Cross-lingual CCG Induction

Kilian Evang
University of Düsseldorf
Germany
evang@hhu.de

Abstract

Combinatory categorial grammars are linguistically motivated and useful for semantic parsing, but costly to acquire in a supervised way and difficult to acquire in an unsupervised way. We propose an alternative making use of cross-lingual learning: an existing source-language parser is used together with a parallel corpus to induce a grammar and parsing model for a target language. On the PASCAL benchmark, cross-lingual CCG induction outperforms CCG induction from gold-standard POS tags on 3 out of 8 languages, and unsupervised CCG induction on 6 out of 8 languages. We also show that cross-lingually induced CCGs reflect known syntactic properties of the target languages.

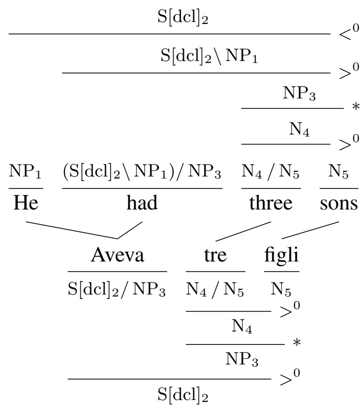


Figure 1: Projection of an English CCG derivation to an Italian translation. The indices distinguish different instances of categories.

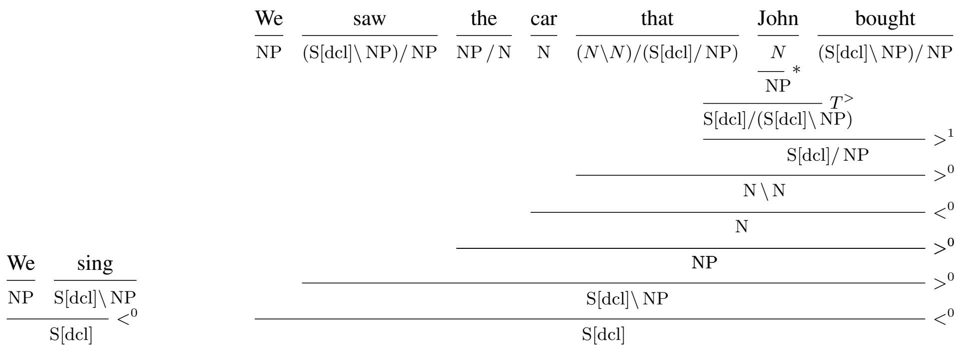
1 Introduction

Combinatory Categorial Grammar (CCG) (Steedman, 2001) is a grammar formalism known for its linguistic elegance and computational efficiency. It has been successfully used for statistical syntactic parsing (Clark and Curran, 2004; Lewis et al., 2016) and has emerged as a leading grammar formalism in semantic parsing (Curran et al., 2007; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011, 2013; Reddy et al., 2014; Artzi et al., 2015; Beschke and Menzel, 2018). Semantic parsing is important because it translates natural language utterances to something that a computer can understand, e.g., database queries, computer commands, or logical formulas, enabling next-generation information systems and knowledge extraction from text, among other applications.

CCGs used in most work to date are either hand-crafted (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2013; Artzi et al., 2015) or extracted from large syntactically annotated corpora (Curran et al., 2007; Reddy et al., 2014). In either case language-specific human effort is required. Acquiring CCGs in an unsupervised way is difficult and does not reach the performance of supervised methods (Bisk and Hockenmaier, 2013). As a result, most research focuses on English and other languages are neglected, meaning that speakers of other languages have delayed or no access to CCG-based semantic parsing technology.

We propose to overcome this bottleneck by inducing CCGs cross-lingually, i.e., transferring an existing grammar from English to other languages via unannotated parallel data. The process is illustrated for one English-Italian sentence pair in Figure 1: the English sentence is parsed by an existing CCG parser and word-aligned to the Italian sentence. Italian words receive categories equivalent to those of the aligned English words, and a semantically equivalent derivation is built for the Italian sentence. With enough derivations projected in this way, they can be used to extract a CCG lexicon and to estimate parameter weights for parsing the target language.

Proceedings of NAACL-HLT 2019, pages 1577–1587
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) (b)

Figure 2: Two examples of CCG derivations.

Unlike previous competitive methods for CCG induction such as Bisk and Hockenmaier (2013), our method does not require the training data to be POS-tagged. It also induces more fine-grained labels. In this paper, we compare the performance of parsers trained using our method to previous induced CCG parsers. We also investigate whether the cross-lingually induced CCG lexicons correspond with linguistic insights about the target languages.

2 Combinatory Categorial Grammar

In categorial grammars (Bar-Hillel, 1953), words and larger constituents share a single space of labels, called *categories*. For example, the intransitive verb *sing* in Figure 2(a) and the verb phrase *saw the car that John bought* in Figure 2(b) have the same category: $\mathrm{S[dcl]\backslash NP}$. Parse trees are conventionally called *derivations* and their nodes depicted as horizontal lines, placed underneath their children.

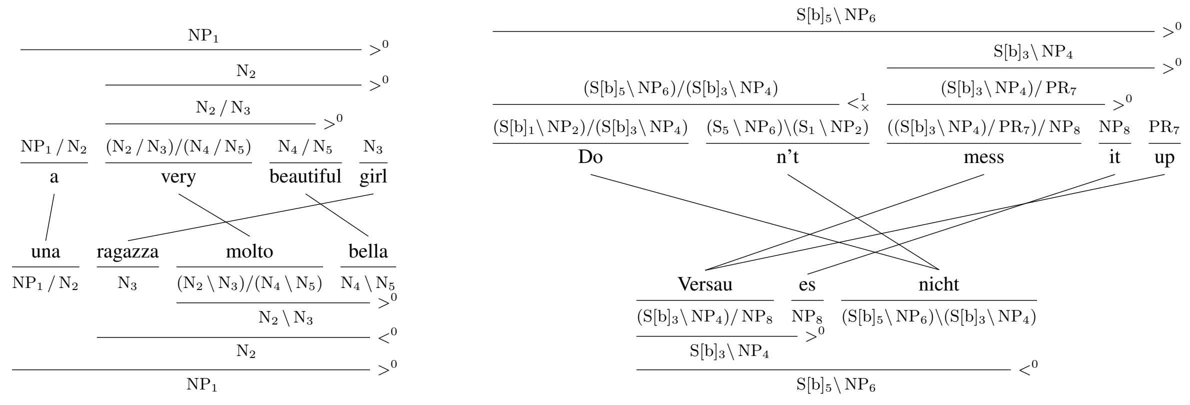
Categorial grammars have only few *basic categories*, typically: N for nouns, NP for noun phrases, PP for argument prepositional phrases, PR for verb particles, and $\mathrm{S}[X]$ for sentences, where X is a feature that indicates the type of sentence or clause, e.g., dcl for declarative sentences or b for infinitives. All other categories are *functional categories*, which contain information about what kinds of arguments constituents with these categories combine with, and what kinds of constituents result. For example, in English, a declarative verb phrase is a constituent that combines with a noun phrase (the subject) to its left to form a declarative sentence. This is expressed by its category: $\mathrm{S[dcl]\backslash NP}$. Similarly, a transitive verb is a constituent that combines with a noun phrase (the object) to its right to form a verb phrase. This results in the functional category $\mathrm{(S[dcl]\backslash NP)/NP}$ for a transitive verb, where the brackets determine the order in which it combines with its arguments.

With such expressive categories, categorial grammars are mainly defined via the lexicon, i.e., which words are associated with which categories. Only few and very general rules are needed to specify how constituents may combine. The basic rules are forward application and backward application ($>^0$, $<^0$). They allow a constituent with a functional category to combine with its argument. Combinatory categorial grammar adds type raising ($T^>$, $T^<$) and generalizes application to (harmonic and crossing) composition ($>^1$, $<^1$, $>^2$, $<^2$...). This allows for dealing with "incomplete" constituents such as the object relative clause *John bought* in Figure 2(b). The object is extracted, thus the transitive verb *bought* cannot combine with the NP it expects to its right. Thanks to type raising and composition, it can nevertheless combine with its subject, resulting in a sentence with an open object argument slot ($\mathrm{S[dcl]/NP}$), which is taken as an argument by the relative pronoun *that*.

Additionally, some unary *type changing* ($*$) rules are used to convert categories, e.g., $\mathrm{N} \Rightarrow \mathrm{NP}$ to convert N to NP when there is no determiner.

3 Derivation Projection

Examples of derivations projected from English to other languages are shown in Figures 1 and 3. Note that we give basic categories indices here to distinguish different instantiations of the same category. For the purposes of derivation projection, different instantiations are treated as different cat-



(a) (b)

Figure 3: Projections of English CCG derivations to Italian and German translations.

egories to ensure that projected derivations are semantically equivalent to the input derivations (e.g., $\mathrm{N}_2 / \mathrm{N}_3 \neq \mathrm{N}_4 / \mathrm{N}_5$).

We now describe our derivation projection algorithm. Given a source derivation, a target sentence, and a word alignment, it attempts to produce a target derivation. Note that target derivations are entirely derived from the data by the algorithm; we do not make use of any hand-crafted language-specific rules.

Input The input to derivation projection consists of a source sentence E with a derivation D_E, a target sentence F which is a translation of E, and a (potentially ambiguous) alignment $\mathcal{A}$ which is a set of 1:N translation units $\langle\langle f\rangle, \mathbf{e}\rangle$ where f is a token in F and $\mathbf{e}$ is a subsequence (not necessarily contiguous) of tokens in E, as well as translation units $\langle\langle\rangle, \langle e\rangle\rangle$, indicating that the English word e is not aligned.

Output Derivation projection may succeed or fail; if it succeeds, the output is a derivation D_F for F.

Auxiliary Definitions $\mathcal{C}$ is the set of all categories. A *category assignment* c for a sequence of tokens $\mathbf{t}$ is a relation such that $c \subseteq \mathbf{t} \times \mathcal{C}$.[1] We write c_E for the category assignment relating tokens in E to the categories they have in D_E; this relation is a function. We write R^*_E for the set of type-changing rules used in D_E. We write $\textsc{RootCat}(D)$ for the category of the root of a derivation D. $\textsc{Parse}$ is a function that takes a sequence of tokens $\mathbf{t}$, a category assignment c for

[1] In a slight abuse of notation, we treat sequences of tokens as sets of tokens when convenient.

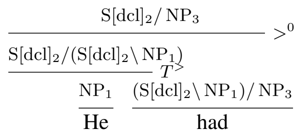


Figure 4: Two source-language categories are merged into one.

$\mathbf{t}$, and a set of type-changing rules R^*. It returns the set of all normal-form CCG derivations (Hockenmaier and Bisk, 2010) that can be built over $\mathbf{t}$ using R^*, forward/backward type raising and harmonic/crossing composition up to degree 2, with possible lexical categories determined by c. To deal with parsing ambiguity during derivation projection, we assume a function $\textsc{Choose}$ that takes a non-empty set of derivations and returns one element. We will say more about it below.

Step 1: Transfer Categories This step assigns categories to the words in F based on the categories of aligned words in E. This is straightforward for 1:1 translation units such as $\langle$tre, three$\rangle$, but 1:N translation units such as $\langle$Aveva, He had$\rangle$ need a bit more care. We define $\textsc{Merge}$ as a partial function from subsequences of E to $\mathcal{C}$. For a single-token subsequence $e \in E$, $\textsc{Merge}(e) = c_E(e)$. For a longer subsequence $\mathbf{e}$, $\textsc{Merge}(\mathbf{e}) = \textsc{RootCat}(\textsc{Choose}(\textsc{Parse}(\mathbf{e}, c_E, R^*_E)))$ (if defined). For example, even though *He had* is not a constituent in Figure 1, it has a parse (shown in Figure 4), and so $\textsc{Merge}(\text{He had}) = \mathrm{S[dcl]}_2 / \mathrm{NP}_3$. We then define a preliminary category assignment for F: $c_F = \{\langle f, \textsc{Merge}(\mathbf{e})\rangle | \langle\langle f\rangle, \mathbf{e}\rangle \in \mathcal{A}, \textsc{Merge}(\mathbf{e})$ is defined$\}$.

Step 2: Transfer Type-changing Rules This step creates a set R^*_F of type-changing rules to be used in D_F. In addition to the type-changing rules used in D_E, we add $\mathrm{N} \Rightarrow \mathrm{NP}$ rules for English determiners that have no corresponding token in the target language. This is a common occurrence, especially with languages which have no articles, such as Czech, or where (some) articles are affixes rather than separate words, such as Swedish. Thus, $R^*_F = R^*_E \cup \{\mathrm{N}_i \Rightarrow \mathrm{NP}_j \,|\, \langle\langle\rangle, \langle e\rangle\rangle \in \mathcal{A}, c_E(e) = \mathrm{NP}_i \,/\, \mathrm{N}_j$ for some $i, j\}$.

Step 3: Flip Slashes This step adapts the directionality of slashes in the assigned categories, because the word order may be different in F than in E. We say that a category C' is a *flip variant* of category C ($\textsc{Flip}(C, C')$) if it is the same as C, except that slashes may lean a different way, as long as subcategories that are modifier categories in C (i.e., are of form X/X or $X\backslash X$, ignoring indices) remain so in C'. For example, in Figure 3(a), the category $(\mathrm{N}_2 \,/\, \mathrm{N}_3)/(\mathrm{N}_4 \,/\, \mathrm{N}_5)$ has a flip variant $(\mathrm{N}_2 \,\backslash\, \mathrm{N}_3)/(\mathrm{N}_4 \,\backslash\, \mathrm{N}_5)$ whereas $(\mathrm{N}_2 \,\backslash\, \mathrm{N}_3)/(\mathrm{N}_4 \,/\, \mathrm{N}_5)$ is not a flip variant because that would destroy the modifier status. In order to be able to construct a derivation for F even with word order different from E, we define a new category assignment: $c'_F = \{\langle f, C'\rangle | \langle f, C\rangle \in c_F, \textsc{Flip}(C, C')\}$. Similarly, we construct a set of type-changing rules with flip variants: $R^{*\prime}_F = \{X' \Rightarrow Y' | X \Rightarrow Y \in R^*_F, \textsc{Flip}(X, X'), \textsc{Flip}(Y, Y')\}$. This constructs more categories and type-changing rules than needed; for example, $(\mathrm{N}_2 \,/\, \mathrm{N}_3)\backslash(\mathrm{N}_4 \,/\, \mathrm{N}_5)$ is a flip variant for *molto* that cannot be used, as the argument category $\mathrm{N}_4 \,/\, \mathrm{N}_5$ does not appear on the left. Such spurious categories are discarded automatically in our implementation.

Step 4: Construct Derivation With c'_F and $R^{*\prime}_F$ constructed, we try to find a parse for F that has the same root category as D_E: $D_F = \textsc{Choose}(\{D | D \in \textsc{Parse}(F, c'_F, R^{*\prime}_F), \textsc{RootCat}(D) = \textsc{RootCat}(D_E)\})$ if defined; otherwise derivation projection fails and no derivation is returned.

Resolving Ambiguity Since parsing in steps 1 and 4 of derivation projection is guided by indexed categories and normal-form constraints, ambiguity primarily arises through ambiguous word alignments, which we use to achieve better projection coverage (see Section 5). For example, in Figure 1, *tre* might also be aligned to *sons*, and *three* to *figli*, giving rise to an additional (incorrect) parse. Our strategy for resolving such ambiguities is to prefer parses whose lexical categories result from word alignments with higher alignment scores. Our current implementations of PARSE and CHOOSE naively order parses by the score of the alignment that produced each lexical target category, greedily from left to right. Future work might improve upon this by ranking parses according to a global score.

4 The Learning Procedure

Given a parallel training corpus of source-target sentence pairs, we parse the source-language part using a source-language parser and run unsupervised word alignment on the entire corpus. Then, for each sentence pair, we run derivation projection using the generated source parses and alignments. If successful, we add the target derivation picked by CHOOSE to a target-language training set. Finally, we use this training set to train a target-language parser in the usual way.

5 Experiments[2]

Target Languages Following prior work, we evaluate the induced CCG parsers in terms of unlabeled attachment score (UAS) on the data of the PASCAL unsupervised grammar induction challenge (Gelling et al., 2012), which includes eight different languages other than English: Arabic, Czech, Danish, Basque, Dutch, Portuguese, Slovenian, and Swedish. For qualitative evaluation, we use German, Italian, and Dutch. We acknowledge the importance of testing our approach on a more typologically diverse range of languages, but leave this for future work.

Training Data To start learning to parse a new language, one needs short and simple example sentences. This is true for human learners, and presumably also for computers. We therefore used the Tatoeba corpus[3] for training, a multilingual parallel corpus gathered by volunteers and aimed at language learners. We extracted English-X sentence pairs for various languages X and tokenized

[2]The training data, code, and configurations are available at https://github.com/texttheater/xlci.

[3]https://tatoeba.org

Parallel corpus	sentences	∅ tokens
eng-ara	19,502	5.8
eng-ces	11,147	6.2
eng-dan	21,409	7.1
eng-deu	244 140	8.1
eng-eus	1,882	6.4
eng-ita	412 427	6.5
eng-nld	44 126	7.5
eng-por	161 126	7.2
eng-slv	835	6.3
eng-swe	24 206	6.4

Table 1: Number of sentence pairs and average number of tokens per target-language sentence in the data extracted from Tatoeba.

them using UDPipe (Straka and Straková, 2017), not making use of the optional multiword token subdivision feature. The resulting parallel corpora are summarized in Table 1.

Source-language Parser To create derivations to project, we needed a suitable parser for our source language, English. Commonly, English CCG parsers are trained on CCGbank (Hockenmaier and Steedman, 2007) or its derivative CCGrebank (Honnibal et al., 2010). However, these treebanks use special categories for punctuation and conjunctions, which would complicate derivation projection. We thus took CCGrebank, automatically transformed it to use normal categories for these cases (an example is shown in Figure 5), and trained the EasyCCG parser (Lewis and Steedman, 2014) on that. The resulting model was used to produce parses for the English portions of our parallel training corpora.

Word Alignments and Derivation Projection For word-aligning the parallel training data, we used GIZA++ with default settings (Och and Ney, 2003). We generated alignments $\mathcal{A}$ for each sentence pair by taking the union of the n-best GIZA++ alignments, trying out different values for n between 1 and 5.

Target-language Parser Again, we used EasyCCG. Its supertagger component is trained on sentences where the words are annotated with categories. We used the projected derivations for that. We used the Polyglot word embeddings (Al-Rfou et al., 2013). Since we do not have supertagged validation sets for the target languages, the number of training epochs was fixed at 3 following initial experimentation. The parser component requires no training, but for decoding, we made some modifications to it to generalize beyond English: instead of a hard-coded set for English, the modified parser uses the set of unary rules used in the projected derivations for the respective language. It also implements all composition rules up to degree 2 rather than an English-specific subset, and it implements Hockenmaier and Bisk's normal-form constraints.

Dependency Conversion For evaluating the induced target-language parsers on the PASCAL benchmark, we have to be able to convert their output derivations to dependency trees, as exemplified in Figure 6. The simplest way to do this is to make arguments dependents of their functors, similar to Koller and Kuhlmann (2009). That is, a word v with the (indexed) category $X|Y\alpha$

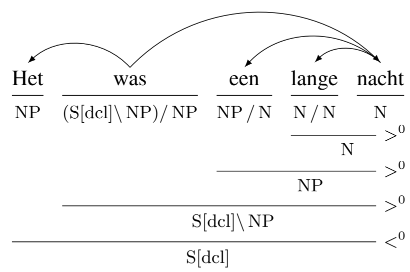


Figure 6: An example derivation and its conversion into a dependency tree.

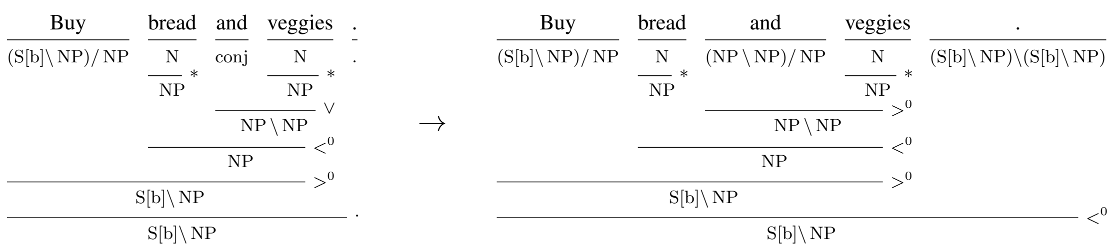


Figure 5: Elimination of special categories and rules for punctuation and coordination.

categories	description	ara	ces	dan	eus	nld	por	slv	swe
$X\|X$	modifier	✓	✓	✓	✓	✓	✓	✓	✓
$\mathrm{NP}\,\|\,\mathrm{N}, \mathrm{NP}\,\|(\mathrm{N}\,\|\,\mathrm{PP})$	determiner	✓	✓		✓	✓	✓	✓	✓
$\{(\mathrm{S}\,\|\,\mathrm{S}), (\mathrm{S}\,\|\,\mathrm{NP})\|(\mathrm{S}\,\|\,\mathrm{NP})\}\|$ $\{\mathrm{S[dcl]}, \mathrm{S[to]}, (\mathrm{S[ng]}\|\,\mathrm{NP})\}$	subordinating conjunction						✓	✓	✓
$\mathrm{S[em]}\|\,\mathrm{S[dcl]},$ $(\mathrm{S[to]}\|\,\mathrm{NP})\|(\mathrm{S[b]}\|\,\mathrm{NP})$	complementizer						✓	✓	✓
$\{\mathrm{N}\,\|\,\mathrm{N}, \mathrm{NP}\,\|\,\mathrm{NP}\}\|(\mathrm{S[dcl]}\|\,\mathrm{NP})$	relative pronoun						✓	✓	✓
$\{\mathrm{PP}, \mathrm{N}\,\|\,\mathrm{N}, \mathrm{NP}\,\|\,\mathrm{NP}, \mathrm{S}\,\|\,\mathrm{S},$ $(\mathrm{S}\,\|\,\mathrm{NP})\|(\mathrm{S}\,\|\,\mathrm{NP})\}\|\,\mathrm{NP}$	adposition							✓	
$\mathrm{S[dcl]}\|\,\mathrm{S[b]},$ $\mathrm{S[\{b,dcl,ng,pt\}]}\|\,\mathrm{S[ng]},$ $\mathrm{S[\{b,dcl,ng,pt\}]}\|\,\mathrm{S[pt]}$	auxiliary verb		✓					✓	

Table 2: Functional categories which in dependency conversion become dependents of their first argument, rather than the other way around, depending on the treebank-specific conventions. Braces denote alternatives.

becomes the head of a word w with category $Y\beta$, where $| \in \{/, \backslash\}$ and α, β stand for any number of additional argument categories with slashes. However, for some categories the head-dependent relation should be inverted. For example, if $X|Y$ is a modifier category, then w becomes the head of v, and any dependents v would get because of additional arguments in X become dependents of w instead. Because dependency treebanks differ in their conventions for attaching certain function words, certain non-modifier categories also need to be treated in this inverted way. They are shown in Table 2. Note that this fine-grained control is only possible because we induce relatively rich CCG categories; by contrast, Bisk and Hockenmaier (2013) use only two basic categories (S and N) and therefore cannot distinguish, e.g., determiners from attributive adjectives (N / N) or *to*-complementizers from auxiliary verbs $((\mathrm{S}\setminus\mathrm{N})/(\mathrm{S}\setminus\mathrm{N}))$. They do apply treebank-specific conversion rules for coordination, which we also implement.

Hyperparameter Tuning We use the PASCAL development data to tune the hyperparameter n which controls how many GIZA++ alignments are used for derivation projection. Table 3 shows how many sentence pairs our parallel training corpus contains for each of the eight languages, how many of the derivations are successfully projected

language sentence pairs		ara 19 502	ces 11 147	dan 21 409	eus 1 882	nld 44 026	por 161 126	slv 835	swe 24 206
$n = 1$	projected	27.4%	30.5%	49.8%	20.6%	36.3%	30.8%	32.7%	48.8%
	ambiguity	1.029	1.044	1.011	1.111	1.046	1.015	1.040	1.014
	UAS	45.9%	43.6%	61.6%	18.4%	65.7%	**64.8%**	26.9%	**65.0%**
$n = 2$	projected	33.6%	36.6%	52.2%	23.8%	40.0%	34.7%	38.4%	52.3%
	ambiguity	1.252	1.230	1.169	1.266	1.092	1.075	1.215	1.143
	UAS	**46.3%**	45.7%	61.2%	25.6%	**65.9%**	64.2%	28.2%	63.2%
$n = 3$	projected	38.1%	40.4%	53.4%	26.0%	41.6%	37.1%	41.8%	54.2%
	ambiguity	1.379	1.364	1.226	1.325	1.118	1.114	1.289	1.193
	UAS	35.8%	**46.4%**	**62.5%**	24.8%	64.3%	63.0%	29.0%	64.6%
$n = 4$	projected	41.8%	43.4%	54.3%	28.9%	42.7%	39.2%	43.6%	55.6%
	ambiguity	1.484	1.474	1.269	1.397	1.142	1.152	1.352	1.232
	UAS	38.1%	45.3%	60.0%	26.1%	65.0%	62.0%	**32.2%**	63.1%
$n = 5$	projected	45.2%	45.9%	55.0%	31.3%	43.8%	41.2%	46.0%	57.0%
	ambiguity	1.592	1.583	1.318	1.461	1.174	1.207	1.409	1.278
	UAS	33.9%	45.8%	60.4%	**27.4%**	64.4%	61.8%	30.2%	63.7%

Table 3: Effects of varying the projection hyperparameter n: percentage of successfully projected source derivations, mean ambiguity (how many target derivations are found per projected source derivation), and UAS of the trained system on the PASCAL development data (max sentence length 15, not counting punctuation).

		ara	cze	dan	eus	nld	por	slv	swe
train tokens (PASCAL)		5 470	436 126	25 341	81 345	78 737	158 648	54 032	61 877
system	input								
BH13 (publ.)	gold POS	**65.1%**	**50.7%**	58.5%	**45.0%**	54.4%	62.9%	**46.4%**	**66.9%**
BH13 (repl.)	gold POS	45.4%	38.3%	25.1%	37.3%	54.9%	51.0%	41.8%	63.2%
BCH15 (publ.)	raw text	43.7%	32.4%	37.7%	35.2%	43.8%	51.6%	23.6%	52.9%
train tokens (Tatoeba)		19 502	11 147	21 409	1 882	44 026	161 126	835	24 206
system	input								
EB16	parallel, POS	26.4%	28.4%	35.8%	22.1%	40.4%	39.4%	27.2%	26.2%
ours	parallel, embeddings	46.8%	44.9%	**63.0%**	29.0%	**61.4%**	**67.8%**	35.0%	63.7%

Table 4: UAS of different systems on the PASCAL test data (max sentence length 15, not counting punctuation).

for each value of n, and how accurately the development data is parsed. The numbers show the importance of having enough training examples: Portuguese, Swedish, Dutch, and Danish are leading in terms of corpus size and parsing accuracy, whereas Basque and Slovene are far behind in both. Arabic is a bit of an outlier, performing worse than Czech despite a considerably larger corpus. The ratio of successfully projected derivations increases as n is increased. This makes for more training data but also more noise; different languages peak at different values for n. Languages with little training data (Slovene, Basque, Czech) most clearly profit from more projected derivations. For the final tests, we set $n \leq 5$ to maximize UAS on the development data for each language.

Baselines We compare with two unsupervised CCG induction system and one other cross-lingual CCG induction system. To our knowledge, Bisk and Hockenmaier (2013) represents the state of the art in unsupervised CCG induction. It does, however, use gold-standard POS tags in the training and testing data. These seem to be essential, as a variant of this system which does not rely on POS tags performed much worse (Bisk et al., 2015). Our system does not rely on POS tags but on parallel data and word embeddings instead, which is an advantage as parallel data and word embeddings may be more readily available than POS tags for new languages. We also compare with the system of Evang and Bos (2016), a cross-lingual system similar to ours which was previously only evaluated on a semantic parsing task, not on syntactic dependencies. For the unsupervised systems, we report published results when trained on the complete PASCAL data. For BH13, we also include our best replication attempt using the original software and training data, falling short of the published results as the exact configurations appear to be lost. For the cross-lingual systems which require parallel training data, we train on the Tatoeba dataset. All test scores are on the PASCAL test set, limited to sentences with at most 15 tokens, not counting punctuation.

Results Test results are shown in Table 4. Despite not using POS tags, our system outperforms the cross-lingually supervised system of Evang and Bos (2016) by a large margin on all languages. It also outperforms the unsupervised system of Bisk et al. (2015) on 6 out of 8 languages, and that of Bisk and Hockenmaier (2013) (which uses POS tags) on 3 out of 8 languages. This is also in spite of these two unsupervised systems being trained on more (albeit not parallel) data, which even included the test data.

6 The Induced Lexicons

Have our cross-lingually trained parsers acquired language-specific knowledge? Based on what we know about the syntactic differences between English, German, Italian, and Dutch, we would expect certain categories to be more prominent in the lexicon for some languages than for others:

1. English word order in transitive clauses is almost always SVO, whereas for German and Dutch, SVO is the typical order for main clauses, and SOV the typical order for subordinate clauses (Dryer, 2013c). Thus, we expect the English parser to almost always assign category $(\mathrm{S[X]} \backslash \mathrm{NP}) / \mathrm{NP}$ to transitive verbs, whereas we expect German and Dutch transitive verbs to be split between $(\mathrm{S[X]} \backslash \mathrm{NP}) / \mathrm{NP}$ and $(\mathrm{S[X]} \backslash \mathrm{NP}) \backslash \mathrm{NP}$.

	Category	English	German	Italian	Dutch
1	$(S[dcl]\backslash NP)/NP$	.0366	.0445	.0256	.0389
	$(S[dcl]\backslash NP)\backslash NP$	.0000	.0056	.0046	.0061
	$(S[b]\backslash NP)/NP$	.0284	.0032	.0147	.0044
	$(S[b]\backslash NP)\backslash NP$	.0000	.0169	.0043	.0151
2	$(S[dcl]\backslash NP)/(S[b]\backslash NP)$	.0237	.0184	.0150	.0180
3	$(S[b]\backslash NP)\backslash PR$	.0000	.0000	.0000	.0000
	$(S[b]\backslash NP)/PR$	.0004	.0000	.0000	.0000
4	N/N	.0309	.0299	.0213	.0316
	$N\backslash N$	.0013	.0018	.0099	.0018
	$(N/N)/(N/N)$	.0016	.0018	.0003	.0012
	$(N\backslash N)/(N\backslash N)$	.0001	.0000	.0008	.0000
5	$S[dcl]$	.0000	.0000	.0012	.0001
	$S[dcl]/NP$	.0004	.0013	.0115	.0007

Table 5: Frequency (per sentence) of lexical categories in the output of different parsers when applied to the Tatoeba data, illustrating learned language-specifics. The averages for English are calculated over all three parallel training corpora.

2. German, Italian and Dutch do not have *do*-support for negation (Miestamo, 2013), so we expect the category $(S[dcl]\backslash NP)/(S[b]\backslash NP)$ to be less common in them than in English.

3. In the infinitive mood, German and Dutch spell particle verbs as one token (e.g., *ausgehen*, *uitgaan*), unlike English which spells them apart (*go out*) (Dehé, 2015). Thus, we expect categories such as $(S[b]\backslash NP)\backslash PR$ or $(S[b]\backslash NP)/PR$ to be nonexistent in German and Dutch but common in English.

4. In Italian, attributive adjectives commonly appear after the noun they modify, whereas in English they almost always appear before (Dryer, 2013b). We thus expect the category $N\backslash N$ to be much more common in Italian than in English. Likewise, for adverbs modifying these adjectives, we expect $(N\backslash N)/(N\backslash N)$ in Italian but not in English (cf. Figure 3(a)).

5. In Italian, subject pronouns are frequently dropped (Dryer, 2013a), so we expect to frequently see verb categories like $S[X]$ and $S[X]/NP$, which are uncommon in English (cf. Figure 1).

To quantify these effects on comparable data for all four languages, we applied our parsers to the Tatoeba data to see how often they predict each category for a word. The relative numbers are shown in Table 5. We find all five expectations confirmed, suggesting that training parsers on projected derivations can indeed teach them specifics of each language's syntax.

7 Related Work

Recent years have seen much interest in cross-lingual learning, that is, learning tagging and parsing models for languages without training data for that language, instead relying on training data or existing systems for another language, and on parallel data to transfer knowledge from one language to the other. This is either done by automatically projecting source-language annotations from the source text to the target text (Yarowsky et al., 2001; Hwa et al., 2005; Tiedemann, 2014; Rasooli and Collins, 2015; Johannsen et al., 2016; Agić et al., 2016; Damonte and Cohen, 2018), sharing parameters between models for different languages (Zeman and Resnik, 2008; Ganchev et al., 2009; McDonald et al., 2011; Naseem et al., 2012; Täckström et al., 2013; de Lhoneux et al., 2018), or automatically translating the text from the source language to the target language and synchronously projecting the annotations (Tiedemann et al., 2014). Our work is an application of the first approach to CCG, which as a grammar formalism provides a more systematic framework for the study of syntax and for compositional interpretation than dependency parsers.

Apart from unsupervised syntactic CCG induction, CCG induction has also been done as part of learning semantic parsers, where supervision typically comes from logical forms, and syntax is treated as latent. Much of this work starts with a manually specified inventory of syntactic categories and only learns the semantic parts (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2013; Reddy et al., 2014; Artzi et al., 2015), whereas we start with no knowledge of the syntactic categories of the target language. Kwiatkowksi et al. (2010); Kwiatkowski et al. (2011); Bisk et al. (2016); Evang and Bos (2016) also learn the syntactic categories but evaluate their parsers only on semantic tasks, so it is unclear how linguistically plausible the induced CCGs are.

Earlier versions of the projection algorithm presented here were used in Evang and Bos (2016) for cross-lingual semantic parsing, and in Abzianidze et al. (2017) for bootstrapping a multilingual CCG treebank.

8 Conclusions and Future Work

Cross-lingual learning is a promising strategy whenever annotated training data for the target language is not available, but annotated training data for a source language as well as a parallel corpus is. This paper has introduced a method to apply this idea to syntactic CCG parsing, based on an algorithm for projecting CCG derivations along word alignments.

Compared to existing work on CCG induction, our method relies on parallel data and word embeddings but obviates the need for POS tags while in many cases outperforming methods that do use POS tags, and with less training data. This should make our method suitable for bringing multilingualism to CCG-based semantic parsers that so far rely on hand-written grammars.

In addition, we have shown that the induced lexicons reflect linguistic knowledge about the target languages. Our method also induces more fine-grained categories than previous approaches. It can thus also be a valuable asset for bootstrapping linguistically informed parsers and CCG treebanks for new languages.

There are various avenues to improving and extending derivation projection: alignment ambiguity could be handled with a global score, and multiple possible parses could be included in the target-language set, potentially improving the tradeoff between the number of projected derivations and the amount of noise. To increase the range of structural differences between languages that can be handled, derivation projection could be extended to consider sub-token units and to handle 1:n translation units in addition to n:1 ones.

Acknowledgments

The author would like to thank Lasha Abzianidze and Johan Bos for invaluable help in planting the seeds for this research, Yonatan Bisk for help with replication, and Laura Kallmeyer, Jakub Waszczuk, and all anonymous reviewers for valuable feedback. This research was supported by the NVIDIA Corporation with a Titan Xp GPU. It was partly carried out within the TreeGraSP project, funded by a Consolidator Grant of the European Research Council (ERC).

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247. Association for Computational Linguistics.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Sebastian Beschke and Wolfgang Menzel. 2018. Graph algebraic combinatory categorial grammar. In *Proceedings of the Seventh Joint Conference on Lexical* *and Computational Semantics*, pages 54–64. Association for Computational Linguistics.

Yonatan Bisk, Christos Christodoulopoulos, and Julia Hockenmaier. 2015. Labeled grammar induction with minimal supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 870–876. Association for Computational Linguistics.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Yonatan Bisk, Siva Reddy, John Blitzer, Julia Hockenmaier, and Mark Steedman. 2016. Evaluating induced CCG parsers on grounded semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2022–2027. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Companion Volume: Proceedings of the Demo and Poster Sessions*, pages 33–36.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual abstract meaning representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155. Association for Computational Linguistics.

Nicole Dehé. 2015. *Particle verbs in Germanic*, volume 1, pages 611–626. De Gruiter Mouton.

Matthew S. Dryer. 2013a. Expression of pronominal subjects. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Matthew S. Dryer. 2013b. Order of adjective and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Matthew S. Dryer. 2013c. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Kilian Evang and Johan Bos. 2016. Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING 2016, the 25th International Conference on Computational Linguistics*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the OPTAFNLP: Volume 1*, pages 369–377.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Jo ao Graça. 2012. The PASCAL challenge on grammar induction. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 64–80, Montreal, Canada.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for combinatory categorial grammars with generalized composition and type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 465–473.

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal, James R. Curran, and Johan Bos. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Anders Johannsen, Željko Agić, and Anders Søgaard. 2016. Joint part-of-speech and dependency projection from multiple sources. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 561–566. Association for Computational Linguistics.

Alexander Koller and Marco Kuhlmann. 2009. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 460–468. Association for Computational Linguistics.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 62–72.

Matti Miestamo. 2013. Symmetric and asymmetric standard negation. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, 2:377–392.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071.

Ole Tange. 2018. *GNU Parallel 2018*. Ole Tange.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Eighteenth Conference on Computational Natural Language Learning*.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.