Deep copycat Networks for Text-to-Text Generation

Julia Ive\textsuperscript{1}, Pranava Madhyastha\textsuperscript{2} and Lucia Specia\textsuperscript{2}
\textsuperscript{1}DCS, University of Sheffield, UK
\textsuperscript{2}Department of Computing, Imperial College London, UK
j.ive@sheffield.ac.uk \{pranava,l.specia\}@imperial.ac.uk

Abstract

Most text-to-text generation tasks, for example text summarisation and text simplification, require copying words from the input to the output. We introduce copycat, a transformer-based pointer network for such tasks which obtains competitive results in abstractive text summarisation and generates more abstractive summaries. We propose a further extension of this architecture for automatic post-editing, where generation is conditioned over two inputs (source language and machine translation), and the model is capable of deciding where to copy information from. This approach achieves competitive performance when compared to state-of-the-art automated post-editing systems. More importantly, we show that it addresses a well-known limitation of automatic post-editing – overcorrecting translations – and that our novel mechanism for copying source language words improves the results.

1 Introduction

Text-to-text generation is generally addressed using sequence to sequence neural models (Sutskever et al., 2014). While initially proposed for machine translation (Bahdanau et al., 2015), these models have been shown to perform very well in monolingual variants of these tasks, such as text summarisation and text simplification (Nallapati et al., 2016a; Scarton and Specia, 2018). One limitation of such models is that they tend to be “over creative” when generating outputs, i.e. create a completely new output, which is not what they should do. Different strategies have been used to mitigate this issue, primarily in the context of abstractive text summarisation. The most prominent strategy is the use of pointer networks, which substantially outperform vanilla sequence to sequence models (Nallapati et al., 2016b; Gu et al., 2016; See et al., 2017). Pointer networks (Vinyals et al., 2015) are an extension of attentive recurrent neural network (RNN) architectures to use attention as a pointer to select which tokens of the input sequence should be copied to the output, as a switching mechanism between copying and generating new words.

In this paper we propose copycat, a flexible pointer network framework for text-to-text generation tasks. It differs from previous work in the following main ways: it is based on transformer networks (Vaswani et al., 2017), which have been shown to achieve superior performance in text generation tasks. While Gehrmann et al. (2018b) also provide contrastive experiments using transformers, they however randomly choose one of the attention heads as the copy-distribution, while we let the networks learn through all layers and use the attention heads from the final layer of the decoder to learn the distribution; thereby the decision between copying input words and generating new words is made at the highest level of abstraction. Further, in our model, we do not use coverage penalty and achieve similar performance as Gehrmann et al. (2018b).

For text summarisation, we show that our copycat model performs competitively with comparable models, but achieves a much higher novel n-gram generation rate (wrt original article) and a low repetition rate (within a summary) without any coverage penalty.

We further extend this architecture to the dual-source setting, where it is conditioned on two inputs and can copy from either of them. We apply this approach to automatic post-editing, where the network can generate new words, or copy words from either the source language or the original machine translation. We show that this is a promising approach especially to avoid overcorrecting high quality machine translations and for text domains where terminology (or jargon) should be preserved.
across languages, rather than translated.

To sum up, our **main contributions** are: (i) **copycat** networks – a transformer-based pointer network for text-to-text generation tasks (Section 3.2); and (ii) an extension of this model which can copy from multiple inputs (Section 3.3). Our models produce SOTA results in summarisation. Our datasets and settings are described in Section 4, and the experiments with text summarisation and automatic post-editing, in Section 5.

## 2 Related work

In this section we highlight the most closely related work on abstractive summarisation and automatic post-editing.

### Abstractive Summarisation

SOTA approaches using sequence to sequence models have evolved from vanilla attentional models (Rush et al., 2015; Nallapati et al., 2016a) to more advanced architectures that include the use of pointer networks (Nallapati et al., 2016b; See et al., 2017; Gu et al., 2016; Paulus et al., 2017), reinforcement learning (Paulus et al., 2017; Li et al., 2018; Pasunuru and Bansal, 2018; Chen and Bansal, 2018; Hsu et al., 2018) and content selection (Gehrmann et al., 2018a; Pasunuru and Bansal, 2018; Chen and Bansal, 2018).

Nallapati et al. (2016b) and Gu et al. (2016) propose the use of pointer-generator RNN-based networks to reduce out-of-vocabulary (OOV) rate. This idea was followed by See et al. (2017), which incorporates a coverage mechanism to avoid repetition of input words by keeping track of what has already been covered.

Reinforcement learning (RL) approaches optimise objectives for summarisation in addition to maximum likelihood. Paulus et al. (2017) combine ROUGE and maximum likelihood as training objectives, use a pointer network and – to avoid repetition – introduce intra-attention to the encoder and decoder that attends over the input and continuously generated output separately. Li et al. (2018) use a global summary quality estimator which is a binary classifier aiming to make the generated summaries indistinguishable from the human-written ones. Pasunuru and Bansal (2018) have a loss function based on whether keywords detected as salient are included in a summary, while Hsu et al. (2018) modulate the attention based on how likely a sentence is to be included in a summary, and Chen and Bansal (2018) follow an extractive-abstractive hybrid architecture to first extract full sentences from a document using a sentence-level policy gradient and then compress them.

Gehrmann et al. (2018a) also perform content selection, but on the level of phrases by treating this process as a sequence labelling task, and without RL. They first build a selection mask for the source document and then constrain a decoder on this mask.

The latter two represent the best overall approaches on common datasets, using either an RNN- or a Transformer-based architecture. We show that our **copycat** approach is competitive and is more abstractive (we describe this in the following sections).

### Automatic Post-Editing (APE)

APE is usually addressed by monolingual translation models that “translate” from the raw MT to fixed MT (post-edits, PE). Given the high quality of current neural MT (NMT) outputs, the task has become particularly challenging. Human corrections to those outputs are rare and very context-dependent, which makes them difficult to capture and generalise. This increases the chance of APE systems to overfit or overcorrect new inputs at test time.

SOTA APE approaches tackle the task with the multi-source transformer architectures (Junczys-Dowmunt and Grundkiewicz, 2018; Tebbifakhr et al., 2018). Two encoders encode the source and the raw MT, respectively, and an additional target-source multi-head attention component is stacked on top of the original target-source multi-head attention component. These models are trained with millions of <source, MT, PE> triplets, which does not fit realistic scenarios where only a handful of post-editions exist. Special parallel training corpora with additional MT data are released to help the development (Negri et al., 2018). According to (Chatterjee et al., 2018), these SOTA systems modify up to 30% of the NMT input, out of which only 50% are positive changes, the rest are unnecessary paraphrases or deteriorate the output.

Our **copycat** networks address this problem: they are conservative and make very few careful
corrections to good quality raw MT – as a result of learning predominantly to copy, while still making substantial corrections when the raw MT is not as good.

3 copycat Networks

3.1 Transformer Architecture

We make use of the SOTA sequence to sequence modelling framework – the Transformer architecture (Vaswani et al., 2017). Our implementation is a multi-layer encoder-decoder architecture that uses the tensor2tensor library. We briefly describe the encoder and decoder blocks in what follows.

Encoder block (E): The encoder block is made of 6 layers, each containing two sub-layers of multi-head self-attention mechanism followed by a fully connected feed forward neural network. We follow the standard implementation and employ residual connections between each layer, as well as layer normalisation. The output of the encoder forms the encoder memory which consists of contextualised representations for each of the source tokens ($M_E$).

Decoder block (D): The decoder block is similar to the standard transformer architecture and comprises of 6 layers. It contains an additional sub-layer that performs multi-head attention over the outputs of the encoder block. Specifically, a decoding layer $d_i$ is the result of multi-head attention over the outputs of the encoder which in turn is a function of the encoder memory and the outputs from the previous layer: $A_D \rightarrow E = f(M_E, d_{i-1})$ where, the keys and values are the encoder outputs and the queries correspond to the decoder input.

In all cases we tie the source and target vocabulary. We also use Byte Pair Encoding (BPE) (Sennrich et al., 2016) word segmentation approach to pre-process the data before we fix the vocabulary. This also helps us in reducing the OOV rate and makes the models less dependent on the vocabulary of the training data. We train all our models with the cross entropy loss.

3.2 Pointer-generator Network

We base our pointer-generator on the approach of See et al. (2017), however we significantly depart from their formulation. We first obtain an attention distribution over the source and decoder at each time step. We use a simple dot product $a_t$ over output of the last decoder layer ($A^\text{fin}_{D \rightarrow E}$) and the contextualised encoder representations for each of the source language tokens ($M_E$):

$$a_t = \text{softmax}(A^\text{fin}_{D \rightarrow E}, M_E)$$

where $A^\text{fin}_{D \rightarrow E}$ comes from final ($n = 6$) layer, just before applying layer normalisation and passing it through feed forward layers as shown in Figure 1. $A_{D \rightarrow E}$ already contains information from the source encoder. This differs both the RNN-based implementation in (See et al., 2017) and the copy-transformer (Gehrmann et al., 2018b) formulation, where a randomly chosen attention-head forms the basis for the copy-distribution. The generation probability $Pr_{\text{gen}}$ for each time step is computed from the output of the decoder layer as:

$$Pr_{\text{gen}} = \sigma(W_{\text{gen}}^T A^\text{fin}_{D \rightarrow E})$$

where $W$ is learned parameters and $\sigma$ is the sigmoid function which restricts $Pr_{\text{gen}} \in [0, 1]$.

We then follow (Vinyals et al., 2015) and use $Pr_{\text{gen}}$ as the switching function to decide between generating a word from the decoder vocabulary or copying the word directly from the source sequence by using $a_t$ as:

$$Pr = Pr_{\text{gen}} Pr_d(w_d) + (1 - Pr_{\text{gen}})a_t$$

where, $Pr_d(w_d)$ is decoder’s probability of generating output word from the decoder vocabulary.

We empirically observed that getting the new attention representation over the final layer is extremely important for obtaining a good estimate of the attention distribution.

3.3 Dual-source extension

We extend copycat networks for the case of APE where a dual-source encoder-decoder architecture has been shown to obtain superior results (Junczys-Dowmunt and Grundkiewicz, 2018). As previously mentioned, most of the translations in current APE datasets (Chatterjee et al., 2018) already have optimal quality, thereby models that alter these already accurate translations are sub-optimal. Based on this observation we propose a dual-source extension to our copycat model. We add an extra encoder ($E_s$) that encodes the sentence in source language and
the standard encoder \( (E) \) which encodes the machine translated output. We also add an additional decoder layer \( A_{D \rightarrow S} = f(M_{E_s}, A_{D \rightarrow E}) \), where \( A_{D \rightarrow E} \) is the standard decoding layer.

We maintain the pointer generator over the final decoder layer \( (n = 6) \) – \( A_{\text{fin}}^{D \rightarrow E} \), similarly to Equation 3.

**Dual-source with double attention** We further extend the dual-source copycat network such that it now attends over both the source-language encoder and the translated output encoder, and therefore words from the source language can also be copied. We believe that copying words from source language would especially be useful for APE. Some corrections may involve retaining the original words in the source language due to a variety of reasons, including the use of proper nouns and jargon. Our extension involves modification in two components:

- **Attention over source**: We build on Equation 1 and add a similar attention over the source encoder as:
  \[
  a_s = \text{softmax}(A_{D \rightarrow S}^{\text{fin}} \cdot M_{E_s}) \quad (4)
  \]
  where \( M_{E_s} \) is the source language encoder outputs.

- **Pointer-generation with multiple sources**: As there are three signals of information that influence the output decision – copy from source language, copy from the translated language and generate from decoder – we treat them as three different tasks. We thus modify Equation 2 with a normalised exponential function \( (\text{softmax}) \) instead of the sigmoid in the former. This results in:
  \[
  Pr_{\text{gen}} = \frac{\exp (W_{\text{gen}} \cdot A_{D \rightarrow S}^{\text{fin}})}{\sum_{j \in \text{tasks}} \exp (W_j \cdot A_{D \rightarrow S}^{\text{fin}})}
  \quad (5)
  \]
  where \( \text{tasks} = \{ \text{gen, trg, src} \} \); \( \text{trg} \) is for attention over the target language encoder, and \( \text{src} \) is for attention over the source language encoder. Notice that we use \( A_{D \rightarrow S}^{\text{fin}} \) as this includes both information from the target language encoder and the source language encoder.

- **Switching function**: Based on the probabilities from Equation 5, the new \( Pr \) is:
  \[
  Pr = Pr_{\text{gen}}Pr_{\text{d}}(w_d) + Pr_{\text{trg}}a_t + Pr_{\text{src}}a_s
  \]

Figure 1 shows the variants of the copycat architectures.

**4 Data and Settings**

**Summarisation** We use the CNN/Daily Mail dataset, the most widely use dataset for abstractive summarisation (Hermann et al., 2015; Nallapati et al., 2016b). It contains online news articles...
(800 tokens on average) and their abstracts (60 tokens on average). We used scripts provided by See et al. (2017) to obtain the same version of the the data, which has around 300K training pairs, 13K validation pairs and 11,490 test pairs. We apply BPE for word segmentation approach with 32K merge operations (Sennrich et al., 2016). During training and test we truncate articles to 400 BPE units as is standard practice (See et al., 2017).

APE Our APE datasets consist of target, human post-edit> triplets. We use two variants: (i) the English-German (EN-DE) dataset from the IT domain provided by the WMT18/WMT19 APE shared task, and translated by a high-performing NMT system (45.8 BLEU). It contains 13,442, 1,000 and 1,023 training, development and test triplets, respectively. (ii) the English-Latvian (EN-LV) dataset from the life sciences domain made available by Specia et al. (2018) containing 12,936 training / 1,000 development / 1,448 test triplets. Here, the outputs from NMT are of a lower quality (38.4 BLEU).

We use HTER (Snover et al., 2009) as the main metric and BLEU (Papineni et al., 2002) as the secondary metric, as in the WMT APE shared tasks (Chatterjee et al., 2018). HTER is defined as the minimum number of edits (substitution, insertion, deletion and shift) required to change an MT hypothesis so that it exactly matches a human post-edition of this hypothesis. BLEU measures n-gram precision between MT hypotheses and post-edits.

The EN-DE task is very challenging as the training data contains very few edits (0.15 HTER) and contain few context-dependent human corrections that are difficult to capture with machine learning methods. The EN-LV dataset has twice more edits (0.29 HTER). Actually, 36% of the German NMT data requires no corrections, and only 14% of the Latvian NMT data requires no correction.

Hyperparameters: For both tasks, we use Adam as optimiser (Kingma and Ba, 2014) and train the model until convergence with a fixed batch size of 50. Across experiments, we use shared source and target embeddings. We build on the tensor2tensor implementation using the transformer_base_v2 parameters for summarisation as best practise. The transformer_base_v1 parameters with a learning rate of 0.05 with 8K warmup steps are used for APE. For other hyperparameters we use the default values in tensor2tensor.

For both tasks we apply early stopping with the patience of 10 epochs based on the validation ROUGE-L or BLEU score, for summarisation and APE respectively. For summarisation, we use a randomly chosen validation subset of 2k abstracts. During decoding, we use a beam search of size 5 and tune the α parameter (length penalty) on the validation subset. During decoding for APE, we use a beam search of size 10.

5 Results

In this section we present results of our experiments, first for summarisation (Section 5.1) and then for APE experiments (Section 5.2).

5.1 Summarisation

Table 1 shows results of our summarisation experiments. We report F1-scores for a standard set of ROUGE (Lin, 2004) scores: ROUGE-1, ROUGE-2 and ROUGE-L. They measure unigram, bigram recall and the longest in-sequence common to the generated and the reference abstracts, respectively. The scores are measured with the pyrouge toolkit.4

We also report results for a range of conceptually different systems, including the LEAD-3 baseline (the top 3 sentences of the source). copycat performs on par with the CopyTransformer. Our generated sentences for both systems are of 62 tokens on average, with references of 58 tokens on average.

To understand the model, we look at two aspects (i) the level of repetition within the generated abstracts, and (ii) the level of transfer from original to summarised text (i.e. the novelty rate).

Figure 2 reports the results of our analysis of repetitions in the outputs of our system and the reference abstracts. For unigrams, our model is comparable to the coverage mechanism of See et al. (2017). However, we note that our model is less effective than the one in See et al. (2017) for higher order n-grams, where repetitions are almost completely eliminated. This is done at the cost

---

3http://www.statmt.org/wmt19/ape-task.html. We also experimented with the additional – English-Russian – WMT19 dataset, but its quality is very questionable, as confirmed in personal communication with the task organisers.

4We use the following wrapper: https://github.com/tensorflow/tensor2tensor#summarization
Table 1: Summarisation results on the CNN/Daily Mail dataset according to three standard metrics. LEAD-3 is a strong baseline which simply copies the first three sentences of the input. The second group contains RNN-based networks, while the last group contains transformer-based networks. All external results are from the original papers, which are briefly described in Section 2. Our results are significant with a 95% confidence interval of at most 0.25 as provided by the official ROUGE script.

<table>
<thead>
<tr>
<th>Method</th>
<th>ROUGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD-3 Baseline</td>
<td>40.10 15.50 36.30</td>
</tr>
<tr>
<td>Pointer-Generator (See et al., 2017)</td>
<td>36.44 15.66 33.42</td>
</tr>
<tr>
<td>Pointer-Generator + Coverage (See et al., 2017)</td>
<td>39.53 17.28 36.38</td>
</tr>
<tr>
<td>ML + RL (Paulus et al., 2017)</td>
<td>39.87 15.82 36.90</td>
</tr>
<tr>
<td>Saliency + Entailment reward (Pasunuru and Bansal, 2018)</td>
<td>40.43 18.00 37.10</td>
</tr>
<tr>
<td>Saliency to guide network (Li et al., 2018)</td>
<td>38.95 17.12 35.68</td>
</tr>
<tr>
<td>Inconsistency loss (Hsu et al., 2018)</td>
<td>40.68 17.97 37.13</td>
</tr>
<tr>
<td>Sentence Rewriting (Chen and Bansal, 2018)</td>
<td>40.88 17.80 38.54</td>
</tr>
<tr>
<td>Bottom-Up Summarisation (Gehrmann et al., 2018a)</td>
<td>41.22 18.68 38.34</td>
</tr>
<tr>
<td>CopyTransformer + Coverage Penalty (Gehrmann et al., 2018a)</td>
<td>39.25 17.54 36.45</td>
</tr>
<tr>
<td>copycat</td>
<td>39.15 17.60 36.17</td>
</tr>
</tbody>
</table>

Table 2: Examples of articles, their copycat and human reference summaries. Bold highlights segments repeated by copycat.

<table>
<thead>
<tr>
<th>Article</th>
<th>copycat</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>... anne frank died of typhus in a nazi concentration camp at the age of 15 ... her older sister , margot frank , died at least a month earlier than previously thought ... they concluded that anne and margot probably did not survive to march 1945 ...</td>
<td>anne frank died of typhus in a nazi concentration camp at the age of 15 . her sister margot frank died at least a month earlier than previously thought . anne and margot probably did not survive to march 1945 .</td>
<td>museum : anne frank died earlier than previously believed . researchers re-examined archives and testimonies of survivors . anne and older sister margot frank are believed to have died in february 1945 .</td>
</tr>
<tr>
<td>Article</td>
<td>copycat</td>
<td>Reference</td>
</tr>
<tr>
<td>... billionaire businessman tony toutouni ... who chronicles his immensely extravagant lifestyle on the photo-sharing site and is usually seen next to stacks of cash and bikini-clad models - admits ‘ it’s not that hard to get any girl you want ’ . his outrageous posts , which have seen him amass 750,000 followers in eight months ...</td>
<td>tony toutouni , 42 . chronicles his devotion extravagant lifestyle on the photo-sharing site . his outrageous posts have seen him amass 750,000 followers in eight months . toutouni says ‘ it’s not that hard to get any girl you want ’</td>
<td>tony toutouni has amassed 750,000 followers on photo-sharing site in eight months thanks to outrageous posts . la-based entrepreneur is endlessly surrounded by supercars , piles of cash and bikini-clad women in pictures . he ‘s friends with controversial instagram playboy dan bilzerian and says ‘ it’s not hard to get any girl you want ’</td>
</tr>
</tbody>
</table>

The percentage of new words (unigrams) introduced by copycat is already in line with or higher than the percentage reported for other models (2.2% in (See et al., 2017), 0.5% Gehrmann et al. (2018a) and 1% in our case). The manual inspection of newly-introduced words confirmed their validity. From this inspection, we observed that the risk of creating new words as a consequence of combining BPE segments is very low (e.g., constadedmind produced by combining con, sta, de, d and mind).

We further manually inspect outputs of copycat systems to gain insights into their performance. Table 2 lists an example of produced outputs. We notice that the model is capable of generating fewer novel n-grams, as compared to the original abstract. Specifically, See et al. (2017) are only able to generate around 10% novel quadrigrams, whereas we generate 22%. Figure 3, which depicts the percentage of novel n-grams and sentences introduced by our copycat models as compared to full articles (without truncation).

The percentage of new words (unigrams) introduced by copycat is already in line with or higher than the percentage reported for other models (2.2% in (See et al., 2017), 0.5% Gehrmann et al. (2018a) and 1% in our case). The manual inspection of newly-introduced words confirmed their validity. From this inspection, we observed that the risk of creating new words as a consequence of combining BPE segments is very low (e.g., constadedmind produced by combining con, sta, de, d and mind).
both to selectively copying relevant parts of an article, as well as making deductions about the content, which often results in an abstract different from the reference abstract. Overall, the model actively attends to different parts of input and is capable of efficiently reorganising the information in the original article.

5.2 APE

We perform experiments with three models: (a) NO-PNT which is a reimplementation of the dual-source Transformer model of (Junczys-Dowmunt and Grundkiewicz, 2018); (b) PNT-TRG, copycat with attention over the MT; (c) PNT-TRG+SRC, copycat with double attention. Our baseline (BASE) is the standard “do nothing”, i.e. the raw MT as provided in the datasets.

As the sizes of the APE datasets are very small, training only using this data is not successful. We thus pre-train all our models, as is a common practise for the APE task (Tebbibafkhr et al., 2018). However, we do not generate new MT data. Instead we mimic MT data: we randomly delete, insert, shift or substitute words in the target side of parallel data to match HTER statistics of a respective APE dataset. For German, we randomly select 500K parallel data (only source and target) from the eSCAPE data provided with the task (Negri et al., 2018). For Latvian, we select those 500K from a corpus created by merging the Europarl (version 8) and EMEA corpora.

The data is truecased using the Moses toolkit scripts (Koehn et al., 2007). We again apply the BPE with 32K merge operations (Sennrich et al., 2016). Parameters of encoders for our dual copycat are shared. We use the label smoothing of 0.1. For the fine-tuning experiments, we reduce the learning rate to 0.001.

Table 4 shows results of our APE experiments. We compare our three setups to the baseline MT quality (BASE) without any APE. We see that the performance of our copycat systems depends on the difficulty of the task: we observe an improvement for Latvian (+1 absolute HTER point improvement) with the lowest baseline quality (HTER= 0.29), and minor improvements for German with a higher baseline quality (on average 0.14 HTER). As a comparison, the best performing SOTA systems achieve only up to 0.4 HTER improvement for the German dataset this is by using millions of training data, combined losses and ensembling techniques, whereas we use minimum external resources (Tebbibafkhr et al., 2018; Junczys-Dowmunt and Grundkiewicz, 2018).

For EN-DE, our PNT-TRG model performs a small number of corrections (mostly accurate) to the outputs: only 50 sentences, compared to the 200-300 sentences modified by the SOTA EN-DE APE systems (Chatterjee et al., 2018). For EN-LV, given the raw translations are of lower quality, our models make more corrections: 330 sentences, which lead to overall improvements.

As a general observation, for both training sets, post-edit operations repeated more than once make around only 10% of all corrections. And this is not taking into account the fact that corrections also occur in different contexts. The majority of repetitive post-edits are corrections to punctuation (e.g., deletion of a comma) or auxiliary words (e.g., insertion/deletion of prepositions or articles). Therefore, we believe that especially in 3http://www.statmt.org/wmt17/translation-task.html
4http://opus.nlpl.eu/EMEA.php
7Our baseline scores differ from those provided by the task organisers because of differences in truecasing strategies.
8For instance, Junczys-Dowmunt and Grundkiewicz (2018) report 16.5/75.44 HTER/BLEU↑ for EN-DE over the 16.84/74.73 “do nothing” baseline.
### Table 3: Examples of correct (two first cases) and incorrect (third case) automatic post-edits produced from the EN-DE dataset by PNT-TRG+SRC.

<table>
<thead>
<tr>
<th>lang</th>
<th>BASE</th>
<th>NO-PNT</th>
<th>PNT-TRG</th>
<th>PNT-TRG+SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LV</td>
<td>30.0 / 59.5</td>
<td>62.5 / 23.3</td>
<td>29.0 / 59.8</td>
<td>28.6 / 59.9</td>
</tr>
<tr>
<td>DE</td>
<td>17.8 / 72.3</td>
<td>18.5 / 71.2</td>
<td>17.9 / 72.3</td>
<td>17.8 / 72.4</td>
</tr>
</tbody>
</table>

Table 4: HTER↓ / BLEU↑ scores for the APE task. Boldface values mark statistically significant improvements over BASE, i.e. the raw MT output. We have performed significance testing via bootstrap resampling using the Multeval tool (Clark et al., 2011), the standard practice in APE.

6 Conclusions

In this paper we propose copycat networks, a novel transformer-based pointer-generator network. For summarisation, we observe that copycat network is not only able to keep the repetition rate low, but also substantially increases the rate of novel n-grams. We further extend the copycat for automatic post-editing by conditioning over dual sources (source language and target language) and introducing the double-attention to copy from both sources. copycat is able to learn (a) more conservative models that copy more and generate less when the raw translations are already good, and (b) more aggressive models that make more corrections when the original translation quality is not as good. This shows that the architectures are flexible and adapt well to different datasets. Our source code is available at: https://github.com/ImperialNLP/CopyCat.

Acknowledgments

The authors thank the anonymous reviewers for their useful feedback. This work was supported by the MultiMT (H2020 ERC Starting Grant No. 678017), MMVC (Newton Fund Institutional Links Grant, ID 352343575) and APE-QUEST (Connecting Europe Facility, project 2017-EU-IA-0151) projects.
References


