Delexicalized transfer parsing for low-resource languages using transformed and combined treebanks

Ayan Das, Mohammad Affan Zafar, Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, WB, India
ayan.das@cse.iitkgp.ernet.in
affanzafar07@gmail.com
sudeshna@cse.iitkgp.ernet.in

Abstract

This paper describes the IIT Kharagpur dependency parsing system in CoNLL-2017 shared task on Multilingual Parsing from Raw Text to Universal Dependencies. We primarily focus on the low-resource languages (surprise languages). We have developed a framework to combine multiple treebanks to train parsers for low resource languages by a delexicalization method. We have applied transformation on the source language treebanks based on syntactic features of the low-resource language to improve performance of the parser. In the official evaluation, our system achieves macro-averaged LAS scores of 67.61 and 37.16 on the entire blind test data and the surprise language test data respectively.

1 Introduction

A dependency parser analyzes the relations among the words in a sentence to determine the syntactic dependencies among them where the dependency relations are drawn from a fixed set of grammatical relations. Dependency parsing is a very important NLP task and has wide usage in different tasks such as question answering, semantic parsing, information extraction and machine translation.

There has been a lot of focus recently on development of dependency parsers for low-resource languages i.e., the languages for which little or no treebanks are available by cross-lingual transfer parsing methods using knowledge derived from treebanks of other languages and the resources available for the low-resource languages (McDonald et al., 2011; Tiedemann, 2015; McDonald et al., 2011; Zeman and Resnik, 2008; Rasooli and Collins, 2015).

The Universal Dependencies (http://universaldependencies.org/) (Nivre et al., 2016) project has enabled the development of consistent treebanks for several languages using an uniform PoS, morphological features and dependency relation tagging scheme. This has immensely helped research in multi-lingual parsing, cross-lingual transfer parsing and the comparison of language structures over several languages.

The CONLL 2017 shared task focusses on learning syntactic parsers starting from raw text that can work over several typologically different languages and even surprise languages for which no training data is available using the common annotation scheme (UD v2). The details of the task are available in the overview paper (Zeman et al., 2017).

For parsing the surprise languages we trained delexicalized parser models. In order to improve performance on the surprise languages we applied syntactic transformation on some source language treebanks based on the information obtained from the "The World Atlas of Language Structures" (WALS) (Haspelmath, 2005) and sample data and used the transformed treebanks to train the parsers for the surprise languages. The details of the treebanks are discussed in Section 3.1.

The rest of the paper is organized as follows. In Section 2 we describe the corpora and resources used to build our system. In Section 3 we describe in details the methods used to train the parser models. In Section 4 we describe the experiments and report the results, and, we conclude in Section 5.

2 Corpus and resources

We used the treebanks (UD v2.0) (Nivre et al., 2017b) which were officially released for the shared task to train our parser models. The dataset

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 182–190,
Vancouver, Canada, August 3-4, 2017.

consists of 70 treebanks on 50 different languages. There are multiple treebanks for some languages such as Arabic, English, French, Russian etc. For the shared task, only the training and development data was released. The small sample treebanks (approximately 20 sentences per language) for the surprise languages were made available separately one week before the test phase.

We have used the pre-trained word vectors of 50 dimensions provided by the organizers to train the parser models. For tokenization and tagging we used the baseline models provided by the organizers. Our parser models were trained using the Parsito parser (Straka et al., 2015) implemented in UDPipe (Straka et al., 2016) text-processing pipeline system.

3 System description

Our parser worked on parsed the tokenized and tagged files (`*-udpipe.conllu`) provided by the organizers rather than the raw text files. We first discuss the steps for training the models for surprise languages in Section 3.1 followed by methods used to train the models for the new parallel treebanks in Section 3.2 and known treebanks in Section 3.3.

3.1 Surprise language

The surprise languages are Buryat (bxr), Kurmanji (Kurdish) (kmr), North Sámi (sme) and Upper Sorbian (hsb) for which sample data of approximately 20 annotated sentences per language was made available. No training or test data is available for the surprise languages.

We have used cross-lingual parser transfer to develop parsers for the surprise languages using the treebanks of resource-rich languages (McDonald et al., 2011). Annotation projection (Hwa et al., 2005) and delexicalized transfer (Zeman and Resnik, 2008) are the two major methods of cross-lingual parser transfer.

However, annotation projection requires parallel data which is not available for the surprise languages. Hence, we used the delexicalized parser transfer method to train parser models for the surprise languages. Training delexicalized parser involves supervised training of a parser model on a source language (SL) treebank without using any lexical features and then applying the model directly to parse sentences in the target language (TL). Zeman and Resnik (2008) and Søgaard (2011) have shown that cross-lingual transfer by delexicalization works best for syntactically related language pairs.

The first step was to identify the languages which are syntactically related to the surprise languages and whose treebanks are in Universal Dependency corpus. We observed that Upper Sorbian being a slavonic language is typologically related to Czech, Polish and to some extent Slovak. North Sámi is spoken in the northern parts of Norway, Sweden and Finland. It belongs to the family of Finno-Ugric languages and hence has typological similarities with Estonian, Finnish and Hungarian. Kurmanji has typological similarities with Persian and Turkish. Buryat is spoken in Monglia. Although none of the languages whose treebanks are available in Universal Dependencies corpus belong to the family of Buryat yet we guessed that Kazakh, Tamil, Hindi and Urdu might have some similarities with Buryat based on the syntactic features and phrasal structures of the languages.

In order to verify our guesses, we tested the delexicalized models trained on individual treebanks on the sample data for the surprise languages and ranked them based on LAS. We observed that our guesses were quite close to the actual results except a few cases. Table 3.1 lists the top-5 languages for each of the surprise language based on LAS score. Encouraged by the above results that support our guesses we explored a transformation-based method to further reduce the syntactic differences between the surprise languages and the corresponding source languages. Besides attempting to reduce the syntactic differences between the languages we also experimented with combining the treebanks for which the individual LAS scores were highest to further boost the LAS on the surprise languages.

3.1.1 Syntactic feature based transformation

Aufrant et al. (2016) have shown that local reordering of words of the source sentence using PoS language model and linguistic knowledge derived from WALS improve performance of the delexicalized transfer parser even for syntactically different SL-TL pairs. The reordering features they use are relative orderings of the adjectives, adpositions, articles (definite and indefinite) and demonstratives with respect to the corresponding modified nouns in the TL.

However, for language pairs that differ in the arrangement of verb arguments, local rearrange-

Buryat	Kazakh (43.14)	Latvian (37.25)	Hindi (37.25)	Tamil (35.95)	Finnish-ftb (35.29)
Kurmanji (Kurdish)	Polish (45.04)	Persian (42.15)	Bulgarian (41.74)	Romanian (40.91)	Czech (40.5)
North Sámi	Finnish (54.42)	Swedish (51.02)	Estonian (48.98)	Norwegian (46.26)	Lithuanian (45.58)
Upper Sorbian	Slovak (71.5)	Slovenian (65.9)	Bulgarian (63.91)	Polish (63.48)	Czech (62.83)

Table 1: Top 5 languages whose delex models gave the highest LAS on the surprise language sample data

	81A	84A	87A	85A	37A	38A	88A	86A	aux	cop	cco-mp	xco-mp	acl	advcl	90A
bxr	SOV	-	pre	post	×	×	pre	pre	post	post	post	-	pre	pre	×
kmr	SOV	-	post	pre	pre	pre	pre	post	pre	post	pre	pre	post	pre	×
sme	SVO	-	pre	post	×	×	×	pre	pre	post	-	pre	post	pre	post
hsb	SVO	-	pre	pre	pre	pre	pre	post	-	-	post	post	post	post	×

Table 2: Ordering of the head-modifier pairs in the target language as derived from the universal dependency treebank statistics. "pre" indicates that the modifier precedes the head, "post" indicates that the modifier succeeds the head and "-" indicates that the ordering cannot be decided. "×" shows that the dependency does apply to the language. Some of the feature identifiers are derived from WALS: 81A - order of subject, object and verb in a sentence, 84A - order of object, oblique and verb, 87A - ordering of ADJ and NOUN, 85A - ordering of ADP and NOUN, 37A/38A - ordering of Definite/Indefinite articles and NOUN, 88A - ordering of Demonstrative and NOUN, 86A - ordering of genitive and NOUN, 90A - ordering of relative clause and VERB

	Our system		Best scores		Our rank
	UAS	LAS	UAS	LAS	
All test data	73.68	67.61	81.30 (Stan.)	76.30 (Stan.)	19
Surprise treebanks	49.98	37.16	58.40 (C2L2)	47.54 (C2L2)	10
Big treebanks	77.32	72.68	85.16 (Stan.)	81.77 (Stan.)	18
New parallel treebanks	74.45	67.42	80.17 (Stan.)	73.73 (Stan.)	17
Small treebanks	59.14	48.33	70.59 (C2L2)	61.49 (C2L2)	22

Table 3: UAS and LAS on blind test as obtained by primary system run and their comparison with the best runs. Stan. and C2L2 refer to the systems submitted by Stanford University and C2L2 (Ithaca) respectively

ments based on PoS tags may not be enough to address the difference between the two languages. Hence, we hypothesize that a reordering of SL sentences based on more generic features such as dependency relations (subject, object, indirect object, clausal complement) might result in improvement in accuracy of the parser.

An example of syntactic transformation In order to illustrate the syntactic difference between two languages we put forth an example for English-Kurmanji (Kurdish) language pair.

The example in figure 1 shows the syntactic difference between the two languages - English and Kurmanji (Kurdish) - and how the transformation of the English sentence makes it syntactically

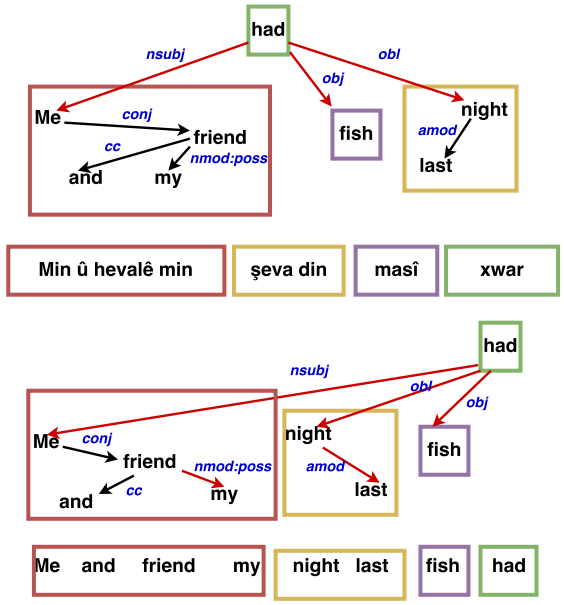

Figure 1: Transformation of English sentence to match the syntax of Kurdish sentence based on the syntactic features of Hindi

closer to Kurmanji (Kurdish). English language sentences have a SVO sentence structure while Kurmanji (Kurdish) has a SOV sentence structure. Moreover, in English the oblique arguments tend to appear after the object in the sentence while in Kurmanji (Kurdish) the oblique arguments tend to appear before the object.

The English sentence "Me and my friend had fish last night" may be translated to Kurmanji (Kurdish) as " *Min û hevalê min şeva din masî xwar (Me and friend my night last fish had)*". In this sentence pair *Me and my friend (Min û hevalê min)* and *fish (masî)* are the subject and the object of the main verb *had (xwar)* and *last night (şeva din)* is the non-core (oblique) argument indicating the time of occurrence of the verb. Also, in Kurmanji (Kurdish), the adjectival modifiers and genitives occur after the modified noun e.g., *hevalê min (friend my)* and *şeva din (night last)*, while in English these modifiers occur before the modified noun e.g. *my friend* and *last night*.

3.1.2 Transformation features

Apart from the features proposed by Aufrant et al. (2016) we obtained the order of subject-object-verb (SOV), order of object-oblique-verb, and the relative order of relative clause, auxiliaries, copula verbs, clausal complements, clausal modifier (adjectival and adverbial) with respect to the modified verb from WALS and the statistics of sample data. The relative ordering of head-modifier pairs based on the features derived from treebanks was determined using the following heuristic. If a particular order appears in at least 90% cases out of the total number of occurrences of the feature (dependency tag) then we use that ordering corresponding to the feature. Else we do not do any transformation based on that feature. We relied on the WALS features and the statistics of the sample data to derive the syntactic features and ignored the features that did not appear in these two sources. For example, although Buryat, Upper Sorbian and Kurmanji have relative clauses we neither did we find mention of the feature in WALS for the language nor did we find that relation in sample data for these languages. Hence, we did not use that relation during transfer. In Table 3.1 we summarize the transformation features and the corresponding orderings used for each surprise language.

We categorized the dependency relations into six classes.

- **Clausal complements :** ccomp, xcomp
- **Subject :** nsubj, nsubj:pass, csubj, csubj:pass
- **Object :** obj, iobj
- **Modifiers :** acl, advcl, amod, aux, case, cop, det (Pronominal type = Article or Demonstrative, and, Definiteness = Definite, Indefinite)
- **Other dependencies :** Dependency labels that do not belong to the above five classes (cc, conj, punct, mwe, foreign etc.).

Among the determiners we only considered the articles and demonstratives. We further divide the members of class *modifiers* into *pre-modifiers* and *post-modifiers* depending upon the position they take in the sentences with respect to the parent word in TL.

3.1.3 Tree-traversal based transformation algorithm

Given a source language sentence $S = \{w_1, \cdots, w_m\}$, where m is the length of S, let T_S be the parse tree of S. The transformation is carried out in two steps.

- **Step 1:** Remove the words corresponding to the dependency relations that do not hold in the TL from the SL parse tree e.g., remove

Demonstratives when North Sámi is the target language.

- **Step 2:** Rewrite the sentence by a tree-traversal method depending upon the ordering of the head-modifier pairs based on the transformation features.

Corresponding to each target language we have separate transformation procedures. The Procedure BuildTree is common for all the target languages. In this procedure we construct the tree data structure where each node in the tree corresponds to a word in the sentence. Each node consists subject, object, clausal complement, *premodifier*, *postmodifier* and *other-modifier* lists. The lists of a node are filled up only by the dependents of the corresponding word in the dependency tree. The subjects, objects and the clausal complements of the word are added to the corresponding lists. While constructing the pre-modifier, post-modifier and other modifier lists the module refers to a look-up table to obtain the order of the modifiers in the TL and place the modifiers in the corresponding lists. All the lists are not necessarily filled up e.g., if none of the dependents hold a subject relation (nsubj or nsubj:pass) with the word then the subject list of the corresponding node remains empty.

We have separate procedures for transforming the SL trees for each TL. The sentences are transformed by traversing the trees according to the ordering of the dependencies in the TL e.g., the subtrees corresponding to the modifiers in the *pre-modifier* list and the modifiers in the *other-modifier* list that appear before the current word in the SL sentence are traversed first, then the word of the current node is added to the transformed word list, followed by traversal of the subtrees corresponding to the modifiers in the *post-modifier* list and the words in *other-modifiers* list that appear after the current word in the SL sentence. Also, if the TL has SVO sentence structure, first the subtree corresponding to the subject is traversed, the verb is added to the transformed list and finally the subtree corresponding to the object is traversed. Procedure TraverseAndTransformTree illustrates the steps used for transforming the SL tree when the the target language follows SOV ordering of verb arguments and the clausal complements occur before the verb.

Language	Treebanks	Transformation	Number of tokens
bxr	Kazakh (kk)	-	547
kmr	Polish (pl)	-	175600
	Slovenian (sl)	37A, 38A, 86A, 87A	
sme	Finnish (fi)	-	187920
	Estonian (et)	88A	
	Lithuanian (lt)	85A, 88A, cop, xcomp, acl, advcl	
hsb	Slovak (sk)	-	273680
	Slovak (sk)	81A, 37A, 38A, 86A, ccomp, advcl	
	Slovenian (sl)	37A, 38A, 86A	

Table 4: The treebanks combined to train the parser models for surprise languages in the primary system. The 'Transformation' column lists the syntactic features on which the source treebanks were transformed. '-' implies untransformed treebank

3.1.4 Steps for training the parser for surprise languages

For each language we used the following steps to train the delexicalized parser:

1. We obtained the syntactic features proposed by Aufrant et al. (2016) from WALS and the sample data.

2. Besides the features obtained in step 1, we also derived some more syntactic features from WALS and sample data statistics such as ordering of subject (S)-object (O)-verb (V)

Procedure BuildTree

input : Source language parse tree T_S
output: Tree data structure T

T = node n_{root}, containing the root word (w_{root}), POS ($pos_{w_{root}}$), empty children list (cl_{root}), parent link (p_{root}) = $null$
for *each word* w_i *in* S *except the root word* **do**
    Form a node n_i containing, w_i, pos_{w_i}, cl_i, p_i, dependency relation with p_i = d_i
    Add n_i to the children list of p_i
    Add n_i to the $pre-modifier$, $post-modifier$ or $other-modifier$ list based on TL features
return T

in a sentence, relative ordering of auxiliaries, copula verbs, clausal complements, adjectival and adverbial clausal modifiers.

3. We transformed all the available treebanks based on the syntactic features described in step 1 and the combination of the features stated in step 1 and 2 using the appropriate transformation procedures.

4. We trained separate delexicalized models for untransformed treebanks and both types of transformations such that corresponding to each source language there are three models - one trained on untransformed treebank and two on transformed treebanks. Universal Dependencies v2.0 corpus consists of 70 treebanks. Hence, after transformation we have 70 $\times$ 3 = 210 treebanks.

5. We ranked the 210 models based on their LAS on the sample data provided for the surprise language and broke ties based on UAS and chose the top 20 treebanks for our next step.

6. We trained 20 models by combining the treebanks in the top-*k* ordering (top-1, top-2,$\cdots$, top-20) and selected the model that gave the highest LAS on the sample data. The treebanks were combined by concatenating the treebank files to form a single treebank e.g. for the top-2 model, we concatenated the two treebanks which ranked first and second with respect to the LAS on the sample data and used the concatenated treebank to train the top-2 model. In Table 3.1.4 we summarize

Procedure TraverseAndTransformTree

input : Source language parse tree data structure T
output: Transformed source language parse tree T_S^R

T_S^R = `RearrangeNodes(`T`)`
return T_S^R
Procedure `RearrangeNodes(`*Root node* n_{root} *of tree datastructure* T`)`
    Rearranged word sequence (S_R) = `TraverseTree(`n_{root}`)`
    $T_S^R = null$
    for i *in* S_R **do**
        i_p^R = index of parent of i in S^R
        Add (i,i_p^R,d_j) to T_S^R
    return T_S^R
Procedure `TraverseTree(`*node* t`)`
    S_R=$null$
    if t *has empty children list* **then**
        Add word (w_t) of t to S_R
    else
        for *Child* c_t *in* cl_t *with clausal complements* ($ccomp, xcomp$) *dependency relation* **do**
            TraverseTree(c_t)
        for *Child* c_t *in* cl_t *with subject* ($nsubj$) *dependency relation* **do**
            TraverseTree(c_t)
        for *Child* c_t *in* cl_t *with* $dobj, iobj$ *dependency relation* **do**
            TraverseTree(c_t)
        for *Child* c_t *in* cl_t *that are* `pre-modifiers` *of* t *or in* `other dependencies` *appearing* ***before*** t *in S* **do**
            TraverseTree(c_t)
        Add word (w_t) of t to S_R
        for *Child* c_t *in* cl_t *that are* `post-modifiers` *of* t *or in* `other dependencies` *appearing* ***after*** t *in S* **do**
            TraverseTree(c_t)
    return S_R

the LAS of our combined treebanks on the sample data. We report only those top-k combinations that have been used in the submitted systems.

3.2 Known language, new parallel treebank

New parallel treebanks were provided for 14 languages in the test data. Out of these 14 languages, we trained the models for German, Hindi, Japanese and Turkish on the single UD treebanks available for each of these languages. Multiple treebanks are available for each of the remaining 10 languages, viz, Arabic, Czech, English, Finnish, French, Italian, Portuguese, Russian and Swedish.

For each language with multiple treebanks we followed the following steps:

1. We combined all the treebanks in that language and trained a parser model on the combined treebank.

2. We tested the combined model and the models trained on the individual treebanks on the development sets of all individual treebanks.

3. We used the combined model for the parallel treebank if it gives uniform UAS and LAS scores across all the development sets and gave significant improvement over the models trained on individual treebanks. Else we used the model trained on the treebank that gave best result across all the treebanks.

We used the combined models for Swedish, English, Finnish, French, Italian, Portuguese, Russian. For Arabic and Czech we used the models trained on the main treebanks (lcode: 'ar' and 'cs', tcode: '0') of the respective languages.

3.3 Known language, known treebank

We trained separate models for each of the 70 Universal Dependecies v2.0 treebanks. We used word, PoS and dependency relation embeddings of 50 dimensions. Apart from these parameters we used the default parameter settings of the UDPipe parser to train our models. The 'small' treebanks for which for which no development data was available, we used the training data itself as development data.

4 Experiments and results

Our system comprises of 88 models. 70 models were trained on the individual treebanks available from http://universaldependencies.org/, 14 models were trained for the new parallel treebanks and 4 models for the surprise language. Given the language code (lcode) and treebank code (tcode), our system identifies the parser model corresponding to the input test treebank and parses the sentences in the treebank file.

The systems were ranked based on macro-averaged LAS. The final evaluation of the parser is on blind test data sets (Nivre et al., 2017a) through TIRA platform set up by Potthast et al. (2014). We submitted 9 systems (software$k$, where $k \in \{2, \cdots, 10\}$). The systems differ in the models trained for the surprise languages. The models corresponding to the known language treebanks and the new parallel treebanks were same in all the systems. Since the test set was blind, the first four systems (software 2 to 5) consisted of a combination of models for the surprise languages that were expected to perform best based on the performance on the sample treebanks. The remaining 5 consisted of models corresponding to combinations of top-k (k= 1, 5, 10, 15, 20) models for each of the surprise languages. Table 3.1.3 lists the treebanks combined to train the models for our primary system. We summarize the macro-averaged LAS scores for the surprise languages for the 8 models in Table 3.3. The highest scoring system for the surprise languages (software2) consists of top-2 model for Buryat, top-10 model for Kurmanji (Kurdish) and top-6 models for North Sámi and Upper Sorbian. The results using the primary system is summarized in Table 3.1 and the macro-average over all submitted softwares are listed in Table 3.3.

5 Conclusion

In this work, we have implemented a system for parsing sentences in several typologically different languages with a special focus on surprise languages for the CoNLL 2017 Shared Task. We have developed a system for combining treebanks to train parsers for surprise languages and applied syntactic transformation of source languages based on the syntactic features of the target languages to improve performance on the target languages. We derived the syntactic features from the WALS and sample data provided. On the surprise languages, the macro-averaged LAS F1-score of our primary system is 37.16 while that of the best performing system (Stanford) is 47.54. However,

Combination	bxr		kmr		hsb		sme	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Top-1	63.1	**43.14**	51.2	45.04	76.52	71.52	63.95	54.52
Top-2	62.18	41.81	52.07	**45.87**	78.26	73.7	70.75	57.82
Top-3	58.17	39.22	45.04	40.91	78.26	74.13	70.75	**59.86**
Top-5	54.25	35.95	47.93	41.74	76.52	71.52	62.59	48.98
Top-6	53.59	32.03	48.6	42.43	77.61	74.13	72.11	59.18
Top-10	59.5	40.52	49.6	43.39	76.96	73.7	72.11	55.1
Top-15	58.17	41.18	40.5	35.54	77.83	**74.57**	70.75	59.18
Top-20	53.59	38.56	45.87	42.15	78.04	73.7	67.35	54.42

Table 5: UAS and LAS scores of models trained on treebank combinations on the surprise language sample data

	bxr	kmr	sme	hsb	Macro-averaged LAS F1 score	Overall macro-averaged LAS F1 score
Primary system (software3)	26.60	32.03	35.25	54.78	37.16	67.61
software2	29.98	32.38	33.27	**55.4**	**37.75**	**67.75**
software4	26.60	32.03	**35.25**	53.37	36.81	67.60
software5	29.98	32.38	32.05	53.37	36.94	67.60
Top-1 (software6)	26.60	32.18	32.83	52.92	36.13	67.56
Top-5 (software7)	29.08	**32.97**	32.88	54.85	37.44	67.63
Top-10 (software8)	28.91	32.38	33.03	53.54	36.96	67.62
Top-15 (software9)	**31.65**	32.27	32.05	53.37	37.33	67.62
Top-20 (software10)	30.44	32.51	32.95	53.23	37.28	67.62

Table 6: Comparison of LAS F1 scores of the submitted systems and their macro-averages on the surprise language test data. The system with highest macro-averaged LAS F1 score (software2) is composed of top-2, top-10, top-6, top-6 models for bxr, kmr, sme and hsb respectively. The *software4* is composed of top-1, top-2, top-3, top-15 models for bxr, kmr, sme and hsb respectively and the *software5* is composed of top-2, top-10, top-15, top-15 models for bxr, kmr, sme and hsb respectively. For the remaining systems (software6-10) we combined the top-5, top-10, top-15 and top-20 treebanks respectively.

the macro-averaged LAS F1-score of our best performing system is 37.75. Our rank with respect to the surprise languages is 10.

The overall macro-averaged LAS F1-score of our primary system is 67.61 as compared the best performing system that has an macro-averaged LAS F1 score of 76.30. The overall macro-averaged LAS F1-score of our best-performing system is 67.75. Our overall rank is 19.

References

Lauriane Aufrant, Guillaume Wisniewski, and Franois 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.

Martin Haspelmath. 2005. *The world atlas of language structures / edited by Martin Haspelmath ... [et al.]*. Oxford University Press Oxford.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering* 11:11–311.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on EMNLP*. Associa-

tion for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 62–72. http://dl.acm.org/citation.cfm?id=2145432.2145440.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017a. 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.

Joakim Nivre et al. 2017b. 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. http://hdl.handle.net/11234/1-1983.

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.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on EMNLP*. Association for Computational Linguistics, Lisbon, Portugal, pages 328–338. http://aclweb.org/anthology/D15-1039.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers.

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.

Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. 2015. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*.

Jörg Tiedemann. 2015. Improving the cross-lingual projection of syntactic dependencies. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*. Linköping University Electronic Press, Sweden, Vilnius, Lithuania, pages 191–199. http://www.aclweb.org/anthology/W15-1824.

D. Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. *NLP for Less Privileged Languages* pages 35 – 35.

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, Hěctor 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.