Surface Realization Shared Task 2018 (SR18): The Tilburg University Approach

Thiago Castro Ferreira and **Sander Wubben** and **Emiel Krahmer**
Tilburg center for Cognition and Communication (TiCC)
Tilburg University
The Netherlands
{tcastrof,s.wubben,e.j.krahmer}@tilburguniversity.edu

Abstract

This study describes the approach developed by the Tilburg University team to the shallow track of the Multilingual Surface Realization Shared Task 2018 (SR18). Based on Castro Ferreira et al. (2017), the approach works by first preprocessing an input dependency tree into an ordered linearized string, which is then realized using a statistical machine translation model. Our approach shows promising results, with BLEU scores above 40 for 4 different languages in development and test sets (English, French, Italian and Spanish) and above 30 for the Dutch and Portuguese languages. The model is publicly available[1].

1 Introduction

This study presents the approach developed by the Tilburg University team for the shallow track of the Multilingual Surface Realization Shared Task 2018 (SR18) (Mille et al., 2018). Given a lemmatized dependency tree without word order information, the goal of this task consists of linearizing the lemmas in the correct order and realizing them as a surface string with the proper morphological form.

For the task, parallel datasets were provided for 10 different languages and we developed our model for 6 out of the 10 languages (Dutch, English, French, Italian, Portuguese, Spanish). We started from the surface realization approach described in Castro Ferreira et al. (2017), where a semantic graph structure is first preprocessed into a preordered linearized form, which is subsequently converted into text using a statistical machine translation model implemented in Moses (Koehn et al., 2007). However for this shared task, instead of a semantic structure, our current approach preprocesses the lemmas of the dependency tree into an ordered linearized version.

Although for the task sufficient parallel corpus data, pairing dependency tree inputs to textual outputs, were made available to train and test our approach, alignments between the source lemmas and the target words were not provided. Since this information is crucial to train our approach, we implemented a method consisting of four consecutive strategies to obtain the alignments.

Except for two languages (Dutch and Portuguese, ironically), our approach showed promising results, with BLEU scores higher than 40 in development and test sets. In the remainder of this paper, we describe the method in more detail: Section 2 explains the alignment method, Section 3 describes the general approach, Section 4 describes the results and discussion of our approach in development and test sets and, finally, Section 5 concludes the study, also describing future work which can be done to improve the model.

2 Alignment

To train and test the models for multilingual surface realization, parallel corpora pairing lemmatized dependency trees and their textual realizations were made available in 10 different languages. However, no word alignments between the two sides were provided, which is a crucial information to train part of our approach. So, to obtain this information, we implemented four sequential alignment strategies.

Before applying these strategies, we first used the spaCy software[2] to tokenize, lemmatize and dependency parse the target texts. Since spaCy

[1] https://github.com/ThiagoCF05/Dep2Text

[2] https://spacy.io/

Proceedings of the First Workshop on Multilingual Surface Realisation, pages 35–38
Melbourne, Australia, July 19, 2018. ©2018 Association for Computational Linguistics

only provides models for 6 out of the 10 covered languages, the approach described in this study is limited to these six. For the Portuguese language, we also parsed the contractions between preposition and determiners (e.g., *da/do* and *na/no*, corresponding to *of the* and *in the* in English) into two single tokens (*de a/de o* and *em a/em o* for the previous examples).

Once the target texts were preprocessed, the first step simply compares the lemmas of the source side with the words on the target side. If a lemma on the source side and a word on the target side matched with each other and not with any other element, they were aligned.

In the second step, we applied the same comparison used in the first step, but now for the lemmas of the target words. If lemmas on source and target sides only matched each other and no other element, the source lemma was aligned to the corresponding target word.

The third step aimed to solve situations where a source lemma matches more than one element on the other side, by aligning the source and target lemmas with the same dependency tags which only matched each other.

Finally, the fourth step matched the remaining source and target lemmas of a parallel instance with the shortest string distance.

Based on the alignment between source and target sides of a parallel instance, we trained our approach, as described in the following section.

3 Model

Our model is based on the NLG approach introduced in Castro Ferreira et al. (2017), where a semantic graph structure is first preprocessed into a preordered linearized form, which is then converted into its textual counterpart using a statistical machine translation model implemented with Moses. However for this task, instead of a semantic structure, our approach takes as input a lemmatized dependency tree. In the next sections, we explain the preprocessing and translation phases in more detail.

3.1 Preprocessing

The preprocessing method consists of two steps: linearization and partial realization.

Linearization aims to linearize a dependency tree input without punctuation nodes into an ordering string format. Our approach is similar to the 2-step classifier introduced in Castro Ferreira et al. (2017). Its pseudo-code is depicted in Algorithm 1.

The approach starts by deciding which first-order child nodes are most likely to be before and after its head node (lines 1-13). It uses a maximum entropy classifier ϕ_1, trained for each language based on the relevant aligned training set. As features, this classifier uses the lemmas as well as the dependency and part-of-speech tags of the head and child nodes.

Once the nodes are split into a group of nodes before and another group of nodes after their heads, each one of these groups is ordered with an algorithm similar to the MergeSort one (lines 14-24 and function $SORT$). To decide the order of two child nodes of a same group, we use a second maximum entropy classifier ϕ_2, also trained for each language based on the corresponding aligned training set. As features (line 44), it uses the lemmas as well as the dependency and part-of-speech tags of the head and the two child nodes involved in each comparison.

Partial realization aims to partially realize the lemmas in the linearized representation. For each language, it uses a lexicon created based on the aligned information extracted from the datasets, as explained in Section 2. Given a lemma and its features, our approach looks for the most likely morphological form in the lexicon.

3.2 Translation

For each one of the 6 languages which our approach covers, we built a phrase-based machine translation model using the Moses toolkit (Koehn et al., 2007). The MT model aims to convert a linearized dependency tree generated during the preprocessing step into text, adding the proper punctuation marks.

Most of the model settings were copied from the Statistical MT system introduced in Castro Ferreira et al. (2017). At training time, we extract and score phrases up to the size of nine tokens. As feature functions, we used direct and inverse phrase translation probabilities and lexical weighting, as well as word, unknown word and phrase penalties. These feature functions were trained using alignments from the training set obtained by MGIZA (Gao and Vogel, 2008) (not by the ones extracted according to Section 2). Model weights were tuned on the development data using 60-batch

Algorithm 1 Linearization method

```
Require: $depTree$
function LINEAR($root$, $orderId$)
    $before \leftarrow \emptyset$
    $after \leftarrow \emptyset$
    $edges \leftarrow getEdges(depTree, root)$
    for all $edge \in edges$ do
        $node \leftarrow edge.node$
        $features_1 \leftarrow f_1(depTree, root, node)$
        if $\phi_1(features_1) == before$ then
            $before \leftarrow before \cup node$
        else
            $after \leftarrow after \cup node$
        end if
    end for
    $before \leftarrow$ SORT($before$)
    for all $node \in before$ do
        $orderId \leftarrow$ LINEAR($node$, $orderId$)
    end for
    $root.orderId \leftarrow orderId$
    $orderId \leftarrow orderId + 1$
    $after \leftarrow$ SORT($after$)
    for all $node \in after$ do
        $orderId \leftarrow$ LINEAR($node$, $orderId$)
    end for
    return $orderId$
end function

function SORT($nodes$)
    if $|nodes| < 2$ then
        return $nodes$
    end if
    $half \leftarrow |nodes|/2$
    $end \leftarrow |nodes|$
    $nodes_1 \leftarrow$ SORT($nodes[0, half)$)
    $nodes_2 \leftarrow$ SORT($nodes[half, end]$)
    $ordNodes \leftarrow \emptyset$
    while $|nodes_1| > 0$ or $|nodes_2| > 0$ do
        if $|nodes_1| == 0$ then
            $ordNodes \leftarrow ordNodes \cup$ POP($nodes_2$)
        else if $|nodes_2| == 0$ then
            $ordNodes \leftarrow ordNodes \cup$ POP($nodes_1$)
        else
            $node_1 \leftarrow$ POP($nodes_1$)
            $node_2 \leftarrow$ POP($nodes_2$)
            $features_2 \leftarrow f_1(depTree, node_1, node_2)$
            if $\phi_2(features_2) == before$ then
                $ordNodes \leftarrow ordNodes \cup node_1$
                $ordNodes \leftarrow ordNodes \cup node_2$
            else
                $ordNodes \leftarrow ordNodes \cup node_2$
                $ordNodes \leftarrow ordNodes \cup node_1$
            end if
        end if
    end while
    return $ordNodes$
end function

LINEAR($depTree.root$, 0)
```

Language	BLEU
Dutch	35.26
English	58.92
French	59.28
Italian	50.33
Portuguese	54.76
Spanish	54.88

Table 1: BLEU scores of our approach in the tokenized development sets.

Language	BLEU	DIST	NIST
Dutch	32.28	57.81	8.05
English	55.29	79.29	10.86
French	52.03	55.54	9.85
Italian	44.46	58.61	9.11
Portuguese	30.82	60.70	7.55
Spanish	49.47	51.73	11.12

Table 2: BLEU, DIST and NIST scores of our approach in the original (non-tokenized) test sets.

MIRA (Cherry and Foster, 2012) with BLEU as the evaluation metric. A distortion limit of 6 was used for the reordering models. We used two lexicalized reordering models: a phrase-level (phrase-msd-bidirectional-fe) (Koehn et al., 2005) and a hierarchical-level one (hier-mslr-bidirectional-fe) (Galley and Manning, 2008). At decoding time, we used a stack size of 1000. To rerank the candidate texts, we used a 5-gram language model trained on the EuroParl corpus (Koehn, 2005) using KenLM (Heafield, 2011).

4 Results and Discussion

Table 1 summarizes the BLEU scores we obtained on the tokenized development data for the 6 relevant languages. For all languages (except Dutch) our approach yielded BLEU scores of 50 or higher, with the highest results obtained for French (with a BLEU score of 59).

Table 2 depicts the BLEU, DIST and NIST scores of our approach on the test sets for the 6 target languages. For most languages, the BLEU scores on development and test set are comparable, albeit somewhat lower. The scores for Portuguese, however, are substantially lower, which we explain as follows. In contrast to the results on the development set, computed by the authors for the lowercased tokenized version of the set, the scores on the test, generated by the organizers, computed the metrics comparing the generated texts with the lowercased and non-tokenized gold-standards. Although we parsed the contractions between preposition and determiners in this language to align source and target data (as explained in Section 2), our approach did not generate these contractions. That is the case, for instance, in the sentence *"greve **na** televisão pública francesa"* (i.e., *strike on the French public television*), generated by our model with the parsed contractions: *"greve **em a** televisão pública francesa"*. We assume this problem explain most of the drop in the BLEU score of the test set in comparison with the development one.

The low scores for Dutch in both development and test set might be due to the way non-segmented words of this language were represented on the source side of the datasets, i.e., their units were split by an underscore. During the surface realization process, our approach did not realize this representation in its correct form, as in the case of the sentence *"Mijn **basis_niveau** is flink omhoog gegaan."*, where the correct form of *basis_niveau* is *basisniveau*. This may have negatively affected the performance of our approach.

5 Conclusion

This study described a shallow surface realizer for 6 languages in the Surface Realization Shared Task 2018 (SR18), with promising results. In future work, we aim to fix the problem of non-segmented words in the Dutch language, as well as the contraction generation in the Portuguese one. Moreover, we aim to evaluate the performance of Neural Machine Translation models in comparison with the statistical used here, in the veins of Castro Ferreira et al. (2017) for AMR-to-text.

References

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. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT'12, pages 427–436, Montreal, Canada. Association for Computational Linguistics.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP'08, pages 848–856, Honolulu, Hawaii. Association for Computational Linguistics.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP'08, pages 49–57, Columbus, Ohio. Association for Computational Linguistics.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL'07, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

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 1st Workshop on Multilingual Surface Realisation (MSR), 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–10, Melbourne, Australia.