The Role of Linguistic Features in Domain Adaptation: TAG Parsing of Questions

Aarohi Srivastava[1] **Robert Frank**[2] **Sarah Widder**[3] **David Chartash**[4]
Departments of [1]Computer Science and [2]Linguistics,
[3]Program in Cognitive Science, and [4]Center for Medical Informatics
Yale University
aarohi.srivastava|bob.frank|sarah.widder|david.chartash@yale.edu

Abstract

The analysis of sentences outside the domain of the training data poses a challenge for contemporary syntactic parsing. The Penn Treebank corpus, commonly used for training constituency parsers, systematically undersamples certain syntactic structures. We examine parsing performance in Tree Adjoining Grammar (TAG) on one such structure: questions. To avoid hand-annotating a new training set including out-of-domain sentences, an expensive process, an alternate method requiring considerably less annotation effort is explored. Our method is based on three key ideas: First, pursuing the intuition that "supertagging is almost parsing" (Bangalore and Joshi, 1999), the parsing process is decomposed into two distinct stages, supertagging and stapling. Second, following Rimell and Clark (2008), the supertagger is trained with an extended dataset including questions, and the resultant supertags are used with an unmodified parser. Third, to maximize improvements gained from additional training of the supertagger, the parser is provided with linguistically-significant features that reflect commonalities across supertags. This novel combination of ideas leads to an improvement in question parsing accuracy of 13% LAS. This points to the conclusion that adaptation of a parser to a new domain can be achieved with limited data through the careful integration of linguistic knowledge.

1 Introduction

The performance of contemporary syntactic parsers for natural language depends crucially on the availability of training data that matches the sentences on which the parser will be tested. In the realm of constituency parsing, by far the most common corpus used for training is the Penn Treebank (PTB) (Marcus et al., 1993), specifically the subset drawn from the Wall Street Journal (WSJ). It is a truism that the sentences in the WSJ are not an accurate representation of the entirety of English, and indeed the distribution of sentence types in the WSJ differs dramatically from language found in other domains. In particular, interrogative sentences (questions) are quite rare in the WSJ. It is unsurprising, then, that parsers trained on the PTB WSJ corpus perform poorly on questions, sometimes suffering reductions in accuracy of up to 20% (Petrov et al., 2010). However, questions are common elsewhere and indeed are a highly relevant sentence type for a range of NLP applications, such as question answering.

One way to resolve this difficulty involves the dedication of considerable resources to augmenting the training data set with additional hand-annotated parses of the questions. The work reported in this paper explores an alternative method that requires less annotation effort and makes use of three key ideas. First, we follow Bangalore and Joshi (1999) in decomposing the parsing process into two stages: supertagging, where lexically-associated pieces of structure are assigned to each word, and stapling, where these supertags are composed to form a parse tree. Second, we build on the work of Rimell and Clark (2008), where improvements to a supertagger trained with an extended dataset that is less costly to produce lead to improvements in parsing performance using an unmodified parser. However, we find that the parsing benefit that results from improved supertagging can only be maximized when the parser is structured so as to be sensitive to linguistically relevant properties of the supertags. As a result, a necessary third key idea is to use a parser whose input is characterized in linguistic terms that cross-cut the supertag set. This fosters the ability of the parser to generalize across linguistically related, but superficially distinct, sentence types. With the goal of increasing efficiency, following these ideas, a significant increase in parsing accuracy can be seen with a relatively small set of questions

for training.

Because we are interested in extracting details of the sentence's interpretation, such as those conveyed through long-distance dependencies, we make use of the Tree Adjoining Grammar (TAG) formalism. TAG is a mildly context-sensitive lexicalized grammar formalism, where the units associated with each word, called *elementary trees*, are pieces of phrase structure that encode detailed information about the word's combinatory potential. Past work (Kasai et al., 2018) has shown that the rich structural representations underlying TAG parsing allow better recovery of long-distance dependencies than is possible with other approaches. Our domain adaptation depends on the rich structure of TAG elementary trees, as we use linguistically-defined features to encode commonalities across trees that the parser can exploit.[1] TAG elementary trees are composed using two operations, substitution and adjoining. The resulting derivations have a structure similar to those familiar from dependency parsing, and indeed computational methods from dependency parsing can be used to accomplish broad coverage TAG parsing (Kasai et al., 2017). As a result, the proposal made in this paper should be more broadly applicable, outside the problem of TAG parsing.

In the first portion of this paper, we introduce the foundations of TAG and the shift-reduce TAG parser employed (Kasai et al., 2017). We then present our methodology of improving the process of assigning elementary trees (supertags) to the words in a sentence to be parsed, and show how and under what conditions improved supertagging can yield substantial benefits for parsing accuracy.

2 Tree Adjoining Grammar

Tree Adjoining Grammar (TAG) (Joshi et al., 1975), is a lexicalized grammar formalism that generates hierarchical structure through a system of tree rewriting. In a TAG derivation, each word in a sentence is associated with an *elementary tree*, a piece of syntactic structure that encodes the structural constraints that the word imposes on the sentence in which it appears. A TAG elementary tree thereby encodes information about the dependencies headed by a word, as well as the structural positions of the word's dependents. For example,

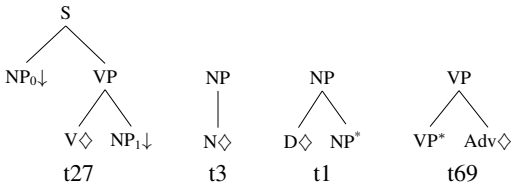


Figure 1: Elementary trees for *Alice read the book quickly*.

a transitive verb like *read* might be associated with the elementary tree t27 on the left of Figure 1, while a name like *Alice* or a noun like *book* would be associated with the elementary tree t3. In these elementary trees, the nodes labeled with the diamond indicate the structural position of the head of the tree. For the verbally-headed tree, the NP nodes that appear along the tree's frontier are the positions for the verb's arguments, i.e., its syntactic dependents. The subscripts on these arguments encode their syntactic relations with the elementary tree's head (0 is subject, 1 is direct object, 2 is indirect object).[2]

Elementary trees are combined using one of two derivational operations: *substitution* and *adjoining*. In substitution, an elementary tree rooted in some category C is inserted into a frontier node in another elementary tree that is also of category C and notated with a down arrow. Thus, to combine the subject NP with the verb in the sentence *Alice read a book*, the NP-rooted elementary tree t3 from Figure 1, headed by *Alice*, is substituted into the NP_0 substitution node in the S-rooted tree t27, headed by *read*.

The second operation, adjoining, introduces recursive structure via a special kind of elementary tree, called an *auxiliary tree*. Auxiliary trees have a distinguished frontier node, the *foot node*, that is of the same category as the root of the tree. The third tree t1 in Figure 1 is an NP-recursive auxiliary tree that would be associated with the determiner *the*. The asterisk on the NP frontier node indicates that it is the tree's foot node. Adjoining works by targeting a node N of category C in some elementary tree using a C-recursive auxiliary tree

[1] In this respect, TAG is similar to Combinatory Categorial Grammar (CCG) (Steedman, 2000), though the lexical units of CCG carry somewhat less information about structural context, as we will discuss below.

[2] These numeric superscripts correspond to "deep" syntactic relations: the subject of a passivized transitive verb will be annotated 1, and operations like dative shift preserve syntactic relations. Though this does not uniquely identify thematic roles of arguments (e.g., unaccusative and unergative subjects are not distinguished), it does provide a richer encoding of predicate-argument dependencies than is provided by usual surface-oriented parses. Recent work has shown that the identity of supertags provides particularly useful information for the task of semantic role labeling (Kasai et al., 2019).

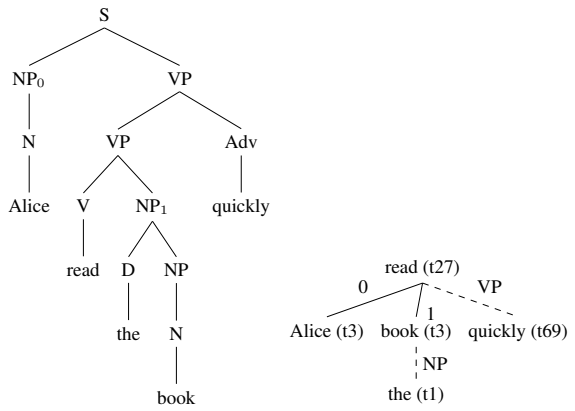


Figure 2: Derived and derivation trees for *Alice read the book quickly*.

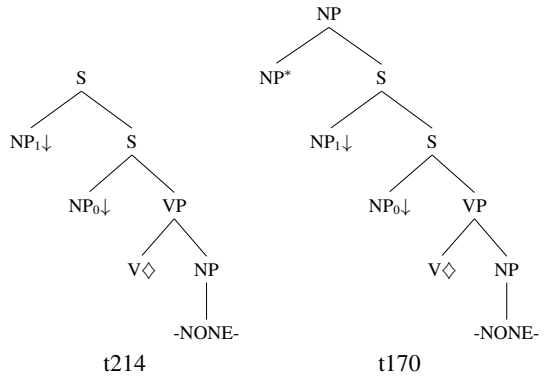


Figure 3: Elementary trees for object questions and object relative clauses.

T. When adjoining applies, the node N is rewritten as the tree T, and N's children are attached (or lowered) as the children of the foot node of T. The determiner tree on the right of Figure 1 can thus adjoin to the NP root of the N-headed tree in the middle of the same figure. In this way, the grammar can generate a structure corresponding to the NP *the book*, which can then be substituted into the NP object substitution node (NP_1) in the transitive verb-headed tree (t27) to derive the entire sentence *Alice read the book*. Similarly, the rightmost elementary tree in the figure, t69, can be adjoined to the VP node in t27 to yield a structure involving adverbial modification. The resulting derived tree structure is given on the left of Figure 2. This derived tree does not, however, represent the derivational steps that were involved in the creation of the structure, which are instead represented in a *derivation tree*. The nodes of the derivation tree correspond to elementary trees, and its edges (dependencies) correspond to substitution and adjoining operations that have applied, i.e., a daughter node is an elementary tree that has been substituted or adjoined into the parent node. Substitution is indicated by solid edges annotated with the index of the substitution site, while adjoining is indicated with dotted edges annotated with the locus of adjoining. The derivation tree for the simple sentence under consideration is given on the right in Figure 2.

TAG shares with the Combinatory Categorial Grammar (CCG) formalism the property of lexicalization: in both formalisms, words are associated with units of structure, elementary trees for TAG and lexical categories for CCG. The presence of rich structure associated with the lexical items is a source of information relevant for a variety of NLP tasks, including semantic analysis and translation, and the use of these formalisms have contributed to performance benefits (Cowan et al., 2006; Xu et al., 2017; Artzi et al., 2015; Nadejde et al., 2017). TAG and CCG differ, however, in the kind of information that the lexical structures encode. In TAG, a verb's elementary tree encodes not only its selected arguments, but also the positions in which they are syntactically realized. Sentences involving long-distance dependencies, such as relative clauses or questions, will therefore involve distinct verbally-headed elementary trees from those used for simple declarative sentences, in which the wh-movement dependency is realized (Frank, 2004). For example, in the question *What did Alice read?*, the displacement of the NP object to the front of the question and its original position filled with a trace node indicated by NONE, as in the Penn Treebank, is represented in the elementary tree t214 on the left in Figure 3. Since the auxiliary verb *did* must appear directly after the fronted NP (NP_1, or *what*, in this case), it adjoins to the S child of NP_1, as shown in Figure 2. In contrast, the verb *read* in the relative clause of the noun phrase *the book that Alice read* would head a different, but related elementary tree, shown on the right in Figure 3, which also includes the fronting of the object, but is itself an auxiliary tree that can adjoin to the NP it modifies.

In contrast, CCG lexical categories do not encode the different realizations of a verb's arguments found in declaratives, questions or relatives. In all such cases, a transitive verb would be assigned the lexical category $(s\backslash np)/np$. What differs are the categories assigned to the object (np in simple sentences, $s/(s/np)$ for the question word,

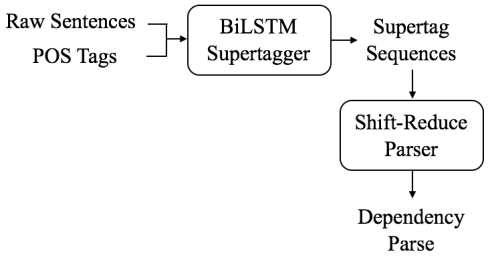


Figure 4: The TAG Parser pipeline.

and (np\np)/(s/np)) for the relative pronoun), as well as the way in which these elements combine with the verb.

3 Supertagging and Parsing

This study uses the TAG supertagger and parser developed by Kasai et al. (2017). The supertagger-parser pipeline is shown in Figure 4. Raw sentences and part of speech tags are given as input to the TAG supertagger, which outputs predicted supertags (i.e., elementary trees) for each word. These predicted elementary trees are given as input to the (unlexicalized) TAG parser, which outputs predicted parses with labeled dependencies among the elementary trees. We briefly review the architecture developed by Kasai et al. (2017). For more details, the reader should consult the original paper.

3.1 Supertagger Architecture

As discussed above, a simple transitive verbal predicate such as *read* might have a different elementary tree depending on the context: t27 as the main predicate of a declarative sentence, or t214 in an interrogative sentence. The same word might have other elementary trees in other constructions, such as subject and object relatives, meaning that the determination of the correct tree requires sensitivity to information that is not local in the string (Kasai et al., 2017). To address the need for long-distance dependency information, the supertagging model makes use of Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997), a recurrent network architecture which is constructed to avoid the vanishing/exploding gradient problem. Specifically, the supertagger developed by Kasai et al. (2017) employs a one-layer bidirectional LSTM network. This architecture processes the input sentence both from beginning to end and from end to beginning. The output of these LSTM units at each time step are concatenated, fed into an affine transformation, and then fed into a softmax unit, yielding a probability distribution over the 4,727 elementary trees that exist in the TAG-parsed corpus we employ, which was extracted from the PTB corpus (Chen et al., 2005). Each word is given to the network as in Kasai et al. (2018): the concatenation of a 100-dimensional GloVe embedding (Pennington et al., 2014), a 5-dimensional embedding of a predicted part of speech tag, and a 30-dimensional character-level representation of the word. The network is trained by optimizing the negative log-likelihood of the observed sequences of supertags.

3.2 Shift-Reduce Parsing Algorithm

Parsing is done using the arc-eager system of shift reduce parsing introduced in the MALT parser (Nivre et al., 2006). This system maintains a stack, buffer, and the set of dependency relations derived so far as the current state. These dependency relations consist of the substitutions and adjoinings that have already occurred between elementary trees. Initially, the buffer holds the sequence of tokens in the sentence, and the transitions terminate when the buffer is empty. At each state, the arc-eager system may choose one of four operations: LEFT-ARC, RIGHT-ARC, SHIFT, and REDUCE, defining ways in which the top elements of the stack and buffer may be manipulated. The TAG parser further divides LEFT-ARC and RIGHT-ARC into seven types according to the derivational operation involved, whether substitution or adjoining, and the location at which the operation takes place (Kasai et al., 2017).

The parser implemented by Kasai et al. (2017) uses a two-level feed-forward network that is trained to predict the operation that should be taken, given the top five elements of the stack and buffer. A noteworthy aspect of the parser is that these data structures contain only supertag information, not the identities of the words in the sentence being parsed. Each supertag is given to the network as a one-hot vector, which is then embedded into a more compact representation, together with vectors that indicate any substitution operations that have already been performed on the supertag. These vector representations of the top elements of the stack and buffer are concatenated and fed to the network, which yields a probability distribution over the possible transition actions. The parser is decoded using a beam search.

3.3 Feature Embeddings

Friedman et al. (2017) explore the benefits of a different input representation for the same parser, involving feature-based embeddings of the elementary trees. These feature-based embeddings are vectors that encode linguistically-defined dimensions of information about the elementary trees specified by Chung et al. (2016). These dimensions include structural properties of the elementary tree (category of the root and head and the category and direction of substitution nodes), subcategorization frame, and grammatical properties (passive, particle shift, wh-movement). The rationale for training a parser with feature embeddings is to allow the network to exploit relationships between trees, and to be able to generalize parsing actions across related contexts. This is particularly useful for cases like passivization and wh-movement, in which the argument structure of the root remains the same, but there are changes in syntax which are reflected in the elementary trees. Friedman et al. (2017) compare the parsing models using both one-hot and featural representations of supertags with respect to parsing performance on PTB sentences, but only saw a "slight improvement" (approximately 0.2% improvement in LAS). However, in the case of adapting to new domains, learning this kind of linguistic information may bridge the gap between the original data domain and the new domain, as it will allow sharing of information about parsing actions for related structures. We explore the importance of providing linguistically-rich feature embeddings to the parser to aid in improving parsing accuracy in the new domain of interrogatives despite never training the parser on sentences from the new domain, especially when limited data is used.

4 Methods

4.1 Background

The most direct approach to adapting a parser for new domains would be to generate a new, hand-annotated dataset that included instances of the new sentence type, which could be used to train a supertagger and parser. Such a process would, however, involve a substantial annotation effort for each new domain. We instead build on the approach of domain adaptation taken by Rimell and Clark (2008). The viability of Rimell and Clark's approach rests on the assumption that "supertagging is almost parsing" (Bangalore and Joshi, 1999). If a parser is provided with a correct set of supertags, it should perform better even on sentence types outside the domain on which it was trained. We therefore focus on retraining the TAG supertagger with a hand-annotated set of questions to which TAG elementary trees have been assigned to each word, but for which parses have not been generated. This hand-annotation process is less expensive than the creation of full parses. As we shall see, this procedure results in improvements in both supertagging and parsing accuracy without ever training the parser on an augmented dataset of questions.

4.2 Data

The question set used in this study contains 350 of the questions used by Rimell and Clark (2008). Their dataset was drawn from the training data provided for the TREC 9-12 Competitions.

4.3 Supertagger Training and Evaluation

To train the TAG BiLSTM supertagger, gold standard part of speech (POS) and supertag sequences were first created for the 350 question set. POS tags were assigned to the 350 questions using the Stanford CoreNLP (Manning et al., 2014) web-based POS tagging tool. These tags were then checked and corrected by hand to create gold standard POS tags.

Next, elementary trees were assigned to the sentences by hand. To make sure these hand annotations were compatible with and followed the same conventions as the method of supertag assignment for the PTB data used to train the parser, the PTB annotation guidelines (Bies et al., 1995) and the gold standard supertag data (Chen et al., 2005) were frequently reviewed. Stanford Tregex (Levy and Andrew, 2006) was used to find relevant trees (e.g., declarative forms of the questions, relative clauses with a similar structure) in the WSJ corpus. Through these methods, ambiguities regarding assignment of elementary trees were resolved. Hand annotation was primarily done by one author, and another author verified or corrected the hand annotations.

In essence, the hand annotation process was conducted as follows. Given the question "What did Alexander Graham Bell (AGB) invent?" the supertag sequence for the corresponding declarative was first determined (Figure 5). From this, the supertag sequence for the question would be cre-

ated. The biggest change is that the tree for the predicate, *invent*, must reflect the wh-movement (Figure 6).

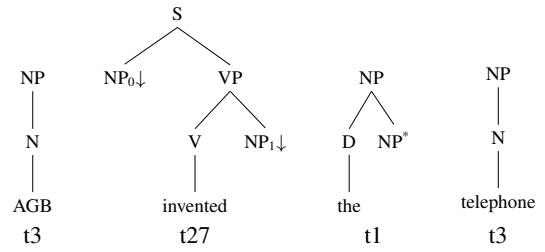


Figure 5: AGB invented the telephone.

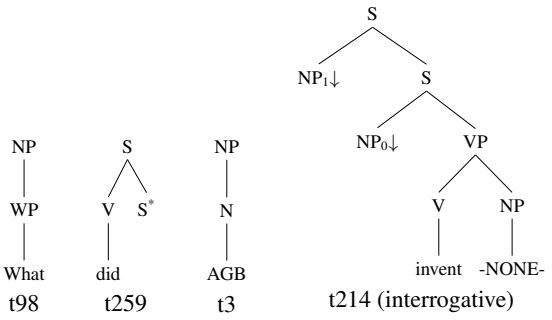


Figure 6: What did AGB invent?

As can be seen, the interrogative elementary tree t214 can be derived from the declarative elementary tree t27. NP_1 has been fronted, and the added auxiliary *did* will adjoin directly after NP_0 at the second S node. Appendix A contains more information about the conventions that were followed in assigning supertags in several common types of questions.

The BiLSTM supertagger was trained with two regimens. In one, only the original PTB training set (WSJ sections 01-22) was provided. In the other, the supertag sequences associated with the hand-tagged questions were added to the PTB data. Rimell and Clark (2008) added ten copies of their 1,328 training questions, adding 13,280 questions to the 39,832 PTB training sentences. Due to the smaller number of hand-tagged questions used for training in this study, 35 exact copies of the training questions were added to the PTB training sentences. This yielded a total of 49,632 sentences in the training set. Through a developmental stage of training and testing, it was determined that 35 copies was optimal to have the highest possible accuracy of supertagging questions without overfitting or reducing accuracy of supertagging PTB sentences. Supertagger training and testing was done using five-fold cross-validation. For each of the five folds, a unique subset of 70 questions was saved for testing, and the remaining 280 questions were used for training. We report mean accuracy over these five folds.

4.4 Parsing Evaluation

In order to analyze parsing performance of questions, gold parses were created for a small test set of 48 questions, each associated with a unique supertag sequence. These questions were not among those used for the training of the supertagger. As before, the assignment of gold parses was done through careful consultation of the PTB annotation guidelines (Bies et al., 1995), as well as the existing TAG-parsed version of the PTB.

For the TAG parser, creation of gold parses requires not only the gold supertag sequences, but also the dependency relations (for UAS and LAS) and the arc labels (for LAS). Two additional columns of information must be added when creating a gold parse as opposed to a gold supertag sequence for a sentence, as shown below. As a result, creating gold supertag sequences is less time-intensive than creating gold parses.

	Word	Supertag	Rel	Arc Label
1	What	t612	2	adjoin
2	continent	t3	5	1 (object)
3	is	t259	5	adjoin
4	India	t3	5	0 (subject)
5	on	t2911	0	root

Two parsing models were explored, both trained only on the PTB TAG parses: (1) the parser model proposed by Kasai et al. (2017) that was trained using one-hot vector embeddings of the elementary trees (henceforth -F), and (2) an identical parser trained with Friedman et al.'s elementary tree feature embeddings (henceforth +F). Decoding for both parsers was done using beam search with a beam size of 16. For each model, three different scenarios were tested, varying in the nature of the supertag input received for the questions to be parsed: (1) supertags given by the original PTB-trained BiLSTM supertagger model (Kasai et al., 2017) (henceforth PTB), (2) supertags given by a supertagger model trained with an augmented dataset of questions and PTB sentences (henceforth PTB+Q), and (3) hand-annotated gold supertags (henceforth Gold). The accuracy of parses in each of the six cases are reported in Section 5.2.

5 Results and Discussion

5.1 Supertagging Results

Supertagging results for the set of 350 questions and the PTB test set are reported separately in Table 1. The PTB-trained supertagger gave an accuracy of 79.61% for the set of 350 questions (an average over the five folds of cross-validation, weighted by the number of words in each fold), and 91.50% for the PTB test set. This PTB-trained supertagger frequently made three types of errors when assigning elementary trees to questions:

1. Incorrect wh-phrase construction: The correct elementary tree for the wh-determiner (e.g., *what* in *what book*) should contain a right NP* adjunction node to adjoin to the NP *book* (as in t1 assigned to *the* in *the book*, Figure 1). Instead, the elementary tree assigned to *book* by the PTB-trained supertagger would incorrectly contain a left NP* adjunction node to facilitate adjunction to the wh-phrase, or the verbal predicate's elementary tree would have two NP substitution nodes into which the wh-determiner and the noun could be inserted separately.

2. Incorrect tree for auxiliary verb: Auxiliary verbs (e.g., *did*) were treated as in a declarative sentence, heading a VP-recursive auxiliary tree t23. Because the auxiliary should appear immediately following the fronted NP and before the subject, the adjunction of the verb should instead take place at S (cf. tree t214 in Figure 3), as in tree t259.

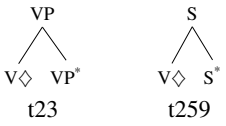


Figure 7: t23 (declarative) vs. t259 (interrogative)

3. Incorrect tree for verbal predicate: The main predicate of the sentence was assigned a declarative elementary tree rather than a question version (i.e., neither fronting nor the NP-NONE trace were expressed in the elementary tree). For a transitive sentence, this means t27 (Figure 1) was assigned to the verbal predicate rather than t214 (Figure 3).

	Questions	PTB Test
PTB training	79.61	91.50
PTB+Q training	95.17	91.64

Table 1: Supertagging Accuracy. Rows indicate training set, whether augmented or not.

		PTB	PTB+Q	Gold
UAS	-F	90.80	90.53	94.60
	+F	91.14	90.51	96.00
LAS	-F	89.63	89.39	94.07
	+F	90.00	89.39	95.81

Table 2: PTB Test Parsing Accuracy. Columns indicate training set for supertagger (or gold supertags) that provide input to the parser. ±F indicates the presence or absence of feature-based supertag embeddings in the input to the parser.

		PTB	PTB+Q	Gold
UAS	-F	81.84	86.70	91.04
	+F	86.18	93.86	99.74
LAS	-F	79.79	85.67	90.53
	+F	83.88	93.09	99.74

Table 3: Question Parsing Accuracy. Columns indicate training set for supertagger (or gold supertags) that provide input to the parser. ±F indicates the presence or absence of feature-based supertag embeddings in the input to the parser.

For the PTB+Q trained supertagger, supertagging accuracy improved, particularly in regards to the three common errors outlined above. On average, supertagging accuracy increased substantially for the question test sets. At the same time, supertagging accuracy on the PTB test set was maintained, indicating that when augmentation is done appropriately, additional training on types of constructions rare in a corpus does not adversely affect supertagging performance on the original corpus.

5.2 Parsing Results

Table 2 reports parsing accuracy on the PTB test set for each of the six parser input conditions described in Section 4 (varying by supertag input and presence or absence of feature-embeddings).[3] We see that the addition of the question data to the supertag's training data (PTB+Q) has a minimal effect on parser performance on the PTB test sentences. Similarly, as found by Friedman et al. (2017), the addition of feature embeddings results in a very small improvement in parsing accuracy, if at all.

[3]Following the standard in the TAG parsing literature, these values do not include accuracy for punctuation.

More relevant for the current topic of discussion is the parsing performance of questions, which is reported in Table 3 for each of the six parser input conditions. We first note that while labeled parsing accuracy (LAS) for the -F parser improved from 79.79% to 85.67% when going from PTB to PTB+Q supertagger training, we see an even more dramatic increase when the feature-trained (+F) parser is used: in this case, parsing accuracy increases to 93.09%. As discussed in Section 3.3, the feature embeddings provide linguistic information over which the parser can generalize from one type of structure to another. Because of the rarity of questions in the PTB, many of the correct supertags used when hand-annotating the question set are also rarely present in the gold standard supertag data for the PTB WSJ corpus (Chen et al., 2005). As a result, the TAG parser (trained only on the PTB WSJ corpus) was not equipped to properly handle these supertags. Thus, while the parsing accuracy increased when given PTB+Q-trained supertags, the improvement is not as large as it might be due to the parser repeatedly encountering uncommon supertags that it was unable to correctly staple together. When the +F parser was used, the parser had learned the knowledge required to better deal with these less common supertags, and parsing accuracy improved from 83.88% to 93.09%. It is notable that this improvement is super-additive: the improvement on LAS (13.3%) is greater than the sum of the individual improvements obtained by using the improved supertagger (PTB+Q) alone (5.88%) or using feature-embeddings (+F) in the parser (4.09%). Thus, we find that with our approach to domain adaptation, when coupled with representations that encode linguistic commonalities across different types of structures, accuracy can increase to a level comparable to the parsing accuracy of the original domain. It is also notable that, when training the supertagger, so few questions (350) are needed to see a significant increase in both supertagging and parsing accuracy (by 15% and 13%, respectively).

Table 4 breaks errors in parsing questions into two categories. The error category of "incorrect wh-phrase" relates to parses of questions that failed to adjoin a wh-determiner to its corresponding noun phrase, or that incorrectly substituted a wh-phrase as an argument of the corresponding predicate. The "missing root" category relates to parses that omit assigning any term in the sentence as the root of the dependency parse, most likely due to complexity or rareness of the correct root word's elementary tree. The number and types of parsing errors deriving from the presence of uncommon supertags in questions (e.g., a parse missing a root) persist in the -F parser. In contrast, these errors are minimal for the +F parser. Treatment of the wh-phrase construction was a specific focus of training the supertagger on questions, and while errors in this category decreased (cf. Table 4) for both parsers once the improved supertags were given, the feature-trained (+F) parser was better able to handle these constructions, and errors decreased much more.

		PTB	PTB+Q	Gold
incorrect wh-phrase	-F	19	9	7
	+F	16	3	3
missing root	-F	16	25	23
	+F	1	0	0

Table 4: Summary of Parsing Evaluation for Questions. ±F indicates the presence or absence of feature-based supertag embeddings in the input to the parser.

It is important to note that, although the number of sentences with a missing root increases from the PTB to PTB+Q trained supertagger, the reason for having a missing root changes. Given the correct (often rarer) supertag for the root in the PTB+Q case, the -F parser is now not equipped to properly combine other trees with it, so the root is skipped. This leads to higher numbers of missing root errors for both PTB and PTB+Q. However, such errors do not occur in the +F parser, as sensitivity to features allows the parser to be better equipped to compose even rare trees correctly. We find then that the statement "supertagging is almost parsing" (Bangalore and Joshi, 1999) is true only when the linguistic content of supertags is known to the parser. When the parser receives correct supertags (gold) and is equipped to handle them properly since it was trained with feature embeddings, it yields near-perfect parses (99.74%).

6 Future Work

We anticipate that the approach of domain adaptation for supertagging and parsing explored here can be applied to other domains. For example, imperatives are another sentence type nearly absent from newspaper corpora, but which are nonetheless a crucial type of input to NLP systems such as

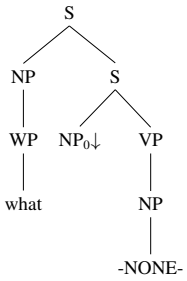


Figure 8: Proposed elementary tree for "what" predicate

virtual assistants. Another domain to which this method might be applied involves biomedical and clinical text (cf. Rimell and Clark 2009), which pose a challenge for information retrieval systems due to the domain-specific vocabulary abbreviations and distinctive syntactic structures, such as null subjects, asyndetic coordination, and fragments.

(a) *abbreviations*: 8 yo M no PMH presents with n/v/F and fever x4 days

(b) *null subjects*: presents with shortness of breath

(c) *asyndetic coordination*: VS notable for fever to 103F, tachycardia, tachypnea

(d) *fragments*: non-toxic though appears ill

In addition, because questions are not well-represented among the original PTB training corpus for the parser, questions on which the parser was tested sometimes involved novel supertags that were absent from the grammar extracted from the PTB. For example, copular sentences with NP predicates (like *Mardi Gras is a festival*) can front the predicate to form a question (as in *What is Mardi Gras?*). The appropriate elementary tree for such cases should be the one given in Figure 8, with the clausal predicate *what* appearing in fronted position. However, no such elementary tree exists among those that were extracted from the PTB by Chen et al. (2005). Consequently, in order to better parse all types of questions, and more generally sentences from other domains, it will be necessary to allow for the creation and feature decomposition of new elementary trees.

7 Conclusion

In this study, we explored an approach to domain adaptation for TAG parsing in the context of questions. We extended Rimell and Clark's approach for improving parsing by improving supertagging. We found first of all that improvements in TAG supertagging, despite the larger number of supertags involved as compared with CCG, are possible through a relatively limited hand-annotation effort. Supertagging accuracy of questions increased by 15%, without sacrificing supertagging accuracy on the original corpus data. Furthermore, while this approach is also successful in improving parsing performance, its effectiveness is maximized when the parser makes use of linguistically-informed representations of supertags. Strikingly, previous work (Friedman et al., 2017) found that the introduction of hand-coded linguistic features in the supertag representations given to the parser does not yield significant benefits in parsing performance. However, our current results suggest that the addition of linguistic features can constitute a crucial source of information when processing structures that are underrepresented in the training data. A parser trained with linguistically-defined feature decompositions of the supertags can better handle those supertags that are uncommon in the data it was trained on. In such cases (e.g., questions), the parser is able to exploit abstract commonalities with related structures, such as relative clauses, that do occur frequently in the training data. Without such linguistically structured representations, considerably more effort would need to be expended to annotate parses in the new domain of questions. We see then that neural methods are not immune to the need for the careful incorporation of hand-coded linguistic features, particularly in addressing problems of domain adaptation.

8 Acknowledgement

The authors are grateful to Jungo Kasai for his crucial advice and technical support throughout this work. We would also like to thank the members of the CLAY lab at Yale, who provided valuable feedback.

References

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.

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for treebank II style Penn Treebank project. University of Pennsylvania.

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2005. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299.

Wonchang Chung, Siddhesh Suhas Mhatre, Alexis Nasr, Owen Rambow, and Srinivas Bangalore. 2016. Revisiting supertagging and parsing: How to use supertags in transition-based parsing. In *12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 12)*, pages 85–92.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241.

Robert Frank. 2004. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Cambridge, MA.

Dan Friedman, Jungo Kasai, Thomas R. McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for TAG parsing. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 122–131. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aravind K Joshi, Leon S Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163.

Jungo Kasai, Robert Frank, R. Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based tag parsing with neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jungo Kasai, Dan Friedman, Robert Frank, Dragomir Radev, and Owen Rambow. 2019. Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 2231–2234.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language CCG supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 68–79, Copenhagen, Denmark. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 475–484. Association for Computational Linguistics.

Laura Rimell and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Pauli Xu, Robert Frank, Jungo Kasai, and Owen Rambow. 2017. TAG parsing evaluation using textual entailments. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 132–141. Association for Computational Linguistics.

A Appendix: Assigning TAG Supertags to Questions

This appendix lays out the linguistic assumptions and analytic decisions that were made for question supertagging and parsing. Within the 350 questions, four basic question types, expressed in a generalized form below, were most common.[4]

a. How many/much ... ?

b. What (NP) is NP ?

c. What (NP) VP ?

d. What (NP) is NP+IN ?

Below we briefly present our assumptions for each type.

How many/much ... ?

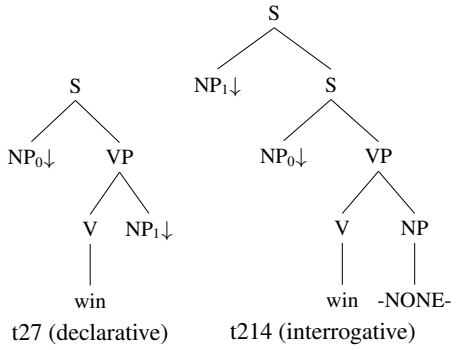
An example of this type of question is:

(1) How many battles did she win?

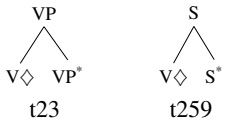
It is first useful to examine the declarative version closest to this sentence:

(2) She did win five battles.

The key difference between the interrogative version (Sentence 1) and the declarative version (Sentence 2) is the change in order, akin to that of wh-movement. Thus, the elementary tree for the verbal predicate *win* in this question must include the noun phrase trace, as in t214:



As can be seen, the interrogative elementary tree t214 can be derived from the declarative elementary tree t27. NP_1 corresponds to *five battles*. NP_1 in t27 has been replaced by the NP-NONE trace in t214, since it has moved to the beginning of the sentence (fronting). To show this, an additional S node has been added to the top of the tree. Another key difference adopted as a convention is the treatment of *did*. In the declarative sentence, *did* is assigned t23, a VP-recursive auxiliary tree. However, in the interrogative version, *did* is assigned t259, an S-recursive auxiliary tree. The difference is shown below:



This is because of the placement of the additional S node in t214. The auxiliary verb *did* must come between the object (NP_1) and subject (NP_0) of the question, as shown in t214.

What (NP) is NP ?

An example of this type of question is:

(3) What is the capital of Kentucky?

with the corresponding declarative sentence:

(4) Frankfort is the capital of Kentucky.

The supertags assigned to Sentence 3 are shown in Figure 9, and the supertag for the predicate is t668 in Figure 10.

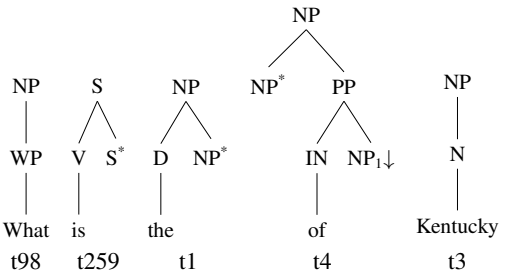


Figure 9: Elementary trees for Sentence 3.

There are two key concepts behind this type of question. First, as for the auxiliary verb *did* in the earlier question type, t23 becomes t259 in the context of questions due to the necessity of adjoining to the S node in a position above the subject. Second, we notice in a copular sentence there is no verb to head the elementary tree, i.e., to project the main S node that serves as the root of the derivation. Instead, the noun *capital* plays the role of predicate of the sentence, and is assigned an S-rooted elementary tree, t668. Figure 10 illustrates the similarity of the two elementary trees assigned

[4] Part of speech tags are taken from the PTB.

to the predicate nominal *capital* in declarative and interrogative forms, with the interrogative t668 encoding the NP-NONE trace.

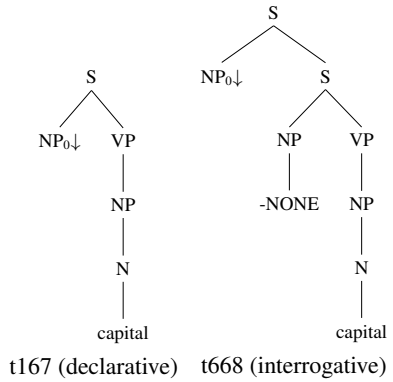


Figure 10: Elementary trees for "capital" in Sentences 4 and 3, respectively.

What (NP) VP ?

Sentence 5 gives an example of this type of question.

(5) What car company invented the Edsel?

(6) Ford invented the Edsel.

The sequence of elementary trees assigned to this sentence is shown in Figure 11. Although there is no change in word order when converting from the interrogative to the declarative version of this sentence, the verbally-headed elementary tree follows the practice of placing a trace in subject position and displacing the subject to a higher position, as done in the PTB.

Earlier, t214 was used for the question version of the transitive verb *win*'s elementary tree. The difference between t214 and t335 is whether it was the object or subject that was fronted to form the question. Distinct elementary trees are necessary for each possible position of extraction for a given pattern of transitivity.

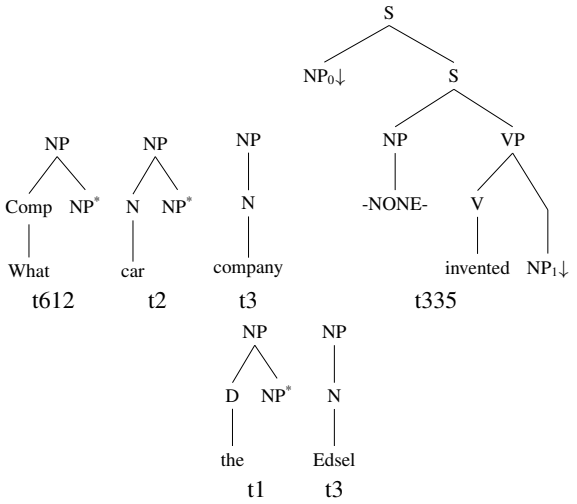


Figure 11: Elementary trees assigned to Sentence 5.

What (NP) is NP+IN ?

The final question type we consider here is as follows:

(7) What city is Logan Airport in?

Unlike copular questions, in which a noun phrase is the main predicate, in Sentence 7 the main predicate is the preposition *in*. As a result, this preposition constitutes the head of the S-rooted elementary tree, as shown in Figure 12, where *what city* substitutes into the NP_1 node (object), and *Logan Airport* substitutes into the NP_0 node (subject).

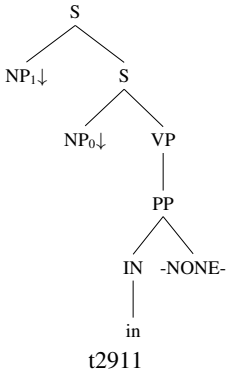


Figure 12: Elementary tree used in Sentence 7.