+ACS model when presented
with the same sample as in Figure 3a. On the right, a column with the \(p_{gen}^t\) values for
every time step is shown. Colors darkening to black mean that the model is weighting the
attention distribution more than the alphabet one; i.e., a copying step.

Notice that each i-th row is shifted one time step to the right before the decoder uses
it to generate the i-th output letter, as shown in Equation (7). It is clearly noticeable that
the network learned to precisely focus on specific letters, and all attention distributions
were more peaked with respect to what happened using the ED+A model (Figure 3a). For
example, in the upper left corner, we see that the network almost perfectly chose to focus
on every letter in “Thesaurus” while reporting it to the output. For the same time steps,
the \(p_{gen}^t\) values are close to 0, confirming that the model was copying. We can therefore
state that the ED+ACS model was more effective than the ED+A model.

6. Conclusions

In this paper, we showed the advantages of using a character-based sequence-to-
sequence model to perform DIT tasks. Using a reduced vocabulary lets the model circum-
vent the rare word issue, as every word is in fact a sequence of characters. Moreover, the
copying and shifting mechanisms introduced in this work give the network the ability to
bring input portions directly to the output. This eliminates the need to leverage non-neural
scripts to pre-process and post-process data, as usually happens when dealing with lexical-
ization and tokenization steps. Therefore, our work demonstrates that a fully end-to-end
system can achieve results that are comparable to top-performing systems. In addition, it
is not constrained to a specific domain, allowing for the creation of more general models.

Acknowledgments: We thank the anonymous reviewers for helping us to improve our paper. This research has been partially carried on in the context of the Visiting Professor Program of the Italian Istituto Nazionale di Alta Matematica (INdAM).

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/tuetschek/e2e-dataset (accessed on 22 March 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References


41. Williams, R.J.; Zipser, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Comput.* 1989, 1, 270–280. [CrossRef]

