Many Languages, One Parser

Waleed Ammar♦  George Mulcaire♥  Miguel Ballesteros♠  Chris Dyer♦  Noah A. Smith♥
♦School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
♥Computer Science & Engineering, University of Washington, Seattle, WA, USA
♠NLP Group, Pompeu Fabra University, Barcelona, Spain
wammar@cs.cmu.edu, gmulc@uw.edu, miguel.ballesteros@upf.edu
cdyer@cs.cmu.edu, nasmith@cs.washington.edu

Abstract

We train one multilingual model for dependency parsing and use it to parse sentences in several languages. The parsing model uses (i) multilingual word clusters and embeddings; (ii) token-level language information; and (iii) language-specific features (fine-grained POS tags). This input representation enables the parser not only to parse effectively in multiple languages, but also to generalize across languages based on linguistic universals and typological similarities, making it more effective to learn from limited annotations. Our parser’s performance compares favorably to strong baselines in a range of data scenarios, including when the target language has a large treebank, a small treebank, or no treebank for training.

1 Introduction

Developing tools for processing many languages has long been an important goal in NLP (Rosner, 1988; Heid and Raab, 1989) but it was only when statistical methods became standard that massively multilingual NLP became economical. The mainstream approach for multilingual NLP is to design language-specific models. For each language of interest, the resources necessary for training the model are obtained (or created), and separate parameters are fit for each language separately. This approach is simple, effective and grants the flexibility of customizing the model and features to the needs of each language, but it is suboptimal for theoretical as well as practical reasons. Theoretically, the study of linguistic typology tells us that many languages share morphological, phonological, and syntactic phenomena (Bender, 2011); therefore, the mainstream approach misses an opportunity to exploit relevant supervision from typologically related languages. Practically, it is inconvenient to deploy or distribute NLP tools that are customized for many different languages because, for each language of interest, we need to configure, train, tune, monitor, and occasionally update the model. Furthermore, code-switching or code-mixing (mixing more than one language in the same discourse), which is pervasive in some genres, in particular social media, presents a challenge for monolingually-trained NLP models (Barman et al., 2014).

In parsing, the availability of homogeneous syntactic dependency annotations in many languages (McDonald et al., 2013; Nivre et al., 2015b; Agić et al., 2015; Nivre et al., 2015a) has created an opportunity to develop a parser that is capable of parsing sentences in multiple languages, addressing these theoretical and practical concerns. A multilingual parser can potentially replace an array of language-specific monolingually-trained parsers (for languages with a large treebank). The

1As of 2007, the total number of native speakers of the hundred most popular languages only accounts for 85% of the world’s population (Wikipedia, 2016).

2While our parser can be used to parse input with code-switching, we were not able to evaluate this capability due to the lack of appropriate data.

3Although multilingual dependency treebanks have been available for a decade via the 2006 and 2007 CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007), the treebank of each language was annotated independently and with its own annotation conventions.
same approach have been used in low-resource scenarios (with no treebank or a small treebank in the target language), where indirect supervision from auxiliary languages improves the parsing quality \cite{Cohen2011,McDonald2011,Zhang2015,Duong2015a,Duong2015b,Guo2016}, but these models may sacrifice accuracy on source languages with a large treebank. In this paper, we describe a model that works well for both low-resource and high-resource scenarios.

We propose a parsing architecture that takes as input sentences in several languages\footnote{We discuss data requirements in the next section.} optionally predicting the part-of-speech (POS) tags and language ID if needed. The parser is trained on the union of available universal dependency annotations in different languages. Our approach integrates and critically relies on several recent developments related to dependency parsing: universal POS tagsets \cite{Petrov2012}, cross-lingual word clusters \cite{Naseem2012}, selective sharing \cite{Naseem2012}, universal dependency annotations \cite{McDonald2013,Nivre2015,Nivre2015a,Agić2015}, advances in neural network architectures \cite{Chen2014,Dyer2015}, and multilingual word embeddings \cite{Gardner2015,Guo2016,Ammar2016}. We show that our parser compares favorably to strong baselines trained on the same treebanks in three data scenarios: when the target language has a large treebank (Table 3), a small treebank (Table 7), or no treebank (Table 8). Our parser will be made publicly available on github\footnote{https://github.com/clab/language-universal-parser}.\footnote{We do not use the co-reference disambiguation resource.}

2 Overview

Our goal is to train a dependency parser for a set of target languages $L^t$, given universal dependency annotations in a set of source languages $L^s$. Ideally, we would like to have training data in all target languages (i.e., $L^t \subseteq L^s$), but we are also interested in the case where the sets of source and target languages are disjoint (i.e., $L^t \cap L^s = \emptyset$). When all languages in $L^t$ have a large treebank, the main-stream approach has been to train one monolingual parser per target language and route sentences of a given language to the corresponding parser at test time. In contrast, our approach is to train one parsing model with the union of treebanks in $L^s$, then use this single trained model to parse text in any language in $L^t$, hence the name “many languages, one parser” (MALOPA). In a MALOPA parser, we strike a balance between: (1) enabling cross-lingual model transfer via language-invariant input representations; i.e., coarse POS tags, multilingual word embeddings and multilingual word clusters, and (2) tweaking the behavior of the parser depending on the current input language via language-specific representations; i.e., fine-grained POS tags and language embeddings.

In addition to universal dependency annotations in source languages (see Table 1), we use the following data resources for each language in $L = L^t \cup L^s$: \footnote{See \S 3.6 for details.}

- universal POS annotations for training a POS tagger \footnote{Our best results make use of this resource. We require that all languages in $L$ are (transitively) connected. The bilingual dictionaries we used are based on unsupervised word alignments of parallel corpora, as described in \cite{Guo2016}. See \S 3.3 for details.} 
- a bilingual dictionary with another language in $L$ for adding cross-lingual lexical information \footnote{See \S 3.4 for details.} 
- language typology information \footnote{Our best results make use of this resource. See \S 3.3 for details.} 
- language-specific POS annotations \footnote{Our best results make use of this resource. See \S 3.5 for details.} and
- a monolingual corpus \footnote{This is only used for training word embeddings with ‘multiCCA’, ‘multiCluster’ and ‘translation-invariance’ in Table 6.} 

Novel contributions of this paper include: (1) using one parser to substitute an array of monolingually-trained parsers without sacrificing accuracy on languages with a large treebank, (2) an effective neural network architecture for using language embeddings to improve multilingual parsing, and (3) a study of how predicted language IDs affect the performance of a multilingual dependency parser.
While not the primary focus of this paper, we also show that a variant of our parser outperforms previous work on multi-source cross-lingual parsing in low resource scenarios, where languages in \( L^t \) have a small treebank (see Table 7) or where \( L^t \cap L^s = \emptyset \) (see Table 8). In the small treebank setup with 3K token annotations, we show that our parser consistently outperforms a strong monolingual baseline with 5.7 absolute LAS points per language, on average.

3 Parsing Model

Recent advances suggest that recurrent neural networks, especially long short-term memory (LSTM) architectures, are capable of learning useful representations for modeling problems of sequential nature (Graves et al., 2013; Sutskever et al., 2014). In this section, we describe our language-universal parser, which extends the stack LSTM (S-LSTM) parser of Dyer et al. (2015).

3.1 Transition-based Parsing with S-LSTMs

This section briefly reviews Dyer et al.’s S-LSTM parser, which we modify in the following sections. The core parser can be understood as the sequential manipulation of three data structures:

- a buffer (from which we read the token sequence),
- a stack (which contains partially-built parse trees),
- and
- a list of actions previously taken by the parser.

The parser uses the arc-standard transition system (Nivre, 2004).\(^{11}\) At each timestep \( t \), a transition ac-

| German (de) | English (en) | Spanish (es) | French (fr) | Italian (it) | Portuguese (pt) | Swedish (sv) |
|-------------|-------------|-------------|------------|-------------|----------------|-------------|
| train       | 14118       | 39832       | 14138      | 14511       | 6389           | 9600        |
| dev.        | 801         | 14011       | 1596       | 399         | 489            | 288         |
| test        | 1001        | 271         | 504        | 14118       | 12543          | 14187       |

3.2 Token Representations

The vector representations of input tokens feed into the stack-LSTM modules of the buffer and the stack.\(^{12}\)

Table 1: Number of sentences (tokens) in each treebank split in Universal Dependency Treebanks version 2.0 (UDT) and Universal Dependencies version 1.2 (UD) for the languages we experiment with. The last row gives the number of unique language-specific fine-grained POS tags used in a treebank.

The parser state at time \( t \) is given by:

\[ p_t = \max \{ 0, W_s b_t + a_t \} \]

where the matrix \( W \) and the vector \( W_{bias} \) are learned parameters. The parser state \( p_t \) is then used to define a categorical distribution over possible next actions \( z \).

\[ p(z | p_t) = \frac{\exp \left( g_T p_t + q_z \right)}{\sum_{z'} \exp \left( g_{z'} p_t + q_{z'} \right)} \]

where \( g_z \) and \( q_z \) are parameters associated with action \( z \). The selected action is then used to update the buffer, stack and list of actions, and to compute \( b_{t+1}, s_{t+1} \) and \( a_{t+1} \) accordingly.

The model is trained to maximize the log-likelihood of correct actions. At test time, the parser greedily chooses the most probable action in every time step until a complete parse tree is produced.

The following sections describe our extensions of the core parser. More details about the core parser can be found in Dyer et al. (2015).

11In a preprocessing step, we transform non-projective trees in the training treebanks to pseudo-projective trees using the “baseline” scheme in Nivre and Nilsson, 2009. We evaluate against the original non-projective test set.

12A stack-LSTM module is used to compute the vector representation for each data structure, as detailed in Dyer et al. (2015).

The total number of actions is \( 1 + 2x \) number of unique dependency labels in the treebank used for training, but we only consider actions which meet the arc-standard preconditions in Fig.2.
### Table 2: Parser transitions indicating the action applied to the stack and buffer at time \( t \) and the resulting stack and buffer at time \( t + 1 \).

| Stack \(_t\) | Buffer \(_t\) | Action          | Dependency | Stack \(_{t+1}\) | Buffer \(_{t+1}\) |
|-------------|-------------|-----------------|------------|-----------------|-----------------|
| \( u, v, S \) | \( B \) | REDUCE-RIGHT(\( r \)) | \( u \rightarrow v \) | \( u, S \) | \( B \) |
| \( u, v, S \) | \( B \) | REDUCE-LEFT(\( r \)) | \( u \leftarrow v \) | \( v, S \) | \( B \) |
| \( S \) | \( u, B \) | SHIFT           | —          | \( u, S \) | \( B \) |

For monolingual parsing, we represent each token by concatenating the following vectors:

- a fixed, pretrained embedding of the word type,
- a learned embedding of the word type,
- a learned embedding of the Brown cluster,
- a learned embedding of the fine-grained POS tag,
- a learned embedding of the coarse POS tag.

For multilingual parsing with MALOPA, we start with a simple delexicalized model where the token representation only consists of learned embeddings of coarse POS tags, which are shared across all languages to enable model transfer. In the following subsections, we enhance the token representation in MALOPA to include lexical embeddings, language embeddings, and fine-grained POS embeddings.

### 3.3 Lexical Embeddings

Previous work has shown that sacrificing lexical features amounts to a substantial decrease in the performance of a dependency parser \cite{Cohen2011, Tackstrom2012, Tiedemann2015, Guo2015}. Therefore, we extend the token representation in MALOPA by concatenating learned embeddings of multilingual word clusters, and pretrained multilingual embeddings of word types.

#### Multilingual Brown clusters

Before training the parser, we estimate Brown clusters of English words and project them via word alignments to words in other languages. This is similar to the ‘projected clusters’ method in \cite{Tackstrom2012}. To go from Brown clusters to embeddings, we ignore the hierarchy within Brown clusters and assign a unique parameter vector to each cluster.

### 3.4 Language Embeddings

While many languages, especially ones that belong to the same family, exhibit some similar syntactic phenomena (e.g., all languages have subjects, verbs, and objects), substantial syntactic differences abound. Some of these differences are easy to characterize (e.g., subject-verb-object vs. verb-subject-object, prepositions vs. postpositions, adjective-noun vs. noun-adjective), while others are subtle (e.g., number and positions of negation morphemes). It is not at all clear how to translate descriptive facts about a language’s syntax into features for a parser.

Consequently, training a language-universal parser on treebanks in multiple source languages...
We modify the parsing architecture as follows:

- compute the language embedding where
- ID turns out to work best in our experiments.

Inspired by Naseem et al. (2012), we address this problem by informing the parser about the input language it is currently parsing. Let \( I \) be the input vector representation of a particular language. We consider three definitions for \( I \):

- one-hot encoding of the language ID,
- one-hot encoding of individual word-order properties and
- averaged one-hot encoding of WALS typological properties (including word-order properties).

It is worth noting that the first definition (language ID) turns out to work best in our experiments.

We use a hidden layer with \( \tanh \) nonlinearity to compute the language embedding \( I' \) as:

\[
I' = \tanh(L \times 1 + L_{bias})
\]

where \( L \) and \( L_{bias} \) are additional model parameters. We modify the parsing architecture as follows:

- include \( I' \) in the action vector representation (which feeds into the stack-LSTM module of the buffer and the stack as described in §3.1), and
- redefine the parser state at time \( t \) as

\[
p_t = \max \{0, W[s_t; b_t; a_t; I'] + W_{bias}\}
\]

Intuitively, the first two modifications allow the input language to influence the vector representation of the stack, the buffer and the list of actions. The third modification allows the input language to influence the parser state which in turn is used to predict the next action. In preliminary experiments, we found that adding the language embeddings at the token and action level is important. We also experimented with computing more complex functions of \( (s_t, b_t, a_t, I') \) to define the parser state, but they did not help.

3.5 Fine-grained POS Tag Embeddings

Tiedemann (2015) shows that omitting fine-grained POS tags significantly hurts the performance of a dependency parser. However, those fine-grained POS tagsets are defined monolingually and are only available for a subset of the languages with universal dependency treebanks.

We extend the token representation to include a fine-grained POS embedding (in addition to the coarse POS embedding). We stochastically dropout the fine-grained POS embedding for each token with 50% probability so that the parser can make use of fine-grained POS tags when available but stay reliable when the fine-grained POS tags are missing.

3.6 Predicting POS Tags

The model discussed thus far conditions on the POS tags of words in the input sentence. However, gold POS tags may not be available in real applications (e.g., parsing the web). Here, we describe two modifications to 1) model both POS tagging and dependency parsing, and 2) increase the robustness of the parser to incorrect POS predictions.

**Tagging model.** Let \( x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_{2n} \) be the sequence of words, POS tags and parsing actions, respectively, for a sentence of length \( n \). We define the joint distribution of a POS
tag sequence and parsing actions given a sequence of words as follows:

\[
p(y_1, \ldots, y_n, z_1, \ldots, z_{2n} \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} p(y_i \mid x_1, \ldots, x_n) \\
\times \prod_{j=1}^{2n} p(z_j \mid x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_{j-1})
\]

where \( p(y_i \mid \ldots) \) is defined in Eq. 2 and \( p(y_i \mid x_1, \ldots, x_n) \) uses a bidirectional LSTM (Graves et al., 2013). Huang et al. (2015) show that the performance of a bidirectional LSTM POS tagger is on par with a conditional random field tagger.

We use slightly different token representations for tagging and parsing in the same model. For tagging, we construct the token representation by concatenating the embeddings of the word type (pre-trained), the Brown cluster and the input language. This token representation feeds into the bidirectional LSTM, followed by a softmax layer (at each position) which defines a categorical distribution over possible POS tags. For parsing, we construct the token representation by further concatenating the embeddings of predicted POS tags. This token representation feeds into the stack-LSTM modules of the buffer and stack components of the transition-based parser. This multi-task learning setup enables us to predict both POS tags and dependency trees in the same model. We note that pretrained word embeddings, cluster embeddings and language embeddings are shared for tagging and parsing.

**Block dropout.** We use an independently developed variant of word dropout (Iyyer et al., 2015), which we call block dropout. The token representation used for parsing includes the embedding of predicted POS tags, which may be incorrect. We introduce another modification which makes the parser more robust to incorrect POS tag predictions, by stochastically zeroing out the entire embedding of the POS tag. While training the parser, we replace the POS embedding vector \( e \) with another vector (of the same dimensionality) stochastically computed as: \( e' = (1 - b) / \mu \times e \), where \( b \in \{0, 1\} \) is a Bernoulli-distributed random variable with parameter \( \mu \) which is initialized to 1.0 (i.e., always dropout, setting \( b = 1, e' = 0 \)), and is dynamically updated to match the error rate of the POS tagger on the development set. At test time, we never dropout the predicted POS embedding, i.e., \( e' = e \). Intuitively, this method extends the dropout method (Srivastava et al., 2014) to address structured noise in the input layer.

### 4 Experiments

In this section, we evaluate our MALOPA parser in three data scenarios: when the target language has a large treebank (Table 3), a small treebank (Table 7) or no treebank (Table 8).

**Data.** For experiments where the target language has a large treebank, we use the standard data splits for German (de), English (en), Spanish (es), French (fr), Italian (it), Portuguese (pt) and Swedish (sv) in the latest release (version 1.2) of Universal Dependencies (Nivre et al., 2015a), and experiment with both gold and predicted POS tags. For experiments where the target language has no treebank, we use the standard splits for these languages in the older universal dependency treebanks v2.0 (McDonald et al., 2013) and use gold POS tags, following the baselines (Zhang and Barzilay, 2015; Guo et al., 2016). Table 1 gives the number of sentences and words annotated for each language in both versions. In a preprocessing step, we lowercase all tokens and remove multi-word annotations and language-specific dependency relations. We use the same multilingual Brown clusters and multilingual embeddings of Guo et al. (2016), kindly provided by the authors.

**Optimization.** We follow Dyer et al. (2015) in parameter initialization and optimization. However, we use stochastic gradient updates with an initial learning rate of \( \eta_0 = 0.1 \) in epoch \#0, update the learning rate in following epochs as \( \eta_t = \eta_0 / (1 + 0.1t) \). We clip the \( l_2 \) norm of the gradient to avoid “exploding” gradients. unlabeled attachment score (UAS) on the development set determines early stopping. Parameters are initialized with uniform samples in \( \pm \sqrt{6} / (r + c) \) where \( r \) and \( c \) are the sizes of the previous and following layer in the neural network (Glorot and Bengio, 2010). The standard deviations of the labeled attachment score (LAS) due to random initialization in individual target languages are 0.36 (de), 0.40 (en), 0.37 (es), 0.46 (fr), 0.47 (it), 0.41 (pt) and 0.24 (sv). The standard deviation of the average LAS scores across languages is 0.17.
Table 3: Dependency parsing: labeled attachment scores (LAS) for monolingually-trained parsers and MALOPA in the fully supervised scenario where \( L^t = L^s \). Note that we use the universal dependencies version 1.2 which only includes annotations for \( \sim 13K \) English sentences, which explains the relatively low scores in English. When we instead use the universal dependency treebanks version 2.0 which includes annotations for \( \sim 40K \) English sentences (originally from the English Penn Treebank), we achieve UAS score 93.0 and LAS score 91.5.

When training the parser on multiple languages in MALOPA, instead of updating the parameters with the gradient of individual sentences, we use minibatch updates which include one sentence sampled uniformly (without replacement) from each language’s treebank, until all sentences in the smallest treebank are used (which concludes an epoch). We repeat the same process in following epochs. We found this to help prevent one source language with a larger treebank (e.g., German) from dominating parameter updates, at the expense of other source languages with a smaller treebank (e.g., Swedish).

### 4.1 Target Languages with a Treebank

Here, we evaluate our MALOPA parser when the target language has a treebank.

**Baseline.** For each target language, the strong baseline we use is a monolingually-trained S-LSTM parser with a token representation which concatenates: pretrained word embeddings (50 dimensions), learned word embeddings (50 dimensions), coarse (universal) POS tag embeddings (12 dimensions), fine-grained (language-specific, when available) POS tag embeddings (12 dimensions), and embeddings of Brown clusters (12 dimensions), and uses a two-layer S-LSTM for each of the stack, the buffer and the list of actions. We independently train one baseline parser for each target language, and share no model parameters. This baseline, denoted ‘monolingual’ in Tables 3 and 7, achieves UAS score 93.0 and LAS score 91.5 when trained on the English Penn Treebank, which is comparable to Dyer et al. (2015).

**MALOPA.** We train MALOPA on the concatenation of training sections of all seven languages. To balance the development set, we only concatenate the first 300 sentences of each language’s development section.

**Token representations.** The first MALOPA parser we evaluate uses only coarse POS embeddings to construct the token representation. As shown in Table 3, this parser consistently underperforms the monolingual baselines, with a gap of 12.5 LAS points on average.

Augmenting the token representation with lexical embeddings to the token representation (both multilingual word clusters and pretrained multilingual word embeddings, as described in §3.3) substantially improves the performance of MALOPA, recovering 83% of the gap in average performance.

We experimented with three ways to include language information in the token representation, namely: ‘language ID’, ‘word order’ and ‘full typology’ (see §3.4 for details), and found all three to improve the performance of MALOPA giving LAS scores 83.5, 83.2 and 82.5, respectively. It is

---

\[18\]These embeddings are treated as fixed inputs to the parser, and are not optimized towards the parsing objective. We use the same embeddings used in Guo et al. (2016).

\[19\]We use the same number of dimensions for the coarse POS embeddings as in the monolingual baselines. The same applies to all other types of embeddings used in MALOPA.
Table 4: Recall of some classes of dependency attachments/relations in German.

| Recall % | left | right | root | short | long | nsubj* | dobj | conj | *comp | case | *mod |
|----------|------|-------|------|-------|------|--------|------|------|-------|------|------|
| monolingual | 89.9 | 95.2 | 86.4 | 92.9 | 81.1 | 77.3 | 75.5 | 66.0 | 45.6 | 93.3 | 77.0 |
| MALOPA | 85.4 | 93.3 | 80.2 | 91.2 | 73.3 | 57.3 | 62.7 | 64.2 | 34.0 | 90.7 | 69.6 |
| +lexical | 89.9 | 93.8 | 84.5 | 92.6 | 78.6 | 73.3 | 73.4 | 66.9 | 35.3 | 91.6 | 75.3 |
| +language ID | 89.1 | 94.7 | 86.6 | 93.2 | 81.4 | 74.7 | 73.0 | 71.2 | 48.2 | 92.8 | 76.3 |
| +fine-grained POS | 89.5 | 95.7 | 87.8 | 93.6 | 82.0 | 74.7 | 74.9 | 69.7 | 46.0 | 93.3 | 76.3 |

Table 5: Effect of automatically predicting language ID and POS tags with MALOPA on LAS scores.

| LAS | language ID | coarse POS | target language | average |
|-----|-------------|------------|-----------------|---------|
|     | gold | gold | de | en | es | fr | it | pt | sv |      |
| gold | gold | 78.6 | 84.2 | 83.4 | 82.4 | 89.1 | 84.2 | 82.6 | 83.5 |
| predicted | gold | 78.5 | 80.2 | 83.4 | 82.1 | 88.9 | 83.9 | 82.5 | 82.7 |
| gold | predicted | 71.2 | 79.9 | 80.5 | 78.5 | 85.0 | 78.4 | 75.5 | 78.4 |
| predicted | predicted | 70.8 | 74.1 | 80.5 | 78.2 | 84.7 | 77.1 | 75.5 | 77.2 |

noteworthy that the model benefits more from language ID than from typological properties. Using ‘language ID’, we recover another 12% of the original gap.

Finally, the best configuration of MALOPA adds fine-grained POS embeddings to the token representation. Surprisingly, adding fine-grained POS embeddings improves the performance even for some languages where fine-grained POS tags are not available (e.g., Spanish). This parser outperforms the monolingual baseline in five out of seven target languages, and wins on average by 0.3 LAS points. We emphasize that this model is only trained once on all languages, and the same model is used to parse the test set of each language, which simplifies the distribution or deployment of multilingual parsing software.

**Qualitative analysis.** To gain a better understanding of the model behavior, we analyze certain classes of dependency attachments/relations in German, which has notably flexible word order, in Table 4. We consider the recall of left attachments (where the head word precedes the dependent word in the sentence), right attachments, root attachments, short-attachments (with distance = 1), long-attachments (with distance > 6), as well as the following relation groups: nsubj* (nominal subjects: nsubj, nsubjpass), dobj (direct object: dobj), conj (conjunct: conj), *comp (clausal complements: ccomp, xcomp), case (clitics and adpositions: case), *mod (modifiers of a noun: nmod, nummod, amod, appos), neg (negation modifier: neg).

**Findings.** We found that each of the three improvements (lexical embeddings, language embeddings and fine-grained POS embeddings) tends to improve recall for most classes. Unfortunately, MALOPA underperforms (compared to the monolingual baseline) in some classes: nominal subjects, direct objects and modifiers of a noun. Nevertheless, MALOPA outperforms the baseline in some important classes such as: root, long attachments and conjunctions.

**Predicting language IDs and POS tags.** In Table 5 we assume that both gold language ID of the input language and the gold POS tags are given at test time. However, this assumption is not realistic in practical applications. Here, we quantify the degradation in parsing accuracy when language ID and POS tags are only given at training time, but must be predicted at test time. We do not use fine-grained POS tags were only available for English, Italian, Portuguese and Swedish. Other languages reuse the coarse POS tags as fine-grained tags instead of padding the extra dimensions in the token representation with zeros.

20Fine-grained POS tags were only available for English, Italian, Portuguese and Swedish. Other languages reuse the coarse POS tags as fine-grained tags instead of padding the extra dimensions in the token representation with zeros.

21For each group, we report recall of both the attachment and relation weighted by the number of instances in the gold annotation. A detailed description of each relation can be found at [http://universaldependencies.org/u/dep/index.html](http://universaldependencies.org/u/dep/index.html).
grained POS tags in these experiments because some languages use a very large fine-grained POS tag set (e.g., 866 unique tags in Portuguese).

In order to predict language ID, we use the `langid.py` library (Lui and Baldwin, 2012)
and classify individual sentences in the test sets to one of the seven languages of interest, using the default models included in the library. The macro average language ID prediction accuracy on the test set across sentences is 94.7%. In order to predict POS tags, we use the model described in §3.6 with both input and hidden LSTM dimensions of 60, and with block dropout. The macro average accuracy of the POS tagger is 93.3%. Table 5 summarizes the four configurations: {gold language ID, predicted language ID} × {gold POS tags, predicted POS tags}. The performance of the parser suffers mildly (-0.8 LAS points) when using predicted language IDs, but suffers significantly (-5.1 LAS points) when using predicted POS tags. As an alternative approach to predicting POS tags, we trained the Stanford POS tagger, for each target language, on the coarse POS tag annotations in the training section of the universal dependency treebanks, then replaced the gold POS tags in the test set of each language with predictions of the monolingual tagger. The resulting degradation in parsing performance between gold vs. predicted POS tags is -6.0 LAS points (on average, compared to a degradation of -5.1 LAS points in Table 5). The disparity in parsing results with gold vs. predicted POS tags is an important open problem, and has been previously discussed in Tiedemann (2015).

The predicted POS results in Table 5 use block dropout. Without using block dropout, we lose an extra 0.2 LAS points in both configurations using predicted POS tags.

**Different multilingual embeddings.** Several methods have been proposed for pretraining multilingual word embeddings. We compare three of them:

| multilingual embeddings | UAS | LAS |
|------------------------|-----|-----|
| multiCluster           | 87.7| 84.1|
| multiCCA               | 87.8| 84.4|
| robust projection      | 87.8| 84.2|

Table 6: Effect of multilingual embedding estimation method on the multilingual parsing with MAL-OPA. UAS and LAS scores are macro-averaged across seven target languages.

- multiCCA (Ammar et al., 2016) uses a linear operator to project pretrained monolingual embeddings in each language (except English) to the vector space of pretrained English word embeddings.
- multiCluster (Ammar et al., 2016) uses the same embedding for translationally-equivalent words in different languages.
- **robust projection** (Guo et al., 2015) first pre-trains monolingual English word embeddings, then defines the embedding of a non-English word as the weighted average embedding of English words aligned to the non-English words (in a parallel corpus). The embedding of a non-English word which is not aligned to any English words is defined as the average embedding of words with a unit edit distance in the same language (e.g., ‘playz’ is the average of ‘plays’ and ‘play’).

All embeddings are trained on the same data and use the same number of dimensions (100).

---

24Our implementation of this method can be found at https://github.com/gmulcaire/average-embeddings
25We share the embedding files at https://github.com/clab/language-universal-parser/tree/master
26The embeddings were kindly provided by the authors of Guo et al. (2016) at https://drive.google.com/file/d/0B1z04ix6j3DY3J1MN2NtgyQ/view
Table 8: Dependency parsing: labeled attachment scores (LAS) for multi-source transfer parsers in the simulated low-resource scenario where $L^t \cap L^s = \emptyset$.

| LAS                  | target language | average |
|----------------------|-----------------|---------|
|                      | de   | es   | fr   | it   | pt   | sv   |
| Zhang and Barzilay (2015) | 54.1 | 68.3 | 68.8 | 69.4 | 72.5 | 62.5 | 65.9 |
| Guo et al. (2016)     | 55.9 | 73.0 | 71.0 | 71.2 | 78.6 | 69.5 | 69.3 |
| MALOPA                | 57.1 | 74.6 | 73.9 | 72.5 | 77.0 | 68.1 | 70.5 |

Table 7: Small (3K token) target treebank setting: language-universal dependency parser performance.

| LAS                  | target language | average |
|----------------------|-----------------|---------|
|                      | de   | es   | fr   | it   | sv   |
| monolingual          | 58.0 | 64.7 | 63.0 | 68.7 | 57.6 |
| Duong et al.         | 61.8 | 70.5 | 67.2 | 71.3 | 62.5 |
| MALOPA               | 63.4 | 70.5 | 69.1 | 74.1 | 63.4 |

Small target treebank. **Duong et al. (2015b)** considered a setup where the target language has a small treebank of $\sim 3K$ tokens, and the source language (English) has a large treebank of $\sim 205K$ tokens. The parser proposed in **Duong et al. (2015b)** is a neural network parser based on **Chen and Manning (2014)**, which shares most of the parameters between English and the target language, and uses an $L_2$ regularizer to tie the lexical embeddings of translationally-equivalent words. While not the primary focus of this paper, we compare our proposed method to that of **Duong et al. (2015b)** on five target languages for which multilingual Brown clusters are available from **Guo et al. (2016)**. For each target language, we train the parser on the English training data in the UD version 1.0 corpus (Nivre et al., 2015b) and a small treebank in the target language. Following **Duong et al. (2015b)**, in this setup, we only use gold coarse POS tags, we do not use any development data in the target languages (we use the English development set instead), and we subsample the English training data in each epoch to the same number of sentences in the target language. We use the same hyperparameters specified before for the single MALOPA parser and each of the monolingual baselines. Table 7 shows that our method outperforms **Duong et al. (2015b)** by 1.4 LAS points on average. Our method consistently outperforms the monolingual baselines in this setup, with an average improvement of 5.7 absolute LAS points.

4.2 Target Languages without a Treebank
($L^t \cap L^s = \emptyset$)

**McDonald et al. (2011)** established that, when no treebank annotations are available in the target language, training on multiple source languages outperforms training on one (i.e., multi-source model transfer outperforms single-source model transfer). In this section, we evaluate the performance of our parser in this setup. We use two strong baseline multi-source model transfer parsers with no supervision in the target language:

- **Zhang and Barzilay (2015)** is a graph-based arc-factored parsing model with a tensor-based scoring function. It takes typological properties of a language as input. We compare to the best reported configuration (i.e., the column titled “OURS” in Table 5 of Zhang and Barzilay, 2015).

- **Guo et al. (2016)** is a transition-based neural-network parsing model based on **Chen and Manning (2014)**. It uses a multilingual embeddings and Brown clusters as lexical features. We compare to the best reported configuration (i.e., the column titled “MULTI-PROJ” in Table 1 of Guo et al., 2016).

Following **Guo et al. (2016)**, for each target language, we train the parser on six other languages at

https://github.com/longdt219/universal_dependency_parser/tree/master/data/universal-dependency
the Google universal dependency treebanks version 2 (de, en, es, fr, it, pt, sv, excluding whichever is the target language), and we use gold coarse POS tags. Our parser uses the same word embeddings and word clusters used in Guo et al. (2016), and does not use any typology information.

The results in Table 8 show that, on average, our parser outperforms both baselines by more than 1 point in LAS, and gives the best LAS results in four (out of six) languages.

### 5 Related Work

Our work builds on the model transfer approach, which was pioneered by Zeman and Resnik (2008), who trained a parser on a source language treebank then applied it to parse sentences in a target language. Cohen et al. (2011) and McDonald et al. (2011) trained unlexicalized parsers on treebanks of multiple source languages and applied the parser to different languages. Naseem et al. (2012), Täckström et al. (2013), and Zhang and Barzilay (2015) used language typology to improve model transfer. To add lexical information, Täckström et al. (2012) used multilingual word clusters, while Xiao and Guo (2014), Guo et al. (2015), Søgaard et al. (2015) and Guo et al. (2016) used multilingual word embeddings. Duong et al. (2015b) used a neural network based model, sharing most of the parameters between two languages, and used an $L_2$ regularizer to tie the lexical embeddings of translationally-equivalent words. We incorporate these ideas in our framework, while proposing a novel neural architecture for embedding language typology (see §3.4), and use a variant of the word dropout (Iyyer et al., 2015) for consuming noisy structured inputs. We also show how to replace an array of monolingually trained parsers with one multilingually-trained parser without sacrificing accuracy, which is related to Vilares et al. (2015).

Neural network parsing models which preceded Dyer et al. (2015) include Henderson (2003), Titov and Henderson (2007), Henderson and Titov (2010) and Chen and Manning (2014). Related to lexical features in cross-lingual parsing is Durrett et al. (2012) who defined lexico-syntactic features based on bilingual lexicons. Other related work include Östling (2015), which may be used to induce more useful typological to inform multilingual parsing.

Another popular approach for cross-lingual supervision is to project annotations from the source language to the target language via a parallel corpus (Yarowsky et al., 2001; Hwa et al., 2005) or via automatically-translated sentences (Tiedemann et al., 2014). Ma and Xia (2014) used entropy regularization to learn from both parallel data (with projected annotations) and unlabeled data in the target language. Rasooli and Collins (2015) trained an array of target-language parsers on fully annotated trees, by iteratively decoding sentences in the target language with incomplete annotations. One research direction worth pursuing is to find synergies between the model transfer approach and annotation projection approach.

### 6 Conclusion

We presented MALOPA, a single parser trained on a multilingual set of treebanks. We showed that this parser, equipped with language embeddings and fine-grained POS embeddings, on average outperforms monolingually-trained parsers for target languages with a treebank. This pattern of results is quite encouraging. Although languages underlying share syntactic properties, the individual parsing models must behave, on the surface, quite differently, and our model has the ability to do this while sharing parameters across languages. The value of this sharing is more pronounced in scenarios where the target language’s training treebank is small or non-existing, where our parser outperforms previous cross-lingual multi-source model transfer methods.

### Acknowledgments

Waleed Ammar is supported by the Google fellowship in natural language processing. Miguel Ballesteros is supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA). Part of this material is based upon work supported by a subcontract with Raytheon.
BBN Technologies Corp. under DARPA Prime Contract No. HR0011-15-C-0013. We thank Jiangan Guo for sharing the multilingual word embeddings and multilingual word clusters. We thank Lori Levin, Ryan McDonald, Jörg Tiedemann, Yulia Tsvetkov, and Yuan Zhang for helpful discussions. Last but not least, we thank the anonymous TACL reviewers for their valuable feedback.

References

[Agić et al.2015] Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richard Farkas, Jennifer Foster, Filip Ginter, Iakes Gognaga, Koldo Gojenola, Yoav Goldberg, Jan Hajic, Anders Trerup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Kristin Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missiliä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Pietulainen, Barbara Plank, Prokopis Pyysalo, Wolfgang Sicker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

[Ammar et al.2016] Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. In Arxiv.

[Barman et al.2014] Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In EMNLP 2014 workshop on computational approaches to code switching.

[Bender2011] Emily Bender. 2011. On achieving and evaluating language-independence in NLP. In Linguistic Issues in Language Technology. Special Issue on Interaction of Linguistics and Computational Linguistics.

[Buchholz and Marsi2006] Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In CoNLL-X.

[Chen and Manning2014] Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In Proc. of EMNLP.

[Cohen et al.2011] Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In Proc. of EMNLP.

[Duong et al.2015a] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015a. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In Proc. of ACL-IJCNLP.

[Duong et al.2015b] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015b. A neural network model for low-resource universal dependency parsing. In Proc. of EMNLP.

[Durrett et al.2012] Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In Proc. of EMNLP.

[Dyer et al.2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In Proc. of ACL.

[Gardner et al.2015] Matt Gardner, Kejun Huang, Evangelos Papalexakis, Xiao Fu, Partha Talukdar, Christos Faloutsos, Nicholas Sidiropoulos, and Tom Mitchell. 2015. Translation invariant word embeddings. In Proc. of EMNLP.

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proc. of AISTATS.

[Graves et al.2013] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In Proc. of ICASSP.

[Guo et al.2015] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In Proc. of ACL.

[Guo et al.2016] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In Proc. of AAAI.

[Heid and Raab1989] Ulrich Heid and Sybille Raab. 1989. Collocations in multilingual generation. In Proc. of EACL.

[Henderson and Titov2010] James Henderson and Ivan Titov. 2010. Incremental sigmoid belief networks for grammar learning. JMLR.

[Henderson2003] James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In Proc. of NAACL-HLT.

[Huang et al.2015] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. In arXiv preprint arXiv:1508.01991.
[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In Proc. of JMLR.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proc. of NIPS.

[Täckström et al.2012] Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In Proc. of NAACL-HLT.

[Täckström et al.2013] Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. In Proc. of TACL.

[Tiedemann et al.2014] Jörg Tiedemann, Zeljko Agic, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In Proc. of CoNLL.

[Tiedemann2015] Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In Proc. of Depling.

[Titov and Henderson2007] Ivan Titov and James Henderson. 2007. Constituent parsing with incremental sigmoid belief networks. In Proc. of ACL.

[Vilares et al.2015] David Vilares, Miguel A Alonso, and Carlos Gómez-Rodríguez. 2015. One model, two languages: training bilingual parsers with harmonized treebanks. arXiv preprint arXiv:1507.08449.

[Wikipedia2016] Wikipedia. 2016. List of languages by number of native speakers. http://bit.ly/1LUP5kJ. Accessed: 2016-01-26.

[Xiao and Guo2014] Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In Proc. of CoNLL.

[Yarowsky et al.2001] David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In Proc. of HLT.

[Zeman and Resnik2008] Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In Proc. of IJCNLP.

[Zhang and Barzilay2015] Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In Proc. of EMNLP.