# Revisiting the Binary Linearization Technique for Surface Realization

Yevgenyi Puzikov, Claire Gardent, Ido Dagan, Iryna Gurevych

# Revisiting the Binary Linearization Technique for Surface Realization

**Yevgeniy Puzikov**[1], **Claire Gardent**[2], **Ido Dagan**[3], **Iryna Gurevych**[1]

[1] Ubiquitous Knowledge Processing Lab (UKP-TUDA) and Research Training Group AIPHES,
Department of Computer Science, Technische Universität Darmstadt, Germany
[2] CNRS / LORIA Nancy, France
[3] Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
https://www.ukp.tu-darmstadt.de
gardent@loria.fr
dagan@cs.biu.ac.il

## Abstract

End-to-end neural approaches have achieved state-of-the-art performance in many natural language processing (NLP) tasks. Yet, they often lack transparency of the underlying decision-making process, hindering error analysis and certain model improvements. In this work, we revisit the binary linearization approach to surface realization, which exhibits more interpretable behavior, but was falling short in terms of prediction accuracy. We show how enriching the training data to better capture word order constraints almost doubles the performance of the system. We further demonstrate that encoding both local and global prediction contexts yields another considerable performance boost. With the proposed modifications, the system which ranked low in the latest shared task on multilingual surface realization now achieves best results in five out of ten languages, while being on par with the state-of-the-art approaches in others. [1]

## 1 Introduction

Natural Language Generation (NLG) is the task of generating natural language utterances from various data representations. In this work we consider lemmatized dependency trees as input and focus on the process of transforming a dependency tree into a linearly-ordered grammatical string of morphologically inflected words – the setup which is most commonly known as surface realization (SR) (Langkilde-Geary, 2002; Belz et al., 2011).

Most surface realization approaches fall into two main groups: feature-based incremental generation pipelines and end-to-end neural approaches. To predict a correct token sequence, the former methods start with an empty hypothesis and extend it by ranking possible continuation candidates. These systems use manually-crafted feature sets and lack a principled way of incorporating global context. Neural models, on the other hand, usually encode the whole input to pass the information to the decoder which then generates the output sequence. The two main limitations of these approaches are their reliance on large amounts of training data and less interpretable behavior compared to feature-based methods.

This work builds upon BINLIN, a binary linearization technique proposed by Puzikov and Gurevych (2018). It is a hybrid approach which uses a feature-based neural word ordering module and a sequence-to-sequence morphological inflection component. In terms of prediction accuracy, BINLIN falls short compared to end-to-end neural approaches, but has an advantage of being more intuitive and interpretable. It also supports separate analysis of the syntactic ordering and morphological inflection steps of the surface linearization process. From a research perspective, this offers greater control over the problem-solving procedure.

In this work we extend BINLIN along two orthogonal directions. First, we propose a way to enrich the training data, which largely compensates for the small size of the datasets used in the task. Second, we propose a new input encoding strategy which incorporates both local and global prediction contexts. These modifications bridge the performance gap between BINLIN and end-to-end black-box approaches, while retaining its interpretability advantages.

[1] https://github.com/UKPLab/inlg2019-revisiting-binlin



*Proceedings of The 12th International Conference on Natural Language Generation*, pages 268–278,
Tokyo, Japan, 28 Oct - 1 Nov, 2019. 

| Data split | Language | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ar | cs | en | es | fi | fr | it | nl | pt | ru |
| Train | 6,016 | 66,485 | 12,375 | 14,289 | 12,030 | 14,529 | 12,796 | 12,318 | 8,325 | 48,119 |
| Dev | 897 | 9,016 | 1,978 | 1,651 | 1,336 | 1,473 | 562 | 720 | 559 | 6,441 |
| Test | 676 | 9,876 | 2,061 | 1,719 | 1,525 | 416 | 480 | 685 | 476 | 6,366 |

Table 1: Number of sentences in SR'18 datasets (Mille et al., 2018).

## 2 Task Description

The NLP community organized two Surface Realization Shared Tasks (in 2011 and 2018) which aimed at developing a common representation that could be used by a variety of NLG systems as input (Belz et al., 2011). They used almost identical task definitions, but different datasets. We focus on the latest task (SR'18 (Mille et al., 2018)), because the former was confined to using English data only, while the latter included Arabic, Czech, Dutch, English, Finnish, French, Italian, Portuguese, Russian and Spanish Universal Dependencies (UD, version 2.0) treebanks. [2]

SR'18 offered two different input data representations:

*Shallow Track*: unordered dependency trees consisting of lemmatized nodes with part-of-speech (POS) tags and morphological information, as found in the UD annotations.

*Deep Track*: same as above, but having functional words and morphological features removed and syntactic edge labels mapped into predicate-argument semantic relation labels.

We focus on the Shallow Track, because it covers more languages than the Deep Track (only three), and is therefore more interesting to study the problem of word ordering and morphological inflection as two steps of the surface realization process. The task can be considered as operating under low-resource scenario: Table 1 shows that the treebanks are rather small, which poses a challenge for training complex neural models.

## 3 Related Work

The two best-performing approaches in the task of generating sentences from dependency trees have been feature-based incremental text generation (Bohnet et al., 2010; Liu et al., 2015; Puduppully et al., 2016; King and White, 2018) and techniques performing more global input-output mapping (Castro Ferreira et al., 2018; Elder and Hokamp, 2018). The former approaches traverse the input tree, encode nodes using sparse manually defined feature sets as input representations and generate a sentence by extending a candidate hypothesis with an input word that has the highest score among other input words that have not yet been processed. These approaches rely on the observation that natural language production has a preference for shorter dependencies (Gibson, 2000; White and Rajkumar, 2012; King and White, 2018), which facilitates building sentences incrementally.

The second approach linearizes an input graph structure and treats the resulting sequence as the source string and the corresponding sentence as the target. Since the introduction of encoder-decoder (Cho et al., 2014) and sequence-to-sequence (seq2seq) (Sutskever et al., 2014) neural architectures, this line of work has gained a lot of popularity due to the method's simplicity: the input string is encoded into a dense vector and a sentence is being generated token-by-token from the encoded input representation. From an NLP perspective, one of the main research problems in this paradigm has become the choice of the graph encoding strategy. The most popular method is linearizing it into a sequence of tokens and encoding using a variant of a recurrent neural network (RNN) (Gardent et al., 2017; Castro Ferreira et al., 2017; Konstas et al., 2017). Another prominent approach is using graph-to-text neural networks (Song et al., 2018; Trisedya et al., 2018). These methods have shown good results across various tasks, but in the context of surface realization they produced somewhat mixed results: the former ones were successfully used only when being trained on large amounts of data (Elder and Hokamp, 2018), while the latter ones have been only evaluated on the SR'11 Deep Track data and, while performing better than RNN-type encoders, fell short behind feature-based methods (Marcheg-

[2] http://universaldependencies.org/

| Property | Approaches | |
|---|---|---|
| | Neural | Feature-based |
| Data efficiency | ✗ | ✓ |
| Rich context representation | ✓ | ✗ |
| Interpretability | ✗ | ✓ |
| Language coverage | ✓ | ✗ |

Table 2: High-level comparison of the two most prominent approaches to surface realization.

giani and Perez-Beltrachini, 2018).

Each of these approaches has their advantages and limitations (Table 2). Feature-based systems employ carefully crafted feature templates created using expert knowledge, which makes these approaches more interpretable and data efficient, but difficult to port to other domains or languages. The expressiveness of data representation in these systems is largely determined by the complexity of the feature set, which is another limitation of feature-based approaches. These systems are rather slow to train, since feature extraction is defined over a dynamically changing context.

Deep learning, on the other hand, offers a unified language-agnostic framework to train accurate models when abundant training data is available; they are also fast to train (although hyperparameter tuning routines can take a significant amount of time). However, neural models are less interpretable than their sparse-feature counterparts. Also, low-resource scenario still poses a great challenge to complex neural models. The ADAPT system that achieved the best results in SR'18 task on English data (Elder and Hokamp, 2018) used a data augmentation technique which allowed it to leverage 50 times more data than originally provided by the organizers of the workshop. The authors identified the lack of sufficient training data as the major obstacle to training high-performing neural models and mentioned that the system trained only on the original dataset failed to deliver sensible outputs. These results are supported by the work done in other NLP fields. For example, in the machine translation community researchers have found that neural models have a much slower learning curve with respect to the amount of training data, which usually manifests itself as worse quality in low-resource settings, but better performance in high-resource cases (Koehn and Knowles, 2017). In morphological inflection, when trained on small datasets, seq2seq models with additional external (noisy) alignments perform much better than similar systems which learn the alignment information from scratch (Aharoni and Goldberg, 2017).

The success of the encoder-decoder paradigm has given birth to a prominent research trend of finding various ways of utilizing the abundant data on the web. While looking for ways to acquire more data for training even larger models is a promising research topic, an orthogonal direction is pursuing the question of how to design and train more data-efficient models. Our work focuses on this latter point and attempts to address it via data analysis and algorithm design. Taking this into consideration, we build upon the work done by Puzikov and Gurevych (2018), and attempt to improve their method based on the results of our error analysis.

## 4 Approach Description

Before explaining our work, we briefly recap how BINLIN works. It is a pipeline system which generates a sentence from a dependency tree in two stages:

1. Syntactic ordering: convert dependency tree into a binary tree, then traverse the latter to obtain a sequence of lemmas.

2. Morphological inflection: conjugate each lemma into a surface form.

Figure 1 shows a schematic view of the first stage. It relies on the procedure which first runs a breadth-first search (BFS) algorithm on the input dependency tree to obtain *(head, children)* node groups, corresponding to subtrees of depth one. The head of each subtree is used to initialize a binary tree. Then a binary classifier is used to make decisions of positioning the child nodes to the right/left of the head node. Once all the children have been inserted, the construction of a binary tree for the subtree under consideration is finished and the algorithm moves on to the next subtree.

BINLIN uses a multi-layer perceptron model with a logistic regression function on top to predict the probability of node $n_j$ being positioned to the right ($y = 1$) or left ($y = 0$) of node $n_i$ in a binary tree:

$$p(y = 1) = g([\mathbf{x}_i; \mathbf{x}_j]; \theta) = \frac{1}{1 + e^{\theta \cdot [\mathbf{x}_i; \mathbf{x}_j]}} \quad (1)$$

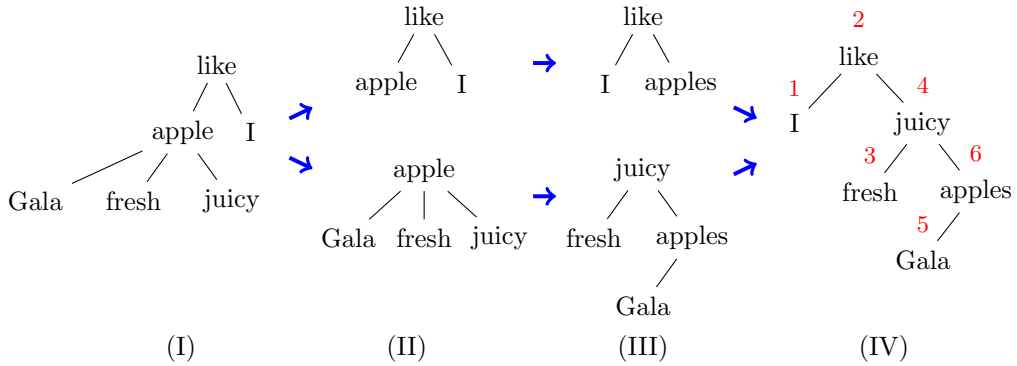


Figure 1: High-level overview of the BINLIN algorithm. Decompose the dependency tree (1) into subtrees of depth one (2), then convert subtrees into equivalent binary trees (3), and merge them. In-order traversal of the merged binary tree (4) produces a sequence of lemmas "I like fresh juicy Gala apple".

Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ are feature representations of $n_i$ and $n_j$, and $\theta$ denotes parameters of the neural network. The decision-making rule is defined by setting a threshold on the output of $g(\cdot)$:

$$decision = \begin{cases} \text{right}, & \text{if } p(y = 1) \geq 0.5, \\ \text{left}, & \text{otherwise.} \end{cases}$$

The algorithm converts local subtrees into binary trees in a bottom-up manner until it reaches the root node. At this point, all dependency nodes have been processed. The constructed local binary trees are merged and the resultant binary tree can be linearized by in-order traversal. Finally, a morphological inflection component, applying a character-level seq2seq model with a hard attention mechanism (Aharoni and Goldberg, 2017), is used to predict a surface form for each lemma in the sequence.

The error analysis of the system outputs provided in (Puzikov and Gurevych, 2018) has shown that the majority of BINLIN's mistakes are caused by the incorrect ordering of lemmas, which is why we focus on the syntactic ordering component and leave the morphological inflection module the same as in the original system.

We argue that there are several directions by which BINLIN could be improved: the way the training data is created and the input encoding schema. In what follows we describe the changes that we made to the original system; the corresponding performance improvements are reported in Section 5.

### 4.1 Modification 1: Training Data Preparation

The first modification we made was improving the way training data for the binary classifier is created. When making training examples, the BINLIN system considers $(n_i, n_j)$ node pairs extracted from subtrees of depth one. For example, in the case of the "I like apple" subtree from Figure 1, one of the training examples the system would add is ("like", "I", *left*), since the word "I" in the sentence is positioned to the left of its head "like".

This procedure seems to be based on the assumption that the system learns position-invariant word order representations, i.e., if the system learns that node $n_j$ should be positioned to the right of $n_i$, it will also be able to deduce that $n_i$ should be inserted to the left of $n_j$. However, it is known that neural networks in general do not have this reasoning ability, and to circumvent this issue, researchers use various data augmentation techniques. For example, in the image processing domain it is common to create additional training images through random rotation, shifts, shear and flips, etc.

In a similar fashion, we propose to enrich the training set with training examples which we call "symmetric": for each $(n_i, n_j, label)$ triple originally considered by BINLIN, we add $(n_j, n_i, op\_label)$ with node positions flipped and having the opposite label. Reusing our previous example: in addition to ("like", "I", *left*), we would add the ("I", "like", *right*) triple to the training set. We run this procedure on all training examples, which effectively doubles the size of the training data.

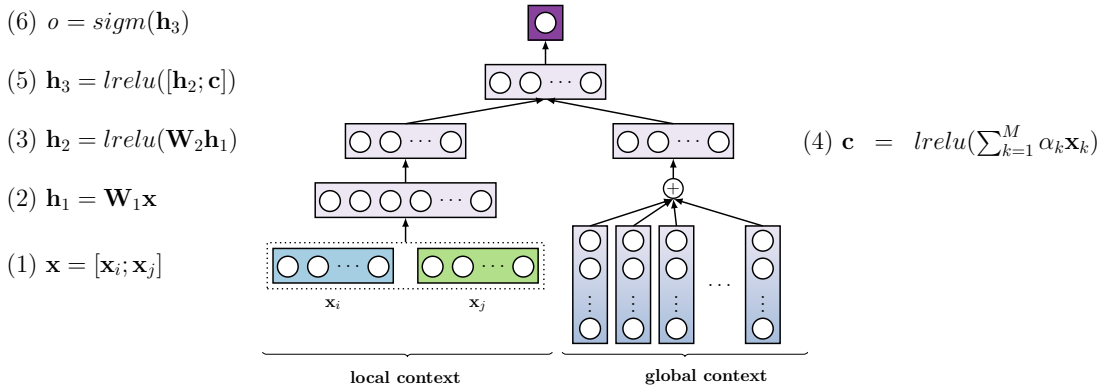


Figure 2: Schematic view of the neural network architecture used as a classifier for the syntactic ordering component of our system.

### 4.2 Modification 2: Encoding Strategy

Figure 2 shows the schematic view of the employed binary classifier; the original BINLIN system would consider the part marked as *local context*, while the remaining *global context* part is our proposed enhancement.

Given a pair of nodes ($n_i$, $n_j$), we first need to extract their features. BINLIN uses a local feature representation of each node, which includes the node itself and graph context in the close proximity – its head and an immediate child. Formally speaking, each node $n_k$ is represented as a vector $\mathbf{x}_k \in \mathbb{R}^{Fd}$, where $F$ denotes the number of extracted features and $d$ is the embedding size. In other words, each dense node representation $\mathbf{x}_k$ is a concatenation of the embeddings for each feature in the feature set $F$. The embedding matrix is denoted as $\mathbf{E} \in \mathbb{R}^{d \times |V|}$, where $V$ is the vocabulary of unique lemmas, POS tags and dependency edge labels, observed in the training data.

The extracted dense feature representations $\mathbf{x}_i$ and $\mathbf{x}_j$ for the two nodes are (1) concatenated to form the input to the classifier, (2) projected onto a lower-dimensional space via a linear transformation, (3) squeezed further via another linear transformation followed by applying the Leaky ReLu function (Maas et al., 2013). The last layer of the BINLIN classifier consists of one node, followed by the sigmoid function.

As mentioned in the previous subsection, in some cases knowing a wider context is crucial for making the correct decision. We decided to enrich the feature representation with a *global context* which encodes all the nodes in the subtree under consideration. We compute it as (4) a weighted sum of the feature representations of these nodes, similar to the attention mechanism of Bahdanau et al. (2015).

We also experimented with the feature set trying to figure out what provides a stronger supervision signal for the binary classifier. The best feature configuration is the same as in the original system with the exception of two additional node features: the number of the node's children and the length of the path from the node to the root of the dependency tree.

## 5 Experiments

The official SR'18 data preserved one-to-one correspondence between sentences and dependency trees, but the alignment between lemmas and surface word forms was omitted, which complicated extracting training data pairs.

Following BINLIN's authors, we used the original UD data files for training all our models (the files contain the same dependency trees as the shared task data, but the order of the tokens is not scrambled and each surface form is aligned with the respective lemma). For a fair comparison with other approaches, system evaluation was done using the official SR'18 data. We used English UD treebank for system development, the evaluation was done on all ten treebanks.

All neural network components were implemented using PyTorch (Paszke et al., 2017). No pretrained embedding vectors or other external resources were used for the experiments. The exact hyper-parameter values for each system component are provided in Appendix A.1.

The syntactic and morphological components were trained separately using the Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.001. We used a batch size of 600 for the syn-

| | | Language | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ar | cs | en | es | fi | fr | it | nl | pt | ru |
| Left/right labels (%) | | 32/68 | 57/43 | 62/38 | 56/44 | 57/43 | 57/43 | 57/43 | 61/39 | 56/44 | 47/53 |
| Node pair | head-dep | 96.27 | 90.26 | 96.14 | 92.67 | 89.21 | 95.14 | 94.26 | 91.51 | 97.29 | 92.85 |
| | dep-dep | 82.81 | 82.78 | 87.64 | 83.85 | 81.43 | 85.12 | 84.98 | 82.52 | 85.6 | 85.56 |

Table 3: The distribution of left/right labels in the training data and the accuracy of predicting a node's relative position with the binary classifier. Two cases are considered: predicting the position of a dependent w.r.t. its head (*head-dep*), and a sibling (*dep-dep*).

| | BLEU | EDIST | NIST |
|---|---|---|---|
| BINLIN | 24.92 | 35.91 | 9.55 |
| + data enrichment | 48.47 | 62.04 | 10.72 |
| + new encoder | 50.67 | 64.05 | 10.82 |
| + new features | 51.15 | 64.78 | 10.82 |
| Upper bound | 65.31 | 85.52 | 11.38 |

Table 4: Cumulative improvements from the modifications to the syntactic ordering procedure that we propose in this work, computed on the English portion of the SR'18 development set.

tactic component and 200 for the morphological component. Both modules were trained for a fixed number of epochs (ten for the syntactic component and 15 for the morphological one).

Since we left the morphological module intact, in Section 5.1 we report evaluation results only for the syntactic ordering component.

### 5.1 Syntactic Ordering

Before evaluating the syntactic ordering module, we conducted a preliminary study in which we tried to validate the dependency locality hypothesis and answer the following question: *Is it possible to accurately predict the relative position of a dependent with respect to its head?*

Table 3 shows the distribution of left/right target labels in the training data and the accuracy of predicting the node's relative position with our system's binary classifier (all proposed modifications applied), both for head-dependent and dependent-dependent node pairs. The latter pairs are relations between sibling nodes in the respective subtree, since at each prediction step the system operates on dependency subtrees of depth one. Note that modeling such relations is a harder task, since siblings in dependency trees do not directly share any grammatical information. However, the surrounding context seems to be enough to make high-accuracy predictions, which supports the dependency locality hypothesis.

We trained the syntactic ordering component and performed its automatic metric evaluation by computing BLEU (Papineni et al., 2002)[3], NIST (Doddington, 2002) and normalized string edit distance (EDIST) scores between the references and system outputs. Note that system outputs contain ordered lemmas, not surface forms, while the references are correctly ordered sequences of inflected surface forms given in the CONLL file.

Table 4 shows the contribution of each of the modifications that we propose in this work; the results are computed on the English SR'18 development set. We also show the maximum metric scores that an ideal syntactic ordering component would get, i.e. an upper bound on its performance. We computed it by retrieving oracle lemma sequences and computing metric scores against the corresponding references. This evaluation was done on English data only, since it was used for system development.

### 5.2 Full Pipeline

We further add the morphological inflection component and evaluate the full pipeline on the SR'18 test data. Table 5 shows the metric scores achieved by the best SR'18 systems (OSU (King and White, 2018) and TILBURG (Castro Ferreira et al., 2018)), BINLIN and the version of BINLIN with the proposed modifications (BINLIN+). We excluded the scores achieved by the ADAPT system, since the system was only evaluated when trained with additional data and such a comparison would not be fair. In order to better assess the performance gains that we obtained from the proposed modifications, the syntactic ordering component of BINLIN+ was trained ten times with different random seeds; we report both the mean scores and standard deviation.

[3]Following the SR protocol, we use the smoothed version and report results using 4-grams (BLEU-4).

| Metric | System | Language | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ar | cs | en | es | fi | fr | it | nl | pt | ru |
| BLEU | OSU | 25.65 | – | **66.33** | **65.31** | **37.52** | 38.24 | – | 25.52 | – | – |
| | TILBURG | – | – | 55.29 | 49.47 | – | **52.03** | **44.46** | 32.28 | 30.82 | – |
| | BINLIN | 16.20 | 25.05 | 29.60 | 32.15 | 23.26 | 20.53 | 23.55 | 22.69 | 24.59 | 34.34 |
| | BINLIN+ | **29.07** ± **0.20** | **54.49** ± **0.26** | 64.70 ± 0.76 | 63.86 ± 0.50 | 37.38 ± 0.59 | 43.40 ± 1.33 | 41.17 ± 0.51 | **50.28** ± **0.53** | **49.01** ± **0.60** | **62.88** ± **0.36** |
| NIST | OSU | 7.15 | – | 12.02 | **12.74** | 9.56 | 8.0 | – | 7.33 | – | – |
| | TILBURG | – | – | 10.85 | 11.11 | – | **9.85** | **9.11** | 8.04 | 7.55 | – |
| | BINLIN | 6.94 | 10.74 | 9.53 | 10.19 | 9.34 | 7.21 | 7.6 | 8.63 | 7.52 | 13.05 |
| | BINLIN+ | **7.53** ± **0.01** | **13.65** ± **0.03** | **12.03** ± **0.05** | 12.59 ± 0.03 | **9.83** ± **0.06** | 8.78 ± 0.11 | 8.63 ± 0.03 | **10.30** ± **0.02** | **9.21** ± **0.05** | **14.43** ± **0.02** |
| EDIST | OSU | 34.37 | – | **68.59** | **59.75** | 47.99 | 44.84 | – | 34.22 | – | – |
| | TILBURG | – | – | 60.32 | 48.47 | – | **51.16** | 46.67 | 37.20 | 40.75 | – |
| | BINLIN | 18.03 | 24.30 | 36.83 | 27.55 | 28.53 | 23.41 | 27.66 | 26.32 | 32.50 | 31.77 |
| | BINLIN+ | **38.11** ± **0.60** | **54.00** ± **0.34** | 66.77 ± 0.77 | 59.00 ± 0.46 | **49.58** ± **0.65** | 45.17 ± 1.10 | **48.50** ± **0.86** | **52.50** ± **0.65** | **59.70** ± **0.78** | **62.80** ± **0.54** |

Table 5: Final results computed on the SR'18 test data. BINLIN+ results include mean scores and standard deviation (scores averaged across ten models trained with different random seed values). Cells with dashes denote cases for which the respective systems have not submitted any output in the shared task.

As can be seen from the table, our modifications bridge the gap between the top-scoring systems and BINLIN. BINLIN+ is the best-performing system for five out of ten languages.

## 6 Error Analysis

In order to better understand the most common errors made by the BINLIN+ system, we manually examined its predictions on the development set. We were focusing predominantly on the syntactic ordering component in its best configuration (i.e. with all the proposed modifications). In what follows we describe the most prominent error types.

**Punctuation**. Generally speaking, the position of punctuation marks is determined not by a specific dependency relation, but rather by discourse-level characteristics of the sentence, since their primary goal is to help the reader interpret text by means of delimiting the contents, dividing it into easy-to-process pieces. Oftentimes there are lexical markers ("so", "because", "although") which signal that, for example, a comma should be inserted before or after a phrase:

- I like chocolate, *because* it is sweet.
- Bryan, you're in, *right*?

However, in UD annotation punctuation marks are considered as dependents of the subtree root. The binary classifier fails to encode discourse information, since it mainly looks for local patterns in head-dependent relations. A more global technique of input encoding might alleviate this issue.

**Contractions**. Spanish, Czech, French, Portuguese, Arabic and Italian treebanks contain annotation of multi-word expressions (MWE). Table 6 shows the number of unique MWE encountered in the training portion of the UD treebanks.

| Language | | | | | |
|---|---|---|---|---|---|
| ar | pt | it | es | cs | fr |
| 3010 | 447 | 361 | 300 | 18 | 12 |

Table 6: Number of multi-word expressions (MWE) in UD treebanks for the languages included in the SR'18 task (only languages with MWE are shown).

The most common case marked as MWE in the UD treebanks is that of contractions which occur when two adjacent words are merged into one. For example, in French the article "les" contracts with the preposition "à" into a compound article "aux". English UD annotation does not contain contractions, which is why when developing BINLIN+ we did not encounter this issue.

Our system predicts the relative position of the contraction elements and attempts to conjugate them separately, but does not perform token merging. The following is an example of a contraction in French:

- Un autel à Jupiter est érigé à l'emplacement de le Temple.
- Un autel à Jupiter est érigé à l'emplacement du Temple.

The first line is what BINLIN+ would predict; the second is what the correct output should be. We suspect that this is the main reason for the performance gap that exists between BINLIN+ and the best-performing approaches on Spanish, French and Italian data.

A possible remedy to this limitation could be modeling syntactic ordering and morphological inflection jointly, but this exploration is out-of-scope for this work. As a quick fix, we added a

| Metric | System | Language | | |
|---|---|---|---|---|
| | | cs | fr | it |
| BLEU | BinLin+ | 54.50 | 43.90 | 40.84 |
| | - w/ MWE | **54.78** | **49.26** | **52.11** |
| NIST | BinLin+ | 13.63 | 8.85 | 8.61 |
| | - w/ MWE | **13.67** | **9.46** | **9.78** |
| EDIST | BinLin+ | 54.00 | 46.20 | 47.57 |
| | - w/ MWE | **54.07** | **49.45** | **53.25** |

Table 7: The result of adding a post-processing step of merging MWE tokens for Czech, French and Italian SR'18 test data.

post-processing step to the outputs of our system [4], whereby we stitch adjacent contraction items into one token. We focused on Czech, French and Italian treebanks, since these languages have very simple contraction cases which we extracted without any knowledge of the respective grammar rules (see Appendix A.2).

Table 7 shows an improvement over all three languages, which suggests that this is indeed a promising direction to investigate in detail, and a more principled treatment of contractions would boost the performance of the system even further.

## 7 Limitations and Future Work

The proposed modifications increase the performance of the baseline BINLIN system significantly, closing the performance gap relative to the state-of-the-art methods to surface realization. Unlike feature-based approaches, the system does not require exhaustive enumeration of the feature templates and is much faster to train. On top of that, it follows a human-designed algorithm, relying on a neural model only to make binary classification decisions which are more transparent than the inner workings of end-to-end neural models. This offers an additional benefit of interpretability and easier debugging. Unlike seq2seq models that occasionally hallucinate content or generate incomprehensible outputs, our system remains faithful to its inputs, since it builds outputs by rearranging input elements and conjugating them.

However, the approach has its limitations. We outline them below and plan to address them in the future.

**Zero-Markov assumption**. The system does not rely on its past predictions when making a current decision. This is a simplifying assumption that sped up system development, but at this point it is a constraint that limits the approach's potential.

**Formalism specificity**. Unlike seq2seq models which can process any input, the approach works only on tree inputs. When changing the input structure one would have to come up with a new graph-to-tree conversion technique. One reassuring fact is that as of now the annotation consistency of available meaning representations is rather low (considering inter-annotator agreement scores), which means that text-based representations like dependencies is the best option one could hope to use in real-life applications.

**Dependent-dependent classification bottleneck**. As can be seen from Table 3, ordering children nodes in a dependency tree is a much harder task, compared to deciding on the position of a child node w.r.t. its head. Most likely this is due to the fact that dependency annotation is not sufficient to make a correct decision, while predicting the order between children nodes might be easier if we change the optimization objective. The masked language modeling approach used in (Liu et al., 2015; King and White, 2018) is very promising in this regard and we plan to investigate it in the future.

## 8 Conclusion

In this work we extended the binary linearization technique for generating sentences from dependency trees. The modifications are motivated by the results of the error analysis of the baseline system and, when applied, significantly improve its accuracy. The resultant system reaches competitive performance in a multilingual setting, while preserving more interpretable behavior and higher data efficiency than the competitors.

## Acknowledgments

This work was supported by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1 and grant GU 798/17-1) and the DFG-funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1). The first author of the paper is supported by the FAZIT Foundation scholarship and the German Federal Ministry of Education and Research (BMBF) as part of the Software Campus program

[4] For the syntactic component that we trained with ten different random seeds, we chose one variant randomly.

under the promotional reference 01IS17050.

We thank Henry Elder and Anastasia Shimorina for the insightful discussions and our colleagues Michael Bugert, Yang Gao, Ji-Ung Lee and Jonas Pfeiffer who provided suggestions that greatly assisted our research.

## References

Roee Aharoni and Yoav Goldberg. 2017. Morphological Inflection Generation with Hard Monotonic Attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The First Surface Realisation Shared Task: Overview and Evaluation Results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France.

Bernd Bohnet, Leo Wanner, Simon Mille, and Alicia Burga. 2010. Broad Coverage Multilingual Deep Sentence Generation with a Stochastic Multi-level Realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 98–106, Beijing, China.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 1–10, Santiago de Compostela, Spain.

Thiago Castro Ferreira, Sander Wubben, and Emiel Krahmer. 2018. Surface realization shared task 2018 (SR18): The Tilburg university approach. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 35–38, Melbourne, Australia.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder-decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, San Francisco, CA, USA.

Henry Elder and Chris Hokamp. 2018. Generating high-quality surface realizations using data augmentation and factored sequence models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53, Melbourne, Australia.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain.

Edward Gibson. 2000. The dependency locality theory: a distance-based theory of linguistic complexity. *Image, Language, Brain: Papers from the First Mind Articulation Project Symposium*.

David King and Michael White. 2018. The OSU realizer for SRST '18: Neural sequence-to-sequence inflection and incremental locality-based linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 39–48, Melbourne, Australia.

Diederik Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada.

Irene Langkilde-Geary. 2002. An Empirical Verification of Coverage and Correctness for a General-purpose Sentence Generator. In *Proceedings of the 2nd International Natural Language Generation Conference*, pages 17–24, Harriman, New York, USA.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based Syntactic Linearization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 113–122, Denver, Colorado.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, USA.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12, Melbourne, Australia.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS 2017 Workshop Autodiff*, Long Beach, California, USA.

Ratish Puduppully, Yue Zhang, and Manish Shrivastava. 2016. Transition-based Syntactic Linearization with Lookahead Features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 488–493, San Diego, California.

Yevgeniy Puzikov and Iryna Gurevych. 2018. BinLin: A simple method of dependency tree linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 13–28, Melbourne, Australia.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, Melbourne, Australia.

Michael White and Rajakrishnan Rajkumar. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255, Jeju Island, Korea.

# A Appendix

## A.1 Hyper-parameter Details

We did not perform any hyper-parameter tuning, all values were chosen based on our intuition and what we have observed in the research literature.

**Morphological Component**. The morphological inflection component was implemented using character-level LSTM networks, a two-layer bidirectional encoder and a one-layer unidirectional decoder with hidden layer sizes of 200. We used 200-dimensional randomly-initialized embeddings.

We did not lowercase the data, maintained a fixed size vocabulary of 250 characters, and for training used only sequences of maximum 30 characters long (both source and target sides).

**Syntactic Ordering Component**. The syntactic ordering component was implemented using feed-forward networks; the corresponding hidden layer sizes were fixed at 100 and 64 dimensions (weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ from Figure 2). We used 200-dimensional randomly initialized word-level case-sensitive embeddings.

### A.2 Contraction Rules

The following tables contain the contraction rules we used as a post-processing step described in Section 6. The rules were created by extracting lines with contractions in the UD CONLL files and analyzing the contraction patterns.

| | |
|---|---|
| à le → au | de lequel → duquel |
| à les → aux | de lesquels → desquels |
| à lequel → auquel | de lesquelles → desquelles |
| à lesquels → auxquels | en les → ès |
| à lesquelles → auxquelles | vois ci → voici |
| de le → du | vois là → voilà |
| de les → des | |

Table 8: French contraction rules.

| | |
|---|---|
| di il → del | a l'→ all' |
| di lo → dello | a le → alle |
| di la → della | a i → ai |
| di l' → dell' | a gli → agli |
| di i → dei | su il → sul |
| di gli → degli | su la → sulla |
| di le → delle | su lo → sullo |
| a il → al | su gli → sugli |
| a lo → allo | con il → col |
| a la → alla | con i → coi |

Table 9: Italian contraction rules.

| | |
|---|---|
| aby by → aby | když by → kdyby |
| Aby by → Aby | Když by → Kdyby |

Table 10: Czech contraction rules.