UNIVERSITY OF ZAGREB

FACULTY OF SCIENCE

DEPARTMENT OF MATHEMATICS

Tina Marić

DEEP MACHINE LEARNING FOR SYNTACTIC ANNOTATION PROJECTION

Master thesis

Mentor:
Goran Igaly

Zagreb, 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

This thesis was made during my internship at the IT University of Copenhagen, that took place from September until the end of December in 2017, under the supervision of professor Željko Agić. I want to thank Željko for all the help, guidance and answers to all of my questions.

Contents

Introduction

The automatic discovery of syntactic structure in natural language texts is instrumental in enabling any sufficiently advanced language technology. Syntactic dependency parsing via supervised learning over hand-annotated texts has seen a stream of major breakthroughs over the last decade or so, where research has enabled highly accurate parsers for over 50 languages, for example via the Universal Dependencies project that makes uniformly annotated training data readily available. Yet, these languages account for only a fraction of the world's written languages. For the remainder, dependency treebanks are either prohibitively small or non-existent. Recently, we are witnessing a resurgence of interest in applying cross-lingual transfer learning to dependency parsing, with a goal of enabling syntactic analysis for these low-resource languages. Thus far, the best such approaches involve annotation projection: the transfer of dependency structures via parallel texts, from resource-rich to low-resource languages. Annotation projection in effect synthesizes the training data in the target languages, enabling the induction of dependency parsers. Since the projection of syntactic dependencies typically takes the form of a static algorithm, it critically depends on the quality of parallel resources, namely the sentence- and word-level alignments. In turn, these alignment procedures are automatic and incur a lot of noise, which contributes to suboptimal parser quality downstream. This work tries to improve some of the existing solutions and proposes an approach to improve the projection of syntactic dependencies through learning to project with deep neural networks.

Chapter 1

Annotations

1.1 Part-of-speech tagging and dependency parsing

In traditional grammar, a part of speech tag (abbreviated form: PoS tag or POS tag) is a category of words (or, more generally, of lexical items) which have similar grammatical properties. Words that are assigned to the same part of speech tag generally display similar behavior. In terms of syntax, they play similar roles within the grammatical structure of sentences and sometimes in terms of morphology. Some part of speech tags for English are given in the Table 1.1. Part-of-speech tagging (abbreviated form: POS tagging or PoS tagging or POST) is the process of assigning a POS tag to every word in a text (corpus) [7]. Example of tagged sentence is in Figure 1.1.

tag	name
ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary
CCONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction

tag	name
SYM	symbol
VERB	verb
X	other

Table 1.1: List of English part of speech tags.

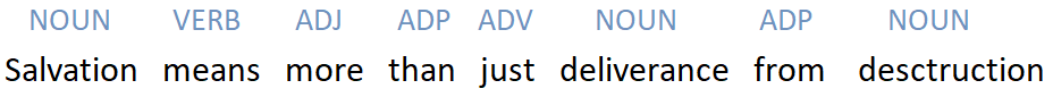


Figure 1.1: Example of sentence with assigned part of speech tags.

A dependency parser analyzes the grammatical structure of a sentence, establishing relationships between ”head” words and words which modify those heads [12]. An example of dependency parsing with TurboParser [2] can be seen in Figure 1.2.

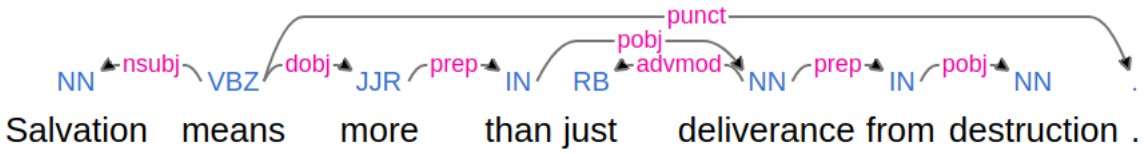


Figure 1.2: Syntactic dependency parse from TurboParser. It outputs part of speech tags using different notation [4].

1.2 Dataset

Our parallel corpus consists of publications from the Watchtower Society, that are being published monthly by Jehovah's Witnesses. They are available in up to 583 languages. We use translations in 26 languages. Accurate parsers exist for 21 of them and we call them source languages (sources). Other 5 will be called target languages (targets). Texts are religious and therefore biased, but the main advantage that we are going to exploit is having resources that are massively multi-parallel.
For each language pair, there are sentences from one language that do not have a translation

in the other. Therefore, Hunalign (Varga et al., 2005) was used to perform conservative sentence alignment. The number of aligned sentences varies from 23 605, for Tamil-Persian pair, up to 96 999, for Dutch-Portuguese pair. Words in aligned sentences were aligned using the IBM1 aligner. Sentences written in source languages were parsed with the arc-factored TurboParser, tagged with the TnT tagger with default settings and then annotated using the format shown in Table 1.2 .

ID	FORM	_	POS	_	_	HEAD	_	_
1	What	_	PRON	_	_	3	_	0:-2.16285 ... 3:2.5009
2	Is	_	VERB	_	_	3	_	0:-1.98414 ... 3:3.15446
3	Salvation	_	NOUN	_	_	0	_	0:-2.76271 ... 4:-1.66405
4	?	_	PUNCT	_	_	1	_	0:-4.5454 ... 3:1.38236

Table 1.2: Example of annotated sentences written in a file.

Chapter 2

Static multilingual projection

In this chapter, we describe the process for syntactic annotation projection presented in the paper Multilingual Projection for Parsing Truly Low-Resource Languages [8]. We recreate presented novel idea for projection to see if we can improve the given "black box" algorithm and how much.

Authors of the paper assume to have n source languages and one target language t. For each pair of translations, their algorithm projects syntactic annotations from n source sentences to the target sentence. Projection step is formalized as a label propagation in a graph structure defined in Definition 2.0.1. Words in sentences represent graph vertices, while graph edges are represented by dependency edges and word alignments.

Definition 2.0.1. *Projection graph is a graph $G = (V, E)$. V is set of vertices that can be decomposed as $V = V_0 \cup ... \cup V_n$ where V_i is the set of words in sentence i. E is a set of edges that are weighted by the function $w_e : E \to \mathbb{R}$.*
Lets denote target vertices as $V_t = V_0$ and source vertices as $V_s = V_1 \cup ... \cup V_n$.
The subgraph $A = (V_s, V_t, E_A)$ induced by all alignment edges is bipartite graph that is connecting V_s and V_t.
The subgraph $G[V_i]$ induced by the set of vertices V_i represents dependency edges between the words of the sentences i.

Definition 2.0.2. *POS projection problem is defined as assigning POS labels to the vertices V_t by transferring the POS labels from vertices $V_1, ..., V_N$ through the alignments A.*

Algorithm 1 Project POS tags

Data: A projection graph $G = (V_s \cup V_t, E)$; a set of POS labels L; a function $p(l \mid v)$ assigning probabilities to labels l for word vertices v.
Result: A POS labeling of V_t
$\widetilde{p} \rightarrow$ empty probability table
$label \rightarrow$ empty label-to-vertex mapping
for $v_t \in V_t$ **do**
  for $l \in L$ **do**
    $\widetilde{p}(l \mid v_t) \propto \sum_{v_s \in V_s} p(l \mid v_s) w_a(v_s, v_t)$
  end
  $label(v_t) \rightarrow argmax_l p(l \mid v_t)$
end

To describe Algorithm 1 for POS projection authors define conditional probability distribution p(l | v).

Definition 2.0.3. *For all source vertices, the probability distributions are obtained by tagging the corresponding sentences in the corpus with POS taggers, assigning a probability of one to the best tag for each word.*

Conditional probability distribution for target vertices $p(l \mid v)$ over POS tags $l \in L$ is defined as:

$$p(l \mid v_t) \propto \sum_{v_s \in V_s} p(l \mid v_s) w_a(v_s, v_t). \tag{2.1}$$

It means that for each target token, i.e., each vertex v, the projection works by gathering evidence for each tag from all source tokens aligned to v, weighted by the alignment score.

The projected tag for a target vertex v t is then:

$$argmax_l p(l \mid v_t) \tag{2.2}$$

The best results in the paper are achieved when both, alignment weights and source tag probabilities are elements of $\{0, 1\}$. Therefore, further on in our work, POS projection is reduced to a simple voting scheme that assigns the most frequent POS tag among the aligned words to each target word.

Definition 2.0.4. *Dependency projection problem is defined as assigning weights to the edges of $G[V_t]$ by transferring the syntactic parse graphs $G[V_1], ..., G[V_N]$ from the source languages through the alignments A.*

While parsing the text written in source languages, probability scores $w_e \in \mathbb{R}$ were assigned to the dependency edges. Scores were standardized to make them comparable

across languages. Standardization centers the scores around zero with a standard deviation of one by subtracting the mean and dividing the result by the standard deviation. The normalization was applied to each sentence. Edge scores are projected from source edges to target edges via world alignments. Alignments vary in quality and according to that they are given scores $w_a \in [0, 1]$. While projecting edges, their scores are scaled with corresponding alignment probabilities.

Let (u_s, v_s) be source edge with score e_e. Edges $u_t, v_t \in V_t$ are such that $(u_s, u_t) \in E_A$ and $(v_s, v_t) \in E_A$ with alignment scores w_{a_1}, w_{a_2} respectively. Score for projected target edge (u_t, v_t) is computed as:

$$w_e(u_s, v_s)w_{a_1}(u_s, u_s)w_{a_2}(v_s, v_t). \tag{2.3}$$

One projected target edge can originate from multiple source edges. In that case, the projected edge is the one that gives the maximum score for the Equation 2.3.

With multiple sources, target edge scores are computed as a sum over the individual sources

$$\sum_{i=1}^{n} \max_{(u_s,v_s)\in V_i} w_e(u_s, v_s)w_{a_1}(u_s, u_s)w_{a_2}(v_s, v_t). \tag{2.4}$$

Result of projection is in a form of a general graph, not necessarily a tree. To get a dependency parse tree graph is decoded using directed maximum spanning tree (DMST) algorithm 2.

Algorithm 2 Project dependencies

Data: A projection graph $G = (V_s \cup V_t, E)$
Result: A dependency tree covering the target vertices V_t
if *project from trees* **then**
  for $i = 0$ *to* n **do**
    $\mathrm{G[V}_i\mathrm{]} \leftarrow \mathrm{DMST}(G[V_i])$
  end
end
for $(u_t, v_t) \in G[V_t]$ **do**
  $\mathrm{w}_e(u_t, v_t) \leftarrow -\infty$
  if $(\cdot, u_t)$ *or* $(\cdot, u_t) \notin E_A$ **then**
    continue
  end
  $w_e(u_t, v_t) \leftarrow \sum_{i=1}^{n} \max_{(u_s, v_s) \in V_i} w_e(u_S, v_s) w_{a_1}(u_s, u_t) w_{a_2}(v_s, v_t)$
end
$\mathrm{G[V}_t\mathrm{]} \leftarrow \mathit{normalize}(G[V_t])$
return $\mathrm{DMST(G[V}_t\mathrm{]}$

In order to efficiently implement projection of dependency trees from source languages to target language, each sentence is represented as a matrix. For a sentence of length n dependency parse tree can be transformed into matrix with dimension $(n + 1) * (n + 1)$. If element a_{ij} of a matrix is non-zero it means that the word with index j is the head of the word with index i. If index j is zero then the word with index i is the root. An example of dependency tree of a sentence is shown in Figure 2.1 and corresponding matrix representation can be seen in Figure 2.2.

ID	FORM	_	POS	_	_	HEAD	_	_
1	Salvation	_	PROPN	_	_	2	_	0:-3.41698 ... 8:-0.346113
2	means	_	VERB	_	_	0	_	0:-1.37355 ... 8:-0.780878
3	more	_	ADJ	_	_	2	_	0:-2.08862 ... 8:-0.504575
4	than	_	SCONJ	_	_	6	_	0:-4.36529 ... 8:0.656906
5	just	_	ADV	_	_	6	_	0:-4.42531 ... 8:1.05511
6	deliverance	_	NOUN	_	_	2	_	0:-3.66814 ... 8:0.784884
7	from	_	ADP	_	_	8	_	1:-0.953412 ... 8:3.47576
8	destruction	_	NOUN	_	_	6	_	0:-4.29034 ... 6:2.89611

Table 2.1: Annotated sentence

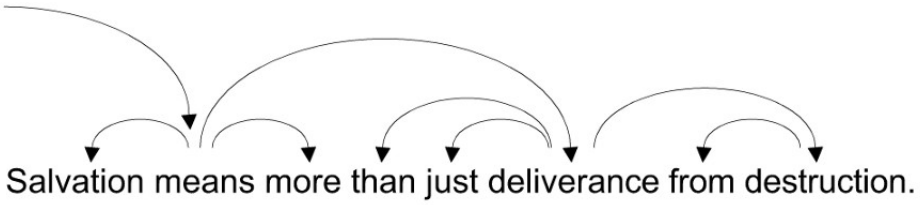


Figure 2.1: Dependency tree of annotated sentence from Table 2.1.

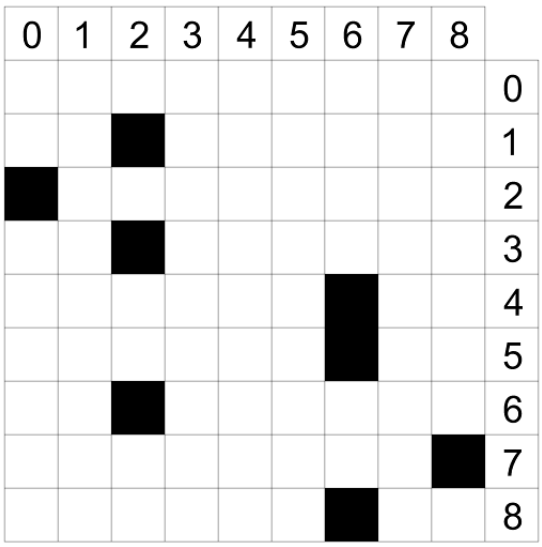


Figure 2.2: Matrix representation of dependency tree of annotated sentence from Table 2.1.

Word alignments are presented in matrix form, as well. If a source sentence has m words and a target sentence has n words, the matrix has a dimension (m+1)*(n+1) and a non-zero element a_{ij} is the score of alignment among word i in a source sentence and j word in a target sentence. All projection steps are shown in Figure 2.3.

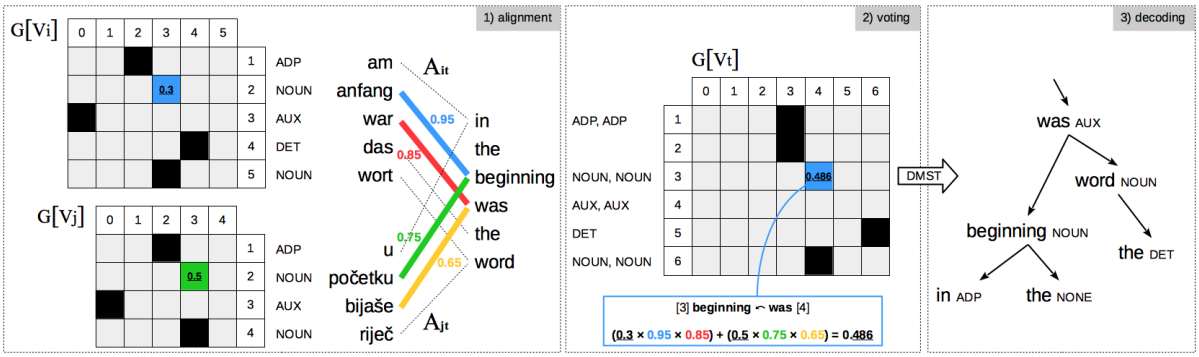


Figure 2.3: Projection of POS tags and dependency trees. The first empty row of described matrix representation is not shown in this image. The figure is borrowed from the paper Multilingual Projection for Parsing Truly Low-Resource Languages [8].

2.1 Coverage effect on projection accuracy

Alignment coverage of a target sentence, for every target-source pair, is defined as a percentage of target words that are aligned with words in the source sentence. To analyze the impact of coverage on the projection accuracy, one target language, Croatian (hr) and three source languages, English (en), Czech (cs) and German (de) were selected. Furthermore, 23 368 sentences that have translations in all four languages were selected. Since projection process depends on word alignments, if one target-source sentence pair has many unaligned words, we assume that projected tree and POS tags are going to be less accurate.

To test presented hypothesis we define several coverage cut-offs and two types of sentence selection. First selection criteria selects target sentence only if alignment coverage is above given cut-off for all three source sentences. The second selects target sentence if it has coverage above given cut-off for at least one source sentence. Figure 2.4 shows the effect of target alignment coverage on a projection accuracy. On x-axis, one can see a minimum expected alignment coverage and on y-axis calculated projection accuracy for selected sentences based on a cut-off. It can be seen that better coverage does provide more accurate projected trees. Also, significant improvement in projection accuracy is visible in the case when all source sentences satisfy defined coverage cut-off and not at least one. Figure 2.5 shows how many sentences were selected for a given cut-off.

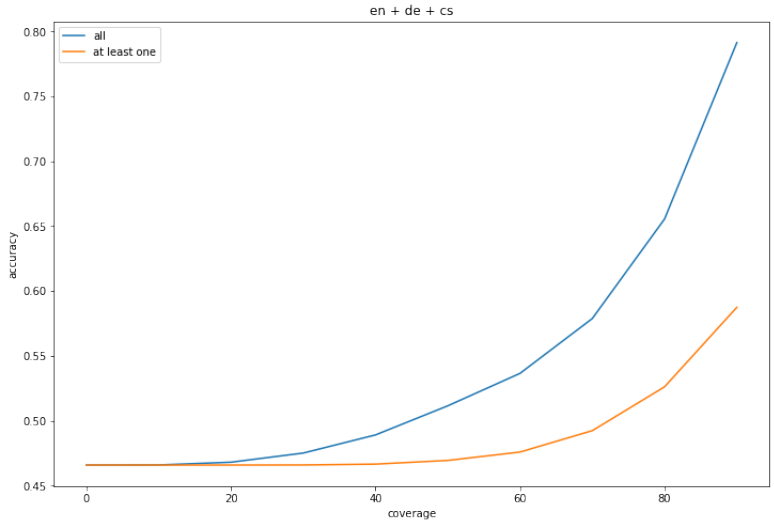


Figure 2.4: Effect of target alignment coverage on projection accuracy. On the x-axis is the minimum expected alignment coverage and on the y-axis is calculated projection accuracy for selected sentences based on a cut-off.

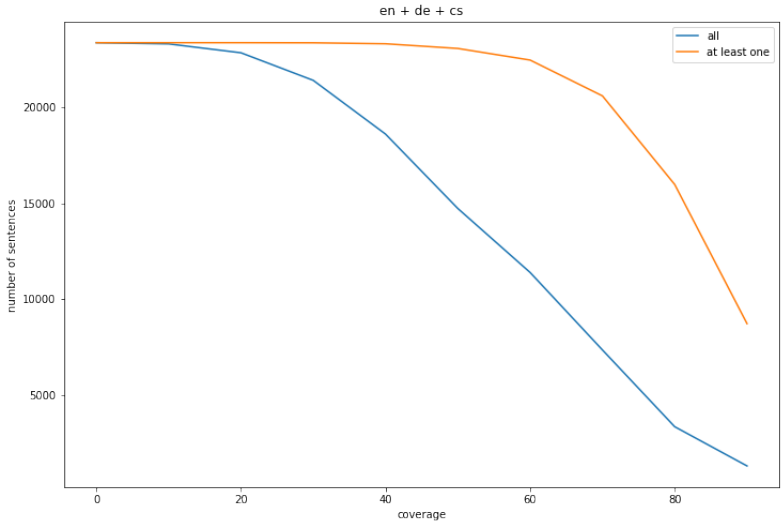


Figure 2.5: A number of selected sentences based on the alignment coverage cut-off shown on the x-axis.

One potential drawback of selecting only high covered sentences could be a change of sentence length distribution in a set of selected sentences. That could have a bad effect on parser training later on. Figure 2.6 shows distribution of selected sentences for each cut-off. It can be seen that the distribution is significantly changed only when the cut-off is 90%.

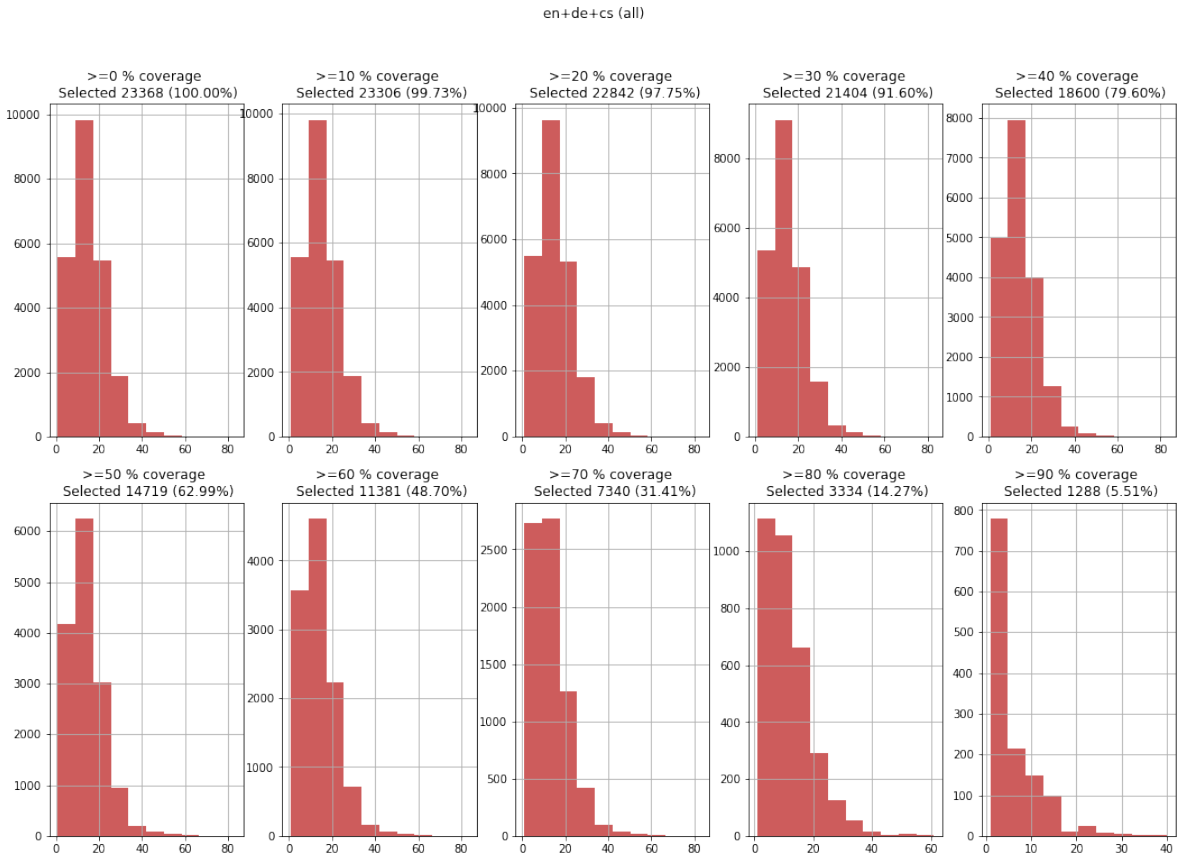


Figure 2.6: Distribution of length of target sentences that were selected for each cut-off.

2.2 The World Atlas of Language Structures (WALS)

The World Atlas of Language Structures (WALS) [9] is a large database of structural (phonological, grammatical, lexical) properties of languages gathered from descriptive materials (such as reference grammars) by a team of 55 authors. For well over a century, linguists have also produced atlases that show the geographical distribution of linguistic features in the dialects of a language. WALS is the first feature atlas on a world-wide

scale. WALS provides an especially significant contribution to the field of areal typology, which seeks to establish whether particular geographical distributions are the result of language contact among neighboring languages. For example, Figure 2.7 shows which order of object and verb is characteristic for languages across the world.

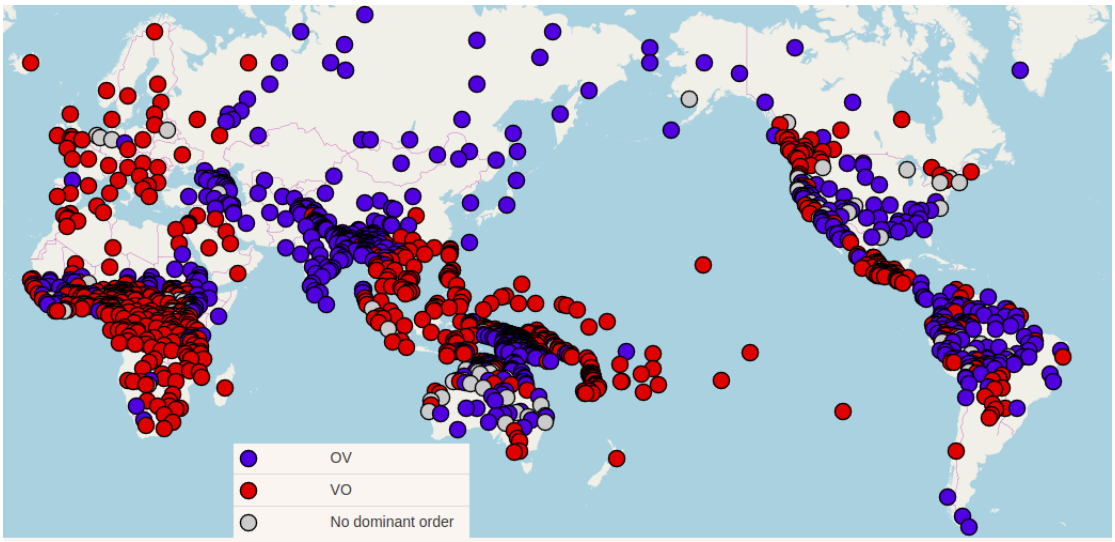


Figure 2.7: Order of object and verb across languages

From dataset used in this study, different language feature values were collected for languages in our data. Based on observed values of selected features, the similarity between each language pair was calculated. Therefore, each language has the corresponding mapping with similarity values that are going to be used in experiments described in Section 2.3. Example of mapping for English can be seen in a Table 2.6.

'fr': 0.37951807228915663	'fa': 0.55263157894736847	'pt': 0.36231884057971014
'pl': 0.37373737373737376	'sv': 0.25287356321839083	'hu': 0.4567901234567901
'ta': 0.67816091954022983	'hi': 0.4391891891891892	'ro': 0.34782608695652173
'no': 0.31578947368421051	'nl': 0.34065934065934067	'cs': 0.44117647058823528
'fi': 0.39393939393939392	'he': 0.42384105960264901	'bg': 0.36170212765957449
'hr': 0.44285714285714284	'es': 0.36969696969696969	'id': 0.50306748466257667
'it': 0.31868131868131866	'sl': 0.33962264150943394	'et': 0.41428571428571431
'ar': 0.61111111111111116	'el': 0.41509433962264153	'da': 0.28358208955223879
'de': 0.32704402515723269		

Table 2.2: Language similarity values for English

2.3 Experiments

While analyzing the data we have realized that, since texts are religious, sentences with two words are usually in the form shown in Table 2.3. They are not informative for tagging or parsing so they are filtered out before starting any experiment.

ID	FORM	_	POS	_	_	HEAD	_	_
1	3	_	NUM	_	_	0	_	0:-1.77388
2	.	_	PUNCT	_	_	1	_	0:-5.0905 1:2.60289
1	10	_	NUM	_	_	0	_	0:-1.77105
2	.	_	PUNCT	_	_	1	_	0:-5.14443 1:2.29314

For every target language we do the experiment following the next steps:

- for each of the source-target sentence pair, project POS tags and dependency edges using word alignments,
- for every token do majority vote on POS tags,
- do decoding on the summed edge scores using DMST,
- rank POS-tagged and dependency parsed target sentence applying two criteria described in the Subsection 2.3,
- select top n sentences, $n \in \{500, 1000, 2000, 5000, 10000, 20000\}$,
- create files from selected sentences for the tagger and parser training,

MarMoT tagger is used for tagging [13], a generic conditional random field (CRF) framework as well as a state-of-the-art morphological tagger. AnnaParser [1] is used for parsing instead of TurboPareser used in the paper [8].

Note 2.3.1. *In order to maintain consistency with the report of results in the paper [8], experiment is also done for each source language observed as a target language.*

Sentence ranking

Authors of the paper [8] randomly subsampled all training sets to a maximum of 20k sentences. In the multi-source systems, this means a uniform sample from all sources up to

20k sentences. As our baseline, we do the same. On the other hand, our new approach includes ranking all sentences based on some criteria. After that n highest ranking sentences are selected, where $n \in \{500, 1000, 2000, 5000, 10000, 20000\}$. As stated before we believe that sentence-pair alignment coverage has a big impact on projection quality. By removing target sentences that have low coverage we are more likely to have better data for the training. First criteria for ranking is calculating average coverage that one target sentence has with the aligned source translations. If there is no translation (aligned sentence) in some source language, we consider that the coverage is 0%. For the second criteria, we want to see if we can make a use of WALS similarity values described in Section 2.2. If some source language is more similar to the target language the translation in that source language can be more informative than the translation in the source language that is a lot less similar to the target. Therefore, before calculating the coverage average, each coverage is multiplied by similarity value.

For each language, calculated similarity values are converted into action probabilities [5]. We use the following form of softmax function

$$\begin{aligned} &l = (l_1, .., l_s), \text{vector of similarity values} \\ &P : \mathbb{R}^s \rightarrow [0, 1]^s \\ &P_j(l) = \frac{\exp\left(\frac{l_j}{temperature}\right)}{\sum_{i=1}^{s} \exp\left(\frac{l_i}{temperature}\right)}. \end{aligned} \tag{2.5}$$

Softmax function is a generalization of the logistic function that "squashes" the s-dimensional vector of real values into the s-dimensional vector of real values in the range $[0, 1]$ that add up to 1.

Temperature is the parameter that is going to be tuned. For high temperatures (temperature $\rightarrow\infty$), all actions have nearly the same probability of $\frac{1}{s}$ and the lower the temperature, the higher similarity values have more affect on the probability. For a low temperature (temperature $\rightarrow 0$), the probability of the action with the highest similarity value tends to 1.

2.4 Evaluation and results

Universal Dependencies v1.2 treebank [14] is used for evaluation. The Universal Dependencies (UD) is a project that is developing cross-linguistically consistent treebank annotation for many languages, with the goal of facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective. The general philosophy is to provide a universal inventory of categories and guidelines to facilitate consistent annotation of similar constructions across languages, while allowing

ID	FORM	LEMMA	UPOSTAG	XPOSTAG	FEATS	HEAD	DEPREL	DEPS	MISC
1	Call	call	VERB	VB	Mood=Imp ...	0	root	_	_
2	me	I	PRON	PRP	Case=Acc ...	1	dobj	_	_
3	if	if	SCONJ	IN	_	5	mark	_	_
4	there	there	PRON	EX	_	5	expl	_	_
5	are	be	VERB	VBP	Mood=Ind...	1	advcl	_	_
6	any	any	DET	DT	_	7	det	_	_
7	questions	question	NOUN	NNS	Number=Plur	5	nsubj	_	SpaceAfter=No
8	.	.	PUNCT	.	_	1	punct	_	_

Table 2.4: Test data format

language-specific extensions when necessary [6]. The test file format is shown in Table 2.4.

The goal is to evaluate the accuracy of parsers trained with selected target sentences. In order not to have parsers assume the existence of POS-annotated corpora, we train the tagger with selected target sentences. Trained tagger is used for tagging sentences from test data. After that, POS column in test data is adjusted according to the tagger output. Furthermore, since only information that we have after projection are projected tags and dependency trees, all columns other than ID, FORM, POS and HEAD are removed from test data. ID, FORM and POS columns are used for parser training and HEAD column is used to calculate parser accuracy. Parser accuracy is calculated as unlabeled attachment score (UAS):

$$acc_{parser} = UAS(parser) = \frac{\text{correctly predicted edges}}{\text{total number of edges}}. \tag{2.6}$$

Tagger accuracy is calculated as:

$$acc_{tagger} = \frac{\text{correctly predicted POS tags}}{\text{total number of words}} \tag{2.7}$$

Accuracies calculated for every language are used to calculate average parser and tagger accuracy. Results are shown in Figures 2.8 and 2.9. It can be seen that the parser trained with the 20 000 highest ranked sentences is more accurate than the parser trained with 20 000 randomly selected sentences. Furthermore, the highest accuracy is achieved when the parser is trained with only 5 000 sentences that have the highest average coverage. With four times fewer sentences, training is faster and therefore more efficient. The same conclusions can be applied to the target languages only. Their average tagger and parser accuracies are shown in Figures 2.10 and 2.11. Results for each language, the number of selected sentences and the type of selection are presented in Table 2.5.

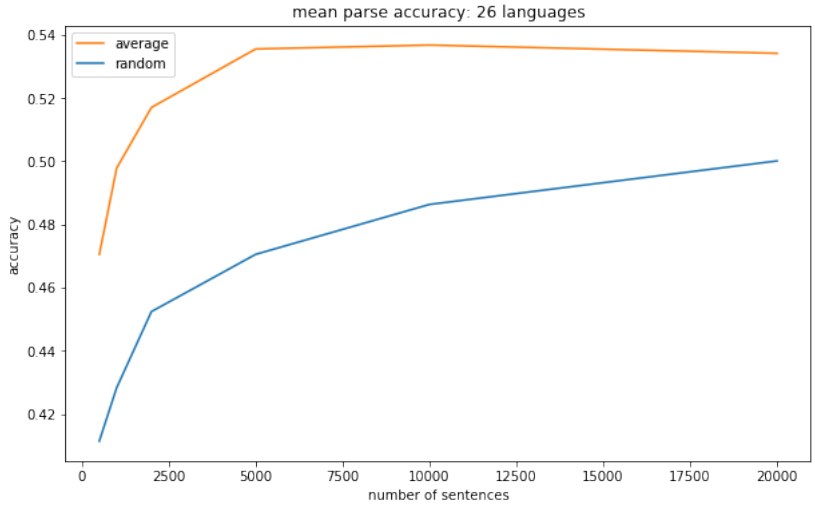


Figure 2.8: Average UAS parser accuracy for all 26 languages.

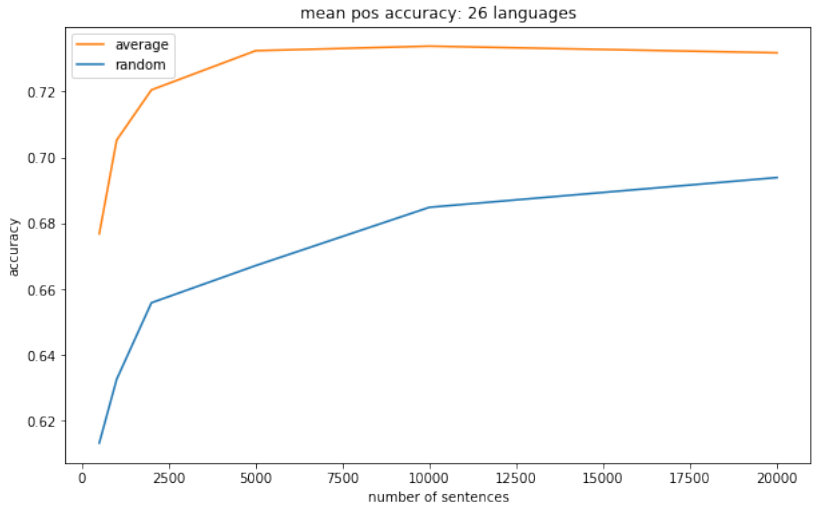


Figure 2.9: Average tagger accuracy for all 26 languages.

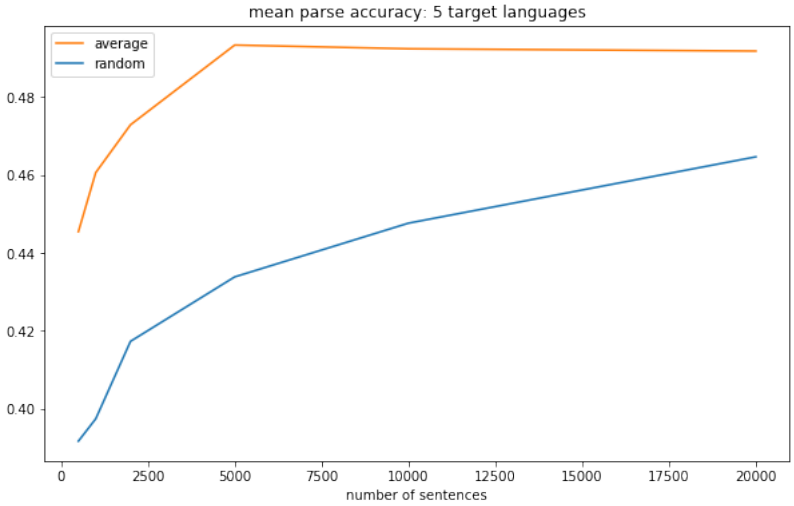


Figure 2.10: Average UAS parser accuracy for 5 target languages.

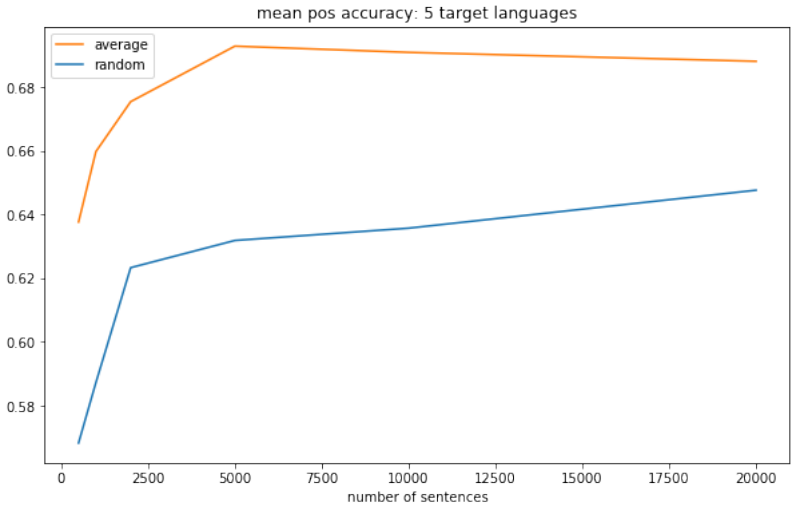


Figure 2.11: Average tagger accuracy for 5 target languages.

	500		1000		2000		5000		10 000		20 000	
	rnd	avg	rnd	avg	rnd	avg	rnd	avg	rnd	avg	rnd	avg
Sources												
Arabic	27.82	29.28	30.37	27.75	29.21	33.64	34.41	35.93	33.36	37.79	34.08	36.91
Bulgarian	33.62	57.97	37.23	61.72	36.77	63.36	41.18	63.72	42.06	60.54	47.23	56.01
Croatian	41.87	47.85	45.82	52.58	43.88	54.55	48.51	57.99	50.88	55.64	50.98	57.31
Czech	36.39	52.09	38.87	55.06	42.58	55.99	42.79	56.40	45.81	56.97	47.31	55.46
Danish	49.54	50.92	51.24	55.73	55.08	56.99	55.66	59.76	57.00	60.81	57.94	59.40
Dutch	49.81	53.75	50.80	54.66	53.11	57.65	56.26	59.10	56.97	60.63	57.73	59.75
English	53.68	51.92	53.05	54.26	54.05	55.19	56.11	57.82	56.78	58.56	58.58	59.09
Farsi	15.74	18.27	17.54	18.39	18.65	18.51	19.13	20.62	19.58	21.69	19.93	21.38
Finnish	29.63	35.21	30.45	40.47	34.99	43.45	35.42	46.18	38.17	48.14	39.73	48.53
French	41.75	49.56	47.39	55.30	49.37	57.75	51.87	60.25	53.79	58.83	54.70	59.42
German	32.47	51.43	33.35	52.96	37.24	53.08	39.39	53.34	41.74	52.50	41.93	50.93
Hebrew	24.64	32.99	23.35	40.29	37.21	47.52	36.07	49.68	42.26	49.67	44.70	49.60
Hindi	15.44	19.19	14.63	18.32	14.41	17.76	15.12	18.12	16.06	17.45	16.83	17.39
Indonesian	44.98	50.74	47.31	52.99	50.46	53.86	51.65	55.04	52.90	55.53	54.67	56.89
Italian	58.61	61.22	59.82	65.27	61.89	65.18	63.79	67.80	64.99	67.36	64.66	67.91
Norweigan	57.13	57.54	59.84	62.55	63.48	65.32	64.32	67.39	66.07	67.42	67.19	68.33
Polish	52.75	57.97	54.74	61.11	58.72	64.80	59.96	66.43	59.68	66.50	62.20	65.26
Portugese	58.16	57.81	60.01	59.66	60.93	61.85	62.81	62.89	62.91	63.80	64.77	64.17
Slovene	44.44	53.52	46.23	57.99	47.29	60.08	50.52	60.49	53.54	61.49	53.30	58.97
Spanish	54.18	58.82	57.50	61.39	59.90	63.36	60.39	65.47	64.28	65.95	65.40	66.36
Swedish	51.00	52.37	55.18	55.48	58.27	57.82	60.98	61.17	61.68	62.07	63.96	63.76
Targets												
Estonian	56.90	53.35	49.37	55.33	56.38	56.38	55.86	59.31	57.53	59.52	58.26	57.32
Greek	45.41	57.89	49.12	59.79	50.11	60.64	52.05	63.67	54.13	61.54	57.87	62.14
Hungarian	34.90	42.79	38.97	45.47	36.84	45.65	39.85	46.20	41.36	47.96	42.09	47.45
Romanian	42.14	50.91	46.13	50.62	47.43	52.86	51.12	55.32	52.35	55.61	54.45	57.28
Tamil	16.49	17.80	15.13	19.11	17.90	20.92	18.05	22.17	18.45	21.57	19.66	21.72
Averages												
All	41.13	47.04	42.82	49.78	45.24	51.70	47.05	53.55	48.63	53.67	50.01	53.41
Sources	41.60	47.64	43.56	50.66	46.07	52.75	47.92	54.55	49.55	54.73	50.85	54.42
Targets	39.17	44.55	39.74	46.06	41.73	47.29	43.39	49.34	44.76	49.24	46.47	49.18

Table 2.5: Parser UAS scores for each language, number of selected sentences and type of selection.

As mentioned in Subsection, 2.3, for the second type of sentence ranking, parameter temperature from Equation 2.5 needs to be tuned in. First, the set of possible temperature values is set. It contains five values: $3.0, 4.0, 5.0, 6.0$ and 7.0. Then, the experiment explained in 2.3 is repeated for every temperature value. Results are show in Figures 2.10 and 2.11. It is visible that second type of selection does not bring improvement in parser accuracy. But we can conclude that similarity languages values obtained from WALS do

not have a bad impact on overall parser accuracy. Results for temperature value 5.0, each language and number of selected sentences are presented in Table 2.6

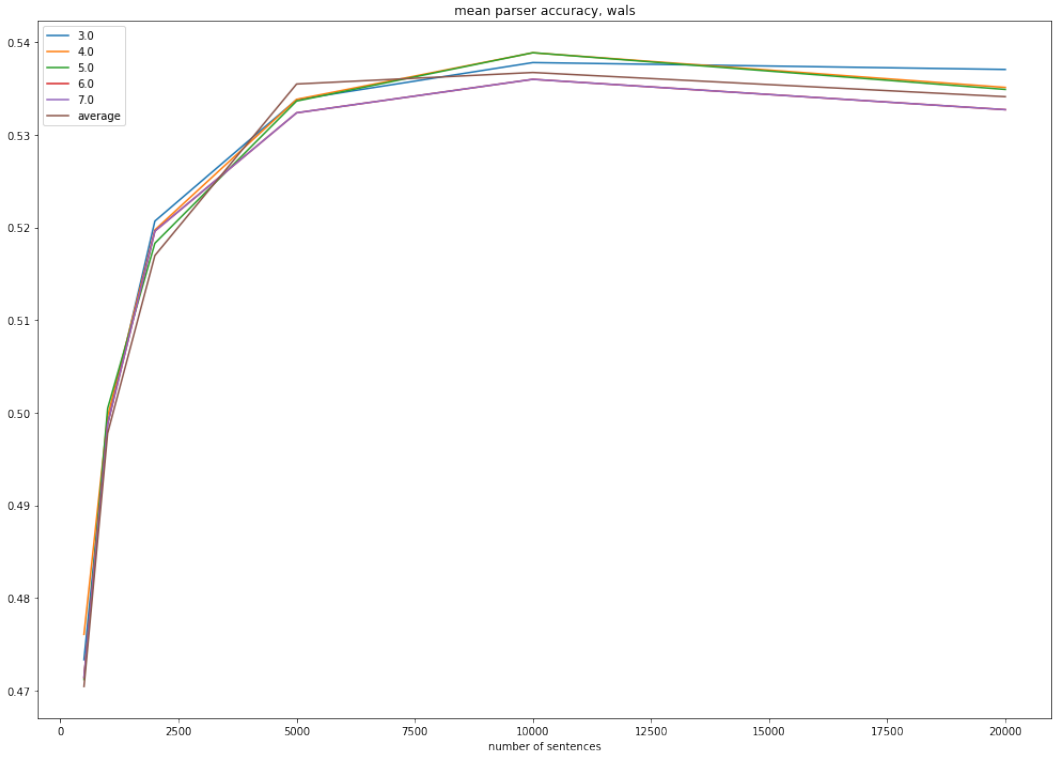


Figure 2.12: Average parser accuracy for different temperature values and different number of selected sentences.

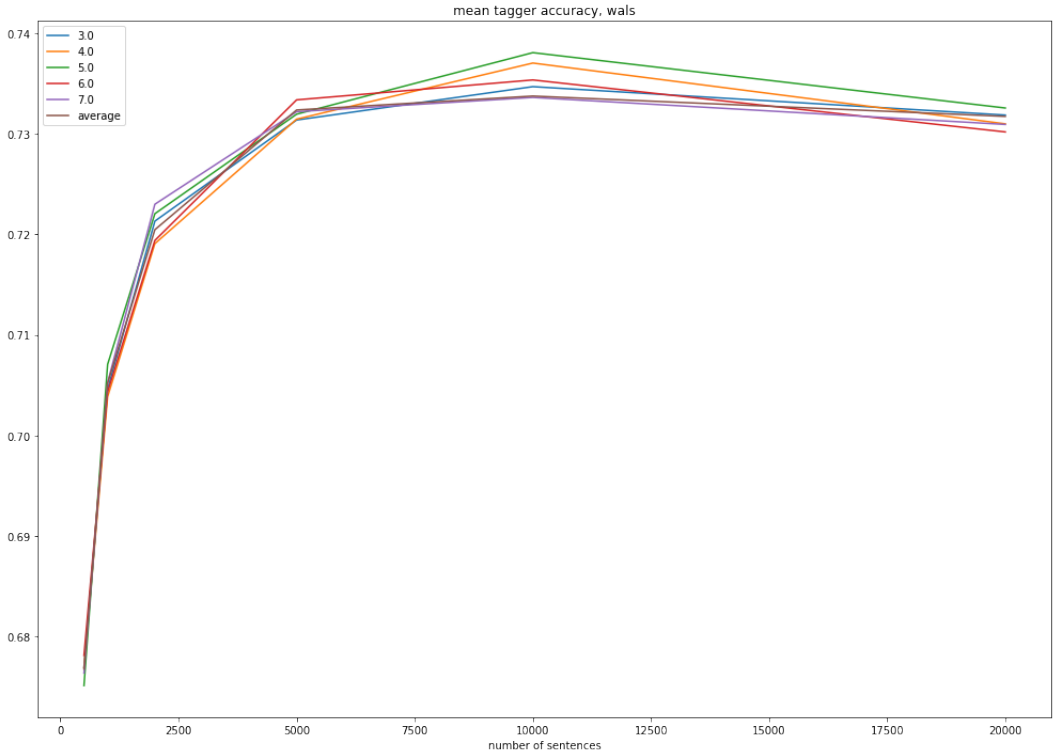


Figure 2.13: Average tagger accuracy for different temperature values and different number of selected sentences.

	500		1000		2000		5000		10 000		20 000	
	avg	wals	avg	wals	avg	wals	avg	wals	avg	wals	avg	wals
Sources												
Arabic	29.28	29.23	27.75	28.27	33.64	31.26	35.93	36.41	37.79	37.83	36.91	36.97
Bulgarian	57.97	57.12	61.72	61.21	63.36	63.22	63.72	63.80	60.54	61.06	56.01	57.07
Croatian	47.85	49.62	52.58	53.26	54.55	55.44	57.99	58.13	55.64	56.34	57.31	58.74
Czech	52.09	51.76	55.06	54.99	55.99	56.47	56.40	56.35	56.97	56.66	55.46	55.15
Danish	50.92	50.20	55.73	54.67	56.99	57.55	59.76	59.62	60.81	61.27	59.40	60.84
Dutch	53.75	53.05	54.66	54.36	57.65	57.19	59.10	59.09	60.63	59.96	59.75	60.13
English	51.92	51.62	54.26	54.17	55.19	55.34	57.82	57.52	58.56	58.35	59.09	59.20
Farsi	18.27	17.28	18.39	18.48	18.51	19.60	20.62	20.55	21.69	21.33	21.38	21.42
Finnish	35.21	36.93	40.47	40.61	43.45	43.48	46.18	47.20	48.14	48.67	48.53	48.18
French	49.56	50.27	55.30	56.98	57.75	57.44	60.25	59.01	58.83	58.63	59.42	59.60
German	51.43	52.75	52.96	53.07	53.08	53.07	53.34	53.04	52.50	53.09	50.93	49.99
Hebrew	32.99	33.94	40.29	40.73	47.52	48.46	49.68	50.66	49.67	51.66	49.60	48.54
Hindi	19.19	19.73	18.32	18.92	17.76	18.31	18.12	18.19	17.45	17.43	17.39	16.71
Indonesian	50.74	51.44	52.99	52.95	53.86	54.38	55.04	55.31	55.53	55.40	56.89	56.52
Italian	61.22	61.30	65.27	65.82	65.18	66.33	67.80	67.22	67.36	67.71	67.91	67.46
Norweigan	57.54	58.45	62.55	63.00	65.32	65.67	67.39	67.30	67.42	67.79	68.33	68.18
Polish	57.97	58.68	61.11	61.88	64.80	63.12	66.43	65.72	66.50	66.56	65.26	66.25
Portugese	57.81	57.85	59.66	59.95	61.85	61.62	62.89	63.32	63.80	64.29	64.17	64.60
Slovene	53.52	54.00	57.99	58.00	60.08	60.98	60.49	60.82	61.49	59.76	58.97	60.39
Spanish	58.82	57.34	61.39	60.42	63.36	64.54	65.47	65.11	65.95	65.71	66.36	66.25
Swedish	52.37	51.99	55.48	56.73	57.82	58.03	61.17	61.00	62.07	62.52	63.76	64.21
Targets												
Estonian	53.35	53.24	55.33	55.75	56.38	55.75	59.31	57.01	59.52	58.16	57.32	57.95
Greek	57.89	57.20	59.79	60.16	60.64	60.30	63.67	62.88	61.54	63.25	62.14	61.75
Hungarian	42.79	41.65	45.47	45.03	45.65	45.94	46.20	46.35	47.96	48.99	47.45	46.28
Romanian	50.91	48.95	50.62	51.92	52.86	51.77	55.32	53.73	55.61	56.84	57.28	56.55
Tamil	17.80	19.51	19.11	19.91	20.92	22.32	22.17	22.17	21.57	21.82	21.72	21.82
Averages												
All	47.04	47.12	49.78	50.05	51.70	51.83	53.55	53.37	53.67	53.89	53.41	53.49
Sources	47.64	47.84	50.66	50.88	52.75	52.93	54.55	54.54	54.73	54.86	54.42	54.59
Targets	44.55	44.11	46.06	46.55	47.29	47.22	49.34	48.43	49.24	49.81	49.18	48.87

Table 2.6: Parser UAS scores for temperature value 5.0, each language and number of selected sentences.

Chapter 3

Deep machine learning

In this chapter, we briefly describe basics of deep neural networks to refer to some concepts and terms used in Section 3.5. Deep learning is a branch of machine learning. It is a re-branded name for neural networks a family of learning techniques that was historically inspired by the way computation works in the brain [11] .

A neural network consists of computational units called neurons. Each neuron has scalar inputs and outputs. Each input is multiplied by an associated weight and all results are summed. A non-linear function, called activation function, is applied to the sum and passed to its output. One neuron is shown in Figure 3.1. Two kinds of networks, feed-forward networks, and recurrent/recursive networks, are going to be described in Sections 3.1 and 3.2.

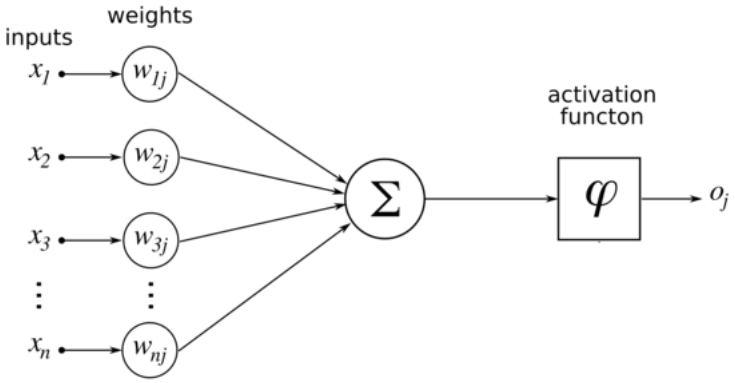


Figure 3.1: Neuron graphical representation.

3.1 Feed forward neural networks

The output of a neuron may feed into the inputs of one or more neurons. Feed forward neural network is being created by connecting neurons into layers. The most common type is called fully connected neural network and its structure can be seen in Figure 3.2.

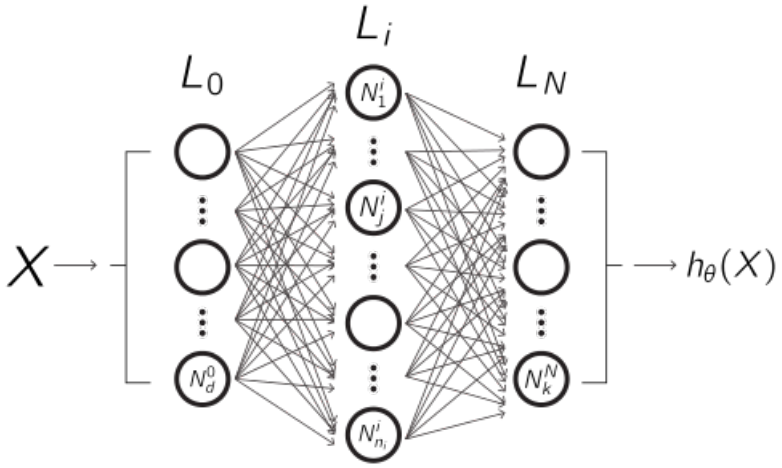


Figure 3.2: Fully connected neural network with $N + 3$ layers, d inputs and k outputs. Layers L_i are called hidden layers, x is input layer and $h(x)$ is output layer. Figure is borrowed from [3].

In mathematical terms each layer can be presented as a function

$$L_i(x) = W_i x + b_i, \tag{3.1}$$

that is followed by non-linear activation function. Neural network inputs are training examples $(x^{(1)}, .., x^{(n)})$ that have corresponding labels $(y^{(1)}, .., y^{(n)})$. The goal of a network is to tune in weight parameters to output $y^{(i)}$ for $x^{(i)}$ as an input, where $i \in \{1, .., n\}$. By tuning parameters, network is defining function $h = L_N \circ ... \circ L_0$ that for given input x outputs desired value.

Training process relays on applying a loss function to the neural network predicted output and the desired value. Loss function assigns a numerical score that represents loss suffered when predicting incorrectly. Loss can be calculated for each input or for a defined number of inputs called batch size.

It was shown by [Cybenko, 1989, Hornik et al., 1989] that network with only one layer is a universal approximator. It can approximate with any desired non-zero amount of error a family functions that include all continuous functions on a closed and bounded subset

of $\mathbb{R}^n$, and any function mapping from any finite dimensional discrete space to another. However, it does not say how easy or hard it is to set the parameters based on training data and a specific learning algorithm, it also does not guarantee that a training algorithm will find the correct function generating our training data and it does not state how large the hidden layer should be. Therefore, there is a benefit in trying out some more complex architectures such as recursive neural networks. [11]

3.2 Recurrent neural networks (RNN)

Textual language data comes in sequences such as letters, words, and sentences. Recurrent neural networks are specialized models for sequential data. Recurrent neural networks allow representing arbitrarily sized sequential inputs in a fixed-size vector. They take as input a sequence of items and produce a fixed size vector that summarizes that sequence [11]. Formally it is defined as a recursive function given in Equation 3.2.

$$A^{(t)} = Wh^{(t-1)} + b + Ux^{(t)} \tag{3.2}$$

The base of the recursion is an initial state vector, h_0, which is also an input to the RNN. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor [15]. In Figure 3.3 unrolled network, for every timestamp t, can be seen.

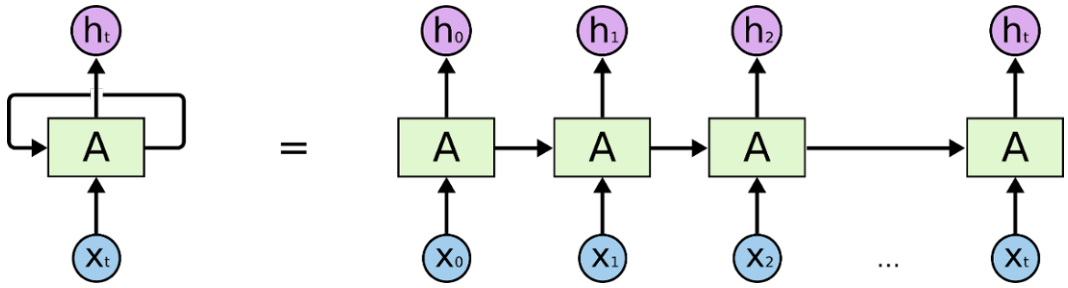


Figure 3.3: An unrolled recurrent neural network. The figure is borrowed from [15].

In theory, recurrent neural networks, because of its recursive definition, are able to connect previous information to the present task. In practice, it turns out that if the gap between the relevant information and the point where it is needed becomes very large, recurrent networks don't seem to be able to learn them [15]. A special type of recurrent network, called long short-term memory network, is able to cope with that problem.

3.3 Long short-term memory (LSTM)

The Long Short-Term Memory (LSTM) neural network is currently the most successful type of RNN architecture. It explicitly splits the state vector h_i into two half, where one half is treated as "memory cells" and the other is working memory. It was the first architecture to introduce the gating mechanism. At each input state, a gate is used to decide how much of the new input should be written to the memory cell, and how much of the current content of the memory cell should be forgotten. Mathematically it is defined in Equation 3.3.

$$
\begin{aligned}
h_t &= o \cdot tahn(c_t) \\
c_t &= f \cdot c_{t-1} + i \cdot z \\
i &= \sigma(W^{xi} x_t + W^{hi} h_{t-1}) \\
f &= \sigma(W^{xf} x_t + W^{hf} h_{t-1}) \\
o &= \sigma(W^{xo} x_t + W^{ho} h_{t-1}) \\
z &= tanh(W^{xz} x_t + W^{hz} h_{t-1})
\end{aligned}
\tag{3.3}
$$

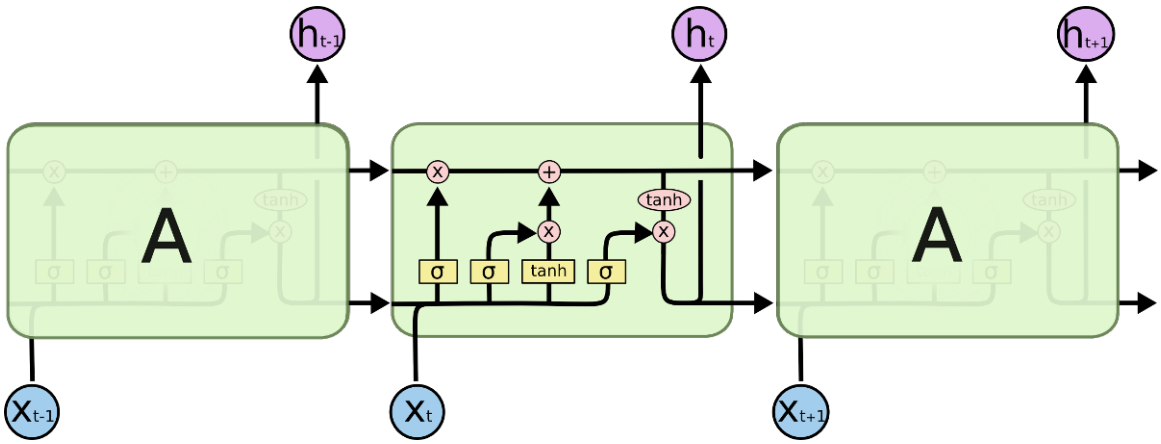
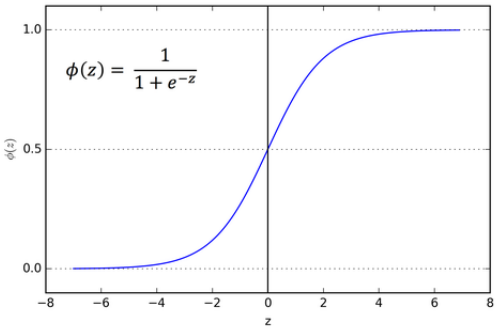
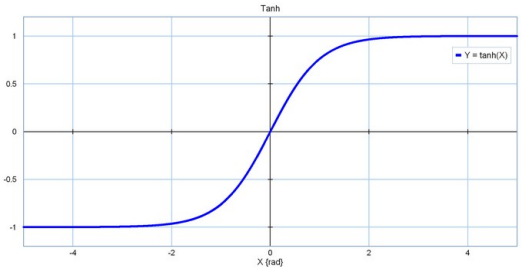


Figure 3.4

The state at time t is composed of two vectors, memory cell state c_t and hidden state h_t. There are three gates, i (input), f (forget) and o (output). All gate values are computed based on linear combinations of the current input x_t and the previous state h_{t-1}, upon which is applied sigmoid activation function. LSTM network has the ability to remove or add information to the memory cell state c_t using input and forget gates. Sigmoid activation function plays a key role in that.



(a) Sigmoid function graph.



(b) Hyperbolic tangent function graph.

Figure 3.5: Graphs of activation functions in LSTM.

Sigmoid function is defined In Equation 3.4 and its graph is shown in Figure 3.5a. It can be seen that it outputs values in the range from 0 to 1. After every gate, there is a multiplication of a vector with the result of the sigmoid function. If the result is close to zero it means that information, which vector holds, is going to be dropped after multiplication and if the result is close to one, all information is going to be kept.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

The memory c_t is computed as a sum of previous memory state (c_{t-1}) multiplied with forget gate result, and update candidate z multiplied by the input gate result. Update candidate is computed as a linear combination of x_t and h_{t-1}, passed through a hyperbolic tangent activation function. Hyperbolic tangent function (tanh) is defined in Equation 3.5 and its graph is shown in Figure 3.5b. It is used to place update candidate values between -1 and 1.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x - e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.5}$$

Finally, the new hidden state h_t, which is also an output of the neural network, is computed based on the previously computed new memory state c_t passed through a tanh activation multiplied by the result of the output gate.

3.4 Data representation

Inputs of neural networks used in experiments, described in Section 3.5, are sentences observed as sequences of words. Sentences have a different number of words whereas

network inputs need to have the same dimension. Therefore, in this section is described how we represent sentences to satisfy that constraint.

Before any data preparation, maximum sentence length value (N) is defined. It is going to enable same dimensional representation of each sentence. To adjust data we use matrix representation of a sentence, similar to one described in Chapter 2 and showed in Figure 2.2. The only difference is that we omit the first row of zeros. Therefore, matrix representation of a sentence with n words has dimension $n * (n + 1)$. For each sentence, its length n is compared with the defined length N and the matrix is adjusted as described below:

- if $n < N$: add $(N - n)$ rows and $(N - n)$ columns with zeros
- if $n == N$: no changes,
- if $n > N$: remove last $(n - N)$ rows and $(n - N)$ last columns.

After described adjusting, every sentence is represented as a two-dimensional matrix whose dimension is $N * (N + 1)$. All sentences used to train network are concatenated to form a three-dimensional matrix whose dimension is $(d, N, N + 1)$ where d is a number of sentences used for training. The third dimension holds information about heads of the words. Example can be seen in Figure 3.6.

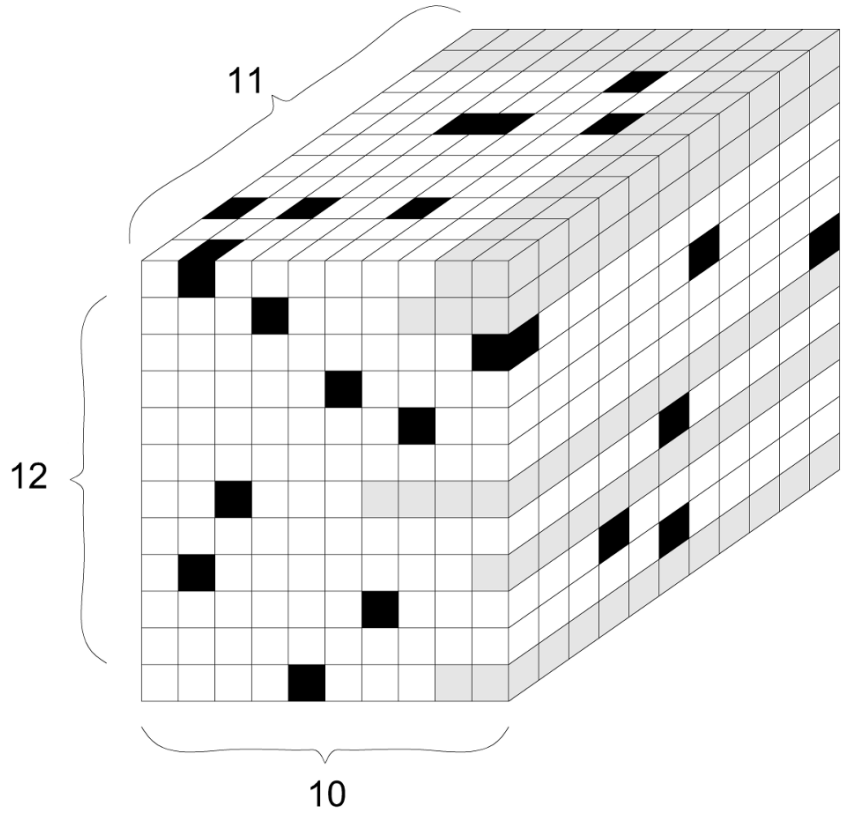


Figure 3.6: Example of representation of data containing 12 sentences with maximum length set at 10. Gray area represents added rows and columns. Matrix at the top is representation of the sentence from Table 2.1. Since its length is 8, two rows and columns are added.

3.5 Experiments

The goal of experiments is to find learning signal in data by training different neural network models. Due to server and time limitation, ten language pairs were selected for experiments. It resulted in 564 966 sentences that are going to be inputs to the network and the same number of translations that are going to be desired outputs. Sentences are processed to be in a form described in a previous section.

Projection from one language to another is observed as:

- sequence-to-sequence learning problem, because of transformation from one sentence into its translation,
- sequence labelling problem, because for each word we want to determine its head of a parse tree. Therefore word is being classified into $(N + 1)$ "classes".

Model is composed of one LSTM layer with sigmoid activation function and one dense (fully connected) layer with softmax activation function. Dense layer is wrapped with the TimeDistributed wrapper. This wrapper applies a layer to every temporal slice of an input. Our input is sequence of N vectors with dimension $(N+1)$. Therefore, dense layer is applied to each of the N timesteps, independently. In our case, it determines the probability that some word is the head in a dependency parse tree. Categorical cross-entropy loss function was used. For predicted output $\tilde{y}$ and true label y it is defined as

$$L_{cross-entropy}(\tilde{y}, y) = -\sum_{i} y_{[i]} log(\tilde{y}_{[i]}). \tag{3.6}$$

It measures the dissimilarity between the true label distribution y and the predicted label distribution.

Several models were trained with a different combination of network parameters to with the aim to find one whose loss function converges to zero as a number of epochs increases. We started out with 512 LSTM number of units (dimensionality of the output) and with 15 epochs. Maximum sentence length was set to 100. Training trough one epoch lasted for approximately two hours. After the tenth epoch loss stars to converge to the 0.39. The number of units was increased to 1024 and 2048, also several batch number values were tried (32, 64, 128) but the behaviour of the loss function stayed the same. Model loss convergence can be seen in Figure 3.7.

Since input data has high dimensionality ($(100 * 101)$ when the maximum length is 100) deeper and wider network architecture should be considered. Such set up would significantly increase the time of training on our server.

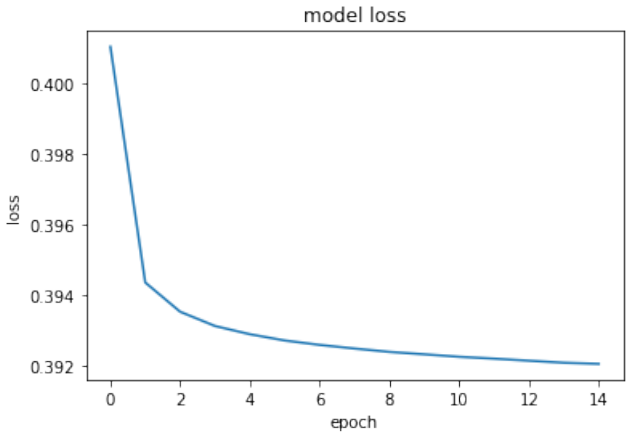


Figure 3.7: Categorical cross-entropy loss after each epoch.

3.6 Future work

Instead of the categorical cross-entropy loss, custom loss function could be used. It would be based on the unlabeled attachment score defined in Equation 2.7. Instead of correctly predicted edges, the function would calculate the number of incorrectly predicted edges. In that case minimization of the loss function would improve projection accuracy. More hidden layers should be added to make network deeper and their output dimension should be higher to make the network wider. That kind of network should be able to better capture learning signal from high dimensional input data.

Furthermore, if the information about a part of speech tags is known when training a parser, parser accuracy increases. Therefore, adding information about part of speech tags in the data representation should also improve the accuracy of network projections of dependency trees.

Bibliography

[1] *AnnaParser*, http://backingdata.org/dri/library/mavenRepo/mate/anna/parser/3.61/, [Online; accessed 25-January-2018].

[2] *ARK Syntactic Semantic Parsing Demo, howpublished = "http://demo.ark.cs.cmu.edu/parse", note = "[online; accessed 25-january-2018]"*.

[3] *Neural nets, howpublished = "http://frnsys.com/ai_notes/machine_learning/neural_nets.html", note = "[online; accessed 25-january-2018]"*.

[4] *Penn Part of Speech Tags, howpublished = "https://cs.nyu.edu/grishman/jet/guide/PennPOS.html", note = "[online; accessed 25-january-2018]"*.

[5] *Softmax function*, https://en.wikipedia.org/wiki/Softmax_function, [Online; accessed 25-January-2018].

[6] *Universal Dependencies*, http://universaldependencies.org/docsv1/introduction.html, [Online; accessed 25-January-2018].

[7] *Wikipedia - Part-of-speech tagging, howpublished = "https://en.wikipedia.org/wiki/Part-of-speech_tagging", note = "[online; accessed 25-january-2018]"*.

[8] Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martınez Alonso, Natalie Schluter, and Anders Søgaard, *Multilingual projection for parsing truly low-resource languages*, (2016).

[9] Martin Dryer, Matthew S. Haspelmath, *The World Atlas of Language Structures Online.* , http://http://wals.info//, 2013, [Online; accessed 25-January-2018].

[10] D. E. Dutkay, D. Han, Q. Sun, and E. Weber, *Hearing the hausdorff dimension*, (2009), http://arxiv.org/abs/0910.5433.

[11] Yoav Goldberg, *Neural network methods in natural language processing*, Morgan publishers, 2016.

[12] Stanford NLP Group, *Software, Stanford Parser, Neural Network Dependency Parser*, `https://nlp.stanford.edu/software/nndep.shtml`, [Online; accessed 25-January-2018].

[13] Thomas Mueller, Helmut Schmid, and Hinrich Schütze, *Efficient higher-order CRFs for morphological tagging*, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (Seattle, Washington, USA), Association for Computational Linguistics, October 2013, pp. 322–332, `http://www.aclweb.org/anthology/D13-1032`.

[14] Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel

Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu, *Universal dependencies 2.1*, 2017, `http://hdl.handle.net/11234/1-2515`, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[15] Christopher Olah, *Understanding LSTM Networks*, `http://colah.github.io/posts/2015-08-Understanding-LSTMs`, [Online; accessed 25-January-2018].

Summary

The purpose of this thesis was to explore cross-lingual transfer learning to dependency parsing, with a goal of enabling syntactic analysis for low-resource languages. The best approaches involve annotation projection: the transfer of dependency structures via parallel texts, from resource-rich to low-resource languages.

In the first chapter, basic concepts of part of speech tagging and dependency parsing are described as well as the way of annotating texts. The first approach to solving an annotation projection problem is described in the second chapter. It is based on the algorithm presented in the paper Multilingual Projection for Parsing Truly Low-Resource Languages [10]. We propose the way of adjusting the existing algorithm which leads to the improvement of results. In the third chapter, the idea how to use neural networks for annotation projection is presented, and also some of the ideas how the work done in this thesis can be extended in the future.

Sažetak

U ovom radu istražuje se prijenosno učenje kroz više jezika s ciljem omogućavanja sintaktičke analize jezika koji nemaju dovoljno označenih podataka za učenje. Najbolji pristupi rješavanju problema uključuju projekciju oznaka sintaktičkih ovisnosti preko paralelnih tekstova, iz jezika koji imaju mnogo označenih podataka za učenje u jezike koji imaju nedovoljno.

U prvom poglavlju opisuju se osnovni pojmovi morfološkog označivanja rečenica i parsanja njihovih ovisnosnih stabala kao i nacin označivanja sintaktičkih ovisnoti. Prvi pristup rješavanja problema projekcije oznaka sintaktičkih ovisnosti je opisan u drugom poglavlju. Zasnovan je na algoritmu predstavljenom u znanstvenom radu Multilingual Projection for Parsing Truly Low-Resource Languages [10]. Predložene su prilagodbe algoritma koje vode poboljšanju rezultata. U trećem poglavlju predstavljena je ideja o upotrebi neuronskih mreža za projekcije oznaka sintaktičkih ovisnosti te nekoliko ideja kojima rad u budućnosti može biti unaprijeđen.

Curriculum vitae

I was born in 1994. in Split, where I have finished primary school and mathematical high school. In 2012. I started an undergraduate programme of Mathematics on Faculty of science at the University of Zagreb. After getting a Bachelor degree In 2015., I have started the Master's programme of Computer science and mathematics which finishes with this thesis.

Životopis

Rođena sam 19. ožujka 1994. godine u Splitu, gdje sam završila osnovnu školu i Matematičku gimnaziju. 2012. godine upisujem preddiplomski sveučilišni studij Matematika na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu, koji završavam 2015. godine stekavši naziv sveučilišna prvostupnica matematike. Iste godine upisujem diplomski studij Računarstva i matematike koji završavam ovim radom.