University of Edinburgh’s Submission to the Document-level Generation and Translation Shared Task

Ratish Puduppully * and Jonathan Mallinson * and Mirella Lapata
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
r.puduppully@sms.ed.ac.uk J.Mallinson@ed.ac.uk mlap@inf.ed.ac.uk

Abstract
The University of Edinburgh participated in all six tracks: NLG, MT, and MT+NLG with both English and German as targeted languages. For the NLG track, we submitted a multilingual system based on the Content Selection and Planning model of Puduppully et al. (2019). For the MT track, we submitted Transformer-based Neural Machine Translation models, where out-of-domain parallel data was augmented with in-domain data extracted from monolingual corpora. Our MT+NLG systems disregard the structured input data and instead rely exclusively on the source summaries.

1 Track 1/2: Natural Language Generation
The Natural Language Generation (NLG) track revolved around systems that take structured data in the form of tabular data from a basketball game as input, and generate a summary of this game in the target language. We entered one multilingual system which outputs summaries in both English and German. A multilingual model allows us to overcome the limited amount of German training data.

We adopted the content selection and planning approach of Puduppully et al. (2019), made extensions to the model and parameterized the decoder with a language tag, indicating the target language. The training was done using the full ROTOwire English dataset and the ROTOwire English-German dataset. We first explain the approach of Puduppully et al. (2019), describe the extensions to their model and show how language tags can be added to the decoder to indicate the target language.

1.1 The Content Selection and Planning Approach of Puduppully et al. (2019)

Puduppully et al. (2019) model \( p(y|r) \) as the joint probability of text \( y \) and content plan \( z \), given input \( r \). They further decompose \( p(y, z|r) \) into \( p(z|r) \), a content selection and planning phase, and \( p(y|r, z) \), a text generation phase:

\[
p(y|r) = \sum_z p(y, z|r) = \sum_z p(z|r) p(y|r, z)
\]

Given input records, probability \( p(z|r) \) is modeled using Pointer Networks (Vinyals et al., 2015). The probability of output text \( y \) conditioned on previously generated content plan \( z \) and input table \( r \) is modeled as follows:

\[
p(y|r, z) = \prod_{t=1}^{\left|y\right|} p(y_t|y_{<t}, z, r)
\]

where \( y_{<t} = y_1 \ldots y_{t-1} \). They use an encoder-decoder architecture with an attention mechanism to compute \( p(y|r, z) \). The architecture is shown in Figure 1.

The content plan \( z \) is encoded into \( \{e_k\}_{k=1}^{\left|z\right|} \) using a bidirectional LSTM. Because the content plan is a sequence of input records, they directly feed the corresponding content selected record vectors \( \{r_j^{cs}\}_{j=1}^{\left|r\right|} \) as input to the LSTM units, which share the record encoder with the first stage. For details of the content selection stage, please refer Puduppully et al. (2019).

The text decoder is also based on a recurrent neural network with LSTM units. The decoder is initialized with the hidden states of the final step in the encoder. At decoding step \( t \), the input of the LSTM unit is the embedding of the previously
predicted word \(y_{t-1}\). Let \(d_t\) be the hidden state of the \(t\)-th LSTM unit. The probability of predicting \(y_t\) from the output vocabulary is computed via:

\[
\beta_{t,k} \propto \exp(d_t^T W_b e_k)
\]

(1)

\[
q_t = \sum_k \beta_{t,k} e_k
\]

\[
d_t^{att} = \tanh(W_d[d_t; q_t])
\]

(2)

\[
p_{gen}(y_t|y_{<t}, z, r) = \text{softmax}_y (W_y d_t^{att} + b_y)
\]

where \(\sum_k \beta_{t,k} = 1\), \(W_b \in \mathbb{R}^{n \times n}\), \(W_d \in \mathbb{R}^{n \times 2n}\), \(W_y \in \mathbb{R}^{n \times |V_y|}\), \(b_y \in \mathbb{R}^{|V_y|}\) are parameters, and \(|V_y|\) is the output vocabulary size.

They further augment the decoder with a copy mechanism, allowing the ability to copy words directly from the value portions of records in the content plan (i.e., \(\{z_k|z\}\)). They experimented with joint (Gu et al., 2016) and conditional copy methods (Gulcehre et al., 2016). Specifically, they introduce a variable \(u_t \in \{0, 1\}\) for each time step to indicate whether the predicted token \(y_t\) is copied \((u_t = 1)\) or not \((u_t = 0)\). The probability of generating \(y_t\) is computed by:

\[
p(y_t|y_{<t}, z, r) = \sum_{u_t \in \{0, 1\}} p(y_t, u_t|y_{<t}, z, r)
\]

where \(u_t\) is marginalized out.

### 1.2 Copying from Table and Plan

We extended the copy mechanism further such that \(u_t\) can take three values: \(y_t\) is generated from the vocabulary \((u_t = 0)\), \(y_t\) is copied from the content plan \((u_t = 1)\) and \(y_t\) is copied from the table \((u_t = 2)\).

**Conditional Copy** The variable \(u_t\) is first computed as a switch gate, and then is used to obtain the output probability:

\[
p(u_t|y_{<t}, z, r) = \text{softmax}(w_u \cdot d_t^{att} + b_u)
\]

\[
\alpha_{t,j} \propto \exp(d_t^T W_{r} c_j)
\]

(3)

\[
p(y_t, u_t|y_{<t}, z, r) =
\begin{cases}
  p(u_t|y_{<t}, z, r) \sum_{k} \beta_{t,k} & u_t = 1 \\
  p(u_t|y_{<t}, z, r) \sum_{j \in \gamma_k} \beta_{t,k} \sum_{k} y_{t \leftarrow r_j} \alpha_{t,j} & u_t = 2 \\
  p(u_t|y_{<t}, z, r) p_{gen}(y_t|y_{<t}, z, r) & u_t = 0
\end{cases}
\]

where \(\sum_{j \in \gamma_k} \alpha_{t,j} = 1\). \(y_t \leftarrow z_k\) indicates that \(y_t\) can be copied from \(z_k\), \(y_t \leftarrow r_j\) indicates that \(y_t\) can be copied from \(r_j\). \(\gamma_k\) indicates records in table corresponding to the \(k\)th record in plan, for example: if \(k\) is ‘PTS’ value of player Jeff Teague, then \(\gamma_k\) corresponds to all the records for the entity Jeff Teague in the table including ‘PTS’, ‘REB’, ‘NAME1’, ‘NAME2’ etc. \(\beta_{t,k}\) and \(p_{gen}(y_t|y_{<t}, z, r)\) are computed as in Equations (1)–(2), and \(w_u \in \mathbb{R}^{3 \times n}\), \(b_u \in \mathbb{R}^{3}\) are parameters.
1.3 Attending to the Table and Content Plan

The output text is generated by attending to both the content plan and the table (See Figure 2.)

\[
\delta_{i,j} \propto \exp(d_i^t W_c r_j^{CS})
\]

\[
s_t = \sum_j \delta_{i,j} r_j^{CS}
\]

\[
d_i^{att} = \tanh(W_d[d_i; q_t; s_t])
\]

\[
p_{gen}(y_t|y_{<t}, z, r) = \text{softmax}_y(W_y d_t^{att} + b_y)
\]

where \(\sum_j \delta_{i,j} = 1\), \(W_c \in \mathbb{R}^{n \times n}\), \(W_d \in \mathbb{R}^{n \times 3n}\), \(W_y \in \mathbb{R}^{n \times |\mathcal{V}_y|}\), \(b_y \in \mathbb{R}^{|\mathcal{V}_y|}\) are parameters, and \(|\mathcal{V}_y|\) is the output vocabulary size.

1.4 Feature for Team Points and Ranking of Player Points

Upon inspection of the ROTOWire game summaries in the development set, we observed that the summaries often describe the statistics of the winning team followed by the statistics of the losing team. The highest ranked players of either team are also often described in sequence in the summaries. Currently, we rely on the word embeddings of the team and player points to help disambiguate the winning from the losing team and to learn the relative performances of the players. We hypothesize that explicitly providing information about the relative performance of players and teams should make the learning easier.

We thus experimented with a feature for the winning/losing team and the ranking of player points within a team. Specifically, we added a binary feature for team records: \(\text{win}\) for each record in the winning team, \(\text{loss}\) for each record in the losing team. We further rank players in a team on the basis of their points and we add a feature indicating their rank in the team. For instance, Kyle Lowry scored the highest number of points in the home team and we add feature \(\text{hometeam-0}\) to each of his records. Player Jahlil Okafor was the second highest scorer in the visiting team and we add the feature \(\text{visteam-1}\) to each of his records and so on.

1.5 Training a Single Multilingual Model

We trained a single model for English and German data-to-text with a common BPE (Sennrich et al., 2015b) vocabulary of 2000 symbols for the output summaries. Player names and values of records in summaries were not BPEd. The target text was prefixed with token indicating the language of output ‘EN’ or ‘DE’. During inference, we forced the model to generate output in the desired language.

1.6 Dataset

We made use of the full ROTOWire English dataset of Wiseman et al. (2017) and the German dataset provided as part of the shared task. The statistics of the dataset are given in Table 1.
Table 1: Count of examples in Training, Development and Test sections of English and German dataset.

| Language | Train | Dev | Test |
|----------|-------|-----|------|
| English  | 3398  | 727 | 728  |
| German   | 242   | 240 | 241  |

Table 2: Automatic evaluation for track 1/2 on the ROTOwire test set using record generation (RG) precision, content selection (CS) precision and recall, content ordering (CO) in normalized Damerau-Levenshtein distance, and BLEU.

| Model | RG P% | CS P% | CS R% | CO DLD% | BLEU |
|-------|-------|-------|-------|---------|------|
| EN    | 91.41 | 30.91 | 64.13 | 21.72   | 17.01|
| DE    | 70.23 | 23.40 | 41.83 | 16.08   | 10.95|

Table 3: Size (number of parallel training sentences) in 100,000 of the EN-DE training data.

| Dataset                  | Size  |
|--------------------------|-------|
| Europarl v9              | 18.39 |
| Common Crawl corpus      | 24.00 |
| News Commentary v14      | 3.38  |
| Document-split Rapid corpus | 14.01 |
| Wikititles               | 13.05 |
| ParaCrawl               | 162.64|
| ROTOwire EN-DE          | 0.033 |
| Total                    | 235.47|

2.1.1 In-Domain Parallel Data

Table 3 highlights the extremely limited amount of in-domain parallel training data used; ROTOwire English-German makes up only 0.001% the parallel training data. To ensure our translation system produces in-domain translation, we supplemented the parallel data with in-domain monolingual data. We used back-translation to translate clean monolingual data from the target language to the source language.

Finding in-domain data for basketball is not trivial, as there are no explicit basketball WMT19 monolingual training sets. Therefore, we extracted in-domain basketball data from the available general-purpose monolingual datasets.

We considered all documents within the News Crawl 2007-2018 dataset and included all sentences which appeared within a document where any of the following conditions were met: (1) Contains a player’s name, as taken from the ROTOwire English-German training data; (2) Contains two team names; (3) the title contains the word NBA. For German, 1.1 million monolingual target sentences were collected, and for English, 4.32 million monolingual target sentences. These sentences were then back-translated via sampling (Edunov et al., 2018) and used to augment the parallel training data.

2.2 Model Description

For our submissions, we used the Transformer model as implemented within OpenNMT-py (Klein et al., 2017). Transformers are state-of-the-art NMT approaches which rely on multi-headed attention applied to both the source and target sentences. All experiments are performed with 6 encoder-decoder layers, with an embedding layer of size 512, a feed-forward layer size of 2048, and
| Model   | RG P% | CS P% | CO R% | DLD% | BLEU |
|---------|-------|-------|-------|------|------|
| EN-DE   | 81.01 | 77.32 | 78.49 | 62.21 | 36.85 |
| DE-EN   | 91.40 | 78.99 | 63.04 | 51.73 | 41.15 |

Table 5: Automatic evaluation for track 3-6 on the ROTOwire test set using record generation (RG) precision, content selection (CS) precision and recall, content ordering (CO) in normalized Damerau-Levenshtein distance, and BLEU.

8 attentional heads. We set the batch size to 4096 tokens and maximum sentence length to 100 BPE subwords. Dropout and label smoothing were also both set to 0.1. All other settings were set their default values as specified in OpenNMT-py. Decoding was performed with a beam size of ~15, length penalty averaging, and the decoder was constrained to block repeating 4-grams. Model selection was done using the BLEU score on the development set.

2.3 Results

Results on the development set in Table 4 show that the inclusion of monolingual data leads to a significant increase in bleu (between 5 and 7 points). Table 5 shows test set results for both English and German target languages. The results were provided by the shared task organizers.

3 Track 5/6: MT + NLG

The MT + NLG track combines the previous tracks, models take in as input both the structured data and the summary in the source language and produce a summary in the target language as output. We chose to disregard the structured data and instead exclusively use the source summary, translating it to the target language. As such this submission to this track is a replication of our MT submission with results shown in Table 5.