Statistical Deep Parsing for Spanish using Neural Networks

Luis Chiruzzo
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
luischir@fing.edu.uy

Dina Wonsever
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
wonsever@fing.edu.uy

Abstract

This paper presents the development of a deep parser for Spanish that uses a HPSG grammar and returns trees that contain both syntactic and semantic information. The parsing process uses a top-down approach implemented using LSTM neural networks, and achieves good performance results in terms of syntactic constituency and dependency metrics, and also SRL. We describe the grammar, corpus and implementation of the parser. Our process outperforms a CKY baseline and other Spanish parsers in terms of global metrics and also for some specific Spanish phenomena, such as clitics reduplication and relative referents.

1 Introduction

The syntactic analysis of sentences is one of the key activities within the Natural Language Processing pipeline, and is often seen as a necessary step that must be performed before extracting deeper information such as semantics. Two main paradigms have historically dominated the work on syntactic analysis: constituency parsing, where the sentence is structured as a tree of linguistically motivated segments called constituents; and dependency parsing, where the sentence is structured as a set of bi-lexical dependencies with each word associated to another word that acts as its head.

Throughout the years there have been attempts to create deeper syntactic models that try to combine both syntactic and semantic notions in the same analysis, for example CCG (Steedman, 1996), HPSG (Pollard and Sag, 1994) or TAG (Joshi, 1985). The process of analyzing a sentence in one of these formalisms is often called deep parsing. Over the last years the use of deep neural networks has advanced the state of the art in many NLP tasks, and though some work has been done for performing deep parsing using these architectures (mainly in English and for CCG), few works have focused on the application of these techniques to other formalisms or other languages. In this work, we present a neural network architecture for deep parsing Spanish sentences using the HPSG formalism.

The rest of the paper is structured as follows: Section 2 describes the grammar and corpus we use. Section 3 introduces our parsing architecture and the baseline we compare it to. Section 4 shows an analysis of results. Section 5 describes relevant related work. Finally, section 6 gives some conclusions and future perspectives.

2 Grammar and Corpus

We use a HPSG style grammar (Pollard and Sag, 1994) adapted to Spanish. HPSG grammars are unification grammars that operate on feature structures. Each word is defined by a feature structure that indicates the combinatorial properties of the word: how it can be combined to other words in order to form phrases. Because of this, these grammars tend to have few rules, which are very generic. When applying a rule, the process of unification validates that both the conditions of the rule and the constraints imposed by the expressions being combined are met, and the resulting phrases inherit some of the features of their children in order to define their own combinatorial properties.

2.1 Feature structure

Our grammar defines morphological, syntactic valence, and semantic role label features, as shown in figure 1. The `HEAD` feature contains the part-of-speech and the morphological attributes of the word such as number and gender. The `VAL` feature contains the local syntactic valence features: what are the expected specifier, complements, or modifier that could be associated to the word, and we add a `CLITICS` feature that is specially important for Spanish as clitic pronouns could act as arguments

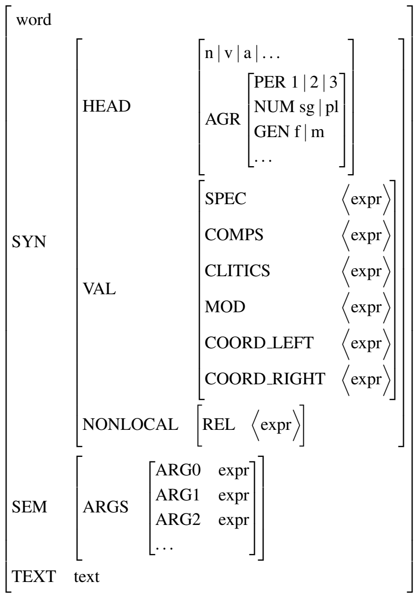

Figure 1: Feature structure for words in our grammar.

of verbs. We also include two features used by conjunctions to guide the analysis of coordinated structures.

In a simplification from the original HPSG theory, the only non-local dependencies modeled in our grammar are the relative clauses that act as modifiers of a noun phrase. This is represented by the REL feature, which allows the grammar to support argument sharing in relative clauses such as *"La manzana que comí" / "The apple I ate"*.

The grammar incorporates semantic features for modeling the semantic role label structure for the sentence. Unlike the way semantics is modeled in standard HPSG (Sag et al., 1999), which uses minimal recursion semantics (Copestake et al., 2005), our approach to semantics uses the simpler PropBank notation (Bonial et al., 2010). The feature SEM includes features derived from PropBank representation: ARG0 for proto-agent, ARG1 for proto-patient, ARGM for modifying adjuncts, etc. Consider, for example, the feature structure for the word *"come" / "eats"* shown in figure 2, which coindexes the syntactic specifier and complement to the corresponding proto-agent and proto-patient semantic arguments.

2.2 Rules

The grammar uses only thirteen combinatorial rules for forming phrases. These rules describe in broad terms how to combine expressions, but the unification process also takes into account the constraints imposed by the expressions as well as the ones

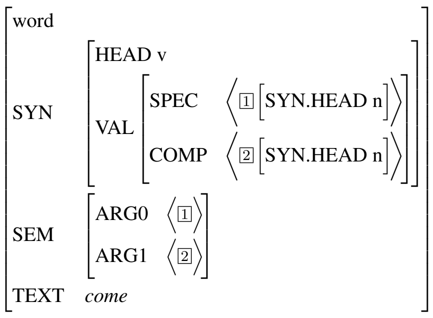

Figure 2: Feature structure for the word *"come" / "eats"*, that expects a noun phrase as specifier (subject) and a noun phrase as complement (direct object), which are also coindexed as proto-agent and proto-patient semantic arguments.

defined in the rules.

For example, consider the head_comp rule defined in figure 3. This rule takes the two expressions shown on the left side of the arrow. The left expression (marked as head) expects at least one complement, which is the right expression (shown only as a coindexation box, meaning any expression that unifies with what the left expression expects). The result is the phrase shown on the right side of the arrow: a phrase that will have the same features as the head expression except the COMP feature, which will expect one less complement. This definition is only a description of what this rule does in particular, but the unification process by which the rules are applied will also ensure that the constraints defined within the expressions are held. If we combine the word *"come" / "eats"* as defined in figure 2 with a complement phrase, the rule application will fail (not unify) if the complement is not a noun phrase.

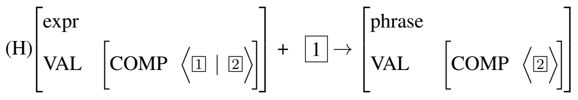

Figure 3: Definition of the head_comp rule.

All rules in our grammar are binary, and they are designed taking into account that the order of constituents in Spanish is not as strict as in English. For example, even though Spanish is officially an SVO language like English, the sentences *"El tren llegó"* and *"Llegó el tren"* are two equally valid ways of saying *"The train arrived"*, where the second case uses a postponed subject. The thirteen rules in our grammar are the following:

- Two rules for attaching a specifier (or subject) to the left or to the right of a head: `spec_head` and `head_spec`.

- Two rules for attaching a complement to the left or to the right of a head: `comp_head` and `head_comp`.
- One rule for attaching a semantic complement to the right of a head: `head_comp_sem`. This rule is used for building more complex verb phrases such as *"ha ido" / "has gone"* or *"comienza a cantar" / "begins to sing"*.
- Two rules for attaching a modifier to the left or to the right of a head: `mod_head` and `head_mod`.
- One rule for attaching a clitic pronoun to the left of a head: `clitic_head`.
- One rule for attaching a relative clause to the right of a head: `head_rel`.
- Two rules for attaching a punctuation marker to the left or to the right of a head: `punct_head` and `head_punct`.
- Two rules for binarizing coordinated structures: `coord_left` and `coord_right`.

Consider the following sample sentence:

[*El niño come una manzana roja*]
(The boy eats a red apple) (1)

The parse tree for sentence 1 using our grammar is shown in figure 4. Although the figure shows a simplified version of the tree (with only one expanded node for the main verb and substituting the non-leaf nodes for the name of the applied rule) each node in the tree is a complete feature structure with reentrancies to other parts of the tree that indicate the syntactic and semantic argument structures.

2.3 Composition of the corpus

Our experiments use a corpus of about half a million words annotated with our grammar based on the AnCora (Taulé et al., 2008) Spanish corpus that contains approximately 17,000 sentences labeled with CFG-style syntactic annotations and other information such as the semantic arguments structure. The corpus was transformed semi-automatically into our grammar (Chiruzzo and Wonsever, 2016, 2018). The transformation process involved using heuristics for identifying heads, binarizing phrases and finding the grammar rules to apply, and reparsing the corpus using unification over feature structures while manually correcting the conversion errors. The training, development and test partitions used in this work are the standard AnCora partitions used for the CoNLL-2009 shared task (Hajič et al., 2009), around 418K words for training, 50K for development and 50K for test.

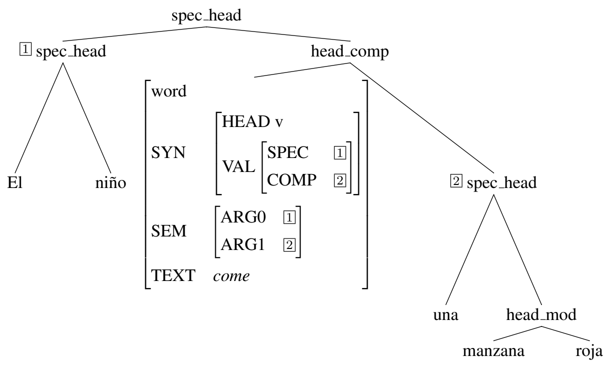

Figure 4: Simplified tree for *"El niño come una manzana roja" / "The boy eats a red apple"*. Only the node for *"come"* is expanded, but each of the nodes in the tree is a feature structure that contains the corresponding combinatorial and semantic information.

3 Parser development

This section describes the LSTM top-down parsing architecture we use in this work and the baselines we compare it to.

3.1 Top-down parser

Our parsing approach divides the parsing process into two steps: first creating a basic binary tree of phrases, where each non-leaf node is annotated with the corresponding rule, and then finding which of the phrases should act as arguments to their respective heads.

3.1.1 Split and rule calculation

The first step is a top-down process that takes as input a sequence of words and decides how to split the sequence in two so that the sub-sequences are valid constituents, indicating which grammar rule should be applied in that node. For example, if we take sentence 1 as input, the output would look like following:

[*El niño*] [*come una manzana roja*]
Rule: `spec_head`

The method proceeds recursively for each constituent that has at least two words. In this case, it will take the sequence *"El niño"* and split it so as to separate the determiner from the noun:

[*El*] [*niño*]

Rule: `spec_head`

The second sub-sequence would be split so as to separate the main verb from the noun phrase complement of the verb, resulting in the following:

[*come*] [*una manzana roja*]

Rule: `head_comp`

This top-down process is iterated until there are no more sub-sequences left to split, effectively transforming the original sentence into a binary tree of words, labeled with the corresponding grammar rules for each non-leaf node. The result is shown in figure 5.

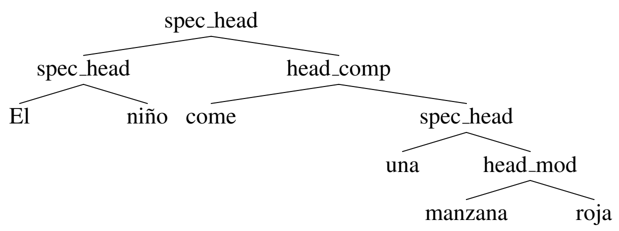

Figure 5: Binary tree with rule labels for the sentence *"El niño come una manzana roja" / "The boy eats a red apple"*, result of the first step of the parsing process.

3.1.2 Arguments calculation

Once this basic tree is created, using the calculated rules it is possible to derive the syntactic features and mark the heads for each node in the tree. With the syntactic features in place, the second step tries to determine which of the syntactic arguments of the predicates should also be set as semantic arguments.

This step considers each head that could be taken as predicate (in our case verbs, nouns and adjectives are possible predicates), and each syntactic argument, and calculates if it should be applied as a semantic feature, and which label it should use. In practice, the method takes the following elements as input:

- The sequence of words corresponding to the top-most constituent that includes the predicate, the head, and the argument.
- The candidate argument being classified.
- The word marked as head.
- The feature that relates the head and the candidate argument.

The result of the method could be `none` (if the candidate should not be considered an argument) or a feature between `arg0` and `argM`. The input values and expected outcomes for our example are the following:

[$\underline{\textit{El niño}}_{SPEC}$ $\underline{\textit{come}}_{HEAD}$ *una manzana roja*] → `arg0`

[*El niño* $\underline{\textit{come}}_{HEAD}$ $\underline{\textit{una manzana roja}}_{COMP}$]→ `arg1`

[*una* $\underline{\textit{manzana}}_{HEAD}$ $\underline{\textit{roja}}_{MOD}$]→ `none`

After this step is finished, the tree will look like the one in figure 4.

3.2 Implementation

Both steps of the process are implemented using neural networks. The first step is a neural network whose input is a sequence of words and whose output is the probability of splitting the sentence at each point in the sequence, and the probabilities of rules for each split point. The method proceeds greedily selecting only the split point with maximum probability and the corresponding rule.

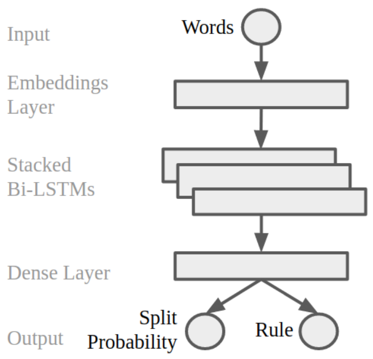

Figure 6: Architecture of the neural network for the sequence split and rule calculation step.

The network (shown in figure 6) is built using stacked bidirectional LSTM layers: the first layer contains a word embeddings model (300 dimensions, trained using `word2vec` for a 6 billion words Spanish corpus); then three layers of stacked bidirectional LSTMs of size 300; then a fully connected layer of size 150; and finally a softmax layer that outputs, for each word in the sequence, the grammar rule to apply or `none` if the word should not be used for splitting in the current step. The probability of the `none` label is used as the probability of splitting at each point in the sequence.

The second step is implemented using a neural network that takes a word (the head), a sequence of words representing the argument to classify, another sequence that represents the outer constituent that contains both the head and the argument, and

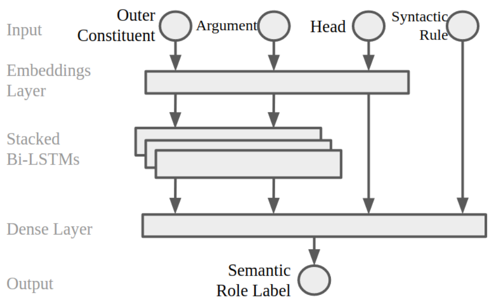

Figure 7: Architecture of the neural network for the argument identification step.

the syntactic rule used to associate the head to the argument. It returns the probability for each argument category (or `none`). The network (as shown in figure 7) has an embeddings layer for the words; then three layers of stacked bidirectional LSTMs of size 150 (in this case they output only one representation for each sequence, instead of one for each word); then a fully connected layer of size 150; and finally a softmax layer that gives the probability distribution for the labels.

We trained several instances of both networks varying the number and sizes of the different layers using `keras` (Chollet, 2015) over `tensorflow` (Abadi et al., 2015), optimizing against the development partition. The number of units for the LSTM and dense layers varied between 150 and 600, and we tried using one, two or three LSTM layers and different values of dropout. Only the results for the best models are reported.

3.3 Baseline CKY parser

In order to compare our parser to more standard HPSG parsing approaches, we implemented another statistical parsing strategy for this grammar, with a bottom-up approach similar to (Matsuzaki et al., 2007): using the CKY algorithm with a probabilistic model to find the best tree. The simplest implementation of the CKY algorithm for HPSG would imply applying all rules to the pairs of expressions found at each step. However, the feature unification method proved to be too slow for doing this in reasonable time, so we relied on a simplification to speed up the process: We designed a set of supertags based on the grammar categories that contain the necessary information to infer the possible rules to apply given a pair of supertags.

Considering sample sentence 1, the main verb of the sentence (*"come" / "eats"*) has a noun phrase specifier to its left (*"El niño" / "The boy"*), which also acts as `ARG0`, and a noun phrase complement to its right (*"una manzana roja" / "a red apple"*), which also acts as `ARG1`. This will be the information conveyed by the supertag for *come*, which is `v-sna0-x-cna1`. The leftmost character is the part-of-speech of the word and the rest is a description of the use of the word in context, where `x` represents the position of the word. This tag means: a verb that expects a noun phrase acting as specifier on its left (`sn`) coindexed with `ARG0`, plus a noun phrase acting as a complement on its right (`cn`) coindexed with `ARG1`. The corresponding supertags for the whole sentence are the following:

El/`d-x` *niño*/`n-sd-x` *come*/`v-sna0-x-cna1`
una/`d-x` *manzana*/`n-sd-x` *roja*/`a-mn-x`

The possible rules to apply to a pair of words depend on the supertags associated to those words. Once they are combined following a certain rule, the resulting phrase will have a new tag that could be used to continue the parsing process. The tag structure is simple enough so that the possible rules to apply can be predicted in terms of regular expressions over the supertags (much faster than the unification method) but also expressive enough so that the invalid combinations are filtered out.

3.3.1 Probabilistic Model

We created a probabilistic model in a way similar to a PCFG. One way of doing this would be considering each possible supertag as a non terminal, the problem with this approach is that the number of supertags is very large: 4146 different supertags in the training corpus. Consequently, the number of times the combination of a particular pair of supertags appears in the corpus is very low (sparsity problem). To mitigate this, we created simpler models for abstracting the non terminals by reducing the information in the supertags and calculated the rule probabilities based on those abstractions. For example, a supertag `v-sna0-x-cna1-csa2` could have the abstract tag `vcs`, that indicates only the POS (verb) and the features it expects (complements and specifier) but not the finer grained information. Our model uses an average of probabilities calculated over these abstract tags, and assigns a very small probability to unseen tag combinations in order to avoid giving rare examples zero probability. Notice that the abstract tags have no effect in the process of determining the possible rules to apply (the full supertag is used for that), but only for estimating the probability of application of a rule.

3.3.2 Supertagger

The CKY algorithm relies on knowing the exact categories of the words to find the valid rules to apply, but when parsing a sentence from scratch that information would not be available. As the number of possible categories per word is large, we trained a supertagger for calculating the most suitable supertags given a sentence. Its inputs are the sentence words and POS-tags and it returns a sequence of supertags. The architecture for this supertagger is similar to the top-down parser: the first layer contains a word embeddings model (300 dimensions) and a POS embeddings model (5 dimensions); then three layers of stacked bidirectional LSTMs of size 450; then a fully connected layer of size 450; and finally a softmax layer that selects one out of 4146 possible supertags. The network achieves 89.1% accuracy over the development corpus and 88.7% for the test corpus considering the top selected tag.

As the performance is not perfect, the sequence of supertags selected might be invalid for forming a tree according to the grammar rules. To mitigate this problem, if a tree is not found we enable two fallback rules with low probability (`head_none` and `none_head`) that combine two arbitrary nodes and take either the left one or the right one as heads. These rules guarantee that a tree will be found, but they make the process much slower as many more subtrees are tried during parsing.

4 Experimental results

In this section we present the experimental results for our LSTM top-down neural network architecture and the baselines in terms of syntactic parsing and semantic role labeling. We also present an analysis of execution time for the different approaches. We compare our method to the CKY baseline defined before and six well established baselines for Spanish parsing. Three of the baselines are the parsers contained in the FreeLing library (Padró and Stanilovsky, 2012), which include both dependency parsing and SRL: Txala (Batalla et al., 2005), Treeler[1] and a LSTM implementation. The other baselines are the pretrained Spanish models Spanish of spaCy[2] (models `es_core_news_sm` and `es_core_news_md`) and UDPipe (Straka and Straková, 2017) (model `Spanish-AnCora`), which only include dependency parsing but no SRL. We show the performance of the (unrealistic) CKY parser using gold supertags as an upper bound for the CKY performance, but also the more realistic CKY using the tags predicted by the supertagger.

4.1 Syntactic parsing

The performance metrics used for the syntactic parsing results are the following:

Constituency F1: The F1 score for the predicted constituents against the expected constituents in a tree. This metric is very strict and penalizes strongly any deviations in the order of application of rules, as swapping the order of only two rules could change many different constituents inside a tree. There are Labeled (L-Cons) or Unlabeled (U-Cons) versions of this metric with or without considering the correct grammar rule.

Dependency Accuracy: As all the rules are binary and they all define a head, it is possible to infer a dependency structure where each word is associated to a corresponding head. This structure contains the same information as the constituent trees except the information about the order of the rules application. This metric calculates the accuracy of assigning each word to its appropriate head, and it is more relaxed compared to constituency F1. Labeled (L-Dep) and Unlabeled (U-Dep) versions of this metric (with or without the grammar rule) are defined.

Model	U-Cons	L-Cons	U-Dep	L-Dep
LSTM Top-down	87.57	82.06	91.32	88.96
CKY Gold tags	77.16	72.72	92.07	92.05
CKY Supertagger	66.08	59.33	83.34	81.03
FreeLing LSTM	-	-	83.15	-
FreeLing Treeler	-	-	83.61	-
FreeLing Txala	-	-	69.75	-
spaCy es_sm	-	-	83.01	-
spaCy es_md	-	-	83.69	-
UDPipe	-	-	82.09	-

Table 1: Results of the syntactic parsing experiments over the test set.

Table 1 shows the performance results for the different experiments. The "CKY Gold tags" model is the unrealistic model that we consider as upper bound for the CKY process, the more realistic version is the "CKY Supertagger". As seen in the table, the best performing model in constituency and dependency metrics is the LSTM top-down approach. The CKY model does not perform well in terms of constituency even when using the gold supertags, probably because it mixes the order of application of rules: it gets only up to 77%, but

[1]http://treeler.lsi.upc.edu/

[2]https://spacy.io

for the dependency metrics (which ignore the rule application order) it gets around 92%. The CKY using the supertagger, as expected, performs worse than that. On the other hand, the performance of the top-down approach is very robust, outperforming even the CKY with gold supertags for the constituency metrics, and achieving comparable results for the dependency metrics.

The top-down parser also outperforms the external baselines for this corpus. Notice, however, that a direct comparison is difficult given that we are using a different grammar formalism. The only metric that could be considered comparable across the different parsers in this context would be the U-Dep, analogous to the UAS metric in dependency parsing, but even for this not all parsers are trained using the same dependency frameworks and they might have differences in how they determine which words are heads. For example, spacy and UDPipe are trained on corpora annotated using the Universal Dependencies framework (Nivre et al., 2016), which prefers using content words over function words as heads (Alonso and Zeman, 2016). In order to compare to these parsers, we post-processed the results and transformed the heads of prepositional phrases, copulas and other structures in order to adapt it to our format. Section 4.4 shows other approaches to the evaluation of some aspects of the results, considering some particular phenomena of the language and how well the different parsers deal with them. In these cases it was also necessary to post-process the results to adapt the different ways the parsers represent these phenomena in order to compare them.

4.2 Semantic role labeling

The performance metric used in this case is the F1 measure for SRL taken as bi-lexical dependencies. We report two versions of the metric: the unlabeled version (U-SRL) measures if the argument was correctly matched as a semantic argument of its head regardless of the selected label, while the labeled version (L-SRL) also considers if the appropriate semantic role label was selected.

Table 2 shows the results for the semantic role labeling experiments. In this case, the (unrealistic) CKY using gold tags is the best performing method, and can be seen as an upper bound for performance. Our top-down approach gets almost as good results for the unlabeled metric, but performs a little worse for selecting the correct labels. The top-down approach also outperforms all the other baselines.

Model	U-SRL	L-SRL
LSTM Top-down	87.68	80.66
CKY Gold tags	88.51	87.51
CKY Supertagger	81.48	75.78
FreeLing LSTM	68.50	60.74
FreeLing Treeler	69.10	61.53
FreeLing Txala	52.17	45.73

Table 2: Results of the semantic role labeling experiments over the test set.

4.3 Execution time

Model	Time (ms)
LSTM Top-down	86.1
CKY Gold tags	288.7
CKY Supertagger	1,237.3
FreeLing LSTM	60.3
FreeLing Treeler	948.5
FreeLing Txala	41.8
spaCy es_sm *(no SRL)*	9.3
spaCy es_md *(no SRL)*	19.8
UDPipe *(no SRL)*	21.8

Table 3: Average time in milliseconds for parsing a sentence in the test set.

Table 3 shows the average time for parsing a sentence in the test set (1,692 sentences) for the different models. The experiments were run on an Intel i7, 2.7GHz, 16GB RAM, without GPU acceleration. The metrics in the table are an average over all sentence lengths, but figure 8 shows a breakdown of execution times for different sentence length ranges (up to 80) for our approaches.

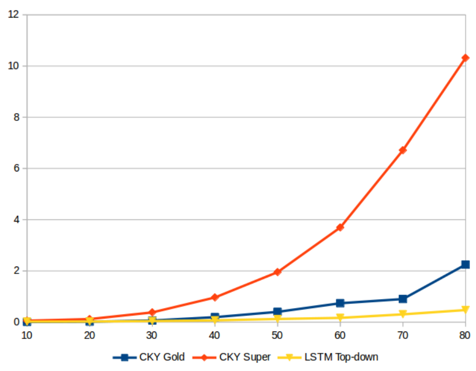

Figure 8: Breakdown of execution time in seconds for different input sizes.

In this case, the parsers based on neural networks seem to have an advantage over the others in terms of speed. The fastest parsers are the spaCy and UDPipe models, but we must take in consideration

		LSTM Top-down	CKY Gold	CKY Supert.	Freeling Txala	Freeling LSTM	Freeling Treeler	spaCy es_sm	spaCy es_md	UDPipe
Postponed Subjects	P	82.22	99.25	**84.25**	60.00	69.74	74.07	60.11	59.92	57.37
	R	64.02	98.89	**76.01**	19.92	65.49	62.73	56.45	59.59	52.39
	F	71.99	99.07	**79.92**	29.91	67.55	67.93	58.23	59.75	54.77
Clitics Identification	P	**98.76**	100.	98.71	96.44	78.25	80.67	84.99	83.47	82.33
	R	**99.03**	100.	95.46	85.83	88.58	88.44	97.38	97.93	96.83
	F	**98.90**	100.	97.06	90.82	83.09	84.38	90.76	90.12	89.00
Clitics Classification	Acc	**85.28**	95.46	83.63	75.10	80.47	80.88	-	-	-
	MP	**76.20**	98.46	65.05	72.03	82.67	83.50	-	-	-
	MR	**66.52**	88.18	57.80	44.02	51.51	54.60	-	-	-
	MF	**70.60**	92.78	61.00	49.20	56.37	58.63	-	-	-
Clitics Reduplication	P	32.69	100.	**75.00**	8.33	22.05	30.23	-	-	-
	R	**35.41**	81.25	18.75	2.08	31.25	27.08	-	-	-
	F	**34.00**	89.65	30.00	3.33	25.86	28.57	-	-	-
Relatives Identification	P	90.60	100.	**92.27**	72.84	76.59	78.23	69.16	70.85	66.49
	R	**89.00**	99.72	81.00	54.95	58.61	59.02	55.08	55.35	52.23
	F	**89.80**	99.86	86.27	62.64	66.41	67.28	61.32	62.05	58.51
Relatives Classification	Acc	**76.12**	81.14	57.67	4.21	43.15	42.06	-	-	-
	MP	**64.73**	84.36	54.32	30.54	40.75	40.76	-	-	-
	MR	**55.32**	78.47	37.82	7.63	24.78	24.39	-	-	-
	MF	**59.02**	72.39	43.29	6.54	29.18	28.62	-	-	-
Relative Referents	P	**73.61**	77.68	67.69	37.23	60.46	63.30	54.17	55.53	49.56
	R	**72.32**	77.47	59.43	28.08	46.26	47.76	43.14	43.55	38.94
	F	**72.96**	77.58	63.29	32.01	52.42	54.44	48.03	48.82	43.61
Coordinations Identification	P	**65.49**	74.02	54.92	24.48	56.53	56.09	41.39	42.47	37.22
	R	**65.22**	77.23	48.25	22.20	53.49	53.00	43.85	44.34	39.59
	F	**65.36**	75.59	51.37	23.28	54.96	54.50	42.59	43.38	38.37

Table 4: Results of the experiments for some language phenomena in Spanish. We show precision, recall and F1 score for identification tasks, and accuracy and macro metrics for classification tasks.

that they only perform dependency parsing and not SRL. Our LSTM top-down approach, and FreeLing LSTM and Txala (rule-based) parsers are in intermediate positions, while the CKY approaches and the FreeLing Treeler parser are way behind. If we break down the execution time of our top-down process, we get that there is a balance in the time spent at each step: 54.1 ms for syntactic parsing and 31.9 ms for argument identification. This could be sped up using a unified architecture.

As seen in figure 8, the top-down approach execution time seems to grow close to linearly while for CKY it grows faster, particularly for the CKY with supertagger. This is because, especially for longer sentences, the probability of obtaining a sequence of tags that does not form a correct tree is much higher as the sentence grows, so the fallback rules have to be enabled more frequently, rendering the process much slower.

4.4 Analysis

Besides the global metrics shown so far, we wanted to test the performance of the parsers for some of the Spanish language characteristics that inspired the grammar rules and behavior in the first place (see Section 2.2). We tested the parsers on how well they detect the following phenomena:

Postponed subjects: Identifying a subject that occurs on the right of the verb, as opposed to the more usual left position.

Clitics identification and classification: Detecting the clitic pronouns that accompany a verb and classifying them according to SRL.

Clitics reduplication: In Spanish, clitic pronouns can be used in lieu of an explicit object, but it is also possible to include both the object and the corresponding clitic at the same time. This is called clitic reduplication or clitic doubling.

Relatives identification and classification: Detecting the relative pronouns and expressions that are attached to some verbs creating relative sentences, and classifying them according to SRL.

Relatives referent identification: Besides identifying the relative pronoun and its verb, the parser must also properly identify the nominal referent for the relative pronoun.

Coordinations identification: Finding chains of (two or more) coordinated elements.

Table 4 shows the results for these experiments over the test set for the different parsers. In the results, we can see that the LSTM Top-down parser performs better for most experiments, except for the postponed subject identification where CKY with supertagger performs better. At least for these phenomena, the results seem to indicate that both parsers outperform the other baselines. However, we must take in consideration that, as mentioned in section 4.1, the output of the different parsers had to be post-processed in order to recognize the different phenomena, so there could be some noise in the evaluation introduced by this transformation.

5 Related work

Most of the work over the last years on deep parsing using neural network architectures has been done for the CCG formalism and for English language. For example (Xu et al., 2015) uses a recurrent neural network for improving the supertagging and parsing accuracy for CCG, while (Ambati et al., 2016) describes a neural networks architecture that performs CCG parsing. For HPSG, the work by (Zhou and Zhao, 2019) is very relevant as they try to derive a HPSG grammar from the Penn Treebanks in English and Chinese and use an self-attention based mechanism followed by a CKY decoder to parse them, obtaining very good results. In our work, however, we focus in HPSG for the Spanish language and define a different architecture for parsing: a top-down approach with LSTMs.

Our approach to HPSG development is similar to another relevant statistical parser for HPSG in English, the Enju parser (Matsuzaki et al., 2007; Zhang et al., 2010), which was created by transforming the English Penn Treebank through a set of rules into HPSG format and used this transformed corpus to train a statistical model. Another relevant precedent for HPSG is the LKB (Copestake, 2002) platform together with the PET (Callmeier, 2000) parser, used to define HPSG grammars in English and other languages (in particular the Spanish Resource Grammar (Marimon, 2010)), although they do not use neural networks for the parsing process. We differ from the Spanish Resource Grammar in that we derive the feature structures from a large corpus instead of building them manually, with the aim of building a purely statistical parser that could, for example, handle slightly ungrammatical sentences or out of vocabulary concepts better.

The CKY with supertagging method we compare to in this work follows a similar approach to the parsing methods used for deep syntactic grammars such as CCG (Curran et al., 2006; Lewis and Steedman, 2014), HPSG (Matsuzaki et al., 2007; Dridan, 2009; Zhang et al., 2010) and TAG (Kasai et al., 2017; Friedman et al., 2017).

Much more work on applying neural network architectures to parsing has been done for the two more classical syntactic paradigms: constituency and dependency parsing. For constituency parsing, (Socher et al., 2013) defines a class of recurrent neural networks that combine pairs of words and builds a tree bottom-up, with the aim of improving sentiment analysis in English sentences. On the other hand, (Vinyals et al., 2015) frames the parsing process as a translation between a sentence in natural language and the bracketed representation of its parse tree. Other authors train neural models to predict the actions to be performed in a transition based shift-reduce constituency parser (Dyer et al., 2016), combining it with a sequence-to-sequence modeling (Liu and Zhang, 2017), or encoding the parsing stack using a recurrent network (Watanabe and Sumita, 2015). In (Cross and Huang, 2016) a transition based constituency parser gets good results for English and French using LSTMs for representing word spans instead of partially derived trees.

Another approach that focuses on determining the word spans in the tree is used in (Stern et al., 2017), which describes a top-down parser that greedily splits a sentence in constituents and assigns labels to them, processing the text spans with LSTMs in order to generate an intermediate representation. This approach is the most similar we found to the one we use, the main differences are that they work for French, they use a standard constituency grammar while ours is a strictly binarized HPSG grammar, and we add a further step for predicting the argument structure of predicates for the generated trees. Related approaches for English include: (Gaddy et al., 2018), that calculates the score and label for each sentence span then uses CKY to find the optimal tree; (Shen et al., 2018), that predicts the syntactic distances for words and builds the tree top-down using these distances.

Compared to English, there are few works that focus on Spanish parsing. For constituency parsing, (Cowan and Collins, 2005) tries two approaches to improve standard PCFG parsers: including morphological information in the probabilistic model, and a reranking method with max-margin criterion trained over a set of global features from the parse trees. Their evaluation against the Cast3LB corpus, a subset of AnCora, achieves a constituent F1 of 83.6 and 85.1 respectively. (Le Roux et al.,

2012) experiments with Spanish parsing using a PCFG with latent annotations with a simplified tagset, achieving 85.47 F1 over the Cast3LB corpus.

There is considerably more work done for dependency parsing and SRL in Spanish, beginning in the CoNLL-X (Buchholz and Marsi, 2006) and CoNLL-2009 (Hajič et al., 2009) shared tasks. The best LAS achieved for Spanish were 82.3 (max span tree approach) and 81.3 (transition based approach). Later on, (Lloberes et al., 2010) describes a dependency grammar and a rule-based dependency parser for Spanish (one of the Freeling parsers (Padró and Stanilovsky, 2012)) transforming the result of a shallow parser, achieving 81.13 UAS and 73.88 LAS. In (Ballesteros et al., 2010) they describe a set of experiments using MaltParser (Nivre et al., 2007) to determine how much corpus size, sentence length or other factors contribute to the dependency Spanish parsing performance. The latest efforts in dependency parsing have generally focused on using the Universal Dependencies (Nivre et al., 2016) framework. For example in CoNLL 2017 (Zeman et al., 2017) and CoNLL 2018 (Zeman et al., 2018) shared tasks on multilingual parsing from raw text to Universal Dependencies the best parsers achieve LAS of 87.29 (Dozat et al., 2017) and 90.93 (Che et al., 2018) respectively for Spanish.

6 Conclusions

We presented a statistical deep parser for Spanish that outputs trees in the HPSG formalism. The parser uses a top-down approach for building the tree followed by a second step for calculating the arguments, both steps are implemented using LSTM neural networks. The parser gets good results for performance compared to a CKY baseline and to other parsing baselines in Spanish, achieving 87.57% unlabeled and 82.06% labeled constituency F1, and 91.32% unlabeled and 88.96% labeled dependency accuracy. It also achieves good performance for SRL, getting 87.68% unlabeled and 80.66% labeled F1, and beats the baselines for some particular Spanish phenomena we analyzed.

Although the results improve over the baseline methods, there is still room for improvement, especially in terms of execution time. We plan to build a unified architecture that could perform both the syntactic and the arguments step at the same time, which could help lower the execution times and it might also help the network generalize better in order to improve the prediction of labels in SRL. It would be also very interesting to try this approach to other languages such as English, and also languages that share some of the characteristics we analyzed such as Italian or French.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Héctor Martínez Alonso and Daniel Zeman. 2016. Universal dependencies for the ancora treebanks. *Procesamiento del Lenguaje Natural*, 57:91–98.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2016. Shift-reduce ccg parsing using neural network models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 447–453.

Miguel Ballesteros, Jesús Herrera, Virginia Francisco, and Pablo Gervás. 2010. Improving parsing accuracy for spanish using maltparser. *Procesamiento del Lenguaje Natural*, 44.

Jordi Atserias Batalla, Elisabet Comelles Pujadas, and Aingeru Mayor. 2005. Txala un analizador libre de dependencias para el castellano. *Procesamiento del Lenguaje Natural*, 35.

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. PropBank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder.*

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the tenth conference on computational natural language learning*, pages 149–164. Association for Computational Linguistics.

Ulrich Callmeier. 2000. Pet–a platform for experimentation with efficient hpsg processing techniques. *Natural Language Engineering*, 6(1):99–107.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Luis Chiruzzo and Dina Wonsever. 2016. Transforming the AnCora corpus to HPSG. In *Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar, Polish Academy of Sciences, Warsaw, Poland*, pages 182–193, Stanford, CA. CSLI Publications.

Luis Chiruzzo and Dina Wonsever. 2018. Spanish HPSG Treebank based on the AnCora Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Ann Copestake. 2002. *Implementing typed feature structure grammars*, volume 110. CSLI publications Stanford.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on language and computation*, 3(2-3):281–332.

Brooke Cowan and Michael Collins. 2005. Morphology and reranking for the statistical parsing of spanish. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 795–802. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

James R Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 697–704. Association for Computational Linguistics.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Rebecca Dridan. 2009. *Using lexical statistics to improve HPSG parsing*. Ph.D. thesis, University of Saarland.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Dan Friedman, Jungo Kasai, R. Thomas McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for TAG parsing. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 122–131, Umeå, Sweden. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18. Association for Computational Linguistics.

Aravind Krishna Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?

Jungo Kasai, Robert Frank, R Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. Tag parsing with neural networks and vector representations of supertags. In *Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Joseph Le Roux, Benoit Sagot, and Djamé Seddah. 2012. Statistical parsing of spanish and data driven lemmatization. In *ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages (SP-Sem-MRL 2012)*, pages 6–pages.

Mike Lewis and Mark Steedman. 2014. Improved ccg parsing with semi-supervised supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

Jiangming Liu and Yue Zhang. 2017. Encoder-decoder shift-reduce syntactic parsing. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 105–114, Pisa, Italy. Association for Computational Linguistics.

Marina Lloberes, Irene Castellón, and Lluís Padró. 2010. Spanish freeling dependency grammar. In *LREC*, volume 10, pages 693–699.

Montserrat Marimon. 2010. The spanish resource grammar. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, pages 17–23, Valletta, Malta.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient hpsg parsing with supertagging and cfg-filtering. In *IJCAI*, pages 1671–1676.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *LREC2012*.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Ivan A Sag, Thomas Wasow, Emily M Bender, and Ivan A Sag. 1999. *Syntactic theory: A formal introduction*, volume 92. Center for the Study of Language and Information Stanford, CA.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Mark Steedman. 1996. A very short introduction to ccg. *Unpublished paper. http://www. coqsci. ed. ac. uk/steedman/paper. html*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Mariona Taulé, M. Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). Http://www.lrec-conf.org/proceedings/lrec2008/.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2773–2781, Cambridge, MA, USA. MIT Press.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179. Association for Computational Linguistics.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. Ccg supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 250–255.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, jr. Jan Hajič, Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl a! nd Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. Zhang, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. Forest-guided supertagger training. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1281–1289. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.