The DipInfoUniTo Realizer at SR'19: Learning to Rank and Deep Morphology Prediction for Multilingual Surface Realization

Alessandro Mazzei
Dipartimento di Informatica
Universit degli Studi di Torino
Corso Svizzera 185, 10153 Torino
mazzei@di.unito.it

Valerio Basile
Dipartimento di Informatica
Universit degli Studi di Torino
Corso Svizzera 185, 10153 Torino
basilei@di.unito.it

Abstract

We describe the system presented at the SR'19 shared task by the DipInfoUnito team. Our approach is based on supervised machine learning. In particular, we divide the SR task into two independent subtasks, namely word order prediction and morphology inflection prediction. Two neural networks with different architectures run on the same input structure, each producing a partial output which is recombined in the final step in order to produce the predicted surface form. This work is a direct successor of the architecture presented at SR'19.

1 Introduction

Surface Realisation (SR) is one of the main tasks involved in Natural Language Generation. SR focuses the final macro-step of the standard NLG pipeline defined by Reiter and Dale (2000), therefore involving the production of producing natural language sentences and longer documents from formal abstract representations. Such input is assumed to come from an external source, such as a macro-planning and micro-planning pipeline, and therefore it will contain all the necessary information to create the final natural language output. Generating a **correct** and **fluent** output in a target natural language is the main responsibility of the SR component. In this paper, we report on the system submitted to the second edition of the Surface Realization Shared Task (Mille et al., 2019, SR'19), organized in the context of the Multilingual Surface Realization Workshop in 2019.

The SR task, in the version proposed at SR'19, considers the surface realization of Universal Dependency (UD) trees, i.e., syntactic structures where the words of a sentence are linked by labeled directed arcs. In particular, UD represents natural language syntax with trees where each node is a word. The labels on the arcs indicate the syntactic relation holding between each word and its dependent words — see an example in Figure 1a. Our approach to the SR task is based on supervised machine learning. In particular, we draw inspiration from Basile (2015), subdividing the task into two independent subtasks, namely **word order prediction** and **morphology inflection prediction**. Two neural networks with different architectures run on the same input structure, each producing a partial output which is recombined in the final step in order to produce the predicted surface form. This work is a direct successor of the architecture presented at last year's edition of the shared task (Mille et al., 2018) and experimented in more detail in (Basile and Mazzei, 2018b). With respect to the last year previous system, there are two major differences: i) we took advantage of a high-performance computing public platform (see acknowledgments) in order to optimize the learning parameters and avoid overfitting; ii) we select the best model by using the Kendall's Tau (Kendall, 1938, τ), a rank correlation measure used to score the word order at the subtree level predicted by the model, at different training epochs (Basile and Mazzei, 2018b). By following the approach of (Basile, 2015), the fitness of the model at each epoch is computed by using the number of incorrect item inversions (intrinsic evaluation), rather than on the downstream task score (see Section 3).

In the following, we refer to our system by using the name *DipInfo-UniTo realizer*.

2 Method

In this section, we detail the two main components developed to approach word order prediction (2.1) and morphology inflection (2.2).

Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019), pages 81–87
Hong Kong, China, November 3rd, 2019.
https://doi.org/10.18653/v1/P17

2.1 Word Ordering

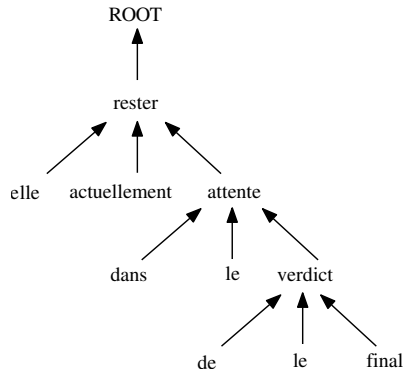
We formulate the task of predicting the correct order of words in a sentence in terms of reordering the subtrees in its syntactical structure. The algorithm works in three steps:

1. splitting the unordered tree into single-level unordered subtrees;

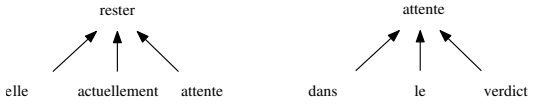
2. predicting the local word order for each subtree;

3. recomposing the single-level ordered subtrees into a single multi-level ordered tree to obtain the global word order.

The first step splits the input UD tree into several single-level unordered trees composed by a head (the root) and all its dependents (the children), similarly to Bohnet et al. (2012).



(a) Tree corresponding to the French sentence "Elle reste actuellement dans l'attente de le verdict final." (*"She is currently waiting for the final verdict."*)



(b) Two subtrees extracted from the main tree.

Figure 1: Splitting the input tree into subtrees to extract lists of items for learning to rank.

An example is shown in Figure 1: from the (unordered) tree representing the sentence "*Elle reste actuellement dans l'attente de le verdict final.*" (1a), each of its component subtrees (limited to one-level dependency) is considered separately (1b). The head and the dependents of each subtree form an unordered list of lexical items. We leverage the flat structure of the subtrees to extract structures that are suitable as input to the learning to rank approach we propose, carried out by the next step of the pipeline.

The second step of the algorithm predicts the relative order of the head and the dependents of each subtree with a *learning to rank* approach. We employ the list-wise learning to rank algorithm *ListNet* (Cao et al., 2007). The limited cardinality of the lists to rank makes it advantageous to use a list-wise approach, as opposed to pair-wise or point-wise approaches, without an unmanageable increase of the computation load. ListNet is a generalized version of the pairwise learning to rank algorithm RankNet (Burges et al., 2005). ListNet employ a list-wise loss function based on the *top-one probability*, i.e., the probability of an element of being the first one in the ranking. The top-one probability model approximates the *permutation probability* model that assigns a probability to each possible permutation of an ordered list. This approximation is necessary to keep the problem tractable by avoiding the exponential explosion of the number of permutations. Formally, the top-one probability of an object j is defined as

$$P_s(j) = \sum_{\pi(1)=j, \pi \in \Omega_n} P_s(\pi)$$

that is, the sum of the probabilities of all the possible permutations of n objects (denoted as Ω_n) where j is the first element. $s = (s_1, ..., s_n)$ is a given list of *scores*, i.e., the position of elements in the list. Considering two permutations of the same list y and z (in the case of the SR task, the predicted order and the reference order) their distance is computed using cross entropy. The distance measure and the top-one probabilities of the list elements are used to compute the loss function:

$$L(y, z) = -\sum_{j=1}^{n} P_y(j) log(P_z(j))$$

A linear neural network model provides the learning environment, using the list-wise loss function above. ListNet takes as input a sequence of ordered lists of feature encoded as numeric vectors. The weights of the network are updated over several epochs by computing distance between the reference ranking and the prediction of the model (list-wise cost function) and passing its value to the gradient descent algorithm for optimization. We used an implementation of ListNet[1] that was previously applied in a surface realization task

[1] https://github.com/valeriobasile/listnet

with a similar supervised setting (Basile, 2015). On top of the core ListNet algorithm, this implementation features a regularization parameter to prevent overfitting.

We manually engineer the features for the supervised learning in the word order module. We use several word-level features encoded as one-hot vectors, namely: the universal POS-tag, the treebank specific POS tag, the morphology features and the head-status of the word (head of the single-level tree vs. leaf). We also include vectorial word representations of two different kinds. Content words are open-class word lemmas, and are represented by language-specific, pre-trained word embeddings. In particular, we employ the multilingual model Polyglot (Al-Rfou' et al., 2013). Function words are closed-class word lemmas, and are encoded as one-hot bag-of-words vectors. An implementation of the feature encoding for the word ordering module of our architecture is available online[2].

The third step of the word ordering algorithm reconstructs the global order (i.e., at the sentence level) from the local order of the one-level trees. Note that this approach works under the hypothesis of *projectivity*. The DipInfo-UniTo realizer cannot predict the correct word order for non-projective sentences. If the local reordering of the one-level tree T_1^h with root h and children $c_1...c_M$ produces an order of nodes $n_1 n_2 ... n_{M+1}$, the hypothesis of projectivity implies that in the global word order the position of all the children of the node n_j will be after the position of the node n_{j-1} and before the position of the node n_{j+1}. So, the node global order (O) of a k-level tree T_k^h rooted by the node h and with children $c_1...c_M$ can be rewritten formally in terms of the local order as:

$$O(T_k^h) = \begin{cases} h & \text{if } k=0 \\ O_{ln}(h, c_1, ..., c_M) & \text{if } k=1 \\ O_{ln}(h, O(T_{k-1}^{c_1}), ..., O(T_{k-1}^{c_M})) & \text{if } k>1 \end{cases}$$

where $O_{ln}(h, c_1, ..., c_M)$ is the permutation learned by the ListNet algorithm from the training set and parametrized over the feature set $F(h, c_1, ..., c_M)$, that is

$$O_{ln}(h, c_1, ..., c_M) \stackrel{def}{=} P_{ListNet}^{F(h,c_1,...,c_M)}(h, c_1, ..., c_M)$$

[2] https://github.com/alexmazzei/ud2ln

2.2 Morphology Inflection

The second half of our proposed architecture is the morphology inflection component. We consider this task an alignment problem at the level of character, and approach it with a *sequence-to-sequence* supervised model. We employ the deep neural network based on a hard attention mechanism introduced by Aharoni and Goldberg (2017). The model consists of a neural network in an encoder-decoder setting. At each training step, the model can either write a symbol to the output sequence, or move the attention pointer to the next state of the sequence. This architecture models the monotonic alignment between the input and output sequences, allowing the freedom to condition the output on the entire sequence in input.

We employ all the morphological features provided by the UD annotation and the dependency relation between the target word and its head. We transform the training CONLL files into a set of $((lemma, features), form)$ tuples, in order to learn the neural inflectional model associating a $(lemma, features)$ to the corresponding $form$. An example of training instance for the morphology inflection module is the following:

```
lemma: rester
features:
  uPoS=VERB
  rel=root
  Number=Sing
  Mood=Ind
  Person=3
  Tense=Pres
  VerbForm=Fin
form: reste
```

Corresponding to the word form *reste*, an inflected form (3rd person, present, indicative) of the lemma *rester* (to remain, to stay).

3 Experiments

Since our approach does not rely on language specific procedures or hand-made rules, we tested it on three languages, namely English, French and Chinese, in order to cover different families of languages. We were not able to provide results for other language for computational time constraints. For word ordering, we ran the system on a virtualized GNU/Linux box with 16-core and 64GB of RAM. The computation time of the word ordering component was around one hour per epoch for

the English language, which had the larger data set among the three languages that we considered. For morphology inflection, we used a GNU/Linux box with NVIDIA Tesla K40c GPU computing capability. Similarly to word ordering, the computation time for each epoch in morphology inflection was around one hour for the English language.

3.1 Pipelines

We designed two processing pipelines for the training and testing phase, as depicted in Figure 2. We applied the pipelines separately for each of the tested languages (EN-FR-ZH).

In the training pipeline, we created two distinct files starting from the UD treebank training files. The first file contains morphological information (that is $((lemma, features), form)$, see Section 2.2), used to create the morphological inflection model with the deep learning architecture described in Section 2.2. The second file contains the vector representation of the tree features (word embeddings or one-hot for function words, morphological features, etc.) and it is used to create the word order model by using the linear neural network architecture described in Section 2.1.

In the testing pipeline, we created two distinct files starting from the test files provided from the organizers. Both files are created with the same procedures of the training pipelines. The first file was used to test the morphological neural model and to create a mapping from the lemma-features pair to the inflected form. The second file was used to test the word order model by providing the local word orders of the subtrees and the global word order at the sentence level. In a subsequent step, the information from the morphological map and from the word ordered trees are merged into one single complete, CONLL-compliant tree structure. Finally, the trees are *detokenized* (see 3.3) in order to produce the sentences that are submitted as the final output of the system.

3.2 Datasets

The rules of the shallow track for the SR'19 do not allow to use external resources to train the surface realizer. However, of lexical resources such as word embedding and neural language models are allowed. In order to investigate about the syntactic information contained in the Universal Dependency format and its appropriateness for the SR task, we decided to focus on information derived from the Universal Dependency project (Nivre et al., 2016), with the only exception of precompiled embeddings to encode of the open-class words.

The task organizers provided twenty training files and twenty development files, derived from the version 2.2 of the UD dataset for the eleven languages included in the shallow track. In particular, modified versions of the original treebanks were provided, where the information about the original word order was replaced by a random ordering. Moreover, the original UD feature set was enriched with new features, i.e. `original_id` containing the original position of the word in the sentence. For a number of specific parts of speech (e.g. `PUNCT`, punctuation), the feature `lin` is added, containing the original relative position of the word with respect to its head. Note that the `lin` feature, in contrast to the `original_id` feature, is present in the test file too.

We decided not to use the `lin` feature, therefore we employ the original versions 2.2 of the treebank files (provided by the shared task organizer) since they contain both the gold word order and the inflected forms of the word. However, during the conversion of the dependency trees into a vector form (see Section 2.1), we ignored the information about word ordering and inflected forms.

For all the three language processed, we decided to use an *holistic* approach to learning, that is, we built one single probabilistic model (i.e. one for word ordering and one for morphology inflection) by using one single training file obtained by merging together all the training file for a specific language.

3.3 Detokenization

In order to produce the final result of the realization, one needs to transform the UD tree produced by the DipInfoUniTo realizer into a single string containing the sentence. Since the final goal of the task is to reproduce an output sentence close to the original sentence, in detokenized form, we post-processed the English and French syntax trees, in two additional phases, namely *contraction* and *space removal*.

In contraction, the sentence was modified in order to produce the contracted form for some specific multi-word constructions. In particular, in French there are two linguistic phenomena to account for, typical of romance languages, namely *articulated preposition* and *clitics*. Since they are

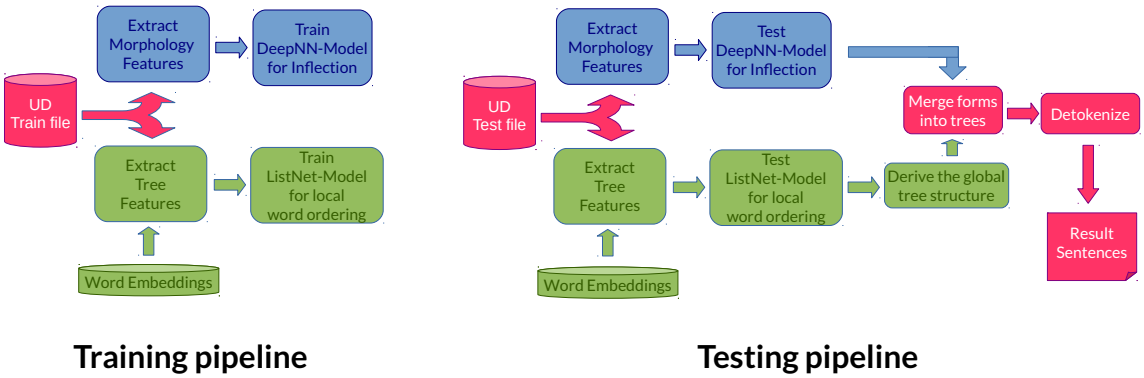


Figure 2: The training and testing pipelines, originally reported in (Basile and Mazzei, 2018a).

special case of multi-word expressions, both articulated prepositions and clitics have a special annotation status into UD treebanks, that we exploited to obtain the contracted form (see (Basile and Mazzei, 2018a) for details).

Moreover, each language has additional specific rules for the treatment of space between words and punctuation. In order to treat this specific cases we used the detokenizer script provided in the *moses* project[3]. The detokenizer provides specific rules for English and French.

3.4 Results

The final results have been produced by training the neural models for word ordering and morphology inflection for exactly 100 epochs and by using the development set provided by the organizer to select the best model. Note that the morphology inflection deep neural network uses a standard accuracy measure to select the best epoch-model. In contrast, the performance of word ordering is measured in terms of average Kendall's Tau (Kendall, 1938, τ), a rank correlation measure used to score the rankings predicted by a specific epoch model for every subtree (cf. (Basile and Mazzei, 2018b)). τ measures the similarity between two rankings by counting how many pairs of elements are swapped with respect to the original ordering out of all possible pairs of n elements:

$$\tau = \frac{\#concordant\_pairs - \#discordant\_tpairs}{\frac{1}{2}n(n-1)}$$

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/detokenizer.perl

.

In Table 1 the official scores of the DipInfoUniTo system for English, French and Chinese datasets are reported, computed in terms of the automatic metrics BLUE, NIST, and DIST. With respect to the other teams, our results score are in the lower half of the leaderboard, raking between $8th$ and $9th$ position depending on metrics and detokenization over the 12 teams participating to the T1-shared task. Since there is no notable difference in the ranking of our system in tokenized and detokenized ranks, we hypothesize that our detokenization procedure is similar to that of the others teams.

It is interesting to note the the best values for BLEU and NIST have been obtained on the `en_pud-ud-test` test file. This fact seems to suggest that our model does not overfit on a specific domain, which could be a consequence of our design choice to produce domain-agnostic models for each language.

Moreover, since the performance of the system for English and French does not correlate to the dataset size, we speculate that there are other linguistic features influencing the performance of the system, e.g., average length of the sentences, or the complexity of the lexicon. More experimentation is necessary to investigate on this speculation.

4 Conclusions and Future Work

In this paper, we described the DipInfoUnito realizer and its participation to the SR'19 competition. With respect to the previous year, we have introduced the evaluation of the models produced

	Detokenized			Tokenized		
	BLEU	NIST	DIST	BLEU	NIST	DIST
en_ewt-ud-test	37.88	10.03	60.10	43.5	11.56	60.13
en_gum-ud-test	39.59	9.82	56.28	44.24	11.15	56.04
en_lines-ud-test	26.83	8.56	52.97	32.42	10.05	53.21
en_partut-ud-test	29.47	7.81	51.03	35.11	9.08	51.15
fr_gsd-ud-test	25.86	8.19	47.48	27.04	9.58	47.33
fr_partut-ud-test	36.77	7.84	55.08	37.69	8.57	54.85
fr_sequoia-ud-test	27.4	8.49	49.13	28.95	9.72	48.70
zh_gsd-ud-test	0.02	0.01	32.10	32.87	11.16	50.57
en_pud-ud-test (OoD)	40.73	10.43	53.53	45.61	11.81	53.26
en_ewt-Pred-HIT-edit (Pred)	0.00	0.00	0.00	43.23	11.44	58.72
en_pud-Pred-LATTICE (Pred)	39.63	10.28	54.61	44.06	11.67	54.42

Table 1: The official scores of the DipInfoUniTo system for English, French and Chinese datasets, in terms of the automatic metrics BLUE, NIST, and DIST. Note that the label OoD stands for *out of domain* and the label Pred stands for *predicted* values of the features values.

at each epoch by the word ordering neural network in the training pipeline in terms of Kendall's Tau. Due to computational constraints, we have been able to run our systems on three languages only, namely English, French and Chinese. The final results rank our system in the the mid-lower part of the final ranking. We believe that a more efficient implementation of the word ordering, i.e., the neural network implementing the ListNet algorithm, could improve the results.

With respect to the problem of generalizing our approach to account for non-projective structure, we intend to develop our work in two directions. First, decomposing the original dependency tree into structures with a wider *domain of locality*. By following the direction by Joshi and Rambow (2003), we plan to model the prediction of local order with more complex structures. Second, as pointed out in (Basile, 2015, Chapter 7), learning the global order of the words rather (or in addition to) their local order. However, learning the word order globally may impact the transparency of the system, therefore a careful balance between performance and explainability must be achieved. On the other hand, global order may alleviate the problem of non-projective sentences, that is currently an issue with the local ordering approach. In future work, we plan to devise a two-step approach to leverage both approaches and learn jointly from both global and local order, e.g., in a multi-task learning fashion.

Aknowledgment

We thank the GARR consortium for the use of GARR Cloud Platform[4]. Valerio Basile was partially funded by Progetto di Ateneo/CSP 2016 (*Immigrants, Hate and Prejudice in Social Media*, S1618_L2_BOSC_01).
Alessandro Mazzei was partially supported by the HPC4AI project, funded by the Region Piedmont POR-FESR 2014-20 programme (INFRA-P call).

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, ACL 2017*, pages 2004–2015.

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *CoNLL*, pages 183–192. ACL.

Valerio Basile. 2015. *From Logic to Language : Natural Language Generation from Logical Forms.* Ph.D. thesis, University of Groningen, Netherlands.

Valerio Basile and Alessandro Mazzei. 2018a. The dipinfo-unito system for srst 2018. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 65–71. Association for Computational Linguistics.

Valerio Basile and Alessandro Mazzei. 2018b. Neural surface realization for italian. In *Proceedings of the*

[4] https://cloud.garr.it

Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Torino, Italy, December 10-12, 2018.

Bernd Bohnet, Anders Björkelund, Jonas Kuhn, Wolfgang Seeker, and Sina Zarrieß. 2012. Generating non-projective word order in statistical linearization. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 928–939. Association for Computational Linguistics.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA. ACM.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA. ACM.

Aravind K. Joshi and Owen Rambow. 2003. A formalism for dependency grammar based on tree adjoining grammar.

M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

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. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, and Leo Wanner. 2019. The Second Multilingual Surface Realisation Shared Task (SR'19): Overview and Evaluation Results. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR), 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Hong Kong, China.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.