Classifiers for data-driven deep sentence generation

Miguel Ballesteros[1], **Simon Mille**[1] **and Leo Wanner**[2,1]
[1]NLP Group, Department of Information and Communication Technologies
Pompeu Fabra University, Barcelona
[2]Catalan Institute for Research and Advanced Studies (ICREA)
`<fname>.<lname>@upf.edu`

Abstract

State-of-the-art statistical sentence generators deal with isomorphic structures only. Therefore, given that semantic and syntactic structures tend to differ in their topology and number of nodes, i.e., are not isomorphic, statistical generation saw so far itself confined to shallow, syntactic generation. In this paper, we present a series of fine-grained classifiers that are essential for data-driven deep sentence generation in that they handle the problem of the projection of non-isomorphic structures.

1 Introduction

Deep data-driven (or stochastic) sentence generation needs to be able to map abstract semantic structures onto syntactic structures. This has been a problem so far since both types of structures differ in their topology and number of nodes (i.e., are non-isomorphic). For instance, a truly semantic structure will not contain any functional nodes,[1] while a surface-syntactic structure or a chain of tokens in a linearized tree will contain all of them. Some state-of-the-art proposals use a rule-based module to handle the projection between non-isomorphic semantic and syntactic structures/chains of tokens, e.g., (Varges and Mellish, 2001; Belz, 2008; Bohnet et al., 2011), and some adapt the semantic structures to be isomorphic with syntactic structures (Bohnet et al., 2010). In this paper, we present two alternative stochastic approaches to the projection between non-isomorphic structures, both based on a cascade of Support Vector Machine (SVM) classifiers.[2] The first approach addresses the projection as a generic non-isomorphic graph transduction problem in terms of four classifiers for 1. identification of the (non-isomorphic) correspondences between fragments of the source and target structure, 2. generation of the nodes of the target structure, 3. generation of the dependencies between corresponding fragments of the source and target structure, and 4. generation of the internal dependencies in all fragments of the target structure. The second approach takes advantage of the linguistic knowledge about the projection of the individual linguistic token types. It replaces each of the above four classifiers by a set of classifiers, with each single classifier dealing with only one individual linguistic token type (verb, noun, adverb, etc.) or with a configuration thereof. As will be seen, the linguistic knowledge pays off: the second approach achieves considerably better results.

Since our goal is to address the challenge of the projection of non-isomorphic structures, we focus, in what follows, on this task. That is, we do not build a complete generation pipeline until the surface. This could be done, for instance, by feeding the output obtained from the projection of a semantic onto a syntactic structure to the surface realizer described in (Bohnet et al., 2010).

2 The Task

The difference in the linguistic abstraction of semantic and syntactic structures leads to divergences that impede the isomorphy between the two and make the mapping between them a challenge for statistical generation. Let us, before we come to the implementation, give some theoretical details on these structures as we picture them and on the possible approaches to the projection of a semantic structure to a syntactic one.

2.1 The Notion of semantic and syntactic structures

As semantic structure, we assume a *shallow semantic* representation that is very similar to the

[1]See, for instance, (Bouayad-Agha et al., 2012).

[2]Obviously, other machine learning techniques could also be used.

PropBank (Babko-Malaya, 2005) and deep annotations as used in the Surface Realisation Shared Task (Belz et al., 2011): the *deep-syntactic layer* of the AnCora-UPF corpus (Mille et al., 2013).

Deep-syntactic structures (DSyntSs) do not contain any punctuation and functional nodes, i.e., governed prepositions and conjunctions, auxiliaries and determiners.[3]

As syntactic structure (in the terminology of Ancora-UPF: *surface-syntactic structures*, SSyntSs), we assume dependency trees in which the nodes are labeled by open or closed class lexemes and the edges by grammatical function relations of the type subject, oblique_object, adverbial, modifier, etc.; cf.[4] See Figure 1 for a contrastive illustration of DSyntS and SSyntS.

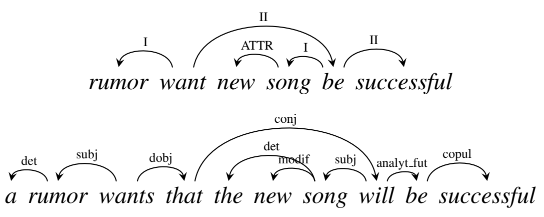
II
I
ATTR
I
II
rumor want new song be successful
conj
det
subj
dobj
det
modif
subj
analyt_fut
copul
a rumor wants that the new song will be successful

Figure 1: DSyntS (above) and SSyntS (below) of an English Sentence.

Note, however, that the proposal outlined below for the projection of non-isomorphic structures is trainable on any multi-layered treebanks where different layers are not isomorphic.

2.2 Projection of DSyntSs onto SSyntSs

In order to project a DSyntS onto its corresponding SSyntS in the course of sentence generation, the following types of actions need to be performed:

1. Project each node in the DSyntS onto its SSynS-correspondence. This correspondence can be a single node, as, e.g., *successful* $\rightarrow$ *successful*, or a subtree (*hypernode*, known as *syntagm* in linguistics), as, e.g., *song* $\rightarrow$ *the song* 'DT NN' (where 'DT' is a determiner and 'NN' a noun) or *be* $\rightarrow$ *that will be* 'IN $\mathrm{V_{AUX}}$ VB' (where 'IN' is a preposition, '$\mathrm{V_{AUX}}$' an auxiliary and 'VB' a full verb). In formal terms, we assume any SSyntS-correspondence to be a hypernode with a cardinality ≥ 1.

2. Generate the correct lemma for the nodes in SSyntS that do not have a 1:1 correspondence in the SSyntS (as 'DT', 'IN' and '$\mathrm{V_{AUX}}$' above).

3. Establish the dependencies within the individual SSyntS-hypernodes.

4. Establish the dependencies between the SSyntS-hypernodes (more precisely, between the nodes of different SSyntS-hypernodes) to obtain a connected SSyntS-tree.

[3]For more details on the SSyntS, see (Mille et al., 2013).

[4]DSyntSs and their corresponding SSyntSs are stored in the 14-column CoNLL'08 format.

3 Classifiers

As mentioned in the Introduction, the realization of the actions 1.– 4. can be approached either in terms of 4 generic classifiers (Section 3.1) or in terms of 4 sets of fine-grained (micro) classifiers (Section 3.2) that map one representation onto another. As also mentioned above, we realize both approaches as Support Vector Machines (SVMs).

3.1 Generic classifier approach

Each of the generic classifiers deals with one of the following tasks.

a. Hypernode Identification: Given a deep syntactic node n_d from the DSyntS, the system must find the shape of the surface hypernode (= syntagm) that corresponds to n_d in the SSyntS. The hypernode identification SVM uses the following features:

> POS of n_d, POS of n_d's head, *voice*, *temp._constituency*, *finiteness*, *tense*, lemma of n_d, and n_d's dependencies.

In order to simplify the task, we define the shape of a surface hypernode as a list of surface PoS-tags. This list contains the PoS of each of the lemmas within the hypernode and a tag that signals the original deep node; for instance:

[VB(deep), $\mathrm{V_{AUX}}$, IN]

b. Lemma Generation. Once the hypernodes of the SSyntS under construction have been produced, the functional nodes that have been newly introduced in the hypernodes must be assigned a lemma. The lemma generation SVM uses the following features of the deep nodes n_d in the hypernodes:

> • *finiteness*, • *definiteness*, • PoS of n_d, • lemma of n_d, • PoS of the head of n_d

to select the most likely lemma.

c. Intra-hypernode Dependency Generation. Given a hypernode and its lemmas provided by the two previous stages, the dependencies (i.e., the dependency attachments and dependency labels) between the elements of the hypernode must be determined (and thus also the governor of the hypernode). For this task, the intra-hypernode dependency generation SVM uses the following features:

• lemmas included in the hypernode, • PoS-tags of the lemmas in the hypernode, • *voice* of the head h of the hypernode, • deep dependency relation to h.

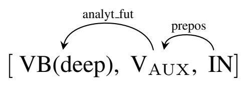

Figure 2: Internal dependency within a hypernode.

d. Inter-hypernode Dependency Generation. Once the individual hypernodes have been converted into connected dependency subtrees, the hypernodes must be connected between each other, such that we obtain a complete SSyntS. The inter-hypernode dependency generation SVM uses the following features of a hypernode s_s:

• the internal dependencies of s_s, • the head of s_s, • the lemmas of s_s, • the PoS of the dependent of the head of s_s in DSyntS

to determine for each hypernode its governor.

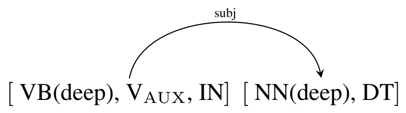

Figure 3: Surface dependencies between two hypernodes.

3.2 Implementation of sets of micro classifiers

In this alternative approach, a single classifier is foreseen for each kind of input. Thus, for the **hypernode identification module**, for each deep PoS tag (which can be one of the following four: 'N' (noun), 'V' (verb), 'Adv' (adverb), 'A' (adjective)), a separate multi-class classifier is defined. For instance, in the case of 'N', the N-classifier will use the above features to assign to the a DSynt-node with PoS 'N' the most appropriate (most likely) hypernode—in this case, [NN(deep), DT]. In a similar way, in the case of the **lemma generation module**, for each surface PoS tag, a separate classifier is defined. Thus, the DT-classifier would pick for the hypernode [NN(deep), DT] the most likely lemma for the DT-node (optimally, a determiner).

For the **intra-hypernode attachments module**, for each kind of hypernode, dynamically a separate classifier is generated.[5] In the case of the hypernode [VB(deep), $\mathrm{V}_{\mathrm{AUX}}$, IN], the corresponding classifier will create a link between the preposition and the auxiliary, and between the auxiliary and the verb, with respectively the preposition and the auxiliary as heads because it is the best link that it can find; cf. Figure 2 for illustration.

[5]This implies that the number of classifiers varies depending on the training set, in the intra-hypernode dependency generation there are 108 SVMs.

Finally, for the **inter-hypernode attachments module**, for each hypernode with a distinct internal dependency pattern, a separate classifier is dynamically derived (for our treebank, we obtained 114 different SVM classifiers because it also takes into account hypernodes with just one token). For instance, the classifier for the hypernode [NN(deep), DT] is most likely to identify as its governor $\mathrm{V}_{\mathrm{AUX}}$ in the hypernode [VB(deep), $\mathrm{V}_{\mathrm{AUX}}$, IN]; cf. Figure 3.

4 Experiments and Results

In this section, we present the performance of the two approaches to DSyntS–SSyntS projection on the DSyntS- and SSynt-layers of the AnCora-UPF treebank (Mille et al., 2013).[6] Table 1 displays the results for the generic classifier for all tasks on the development and the test set, while Table 2 displays the results obtained through the sets of micro classifiers.

Dev.set	#	%
Hypernode identification	3131/3441	90.99
Lemma generation	818/936	87.39
Intra-hypernode dep. generation	545/798	68.30
Inter-hypernode dep. generation	2588/3055	84.71
Test set	#	%
Hypernode identification	5166/5887	87.75
Lemma generation	1822/2084	87.43
Intra-hypernode dep. generation	1093/1699	64.33
Inter-hypernode dep. generation	4679/5385	86.89

Table 1: Results of the evaluation of the generic classifiers for the non-isomorphic transduction.

The results show that for hypernode identification and inter-hypernode dependency generation, the results of both types of classifiers are comparable, be it on the development set or on the test set. However, thanks to the micro classifiers, with the same features, the lemma generation model based on micro classifiers improves by 4 points and the intra-hypernode dependency generation by nearly

[6]Following a classical machine learning set-up, we divided the treebank into: (i) a development set (219 sentences, 3271 tokens in the DSyntS treebank and 4953 tokens in the SSyntS treebank); (ii) a training set (3036 sentences, 57665 tokens in the DSyntS treebank and 86984 tokens in the SSyntS treebank); and a (iii) a held-out test for evaluation (258 sentences, 5641 tokens in the DSyntS treebank and 8955 tokens in the SSyntS treebank).

Dev.set	#	%
Hypernode identification	3133/3441	91.05
Lemma generation	851/936	90.92
Intra-hypernode dep. generation	767/798	96.12
Inter-hypernode dep. generation	2574/3055	84.26
Test set	#	%
Hypernode identification	5169/5886	87.82
Lemma generation	1913/2084	91.79
Intra-hypernode dep. generation	1630/1699	95.94
Inter-hypernode dep. generation	4648/5385	86.31

Table 2: Results of the evaluation of the micro classifiers for the non-isomorphic transduction.

30 points. This means that the intra-hypernode dependency generation task is too sparse to be realized as a single classifier. The micro classifiers are in this case binary, i.e., 2:1, or unary, i.e., 1:1 classifiers, which implies a tremendous reduction of the search space (and thus higher accuracy). In contrast, the single classifier is a multi-class classifier that must decide among more than 60 possible classes. Although most of these 60 classes are diferentiated by features, the differentiation is not perfect. In the case of lemma generation, we observe a similar phenomenon. In this case, the micro-classifiers are multi-class classifiers that normally have to cope with 5 different classes (lemmas in this case), while the unique classifier has to cope with around 60 different classes (or lemmas). Hypernode identification and inter-hypernode dependency generation are completely guided by the input; thus, it seems that they do not err in the same way.

Although the micro classifier approach leads to significantly better results, we believe that it can still be improved. First, the introduction of prepositions causes most errors in hypernode detection and lemma generation: when a preposition should be introduced or not and which preposition should be introduced depends exclusively on the sub-categorization frame of the governor of the deep node. A treebank of a limited size as used in our experiments simply does not contain subcategorization patterns of *all* predicative lexical items (especially of nouns)—which would be crucial. Thus, in the test set evaluation, out of the 171 lemma errors 147 are prepositions and out of the 717 errors on hypernode identification, more than 500 are due to nouns and preposition. The increase of the size of the treebank would therefore be an advantage.

Second, in the case of inter-hypernode dependency, errors are due to the labels of the dependencies more than to the attachements, and are quite distributed over the different types of configurations. The generation of these dependencies suffers from the fact that the SSyntS tag-set is very fine-grained. For instance, there are 9 different types of verbal objects in SSyntS,[7] which capture very specific syntactic properties of Spanish, such as "can the dependent can be replaced by a clitic pronoun? Can the dependent be moved away from its governor? Etc. This kind of information is not of a high relevance for generation of well-formed text. Using a more reduced (more coarse-grained) SSyntS tag set would definitely improve the quality of the projection.

5 Related work

There is an increasing amount of work on statistical sentence generation; see, e.g., (Bangalore and Rambow, 2000; Langkilde-Geary, 2002; Filippova and Strube, 2008). However, hardly any addresses the problem of the projection between non-isomorphic semantic and syntactic structures. In general, structure prediction approaches use a single classifier model (Smith, 2011). But see, e.g., (Carreras et al., 2008), who use different models to predict each part of the triplet for spinal model pruning, and (Björkelund et al., 2010; Johansson and Nugues, 2008), who use a set of classifiers for predicate identification in the context of semantic role labelling. Amalgam (Corston-Oliver et al., 2002), which maps a logical input onto sentences with intermediate syntactic (phrase-based) representation, uses language-specific decision trees in order to predict when to introduce auxiliaries, determiners, cases, etc.

6 Conclusions

We presented two alternative classifier approaches to deep generation that cope with the projection of non-isomorphic semantic and syntactic structures and argued that the micro classifier approach is more adequate. In spite of possible improvements presented in Section 4, each set of micro classifiers achieves results above 86% on the test set. For intra-hypernode dependency generation, it even reaches 95.94% .

Acknowledgments

This work has been partially funded by the European Commission under the contract number FP7-ICT-610411.

[7]There are 47 SSynt dependencies in total, to compare to the 7 dependencies in the DSyntS.

References

Olga Babko-Malaya, 2005. *Propbank Annotation Guidelines*.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 42–48, Saarbrücken, Germany.

Anja Belz, Michael 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 (ENLG)*, pages 217–226, Nancy, France.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Journal of Natural Language Engineering*, 14(4):431–455.

A. Björkelund, B. Bohnet, L. Hafdell, and P. Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics : Demonstration Volume (COLING)*, pages 33–36, Beijing, China.

Bernd Bohnet, Leo Wanner, Simon Mille, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multilevel realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 98–106, Beijing, China.

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. StuMaBa: From deep representation to surface. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation (ENLG)*, pages 232–235, Nancy, France.

Nadjet Bouayad-Agha, Gerard Casamayor, Simon Mille, and Leo Wanner. 2012. Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Transactions on Speech and Language Processing*, 9(2):3:1–3:31.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*, pages 9–16, Manchester, UK.

Simon Corston-Oliver, Michael Gamon, Eric Ringger, and Robert Moore. 2002. An overview of Amalgam: A machine-learned generation module. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG)*, pages 33–40, New-York, NY, USA.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 177–185, Honolulu, Hawaii.

Richard Johansson and Pierre Nugues. 2008. Dependency-based Semantic Role Labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 69–78, Honolulu, Hawaii.

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 (INLG)*, pages 17–24, New-York, NY, USA. Citeseer.

Simon Mille, Alicia Burga, and Leo Wanner. 2013. AnCora-UPF: A multi-level annotation of Spanish. In *Proceedings of the 2nd International Conference on Dependency Linguistics (DepLing)*, pages 217–226, Prague, Czech Republic.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Sebastian Varges and Chris Mellish. 2001. Instance-based Natural Language Generation. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1–8, Pittsburgh, PA, USA.