LIMSI@CoNLL'17: UD Shared Task

Lauriane Aufrant[1,2] **Guillaume Wisniewski**[1]

[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France
[2]DGA, 60 boulevard du Général Martial Valin, 75 509 Paris, France
{lauriane.aufrant,guillaume.wisniewski}@limsi.fr

Abstract

This paper describes LIMSI's submission to the *CoNLL 2017 UD Shared Task*, which is focused on small treebanks, and how to improve low-resourced parsing only by *ad hoc* combination of multiple views and resources. We present our approach for low-resourced parsing, together with a detailed analysis of the results for each test treebank. We also report extensive analysis experiments on model selection for the PUD treebanks, and on annotation consistency among UD treebanks.

1 Introduction

This paper describes LIMSI's submission to the *CoNLL 2017 UD Shared Task* (Zeman et al., 2017), dedicated to parsing Universal Dependencies (Nivre et al., 2016) on a wide array of languages. Our team's work is focused on small treebanks, under 1,000 training sentences. To improve low-resourced parsing, we propose to leverage base parsers, either monolingual or cross-lingual, by combining them with a cascading method: each parser in turn annotates some of the tokens, and has access to previous predictions on other tokens to help current prediction; in the end each token is annotated by exactly one parser. Compared to the official baseline, this combination method yields significant improvements on several small treebanks, as well as a few larger ones.

Overall, according to the official results, our system achieves 67.72 LAS and is ranked 17th out of 33 participants, while the baseline (UDPipe 1.1) achieves 68.35 LAS and is ranked 13th. This is mostly due to huge drops from the baseline on a few languages, for which we submitted one of the official baseline models. Analyzing these drops (see §4.5) unveils that strong annotation divergences remain among UD treebanks of the same language. If for these treebanks we had submitted the exact same models as the baseline submission, our system would have been ranked 9th, achieving 68.90 LAS. In the unofficial, post-evaluation ranking, it is ranked 12th.

In §2, we present the design of our system, the base parsers we use and how we combine them. Official results are reported in §3; the strategy adopted for each group of treebanks is presented in §4, along with per-treebank detailed analyses.

2 System overview

Our system consists of several strategies, summarized in Figure 1: depending on the treebank size, we build and compare several parsers with various designs, and finally submit the parser or parsers combination performing best on development data. For a few languages, we also improve preprocessing by correcting errors in the tokenization predicted by UDPipe (Straka et al., 2016).

Most of the parsers we consider are based on the literature or our previous works, but for some languages, we also experiment with a new method for combining base parsers. This method is designed to better leverage each available resource in a low-resource context. Indeed, state-of-the-art methods for low-resourced parsing generally focus on exploiting one type of resource (e.g. parallel data) to build a model, neglecting the others. On the contrary, our approach aims at using all available resources together, since when data is scarce, we can hardly afford ignoring a given information source.

The main idea of our approach is that various kinds of parsing algorithms and training data (monolingual, cross-lingual, delexicalized, parallel data) provide different, complementary views on the dependency structure of the language at hand. We intend to leverage this complementar-

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 163–173,
Vancouver, Canada, August 3-4, 2017. © 2017 Association for Computational Linguistics

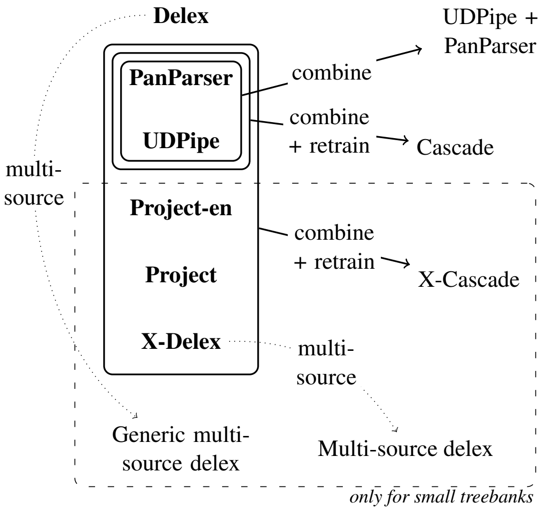

Figure 1: The various strategies contained in our system. The strategies written in bold consist of a single model, the others are combinations of models. For each language, the best strategy is selected on development data.

ity with a selective combination of several base parsers.

For instance, a cross-lingual delexicalized parser intuitively provides insights on the main syntactic structures, presumably shared because of linguistic similarities (typically assessed using linguistic knowledge), while a monolingual parser can learn target-specific structures in target data. On one hand, if monolingual data is too small, it does not contain enough information on the main syntactic structures, and the cross-lingual parser will be more accurate; it should be preferred for this kind of dependencies. On the other hand, knowing the main structure of the sentence can help in identifying fine-grained, target-specific structures; when it is not available in monolingual data, we want the cross-lingual parser to provide this information to the monolingual parser (e.g. as additional features).

Hence, we propose that some part of the syntax is preassigned to each base parser, which is retrained to specialize on that part, and to make use of the syntactic insights provided by other, more generic parsers. We achieve this with a cascading method (Alpaydin and Kaynak, 1998) and preestimated competence regions (Kuncheva, 2004, Chapter 6): each component of the cascade is assigned a competence region, i.e. a subset of the tokens on which it is assumed *a priori* to perform well, it annotates only these tokens and the subsequent parsers are only allowed to complete this partial tree. In the cascade, each component parser trains and annotates the input based on its predecessors; this requires (a) parsers with partial trees as output, (b) to predict parses that include a given set of dependency constraints, and (c) to train parsers to use such constraints. Figure 2 summarizes the method.

The base parsers that we compare and combine are introduced in §2.1, and §2.2 describes how the cascades are implemented and the competence regions are chosen. §2.3 and §2.4 address other technical aspects of our submission: model selection among our multiple strategies, and input preprocessing.

2.1 Base parsers

2.1.1 Training data

In our system, we do not distinguish languages with or without development data, nor surprise languages. For all languages, we use train/tune/dev splits of the UD data (Nivre et al., 2017a), following the splits provided with the official baseline (Straka, 2017).[1] We perform a similar split for the surprise languages, retaining the 10 first sentences for the trainset and the 4 last sentences for the devset.

We always use gold tokenization and segmentation during training, but to improve robustness to noisy tags, all models are trained on treebanks with predicted tags, provided by the task organizers.[2]

We use the word embeddings provided by the organizers, computed on monolingual data preprocessed by UDPipe.[3]

Parallel data from the OPUS platform (Tiedemann, 2012) is preprocessed as follows: for each pair, all corpora are concatenated, tokenized and annotated by UDPipe, and word aligned with fast align (Dyer et al., 2013).

2.1.2 Monolingual

We consider four monolingual parsers:

[1] However, in the case of Uyghur and Kazakh, whose tunesets are particularly small and can hinder our method, we reallocate sentences from trainsets to tunesets, to reach 15 tuning sentences. Still, to prevent train-tune overlaps, the initial tuneset is used when evaluating the official UDPipe model.

[2] Except for the sample treebanks of the surprise languages, for which only gold tags are released.

[3] We also compute such embeddings for the surprise languages, using the same process.

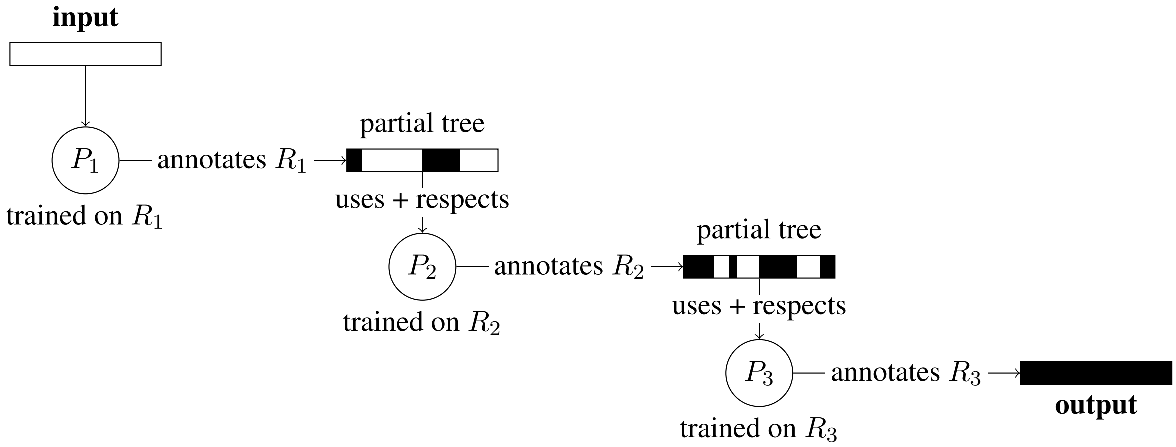

Figure 2: Processing of an input sentence by a 3-component cascade. The cascade contains parsers P_1, P_2 and P_3, which have been respectively assigned the competence regions R_1, R_2 and R_3; each token belongs to exactly one region. On the input sentence, the white areas represent tokens whose head is unknown, while the black areas represent tokens whose head has already been predicted.

UDPipe We apply the official UDPipe 1.1 baseline models (Straka, 2017). For the surprise languages, we train our own model.[4]

PanParser This is an in-house implementation (Aufrant and Wisniewski, 2016) of a transition-based parser, using the ArcEager system and an averaged perceptron. Hyperparameters to tune are the number of epochs, the use of the universal morphological features, the use of word embeddings concatenated to the feature vectors, and the size of the beam (either 8 or 1, i.e. greedy). In any case the parser trains with dynamic oracles, with the restart strategy of Aufrant et al. (2017). Relation labels are predicted in a second step, which enables to use features of the whole parse tree for this prediction.

Delex This is the same as the PanParser models, except that all lexicalized features are removed, including word embeddings.

UDPipe+PanParser As relation labels are sometimes better predicted by PanParser than UDPipe, we also consider combining their outputs at prediction time: we first annotate the input with UDPipe, discard the predicted labels and replace them with labels predicted by PanParser on UDPipe trees.

2.1.3 Cross-lingual

For each treebank under 1,000 training sentences, we apply cross-treebank techniques to build additional parsers.

First, for each target treebank, we transform every source treebank by delexicalizing it and applying the WALS rewrite rules of Aufrant et al. (2016). We then compute, for each such treebank, its similarity to the target treebank, using the KL_{cpos^3} divergence metric (Rosa and Zabokrtsky, 2015).

We select the source among treebanks over 2,000 sentences, by retaining the languages requiring the smallest number of rewrite rules (i.e. the smallest number of divergent WALS features), and then choosing the (transformed) treebank minimizing the KL_{cpos^3} divergence.

When the selected source is of the same language, we use domain adaptation techniques, otherwise we turn to cross-lingual methods. However, domain adaptation was not used in the final submission as it did not bring significant improvements on the corresponding treebanks (French-ParTUT, Galician-TreeGal and Czech-CLTT), and is not detailed here.

We consider five cross-lingual parsers:

Project-en Based on parallel data with English, we use the partial projection technique of Lacroix et al. (2016). Similarly to Lacroix et al. (2016), alignment links are filtered by PoS agreement and non-projective parses are filtered out; for some pairs there are not many trees with high coverage,

[4] We use the same hyperparameters as the smallest UD treebank (Kazakh), including word embeddings pretrained on their trainsets.

so we adopt the following heuristic: we select all trees with coverage over 20% if there are less than 5,000, trees with coverage over 80% if there are more than 5,000 (up to 10,000), or the 5,000 trees with highest coverage otherwise. Training on partial trees is enabled by PanParser; PanParser hyperparameters are tuned on the target tuning data.

Project This model is the same as Project-en, but using parallel data between the selected source and the target. Depending on the language, the language pair may be similar enough to compensate for the reduced amount of parallel data.

X-Delex We train a delexicalized PanParser model on the selected source treebank, transformed by the WALS rules. In order to increase train-test similarity, especially for Uyghur and Kazakh whose tagging accuracy is particularly low (under 70%), we also experimented with artificial noise added to the source tags (either random or designed to match the error types of the target tagger) but it was not conclusive. Hyperparameters are tuned on the target tuning data.

Generic multi-source delex This language-independent model first parses the input with all Delex models, then computes the output tree as a maximum spanning tree over all (unweighted) candidate parses; relation labels result from a vote of all Delex models, for each dependent token.

Multi-source delex This is a language-dependent variant of the previous model, where delexicalized models (excluding the target) are trained on the transformed treebanks, and their contributions to tree combination and vote on labels are weighted by $(1/KL_{cpos^3})^4$, following Rosa and Zabokrtsky (2015). We experiment with three heuristics to reduce the source set: retaining only treebanks over 2,000 sentences, treebanks that minimize the number of rewrite rules, and the top 5 treebanks according to the KL_{cpos^3} metric; the best heuristic depends on the language and is tuned on target data.

Most base parsers are trained in a few hours on CPU, using a single thread (excluding hyperparameter tuning); exceptions are for instance Czech and Russian-SynTagRus, whose treebanks are notably large, and parser projection to Hungarian, due to parallel data much larger than other pairs.

2.2 Cascade combination

Implementing a cascade parser relies on three features of PanParser:

- Training parsers for partial output: training data annotations are filtered according to the competence region, and during training, the model is penalized when it attaches unannotated tokens.
- Predicting parses under constraints: the search space is reduced at prediction time, according to a partial tree.
- Training parsers under constraints: the search space is reduced at training time, by preannotating the training data with all previous components of the cascade.

Hence, each component parser is retrained[5] both to specialize on its competence region, and to take into account the knowledge provided by the previous components. To ensure that the final output is complete, the last component trains on full parses and annotates any remaining token.

As these features do not exist in UDPipe, we do not retrain the UDPipe models, but still include them in the cascade by predicting without constraints, filtering the output according to competence regions, and restoring the constraint trees as a postprocessing step. This way, they annotate more tokens than needed, and do not use the knowledge from previous components, but they can still provide useful knowledge to later components.

Confidence filtering Sometimes the output contains a few noisy dependencies on top of the dependencies belonging to the competence region. To help distinguishing those, we add confidence filtering with ensembling: for each component, 5 parsers are trained, and only the dependencies predicted by at least 3 parsers are retained (using maximum spanning tree techniques for combination).

Relation labels As in base PanParser models, relation labels are predicted in a second step, using features from the whole predicted parse tree. Since each label is predicted independently, the competence regions are computed only *after* training, without retraining: at test time we simply use the label predicted by the competent classifier.

[5] For each retrained component, hyperparameter values are reused from the corresponding base parser.

For each component separately, label prediction is trained on full data, preannotated with parse trees resulting from the whole cascade, and competence regions are computed for these label classifiers.

Competence regions To compute the regions, we group the dependencies into classes according to the PoS of the child and parent (e.g. DETs depending on NOUNs, or NOUNs depending on VERBs), evaluate each base parser on tuning data, and assign each 'PoS-PoS' class to the model that annotates it best. We do not assign classes that are too small (less than 5 occurrences) or have low accuracy (under 0.2). By design, any unassigned class defaults to the last component of the cascade, which in our experiments is always the monolingual PanParser model.

Choice of components We apply the cascade combination method both in monolingual and in cross-lingual configurations. In each case, we try out several subsets of components, training cascades for each subset, and tune this choice separately for the heads and the labels.[6]

In monolingual configurations (denoted Cascade), when the scores of UDPipe and PanParser on tuning data are close enough, we hypothesize that their views may complement one another, and train cascades with the following component candidates: either UDPipe followed by PanParser, or UDPipe alone, or PanParser alone. Hence, we compare 3 cascades on the head prediction task, and 3 computations of competence regions on the label prediction task.

In cross-lingual configurations (denoted X-Cascade), the cascades are trained with the best projected parser (either Project or Project-en, when they exist), followed by X-Delex, the target UDPipe and the target PanParser. We try removing one or both of X-Delex and UDPipe, thus comparing 4 cascades.

2.3 Model selection

For each treebank, we compare all base and cascade parsers, and retain the parser yielding the best LAS on the provided development set (using gold tokenization). However, in some languages this dataset was particularly small and consequently biased, which often led to selecting the wrong model, as will be seen in §4.

For the PUD treebanks, i.e. extra test sets of a UD language but which do not correspond to a given UD treebank, when there are several UD treebanks for the same language, we choose the model whose LAS on own development data is the highest. This corresponds in practice to choosing the largest treebank (in number of sentences), except for Swedish-LinES. This choice is analyzed in detail in §4.5.

2.4 Preprocessing

For most languages, we rely solely on the UDPipe preprocessing as provided by the organizers. Tokenization is customized only for the 3 languages with lowest tokenization accuracy on development data: Vietnamese, Chinese and Japanese.

Vietnamese In the UD 2.0 guidelines, spaces are allowed inside words. While in most treebanks such words are rare, in Vietnamese whitespaces denote syllable boundaries as well as word boundaries, and words containing spaces are consequently much more frequent. The low tokenization accuracy (F_1=83.99) of the baseline UDPipe is mostly due to errors on such words.

UDPipe's tokenization is postprocessed to improve the recognition of words containing spaces. To achieve this, we use a PMI criterion, assuming that pairs of tokens which mostly appear together are very likely a single word. When two consecutive (orthographic) tokens have a very high (resp. low) PMI, they are joined into a single word (resp. split, if UDPipe had joined them). We do not allow words with more than 2 orthographic tokens; in case of conflict the pair with highest PMI is joined.

PMI values are computed using the unigram and bigram counts of orthographic tokens in the crawled monolingual data (after UDPipe tokenization, to segment punctuation); in case of OOV the pair remains unchanged. We set the PMI lower- and upperbounds to $\log 5$ and $\log 400$.

Chinese and Japanese We rely on UDPipe for sentence segmentation, and then use KyTea (Neubig et al., 2011) to tokenize each sentence. KyTea models are trained on UD Chinese and Japanese training treebanks.

For all three languages, the newly tokenized input is then morphologically annotated by UDPipe.

[6] Depending on the data sizes, the cascades train in a few hours to two days on CPU, using 5 threads.

3 Overall results

As part of the *CoNLL 2017 UD Shared Task*, we evaluated our system on the TIRA platform (Potthast et al., 2014). Evaluation runs on the virtual machine took 10.5 hours on a single thread, using up to 6GB RAM.

Table 1 presents our overall results as published by the organizers, compared to the UDPipe 1.1 baseline.

Results rank our model first in tokenization and second in morphological tagging, although these improvements are due to our tokenization improvements on only 3 languages.

Regarding LAS, and according to the various strategies that were adopted, our system presents various behaviors depending on the treebank group, and sometimes even among treebanks of the same group, e.g. surprise languages (see §4.4).

However, the publication of the results, and careful comparison with the baseline UDPipe submission, revealed huge unexpected drops on the PUD treebanks, associated to differences in model selection. Consequently, we also submitted an additional, unofficial run, using the same heuristic as the baseline submission: always retain the model trained on the main treebank of the language. The corresponding scores are reported in Table 1 as 'unoff.'. The score differences are mostly explained by biases in PUD treebanks, towards one of the UD treebanks of the given language (see §4.5).

Consequently, our system is ranked below the baseline in the official run, but above the baseline when taking the PUD bias into account.

4 Analysis

In this section, we present the strategy that was adopted for each treebank, along with detailed analysis on the results. The analysis is conducted separately for the treebanks with custom tokenization, the large (over 10,000 training sentences), medium and small (under 1,000 sentences) UD treebanks, and for PUD treebanks.

Throughout the section, we report both official results from the TIRA platform, and our own measures using the official evaluation script on the released test files (Nivre et al., 2017b). The latter are displayed in italics.

	UDPipe [off.]		LIMSI [off.]		LIMSI [unoff.]	
	F1/LAS	Rank	F1/LAS	Rank	F1/LAS	Rank
Tokenization	98.77	8	98.95 *	1	98.95 *	
All tags	73.74	4	73.86 *	2	73.86 *	
All treebanks	68.35	13	67.72	17	68.90 *	12
Big (55)	73.04	17	73.64 *	13	73.64 *	
PUD (14)	68.33	13	62.24	26	69.07 *	
Small (8)	51.80 *	15	51.71	16	51.71	
Surprise (4)	37.07	11	37.57 *	9	37.57 *	

Table 1: Overall results of the shared task, as published by the organizers. '*' denotes the best scores among the three systems. For each group of treebanks, the number of treebanks it contains is indicated in parentheses. The last column corresponds to the unofficial ranking, which also takes into account later improvements achieved by other teams. The missing ranks are unknown.

4.1 Custom tokenization

For the languages with custom tokenization (Japanese, Chinese and Vietnamese), we use the UDPipe model for parsing. Table 2 displays our improvements on tokenization and the resulting improvements on LAS. It also reports the LAS of the UDPipe models using gold tokenization and sentence segmentation, which singles out the LAS drop due to tokenization issues.

It appears that for all four treebanks, tokenization is an important cause of errors, and that our tokenization improvements (+2 to +4 on F1) result in large LAS improvements (+2 to +8 on F1).

	ja	ja_pud	zh	vi
Trainset size	6,805		3,797	1,330
UDPipe tokenization	89.68	91.06	88.91	82.47
LIMSI tokenization	93.82	94.93	91.35	87.30
UDPipe (gold seg.)	*90.99*	*92.12*	*70.04*	*53.28*
UDPipe LAS	72.21	76.28	57.40	37.47
LIMSI LAS	80.01	82.99	59.98	42.02

Table 2: Tokenization and LAS results on the treebanks with custom tokenization.

4.2 Treebanks over 10,000 sentences

Table 3 reports the scores obtained on the largest treebanks, and the corresponding baselines. It includes evaluation of both UDPipe and PanParser based on gold segmentation, for a better comparison of parsers (as they were trained, tuned and selected using gold segmentation), and an assessment of the LAS drop due to segmentation errors.

		cs	ru_syntagrus	cs_cac	la_ittb	no_bokmaal	fi_ftb	grc_proiel	fr	es_ancora	la_proiel
	Trainset size	65,070	46,373	22,304	15,017	14,911	14,231	14,103	13,825	13,589	13,482
GOLD SEG.	*UDPipe*	*83.76*	*87.55*	*82.47*	*77.74*	*83.94*	*76.07*	*70.69*	*82.15*	*83.97*	*67.22*
	PanParser	*78.65*	*82.41*	*79.80*	*74.78*	*83.38*	*75.49*	*69.03*	*81.33*	*83.50*	*65.36*
TIRA	UDPipe (F1)	82.87	86.76	82.46	76.98	83.27	74.03	65.22	80.75	83.78	**57.54**
	LIMSI (F1)	82.87	86.76	82.46	76.98	83.27	**74.04**	65.22	80.75	83.78	57.51

		es	no_nynorsk	de	hi	ca	it	en	nl	fi	grc
	Trainset size	13,477	13,465	13,412	12,638	12,466	12,196	11,915	11,713	11,606	10,902
GOLD SEG.	*UDPipe*	*81.97*	*82.71*	*71.33*	*86.84*	*85.46*	*85.90*	*80.72*	*71.10*	*75.41*	*56.16*
	PanParser	*80.58*	*81.04*	*72.98*	*85.88*	*84.55*	*85.25*	*79.64*	*70.10*	*73.87*	*52.45*
TIRA	UDPipe (F1)	81.47	81.56	69.11	86.77	85.39	85.27	75.84	**68.90**	73.75	56.04
	LIMSI (F1)	81.47	81.56	**70.89**	**86.82**	85.39	**85.28**	75.84	68.31	73.75	56.04

Table 3: LAS results on the large treebanks. The models selected on development data are underlined. For 3 languages, we selected other models: UDPipe+PanParser for `la_proiel`, a monolingual Cascade (using UDPipe and PanParser) for `hi` and `it`.

		pt_br	bg	sk	pt	ro	hr	sl	pl	ar	nl_lassysmall	eu	he	fa	id	ko	da
	Trainset size	9,180	8,461	8,058	7,914	7,640	7,304	6,154	5,795	5,771	5,738	5,126	4,978	4,558	4,253	4,180	4,163
GOLD SEG.	*UDPipe*	*85.79*	*84.55*	*74.23*	*83.10*	*80.57*	*77.51*	*81.20*	*79.31*	*73.85*	*81.34*	*69.23*	*78.31*	*79.82*	*75.02*	*60.04*	*74.98*
	PanParser	*84.38*	*84.13*	*75.90*	*81.72*	*80.55*	*78.21*	*81.64*	*80.42*	*74.20*	*81.54*	*67.45*	*77.83*	*79.33*	*74.34*	*62.19*	*75.24*
	Cascade		*84.09*	*75.19*			*77.28*	*81.24*	*79.88*		*80.48*	*69.31*	*77.60*	*79.30*	*75.18*	*59.60*	*74.92*
TIRA	UDPipe (F1)	85.36	**83.64**	72.75	82.11	79.88	77.18	81.15	78.78	65.30	78.15	69.15	57.23	79.24	74.61	59.09	73.38
	LIMSI (F1)	85.36	83.22	**74.45**	**82.19**	**80.11**	**78.02**	**81.37**	**79.95**	**65.86**	78.15	**69.21**	57.23	79.24	**74.78**	59.09	**73.85**

		sv	cu	ur	ru	tr	got	sv_lines	en_lines	lv	gl	et	fr_sequoia	sl_sst	el	la	en_partut
	Trainset size	4,087	3,916	3,840	3,657	3,500	3,217	2,601	2,601	2,197	2,162	2,149	2,119	1,816	1,578	1,133	1,035
GOLD SEG.	*UDPipe*	*77.28*	*72.56*	*76.74*	*74.45*	*55.93*	*68.98*	*74.94*	*73.64*	*61.15*	*77.46*	*59.87*	*82.10*	*55.22*	*79.92*	*43.81*	*74.08*
	PanParser	*77.14*	*74.58*	*76.26*	*76.07*	*57.41*	*69.50*	*74.68*	*73.87*	*60.99*	*76.74*	*60.96*	*81.76*	*56.62*	*79.58*	*44.17*	*73.89*
	Cascade	*76.99*	*72.67*	*75.69*	*75.17*	*57.59*	*69.69*	*73.98*	*72.91*	*59.85*		*59.37*	*82.62*	*55.49*	*80.05*	*43.67*	*72.80*
TIRA	UDPipe (F1)	76.73	62.76	**76.69**	74.03	53.19	59.81	74.29	72.94	**59.95**	77.31	58.79	79.98	46.45	79.26	**43.77**	**73.64**
	LIMSI (F1)	76.73	**65.64**	76.65	**75.65**	**55.23**	**60.94**	74.29	72.94	59.81	77.31	**59.80**	**80.55**	**46.71**	**79.38**	43.55	73.60

Table 4: LAS results on the medium treebanks. The models selected on development data are underlined. For `pt`, `ro`, `sl`, `id`, `ur`, `sl_sst` and `en_partut`, we rather selected UDPipe+PanParser.

In most cases, our submission is simply the baseline UDPipe, as we did not focus on improving the system for large treebanks. Thus, no improvement is reported on these treebanks.

Model selection proves successful on all treebanks except Latin-PROIEL and Dutch. In particular, it detects that the German PanParser is significantly better than UDPipe; considering the PanParser LAS on the other large treebanks, this is unexpected and remains to be investigated.

4.3 Treebanks from 1,000 to 10,000 sentences

The results for the medium treebanks are reported in Table 4. Compared to Table 3, it also includes gold segmentation evaluation of the monolingual Cascades, when they were considered.

As treebank size reduces, PanParser is more and more often the best parser; but with smaller development sets, the number of failures to select the best model increases (12 out of 32 treebanks).

The Cascade model provides significant gains on several treebanks, outperforming both UDPipe and PanParser; it is indeed able to extract knowledge from the lowest parser and use it to improve upon the best parser (see Basque, Indonesian, Turkish, Gothic, French-Sequoia and Greek). However, these gains are not consistent, and despite confidence and tuning mechanisms, the method still requires empirical validation.

4.4 Treebanks under 1,000 sentences

Table 5 presents the last group of UD treebanks, the smallest ones, including surprise languages.

For each treebank, we report several scores, now including Delex, which is a promising candidate to parse the smallest treebanks. When cross-lingual methods are used, we indicate the source treebank and the scores of the projected base parsers (except for the surprise languages lacking parallel data), X-Delex and their combination X-Cascade.

For this group, since the development sets are very small, model selection is not reliable and misses the best model in 5 out of 12 cases. Additionally, the even smaller tuning sets lead for the cascades to poor estimation of competence regions. But the main challenge faced by cascades is noisy tags. Indeed, while preliminary experiments on X-Cascade with gold tags were very promising, turning to predicted tags makes the X-Delex models very unreliable (as their only features are unreliable), and they cannot provide useful insights to the cascade anymore, in which case cascades lack interest.

Regarding Uyghur, Kazakh and the surprise languages, for lack of data we decided to reduce the training set in favor of the tuning set, in the hope that better estimated cascades would compensate for worse base parsers. This proves to be a bad strategy in half the cases, and we end up underperforming the baseline by a large margin in Kazakh and Buryat.

However, X-Cascades still achieve significant improvements over their base parsers in Uyghur, North Sámi and Buryat. This suggests that cascading can indeed prove useful in cross-lingual contexts; with the appropriate amount of data and additional effort on estimation of competence regions and on robustness to noisy tags, it may convey much larger gains.

4.5 Model selection for PUD treebanks

The PUD treebanks are the additional test sets provided for 14 of the languages covered by the UD treebanks. They have been annotated separately and do not correspond to a given UD treebank. As such, processing them with systems trained on UD treebanks is prone to domain adaptation issues: the PUD treebanks contain sentences extracted from newswire and Wikipedia, while UD treebanks also cover several other domains. For some of the PUD languages, several UD treebanks are available, which raises the additional question of choosing training data: either one of the UD treebanks, or all of them.

For the PUD treebanks, we submitted baseline UDPipe models in Czech, English, French, Spanish, Finnish, Portuguese, Russian and Swedish, yet on 5 of these treebanks we suffered huge drops (-9 to -34 LAS) from the official baseline submission. For these, the only difference with the baseline is that we selected different treebanks. In order to understand these drops, we performed a systematic evaluation of various preprocessing and parser choices, comparing the same system with different training data.

Table 6 reports, for each PUD treebank, the results when applying each UDPipe pipeline on gold segmentation, and when combining each UDPipe preprocesser (segmentation and tagging) with each UDPipe parser. Experiments with PanParser yield consistent results.

In some cases, important drops seem due to incompatible preprocessing between treebanks of the same language: in English, Portuguese and Swedish, the 'main' scores are much lower and 'variant 1' scores much higher when replacing 'main' preprocessing by 'variant 1'. As the preprocessing provided by the organizers was systematically produced by the 'main' UDPipe model, this certainly affected our submission, which used the provided preprocessing even when not using the 'main' parser.

However, comparing results where treebank choices are consistent, it appears that the 'main' treebank always outperforms the variants by a large margin. This is not only due to differences in tokenization, as this occurs also with gold segmentation.

Additionally, in Table 6, the 'All data' results are those obtained with a UDPipe model trained on the concatenation of all treebanks of the language. In most cases they underperform the 'main' model, which confirms that the variant treebanks add mostly noise to the model, from the PUD perspective.

While there may be various explanations to these accuracy differences, what is surprising here is that the best treebank does not seem consistent with common factors of high accuracy (treebank size, domain similarity, treebank consistency). For instance, Russian contains Wikipedia articles and Russian-SynTagRus news (and Russian-PUD consists of both Wikipedia articles and news), but Russian-SynTagRus is much larger than Russian; as such, parsers trained on Russian-SynTagRus should be more accurate on Russian-PUD than Russian, which they are not. Besides, Russian-SynTagRus performs better than Russian on their own test sets, indicating that the treebank does not have strong self-consistency issues. For these reasons, and the additional cue of incompatible pre-

		hu	uk	fr_partut	gl_treegal	ga	cs_cltt	ug	kk	hsb	kmr	sme	bxr
	Trainset size	864	733	527	510	481	441	75	12	10	10	10	10
GOLD	*UDPipe*	64.67	60.92	78.77	68.50	62.25	72.77	36.66	27.10	32.91*	27.93*	20.72*	12.88*
	PanParser	65.60	61.97	79.78	68.36	63.38	74.94	36.21*	23.48*	47.27*	39.38*	31.99*	28.59*
	Delex	61.97	59.79	74.14	66.01	59.44	66.42	36.54*	22.75*	46.44*	38.27*	32.84*	29.73*
	Cascade	65.46	61.64	79.13	68.75	62.55	73.51	35.83*	23.99*	46.55*	37.80*	31.35*	25.91*
	Source	fi_ftb	sk	fr_sequoia	gl	id	cs_cac	tr	tr	sl	fa	et	et
SEG.	*Project-en*	44.01	52.59			42.48		16.65	13.87	37.77			
	Project	36.61	55.52			38.46		23.27	23.08	43.27			
	X-Delex	41.42	54.75			38.80		22.89	25.63	62.20	43.32	37.58	26.16
	X-Cascade		57.43			60.09		37.35*	22.35*	54.61*	40.26*	37.79*	30.13*
	Multi-source									68.78		41.52	
TIRA	UDPipe (F1)	64.30	60.76	77.38	65.82	61.52	71.64	34.18	**24.51**	53.83	32.35	30.60	**31.50**
	LIMSI (F1)	**65.18**	**61.68**	**78.30**	**65.85**	**61.94**	**73.49**	**34.70**	20.94	**57.79**	**35.59**	**31.03**	25.86

Table 5: LAS results on the small treebanks. For the last 4 columns (surprise languages), 'gold seg.' results use gold segmentation *and* gold tagging. '*' denotes parsers whose monolingual training data is smaller than the data used by the UDPipe baseline, hence important score differences. The models selected on development data are underlined. The Multi-source line reports the scores of two models: 'Generic multi-source delex' for `sme`, and 'Multi-source delex' for `hsb`, with the heuristic retaining only treebanks over 2,000 sentences.

			cs_pud	en_pud	fr_pud	es_pud	fi_pud	it_pud	pt_pud	ru_pud	sv_pud	ar_pud	de_pud	hi_pud	ja_pud	tr_pud
GOLD	SEG.	*UDPipe Main*	81.10$_{+}$	79.58$^{*}_{+}$	75.54$_{+}$	78.40	78.84$_{+}$	84.43$_{+}$	74.32	70.36	73.05$_{+}$	52.78$_{+}$	68.03$_{+}$	52.49$_{+}$	92.12$_{+}$	37.37$_{+}$
		UDPipe Variant 1	75.14	64.67*	68.20	74.38$_{+}$	48.70*	76.62	72.22$_{+}$	61.81$_{+}$	66.58*					
		UDPipe Variant 2	44.67*	65.65	63.57											
SEG.+TAG	MAIN	*UDPipe All data*	79.32	79.05	74.00	77.58	76.49	83.57	74.27	64.49	69.97					
		UDPipe Main	79.80	78.95	73.63	77.65	78.65	83.70	73.96	68.31	70.62	43.14	66.53	50.85	76.28	34.53
		UDPipe Variant 1	76.58	47.30	68.31	68.40	44.99	80.35	59.50	52.36	49.41					
		UDPipe Variant 2	54.65	66.42	68.18											
	VAR1	*UDPipe Main*	75.36	63.42	69.04	70.75	52.80	78.43	62.83	68.18	51.63					
		UDPipe Variant 1	73.03	64.28	66.08	71.30	47.27	75.75	69.82	59.87	65.11					
		UDPipe Variant 2	53.32	55.68	64.39											
	VAR2	*UDPipe Main*	47.31	68.53	63.18											
		UDPipe Variant 1	46.10	45.28	59.20											
		UDPipe Variant 2	41.66	64.82	61.10											
TIRA		UDPipe	79.80	78.95	73.63	**77.65**	**78.65**	**83.70**	**73.96**	**68.31**	**70.62**	43.14	66.53	50.85	76.28	**34.53**
		LIMSI	79.80	78.95	73.63	68.40	44.99	83.69	59.50	52.36	49.41	**43.91**	**68.62**	**50.91**	**82.99**	34.15

Table 6: LAS results on the PUD treebanks. For `lang_pud`, 'main' denotes the `lang` treebank. Variants 1 are `cs_cac`, `en_lines`, `fr_sequoia`, `es_ancora`, `fi_ftb`, `it_partut`, `pt_br`, `ru_syntagrus` and `sv_lines`. Variants 2 are `cs_cltt`, `en_partut` and `fr_partut`.

For each language, the largest treebank is annotated with '$_{+}$' (considering numbers of tokens: `fi_ftb` contains more sentences than `fi`, but they are shorter), and '*' indicates treebanks with important domain adaptation issues (i.e. that contain neither Wikipedia nor news data).

Underlined results denote the training treebanks that we used to annotate the PUD treebanks (not necessarily with a UDPipe model). The baseline UDPipe submission corresponds to 'main'+'main'.

For instance, in the `cs_pud` column, the 3rd row ('gold'+'variant 2') corresponds to gold tokenization and segmentation, tagging with the `cs_cltt` UDPipe model, and parsing with the `cs_cltt` UDPipe model. The 8th row ('var1'+'main') corresponds to tokenizing and tagging with the `cs_cac` UDPipe model, and parsing with the `cs` UDPipe model.

processings, we speculate that the PUD treebanks are indeed biased towards the main treebank of each language, because of annotation scheme discrepancies between treebanks. Such inconsistencies may be due either to one treebank not following the guidelines, or to underspecified aspects of the guidelines that have been interpreted differently by different teams.

Notably, Arabic and Hindi also present huge drops from their scores on UD test sets, even though they trained only on news; it is possible that here as well, the drops are due only to annotation inconsistencies.

Concluding on this hypothesis would require, however, further analysis of the treebank domains: the scores may still be partially explained by actual domain adaptation issues, e.g. the relative size of each domain in multi-domain treebanks, or details of the domains (style, author, date...) that do not appear in the coarse domain categories (news, Wikipedia, fiction...).

Full examination of the annotation inconsistencies is an ongoing work of the UD project, and is out of the scope of this paper, but during our manual analysis we already noticed that plain text (both in the training data, and the raw input of the test data) is in fact already partially preprocessed, for some UD treebanks. For some, multitoken words are already detected and annotated with '_' instead of spaces: this concerns at least Russian-SynTagRus (phrasal conjunctions and numbers), Finnish-FTB (numbers) and Greek (dates). For others (Danish, Finnish-FTB), plain text is already fully tokenized (except for multiword tokens).

This is not an issue in general, because the UD treebanks are self-consistent on this convention, but it affects the ability of the trained models to process actual raw input. This partially explains the PUD bias, since Russian and Finnish main treebanks have actual raw text, as the PUD ones.

5 Conclusion

Our submission to the *CoNLL 2017 UD Shared Task* focuses on low-resourced dependency parsing. Our system is built upon base parsers, and combines them using a cascading algorithm, in order to leverage small and incomplete data.

This shared task was an opportunity to experiment with this new cascading method in realistic conditions; this is particularly interesting, since this method addresses precisely realistic scenarios where available data does not consist in either large monolingual data or large parallel data, but in various amounts of each resource type.

The method has proved useful in many cases, with sometimes large improvements, but the gains are not consistent enough to be reliable and still require further work. The shared task conditions have indeed uncovered several challenges faced by the method: it lacks confidence mechanisms, delexicalized models are too unreliable, and the lack of development data hinders accurate estimation of competence regions for the components of the cascade.

The improvements we achieve are strongly diminished by huge unexpected drops on a few languages; while this affects our ranking in great proportions, it also enables detailed analysis of the new PUD treebanks, and on how and why they are biased towards the main UD treebanks.

Acknowledgments

This work has been partly funded by the French *Direction générale de l'armement* and by the *Agence Nationale de la Recherche* (ParSiTi project, ANR-16-CE33-0021). We thank Joseph Le Roux for fruitful discussions and comments.

References

Ethem Alpaydin and Cenk Kaynak. 1998. Cascading classifiers. *Kybernetika* pages 369–374.

Lauriane Aufrant and Guillaume Wisniewski. 2016. PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing. Technical report, LIMSI-CNRS.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 119–130. http://aclweb.org/anthology/C16-1012.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2017. Don't Stop Me Now! Using Global Dynamic Oracles to Correct Training Biases of Transition-Based Dependency Parsers. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 318–323. http://www.aclweb.org/anthology/E17-2051.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 644–648. http://www.aclweb.org/anthology/N13-1073.

Ludmila I Kuncheva. 2004. *Combining pattern classifiers: methods and algorithms*.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly Easy Cross-Lingual Transfer for Transition-Based Dependency Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1058–1063. http://www.aclweb.org/anthology/N16-1121.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 529–533. http://www.aclweb.org/anthology/P11-2093.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017a. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague. http://hdl.handle.net/11234/1-1983.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017b. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-2184.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Portoro, Slovenia, pages 1659–1666.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the reproducibility of PAN's shared tasks: Plagiarism detection, author identification, and author profiling. In Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*. Springer, Berlin Heidelberg New York, pages 268–299. https://doi.org/10.1007/978-3-319-11382-1_22.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. KLcpos3 - a Language Similarity Measure for Delexicalized Parser Transfer. 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 243–249.

Milan Straka. 2017. CoNLL 2017 Shared Task - UDPipe Baseline Models and Supplementary Materials. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-1990.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Portoro, Slovenia.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

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, Jan Hajič jr., 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, Hĕctor Martínez Alonso, 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, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, 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*. Association for Computational Linguistics.