Towards a Weighted Induction Method of Dependency Annotation

Alina Wróblewska and Adam Przepiórkowski

Institute of Computer Science, Polish Academy of Sciences,
ul. Jana Kazimierza 5, 01-248 Warsaw, Poland
{alina,adamp}@ipipan.waw.pl
http://zil.ipipan.waw.pl/

Abstract. This paper presents a method of annotating sentences with dependency trees which is set within the mainstream of the study on dependency projection. The approach builds on the idea of weighted projection. However, we involve a weighting factor not only in the process of projecting dependency relations (weighted projection) but also in the process of acquiring dependency trees from projected sets of dependency relations (weighted induction). Using a parallel corpus, its source side is automatically annotated with a syntactic parser and resulting dependencies are transferred to equivalent target sentences via an extended set of word alignment links. Projected relations are initially weighted according to the certainty of word alignment links used in projection. Since word alignments may be noisy and we should not entirely rely on them, initial weights are thus recalculated using a version of the EM algorithm. Then, maximum spanning trees fulfilling properties of well-formed dependency structures are selected from EM-scored directed graphs. An extrinsic evaluation shows that parsers trained on induced trees perform comparably to parsers trained on a manually developed treebank.

Keywords: Dependency annotation, cross-lingual projection, weighted induction.

1 Introduction

Supervised methods are very well-established in data-driven dependency parsing and they give the best results so far. However, the manual annotation of training data required by supervised frameworks is a very time-consuming and expensive process. For this reason, intensive research has been conducted on unsupervised grammar induction. However, performance of unsupervised dependency parsers is still significantly below performance of supervised systems. Moreover, performance of unsupervised parsers is also substantially below performance of systems based on cross-lingual projection methods [17].

The cross-lingual projection method has been successfully applied to various levels of linguistic analysis and corresponding NLP tasks. An important area of applying annotation projection is dependency tree projection and parser induction. Experiments with dependency projection were pioneered by [10], who assume that dependencies in one language directly map to dependencies in another language. In order to acquire well-formed dependency trees Hwa and her colleagues apply additional smoothing techniques and aggressive filtering methods. Other research was conducted on projecting only reliable relations and training parsers on partial dependency structures [12,24]. There are also some constraint-driven learning approaches [8,22] which apply projected information to constrain estimation of dependency parsing models. Other related approaches consists in transferring delexicalised parsers between languages [31,17,23] or in multi-source cross-lingual transferring of parsers [17,23,18,26].

The cross-lingual dependency projection may be an alternative method of annotating sentences with dependency trees in less researched languages. The method builds on the assumption that a dependency tree encoding the predicate-argument structure of a sentence largely carries over to its translation since an integrated valency component determines both the number and the kind of complement slots of verbs, nouns, adjectives, etc., and these complement slots are relatively invariant across languages. Furthermore, dependency projection does not take into account the order of words, so it is thus perfectly suited for languages with different word orders.

The main idea behind dependency projection is to automatically parse source sentences and to project acquired dependency trees to equivalent target sentences. Since relations encoded in dependency trees connect tokens, projection of these relations may be sufficiently guided by word alignment which links corresponding tokens in parallel sentences. In the ideal case, projected dependencies constitute valid dependency structures of target sentences.

This paper describes a novel method of annotating Polish sentences with dependency trees which is set within the mainstream of the study on dependency projection. The approach builds on the idea of weighted projection [29]. However, we involve a weighting factor not only in the process of projecting dependency relations (*weighted projection*, Section 2.1) but also in the process of acquiring dependency structures from projected sets of dependency relations (*weighted induction*, Section 2.2). Using a parallel corpus, its English side is automatically annotated with a syntactic parser and resulting dependency relations are transferred to equivalent Polish sentences via an extended set of word alignment links. Projected relations are initially weighted according to the certainty of word alignment links used in projection. Since word alignments may be noisy and we should not entirely rely on them, initial weights are thus recalculated with the EM selection algorithm [7]. Then, maximum spanning trees fulfilling properties of well-formed dependency structures are selected from EM-scored directed graphs. An extrinsic evaluation shows that parsers trained on induced trees perform comparably to parsers trained on a manually developed treebank (Section 3). The novelty of the method proposed here consists in involving a weighting factor in the process of inducing dependency trees.

2 Weighted Induction Method

The weighted induction procedure consists of two successive processes – *projection* of dependency relations followed by *induction* of dependency trees from projected directed graphs (henceforth digraphs). Induced trees are treated as correctly built unlabelled dependency structures.

2.1 Weighted Projection

This section describes weighted projection which is the first step in the entire process of acquiring valid Polish dependency structures. According to the main idea behind weighted projection, arcs making up an English dependency tree are projected via an extended set of word alignment links between English and Polish tokens (*bipartite alignment graph*). Since we aim to project English relations which are restricted to sentence boundaries, only word alignment links within a pair of aligned parallel sentences are considered in projection. Projected arcs constitute initially weighted digraphs.

Bipartite Alignment Graph. Instead of projection only via automatic word alignment links, English dependencies are transferred via a set of links gathered from different automatic word alignments and extended with some additional links. This set of links constitutes a *complete bipartite alignment graph* $BG := (V_{\mathbf{en}} \cup V_{\mathbf{pl}}, E)$, for $E = V_{\mathbf{en}} \times V_{\mathbf{pl}}$. Vertices in BG are decomposed into two disjoint sets $V_{\mathbf{en}}$ and $V_{\mathbf{pl}}$ corresponding to English tokens with a ROOT node and Polish tokens with a ROOT node respectively. Every pair of vertices from $V_{\mathbf{en}}$ and $V_{\mathbf{pl}}$ is adjacent. Bipartite edges are weighted with the function $w : E \rightarrow \{0, 1, 2, 3\}$. The weight w indicates the certainty of an edge and either corresponds to the number of occurrences of this edge in automatic alignment sets or is equal to 0 if it is not present in any alignment set. There are three word alignment sets: two unidirectional alignments and a set of bidirectional alignment links symmetrised with the *grow-diag-final-and* heuristic.[1] The edge between ROOT nodes, which is not present in any set of alignment links, is scored with 1.[2] Weighted bipartite alignment graphs built for each sentence pair are used to project English dependency arcs to Polish sentences.

Projection of Dependency Arcs. English dependency arcs are projected to corresponding Polish tokens via bipartite edges according the following procedure. The projection module takes as input a weighted bipartite alignment graph

[1] To improve the word alignment quality and overcome limitation of alignment scenarios, a method of unidirectional alignments symmetrisation was proposed by [19]. Next to union and intersection, there exist some more sophisticated symmetrisation concepts, e.g., the *grow-diag-final-and* symmetrisation method described in [14].

[2] Scoring the edge between a Polish ROOT and an English ROOT with 1 will guarantee the connectedness of projected digraph.

BG and an English dependency tree $T_{\mathbf{en}}$. The English tree $T_{\mathbf{en}} = (V_{\mathbf{en}}, A_{\mathbf{en}})$ consists of a set of vertices $V_{\mathbf{en}} = \{u_0, u_1, ..., u_m\}$, where u_0 corresponds to the ROOT node, and a set of arcs $A_{\mathbf{en}} \subseteq \{(u', u, gf) | u', u \in V_{\mathbf{en}}, gf \in GF\}$[3] labelled with grammatical functions from GF.

Arcs of the English tree $T_{\mathbf{en}}$ are then iteratively projected to the Polish equivalent sentence. For each Polish lexical node v, its governor node v' is found in the following way. First, an English non-ROOT node u connected with v is looked for in the bipartite alignment graph BG. Then, the governor node u' of u is found in the English tree $T_{\mathbf{en}}$. Finally, the Polish node v' which is connected with u' in BG is identified and recognised as the governor of v.

An arc (v', v, l) between Polish tokens v' and v, which is assigned the label l, can be added to the Polish digraph. The only restriction is that it is not possible to project arcs via bipartite edges which are both weighted with 0. The reason for this limitation is to avoid projection of arcs considered to be the most error prone. However, projection via two edges one of which is weighted with 0 is permitted in order to cover relations between English tokens one of which is not aligned with any Polish token in any of word alignment sets.

For each sentence pair the projection module outputs the set of Polish vertices $V_{\mathbf{pl}}$ and the set of arcs $A_{\mathbf{pl}}$ between these vertices. The set of vertices $V_{\mathbf{pl}} = \{v_0, v_1, ..., v_n\}$ consists of an additional ROOT node v_0 and a set of lexical nodes $\{v_1, ..., v_n\}$, for each vertex v_i corresponding to the ith token of a Polish sentence $S = t_1, ..., t_n$. The vertices from $V_{\mathbf{pl}}$ are connected with arcs from the set $A_{\mathbf{pl}} \subseteq \{(v_i, v_j, l) | v_i, v_j \in V_{\mathbf{pl}}, l = (w_d, w_g, gf, f)\}$, for $w_d, w_g \in \{0, 1, 2, 3\}, gf \in GF, f \in \mathbb{N}_+$. These two sets constitute a Polish digraph in which each node directly or indirectly depends on the ROOT node and the ROOT node does not have any predecessor.

Any projected arc is assigned a label $l = (w_d, w_g, gf, f)$. The first element w_d refers to the weight of a bipartite edge connecting English and Polish tokens with the dependent status. The second element w_g refers to the weight of a bipartite edge connecting English and Polish tokens with the governor status. The third element gf indicates the label of the projected English dependency relation. The projection frequency f indicates the number of English relations labelled with the same grammatical function which are projected to the same two Polish tokens via equally weighted bipartite edges.

Intuitive Weighting of Projected Arcs. Intuitively, an arc between two tokens might be more important than arcs between other tokens if it is projected via bipartite edges with higher scores. Projected arcs are thus scored with initial weights that are estimated based on scores of bipartite edges (w_d and w_g) used in the projection of a particular arc and a projection frequency f. We define the following function $s(v_i, v_j, (w_d, w_g, gf, f)) = w_d + w_g + 2 w_d w_g f$ scoring projected arcs. Initially weighted projected digraphs (or even multi-digraphs since English

[3] The arc (u', u, gf) indicates an edge directed from u' to u and labelled with the grammatical function gf.

arcs are projected via all possible pairs of bipartite edges) provide a starting point to induce final dependency trees.

2.2 Weighted Induction

This section presents weighted induction which is the second step in the process of acquiring Polish dependency structures. The main idea behind weighted induction is to identify the most likely arcs in initially scored projected digraphs and to assign them appropriate weights. Using methods of selecting maximum spanning trees from weighted directed graphs, final well-formed dependency structures, i.e., *maximum spanning dependency trees* (MSDTs), are inferred from weighted projected digraphs. A maximum spanning dependency tree $T = (V', A')$ extracted from a weighted projected digraph $G = (V, A)$, for $A' \subseteq A$ and $V' = V = \{v_0, v_1, ..., v_n\}$, where v_i corresponds to the ith token of a sentence $S = t_1, ..., t_n$ and v_0 is a ROOT node, corresponds to a valid dependency structure if v_0 is the root of T, i.e., $(v_i, v_0, l) \notin A'$, for $v_i \in V'$, $l \in L$, and v_0 has only one successor, i.e., if $(v_0, v_i, l) \in A'$, then $(v_0, v_j, l') \notin A'$, for $v_i \neq v_j$.

Arcs of projected digraphs are assigned initial weights calculated on the basis of weights of bipartite edges used in projection of these arcs. Weights of bipartite edges, in turn, result from automatic word alignment which is prone to errors. We therefore propose a heuristic of recalculating initial arc weights in projected digraphs. The recalculation applies the probability distribution over arcs in k-best MSDTs selected from initially weighted projected digraphs. The probability distribution over selected arcs identified by their feature representations is estimated using the EM-inspired selection algorithm. Projected digraphs with recalculated arc weights are used to induce final dependency structures. A schema of the weighted induction procedure is shown in Figure 1.

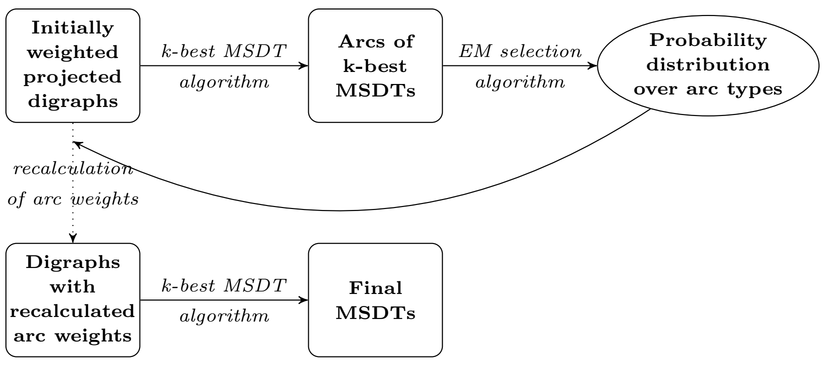
Initially weighted projected digraphs
k-best MSDT algorithm
Arcs of k-best MSDTs
EM selection algorithm
Probability distribution over arc types
recalculation of arc weights
Digraphs with recalculated arc weights
k-best MSDT algorithm
Final MSDTs

Fig. 1. Schema of the weighted induction procedure

***K*-best MSDTs.** The induction procedure starts with the selection of k-best MSDTs from projected digraphs using a slightly modified version of the k-best MSTs selection algorithm by [4].[4] A list of k-best MSDTs of the digraph G is computed with the function *rank* of this algorithm. This function is slightly modified in relation to the original function *rank* which outputs a list of k-best MSTs (c.f., [4], p. 107). In our version, some additional conditions are imposed on candidate MSTs so that they meet properties of well-formed dependency trees. Since not all MSTs fulfil these properties, only valid MSDTs are taken into account in estimation of the probability distribution.

Feature Representations of Arcs. Arcs used in the EM training are represented with their features. Each of related nodes represents a token in a sentence and encodes information about the token's lemma, part of speech tag, and morphological features. Furthermore, any arc is assigned a label and an initial weight. The information available in the arc label and in the related nodes may be used in the feature representation j of this arc. The set of features identifying an arc is given with the function *fr*, i.e., $fr(v_h, v_i, (w_d, w_g, gf, f)) = j$.

Probability Distribution over Arc Types. The probability distribution over arc types is estimated with a version of the EM algorithm defined by [7]. This EM selection algorithm was originally designed to select the most probable valency frames from sets of valency frame candidates. Dębowski's algorithm is adapted for our purposes of identifying the most reliable arcs in sets of arcs in k-best MSDTs found in initially weighted projected digraphs.

Assume we have a training set $\mathcal{B} = \{B_1, ..., B_N\}$, where B_i is a set of arcs in k-best MSDTs coming into the ith vertex, for $i = 1, ..., N$ and N being a number of all nodes in k-best MSDTs. In this setting, the EM selection algorithm estimates model parameters $\theta_t = \left(p_j^{(t)}\right)_{j \in J}$, where t is the iteration number, for $t = 2, ..., T$, and j is a feature representation of an arc, for $j \in J$ and J being a set of all possible arc types in $\mathcal{B}$. The EM-inspired selection algorithm iterates over the formulae in (1) and (2) and defines a series of parameter values θ_2, ..., θ_t until the last iteration. In the first step of each iteration, new parameter values $p_j^{(t)} = P(j|\theta_t)$ are estimated. The second step of each iteration is to estimate values $p_{ij}^{(t)} = P(j|B_i, \theta_t)$, for each $i = 1, ..., N$ and for each possible arc type $j \in J$. In the original version of the EM selection algorithm, the coefficient p_{ij} is a quotient of the probability value p_j of an arc with the type j and the sum of probability values $p_{j'}$ of all arcs in B_i. We modify the way of estimating the coefficient p_{ij} in order to take into account the initial weight $s(v_h, v_i, l)$ of the arc with the type j.

$$p_j^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} p_{ij}^{(t)} \qquad (1)$$

[4] The algorithm by [4] was used for other task related to some extent to our approach, e.g., for re-ranking of parses [9].

$$p_{ij}^{(t)} = \begin{cases} \dfrac{p_j^{(t)} \times s(v_h, v_i, l)}{\sum\limits_{j' \in B_i} p_{j'}^{(t)} \times s(v_{h'}, v_i, l')} & \text{, if } fr(v_h, v_i, l) = j \\ 0 & \text{, otherwise.} \end{cases} \tag{2}$$

The initial parameter values are set to 1, i.e., $p_j^{(1)} = 1$, as in the original approach by [7]. At each iteration, the new parameter values θ_t are calculated as a function of the previous parameter values θ_{t-1} and the training set $\mathcal{B}$. The EM-inspired selection algorithm iterates until the final iteration T is reached.

According to the original procedure by [7], the most likely arc would be selected from the set of possible arcs B_i. However, the most probable incoming arcs for each lexical node do not have to necessarily constitute a valid dependency tree (e.g., a resulting graph may contain a cycle). Therefore, our approach to recalculating weights does not build directly on the selected arcs but on the probability distribution $\left(p_j^{(T)}\right)$ over feature representations of arcs J estimated in the last iteration of the EM selection algorithm.

Recalculation of Arc Weights in Projected Digraphs. The new weight of an arc (v_h, v_i, l) with the feature representation j is calculated as the product of the square root[5] of the previous arc weight and the value p_j (see Equation 3).

$$s^* = \sqrt{s(v_h, v_i, l)} \times p_j, \qquad \text{for } fr(v_h, v_i, l) = j \tag{3}$$

If an arc is not present in any of k-best MSDTs, its probability value is equal to 0. Because there is a risk that some digraph arcs would be assigned 0 and they would have the same priority in the extraction of final MSDTs, their scores s^* are calculated as the product of the square root of the initial arc weight, the lowest value p_j in $\left(p_j^{(T)}\right)$ and an optimisation factor α which further decreases weights of unselected arcs (see Equation 4).

$$s^* = \sqrt{s(v_h, v_i, l)} \times \min_j p_j \times \alpha, \qquad \text{for some } 0 < \alpha < 1 \tag{4}$$

The main idea behind the recalculation is to reward arcs with the probability greater than zero by assigning them higher weights, and to penalise other arcs by assigning them lower weights. Arcs with higher weights are more likely to be selected as part of final dependency trees.

3 Experiments and Evaluation

To test the method outlined above, we conduct an experiment consisting in the projection of English dependency relations to Polish sentences and in the in-

[5] Arcs in projected digraphs are assigned initial weights from $\mathbb{N}_+$. In order to diminish the difference between initial weights and probability values, and therefore to raise the importance of relatively low probability values, initial weights are square rooted.

duction of Polish dependency trees. Since there is no Polish-English parallel corpus annotated with gold-standard dependency trees, we may evaluate neither the induction procedure itself nor the quality of induced trees. Instead, we perform an extrinsic evaluation to see to what extent induced trees affect performance of a parser trained on them.

3.1 Data and Preprocessing

The experiment is conducted on a large collection of Polish–English bitexts gathered from publicly available sources: *Europarl* [13], *DGT-Translation Memory* [25], *OPUS* [27] and *Pelcra Parallel Corpus* [21]. After tokenisation, sentence segmentation and sentence alignment, bitexts are used to produce automatic word alignment links using the statistical machine translation system MOSES [15]. Three sets of alignment links are generated: Polish-to-English, English-to-Polish and a set of links from both unidirectional alignments selected with the *grow-diag-final-and* method implemented as part of the MOSES system.

To parse the English side of the parallel corpus, we use the handcrafted wide-coverage English *Lexical Functional Grammar* [6,3], using the *Xerox Linguistic Environment* [5] as a processing platform.[6] The most probable LFG analyses are converted into dependency trees using a conversion procedure similar as in [20]. The conversion of permitted LFG analyses results in a collection of 4,946,809 English dependency trees, which constitute the subject matter of projection.

3.2 Automatic Induction of Polish Dependency Trees

Given three sets of word alignment links, English dependency trees, and Polish sentences enriched with morphosyntactic information using the *Pantera* tagger [1], the projection module (see Section 2.1) outputs 4,946,809 initially weighted digraphs. Then, the induction module (see Section 2.2) extracts 4,615,698 sets of k-best MSDTs (for $k = 10$) from the entire set of initially weighted digraphs, estimates the probability distribution over arc types in these k-best MSDTs within 10 iterations of the EM selection algorithm, recalculates initial arc weights in projected digraphs and acquires final MSDTs from these digraphs.

Since the final MSDTs are labelled with English grammatical functions, we treat them as unlabelled dependency structures at this point. Arcs in these

[6] Similarly as publicly available data-driven dependency parsers, the *XLE* parser for English may deal with some ungrammatical sentences. It applies the shallow parsing (or chunking) technique to identify well-formed chunks (constituents) in a problematic sentence and then composes them linearly into a FIRST-REST structure marked as FRAGMENTS. Hence the English *XLE* parser marks dubious sentences as FRAGMENTS in contrast to some other parsers which do not distinguish proper sentences from problematic strings of tokens. Since the aim of the current experiment is to build a bank of Polish dependency structures for strings of tokens considered as well-formed sentences or phrases, analyses marked as FRAGMENTS are not taken into account in projection.

unlabelled dependency trees are then assigned Polish dependency labels derived from projected English grammatical functions and morphosyntactic features of related Polish tokens using a set of predefined labelling rules. The entire induction procedure outputs a bank of 3,958,556 labelled dependency structures on which a Polish dependency parser may be trained.

3.3 Evaluation Experiment

We use the *Mate* system [2] in our evaluation experiment. The performance of the *Mate* parser trained on automatically induced trees is evaluated against a set of 822 dependency trees (*manual test*) taken from the Polish dependency treebank [28].[7] Furthermore, we provide a version of these test trees with automatically generated part of speech tags and morphological features (*automatic test*). In addition to these test sets, the parser is evaluated against a set of 100 relatively complex trees (*additional test*).[8]

Table 1 reports results of the *Mate* parser trained on induced dependency trees.[9] Parsing performance is measured with two evaluation metrics: *unlabelled attachment score* (UAS) and *labelled attachment score* (LAS) as defined by [16]. These results are compared with the performance of a *supervised* parser trained on a part of the Polish treebank.

A parser trained on automatically induced trees (*induced*) in one iteration[10] achieves 73.7% UAS if tested against the manual test set, 72.8% UAS if tested against the automatic test trees and 63.5% UAS if tested against the additional test trees. These results are significantly below the performance of a parser (*supervised*) trained on trees from the Polish dependency treebank.

[7] [28] provide a detailed description of the schema used to annotate Polish sentences with dependency tree representations.

[8] Additional test sentences were randomly selected from some Polish newspapers. The selected sentences are quite long and contain 15.3 tokens per sentence on average. They were first automatically tokenised, lemmatised and part of speech tagged, and then manually annotated with dependency trees by two experienced linguists. These linguists also corrected possible errors in lemmatisation and tagging, but not discrepancies in tokenisation.

[9] The reported experiment was preceded by a preliminary experiment. This experiment consisted in comparing performance of dependency parsers trained on two sets of MSDTs selected from a limited set of 1.1 million projected digraphs: (1) 1,000,797 MSDTs selected from projected digraphs with initially weighted arcs, and (2) 924,733 MSDTs selected from projected digraphs with EM-recalculated arc weights. According to the preliminary results the parser trained on the trees with EM-recalculated arc weights outperformed the baseline parser trained on the initially scored trees by 2.4 percentage points.

[10] Preliminary experiments show that the parsing performance decreases with the increasing number of iterations used to train the *Mate* parser. The decrease in parsing performance may be due to noise which is learnt in successive iterations. Therefore, we limit the number of *Mate* iterations to one.

Table 1. Performance of parsers trained with the *Mate* parsing system on the Polish dependency trees acquired with the weighted induction method (*induced*), induced and labelled (*labelled*), labelled and modified (*modified*), and labelled, modified and filtered (*filtered*). Settings of model training: one iteration, the heap size of 100 million features, the threshold of the non-projective approximation of 0.2. The *supervised* model is trained on 7405 trees from the Polish dependency treebank. Setting of supervised model training: 10 iterations, the heap size of 100M, the threshold of 0.2. Validation data sets: *Manual Test* – the set of 822 treebank trees; *Automatic Test* – the set of 822 treebank trees with automatic morphosyntactic annotations of tokens; *Additional Test* – the set of 100 sentences manually annotated with dependency trees.

Model	**Data**	**Manual Test**		**Automatic Test**		**Additional Test**	
		UAS	LAS	UAS	LAS	UAS	LAS
induced	3958556	73.7	–	72.8	–	63.5	–
labelled	3958556	74.6	69.4	74.0	68.1	63.7	58.3
modified	3958556	85.1	79.2	84.0	77.3	74.3	68.5
filtered	2352940	86.0	80.5	84.7	78.3	**76.1**	**70.3**
supervised	7405	**92.7**	**87.2**	**88.4**	**81.0**	76.0	69.5

Following Hwa's idea of improving automatically induced trees, we define 45 labelling rules and 31 correction rules.[11] Even if the induction process seems to be straightforward, there are still some Polish-specific morphosyntactic phenomena or linguistic structures diversely annotated in both languages the annotation of which may not result from the English dependency tree. The *Mate* parser trained on induced trees labelled with Polish dependency types (*labelled*) achieves 74.6% UAS and 69.4% LAS if tested against the manual test trees, 74% UAS and 68.1% LAS if tested against the automatic test trees and 63.7% UAS and 58.3% LAS if tested against the additional test trees. The *Mate* parser trained on induced dependency trees modified with predefined rules performs significantly better – 85.1% UAS and 79.2% LAS if tested against the manual test trees, 84% UAS and 77.3% LAS if test against the automatic test set and 74.3% UAS and 68.5% LAS if tested against additional test trees. These results are still below parsing performance of the supervised parser. Note, however, that in the third – more realistic – scenario on evaluating the parser on real data, the more useful measure LAS shows that the results of the semi-supervised procedure described here are directly comparably to the more costly supervised procedure.

Filtering is one of the most common optimisation techniques in projection-based approaches. Our results show that filtering of possibly incorrect trees does not contribute significantly to improving parsing performance since only two simple filtering criteria are used: percentage of non-projective arcs and percentage of arcs labelled with a default function *dep*. The best parsing results (*filtered*) are achieved if we reject trees with more than 30% of non-projective arcs and with more than 10% of *dep*-labelled arcs – 86% UAS and 80.5% LAS if tested against the manual test set, 84.7% UAS and 78.3% LAS if tested against

[11] Due to lack of space, a detailed presentation of all individual rules is not possible here. A general presentation of labelling and correction rules may be found in [30].

automatic trees and 76.1% UAS and 70.3% LAS if tested against the additional trees. The results of evaluation against the additional trees show that dependency parsers developed in an automatic way as described here may rival fully supervised – and, hence, more costly – parsers.

4 Conclusion

This paper presented a novel weighted induction method of obtaining Polish dependency structures. The weighted induction procedure consists of two main steps: projection of dependency relations and induction of well-formed dependency trees. The projection step resembles cross-lingual dependency projection pioneered by [10]. However, it is not required in our approach that projection results in dependency trees as in [10] or partial dependency structures as in [12] or [24]. Instead, all possible dependency arcs are projected and they constitute initially weighted digraphs. Previous approaches do not need any further steps after projection of dependency relations since projected trees (or tree fragments) are considered to be the final data for parser training. In our approach, projected digraphs may contain noisy arcs that should not be used in parser training. We thus proposed a method of recalculating initial arc weights and selecting the final MSDTs from the projected digraphs with recalculated arc weights. The weighted induction method allows to annotate most of sentences with proper dependency trees that could not necessarily be acquired in case of direct projection. Hence the aggressive filtering techniques are not applicable and weighted induction does not lead to a huge loss of data as in [11]. The well-formed induced MSDTs are thus presumably more appropriate than direct projections for parser training.

Results of an extrinsic evaluation consisting in training the *Mate* parser on induced trees are very encouraging. Even if they are mostly a little below the performance of the supervised parser, when tested on a homogenous set of rather short sentences from the treebank on which the parser was trained, a test against a small set of long and complex trees shows that a parser trained on so-induced trees may exceed the supervised upper bound. As this projection-based result was achieved with much less manual work than in the supervised scenario – construction of a few dozen labelling and correction rules as opposed to annotating thousands of sentences – we conclude that for the purpose of developing dependency parsers, the method described here rivals the supervised scenario.

While our experiment considered the Polish-English language pair, the weighted induction method may be applied to obtain dependency structures for other resource-poor languages which do not have any annotated data but have a reasonable number of sentences which are parallel with their translations in a resource-rich language. The weighted induction method was tested on the task of obtaining dependency structures, but it may also apply to other projection tasks, e.g., semantic role labelling or word sense disambiguation.

Acknowledgements. The presented research was supported by grant no POIG.01.01.02-14-013/09 from Innovative Economy Operational Programme co-financed by the European Union (European Regional Development Fund).

References

1. Acedański, S.: A morphosyntactic Brill tagger for inflectional languages. In: Loftsson, H., Rögnvaldsson, E., Helgadóttir, S. (eds.) IceTAL 2010. LNCS (LNAI), vol. 6233, pp. 3–14. Springer, Heidelberg (2010)
2. Bohnet, B.: Very High Accuracy and Fast Dependency Parsing is not a Contradiction. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 89–97 (2010)
3. Bresnan, J.: Lexical-Functional Syntax. Blackwell, Oxford (2001)
4. Camerini, P.M., Fratta, L., Maffioli, F.: The K Best Spanning Arborescences of a Network. Networks 10, 91–110 (1980)
5. Crouch, D., Dalrymple, M., Kaplan, R., King, T., Maxwell, J., Newman, P.: XLE Documentation. Palo Alto Research Center (PARC), Palo Alto (2011)
6. Dalrymple, M.: Lexical-Functional Grammar. Syntax and Semantics, vol. 34. Academic Press (2001)
7. Dębowski, Ł.: Valence extraction using EM selection and co-occurrence matrices. Language Resources and Evaluation 43(4), 301–327 (2009)
8. Ganchev, K., Gillenwater, J., Taskar, B.: Dependency Grammar Induction via Bitext Projection Constraints. In: Proceedings of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, vol. 1, pp. 369–377 (2009)
9. Hall, K.: k-best Spanning Tree Parsing. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 392–399 (2007)
10. Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., Kolak, O.: Bootstrapping Parsers via Syntactic Projection across Parallel Texts. Natural Language Engineering 11(3), 311–325 (2005)
11. Jiang, W., Liu, Q.: Automatic Adaptation of Annotation Standards for Dependency Parsing – Using Projected Treebank as Source Corpus. In: Proceedings of the 11th International Conference on Parsing Technologies, IWPT 2009, pp. 25–28 (2009)
12. Jiang, W., Liu, Q.: Dependency Parsing and Projection Based on Word-Pair Classification. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 12–20 (2010)
13. Koehn, P.: Europarl: A Parallel Corpus for Statistical Machine Translation. In: Proceedings of the 10th Machine Translation Summit Conference, pp. 79–86 (2005)
14. Koehn, P.: Statistical Machine Translation. Cambridge University Press (2010)
15. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open Source Toolkit for Statistical Machine Translation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 177–180 (2007)
16. Kübler, S., McDonald, R.T., Nivre, J.: Dependency Parsing. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers (2009)
17. McDonald, R., Petrov, S., Hall, K.B.: Multi-Source Transfer of Delexicalized Dependency Parsers. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 63–72 (2011)

18. Naseem, T., Barzilay, R., Globerson, A.: Selective Sharing for Multilingual Dependency Parsing. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers, vol. 1, pp. 629–637 (2012)
19. Och, F.J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1), 19–51 (2003)
20. Øvrelid, L., Kuhn, J., Spreyer, K.: Improving Data-Driven Dependency Parsing Using Large-Scale LFG Grammars. In: Proceedings of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (Conference Short Papers), pp. 37–40 (2009)
21. Pęzik, P., Ogrodniczuk, M., Przepiórkowski, A.: Parallel and spoken corpora in an open repository of Polish language resources. In: Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pp. 511–515 (2011)
22. Smith, D.A., Eisner, J.: Parser Adaptation and Projection with Quasi-Synchronous Grammar Features. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 822–831 (2009)
23. Søgaard, A.: Data point selection for cross-language adaptation of dependency parsers. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers, vol. 2, pp. 682–686 (2011)
24. Spreyer, K.: Does It Have To Be Trees? Data-Driven Dependency parsing with Incomplete and Noisy Training Data. Ph.D. thesis, Universität Potsdam (2011)
25. Steinberger, R., Eisele, A., Klocek, S., Pilos, S., Schlüter, P.: DGT-TM: A freely Available Translation Memory in 22 Languages. In: Proceedings of the 8th International Conference on Language Resources and Evaluation, pp. 454–459 (2012)
26. Täckström, O., McDonald, R., Nivre, J.: Target Language Adaptation of Discriminative Transfer Parsers. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1061–1071 (2013)
27. Tiedemann, J.: Parallel Data, Tools and Interfaces in OPUS. In: Proceedings of the 8th International Conference on Language Resources and Evaluation, pp. 2214–2218 (2012)
28. Wróblewska, A.: Polish Dependency Bank. Linguistic Issues in Language Technology 7(1), 1–15 (2012)
29. Wróblewska, A., Przepiórkowski, A.: Induction of Dependency Structures Based on Weighted Projection. In: Nguyen, N.-T., Hoang, K., Jędrzejowicz, P. (eds.) ICCCI 2012, Part I. LNCS (LNAI), vol. 7653, pp. 364–374. Springer, Heidelberg (2012)
30. Wróblewska, A., Przepiórkowski, A.: Projection-based Annotation of a Polish Dependency Treebank. In: Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, pp. 2306–2312. ELRA, Reykjavík (2014)
31. Zeman, D., Resnik, P.: Cross-Language Parser Adaptation between Related Languages. In: Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pp. 35–42 (2008)