A Pilot Arabic CCGbank

Stephen A. Boxwell, Chris Brew

The Ohio State University
Columbus OH, USA
boxwell@1ing.ohio-state.edu, cbrew@1ing.ohio-state.edu

Abstract

We describe a process for converting the Penn Arabic Treebank into the CCG formalism. Previous efforts have yielded CCGbanks in English, German, and Turkish, thus opening these languages to the sophisticated computational tools developed for CCG and enabling further cross-linguistic development. Conversion from a context free grammar treebank to a CCGbank is a four stage process: head finding, argument classification, binarization, and category conversion. In the process of implementing a basic CCGbank conversion algorithm, we reveal properties of Arabic grammar that interfere with conversion, such as subject topicalization, genitive constructions, relative clauses, and optional pronominal subjects. All of these problematic phenomena can be resolved in a variety of ways - we discuss advantages and disadvantages of each in their respective sections. We detail these and describe our categorial analysis of each of these Arabic grammatical phenomena in depth, as well as technical details on their integration into the conversion algorithm.

1. Introduction

In recent years, Arabic has become an increasingly important language for natural language processing. Applications like machine translation, information retrieval, and semantic processing are in high demand, but are technically difficult because Arabic has many properties that make it resistant to traditional statistical and finite-state approaches. A highly expressive formalism like CCG can capture many grammatical phenomena, like long-range dependencies, that simpler formalisms cannot. Furthermore, a wide variety of high-quality NLP tools exist for CCG, and an Arabic CCGbank would make this technology available to Arabic for the first time. In this paper, we describe the process of automatically creating an Arabic CCGbank from the Penn Arabic Treebank (Maamouri et al., 2004a) by following the process for constructing the English CCGbank (Hockenmaier and Steedman, 2007) from the English Penn Treebank (Marcus et al., 1993).

Section 2 describes related efforts to construct CCGbanks in other languages. Section 3 gives a brief overview of Combinatory Categorial Grammar (CCG) and details some of its advantages over a context-free grammar style formalism like that in the Penn Arabic Treebank. Section 4 describes the basic conversion algorithm, based heavily on the one used for the English CCGbank (Hockenmaier and Steedman, 2007), and section 5 describes some of the grammatical phenomena in the treebank that are handled inadequately by the basic algorithm and require special attention. Finally, we conclude in section 6 with the ongoing efforts to produce a complete Arabic CCGbank.

2. Related Work

Converting treebanks into other grammatical formalisms has been a topic of much research interest in recent years. Work has been done on converting treebanks in other languages into CCG, including Turkish (Çakıcı, 2005) and German (Hockenmaier, 2006). With regard to Arabic, the Penn Arabic Treebank (Maamouri et al., 2004b) has already been the subject of much conversion work, including the extraction of a Tree Adjoining Grammar treebank (Habash and Rambow, 2004) and an LFG Treebank (Tounsi et al., 2009). This work has been augmented by efforts to improve the Arabic treebank itself (Maamouri et al., 2008; Maamouri et al., 2009; Smrž et al., 2008). The present work is, to the best of our knowledge, the first attempt to convert the Penn Arabic Treebank into CCG.

3. Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a syntactic framework that describes words and phrases in terms of their combinatory potential. For example, words like “the” are of the category np/n, or “the kind of word that would make a noun phrase if it could combine with a noun to the right”, The left side of the category represents the result, the right side represents the argument, and the direction of the slash represents where the argument must be with respect to the word in question (i.e., to the left or the right). Categories can be nested inside of each other to an arbitrary depth: verbs used transitively like “ate” are given the syntactic category (s\np)/np, or, “the category that would represent a sentence if it could combine with a noun phrase to its right and then a noun phrase to its left”. The categories then combine with each other to form larger constituents (shown in figure 1).

CCG also features a transparent way to resolve local and long-range dependencies. The dependency graph of *the man ate the steak* is shown in figure 2. This analysis uses only local dependencies. In other cases, however, like the English relative clause, the relationship between words must be handled by an intermediary category. CCG can represent these constructions in a manner that reflects their relationship with their corresponding declarative sentence. Consider the phrase *the man who ate the steak*. In this sentence, *the man* has the same semantic relationship with the verb *ate* as it does in the sentence from figure 1. Using argument coindexation, CCG allows *the man* to have the same syntactic relation as well (figures 3 and 4).

CCG encodes several properties of language that are absent from the Penn Arabic Treebank. First, CCG requires that the lexical heads of each span be made explicit. The Penn

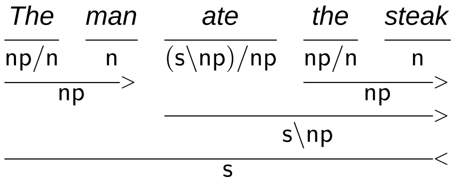

Figure 1: A simple CCG derivation, showing how CCG categories combine to make a sentence.

Category	Argument	From	To
np/n	1	the	steak
np/n	1	the	man
(s\np)/np	1	ate	man
(s\np)/np	2	ate	steak

Figure 2: A dependency graph of *the man ate the steak*, implicit in the derivation in figure 1.

Arabic Treebank, however, does not include this information. Second, CCG makes a sharp distinction between syntactic arguments and adjuncts. PATB makes some effort with regard to this (through the use of the -CLR "closely related" tag), but most constituents are left unmarked. Third, CCG requires a binary branching structure, while PATB annotators are free to include an arbitrarily large number of daughters for each parent node. All three of these differences must be resolved before we assign CCG categories to constituents on PATB trees.

4. The General Treebank Conversion Algorithm

The English CCGbank was created automatically from the Penn Treebank. The algorithm used to convert the treebank to CCG (which we adapt to the Penn Arabic Treebank here) consists of four cascading steps: head identification, argument/adjunct distinguishing, binarization, and mapping to CCG categories. In this section, we will examine each in turn.

Please note that the syntax trees in the following sections are intended to be read from left to right, not from right to left (as with normal Arabic script). The script on the individual nodes can be read from right to left as normal. All example sentences and fragments are taken from the Penn Arabic Treebank, with some simplifications for reasons of space. English glosses are provided for the benefit of non-Arabic speakers.

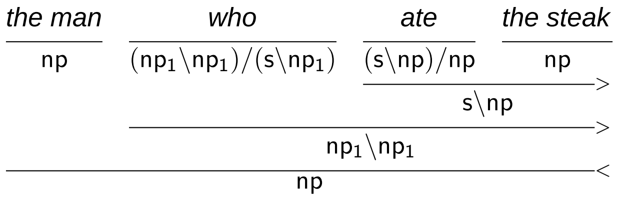

Figure 3: CCG's treatment of relative clauses. The coindexation on the relativizer category allows the verb's subject dependency to resolve to *the man*.

category	arg	from	to
(np\np)/(s\np)	2	who	ate
(np\np)/(s\np)	1	who	man
(s\np)/np	**1**	**ate**	**man**
(s\np)/np	2	ate	steak

Figure 4: A dependency graph for *the man who ate the steak*, shown in figure 3. Notice that the subject dependency of *ate* is resolved to *man*, reflecting their semantic relation.

4.1. Head Identification

The first step in converting the Penn Arabic Treebank into CCG is to identify the head (or heads) of each constituent. First, we devise a simple set of baseline heuristics to identify the heads of the most common constituents. An exhaustive list of these heuristics is given in figure 5. Using these heuristics, we can tag 97.99% of the constituents in the treebank and achieve 100% coverage on 52.7% of the trees in the treebank. An example of a head-tagged constituent is given in figure 6.

There remain, after this initial pass, a number of problematic cases that are not covered by our heuristics. Although these cases are diverse in nature, many involve the annotation of gerunds. A gerund is a verb that functions like a noun, and is therefore annotated as a noun for the purposes of the Penn Arabic Treebank. Consider the case shown in figure 7. It is clear that *sustaining* is the head of its constituent, but it would be inelegant and potentially harmful to propose a heuristic in which a noun could act as the head of a VP when this role is generally played by a verb.

To account for these exceptional cases, we use the baseline head-finding heuristics to extract features for a Maximum-Entropy classifier that could predict the heads of phrases that are not covered by the heuristics. The features used are as follows:

- **Grammatical Category**. The grammatical category (NN, NP, VB, etc) of each candidate daughter node, and the parent node.
- **Predicate**. A binary indicator feature showing whether the candidate daughter node is identified with the PRED tag by the treebank annotators – a strong predictor of headship for VP and S nodes.
- **Location**. A feature indicating whether the candidate daughter node is the first or last node in the constituent.

After training this classifier, we perform a second pass over the constituents that were not tagged the first time and identify the most likely head node.

4.2. Distinguishing Arguments from Adjuncts

The second step in the pipeline is to sort the remaining daughter nodes of each constituent into arguments and adjuncts. This is crucial to the conversion process, as CCG requires a sharp argument-adjunct distinction. To achieve

Parent Category	Category of Head
ADJP	JJ
NP	NN, NNP, NNS, NP
PP	IN
S	VP, -PRD, S
VP	VBD, VBP, VBN, VB
QP	CD
SBAR	WHNP, WHADVP, IN
CONJP	CC

Figure 5: Heuristics used to identify the headwords of constituents. Using only this short list of heuristics, we can tag 97.99% of the constituents in the treebank.

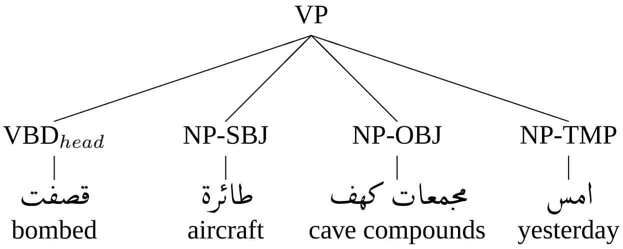

Figure 6: The verb is correctly identified as the head of the VP.

this, we apply a similar technique to that used in section 4.1: we first enumerate a list of heuristics to identify likely arguments and adjuncts, then handle the exceptional cases with a Maximum Entropy model trained over the the cases predicted by the heuristics. A selection of the heuristics used can be found in figure 9. These heuristics can account for 95.06% of the constituents in the corpus. The rest are predicted by another Maximum Entropy classifier. The features for the argument-adjunct classifier are identical to those of the head finder. The output of this step is shown in figure 8.

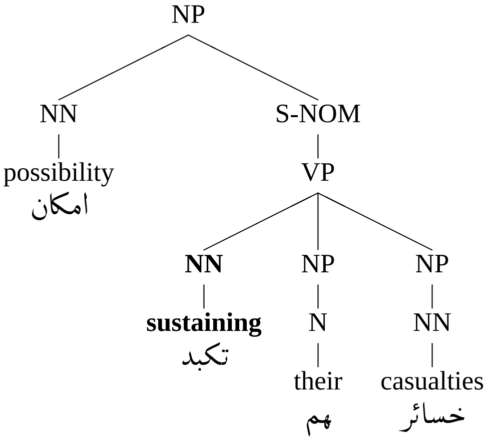

Figure 7: A fragment of ANN20020115.0001.10, meaning *the possibility of their sustaining casualties*. While this annotation is grammatically insightful, this leads us to the uncomfortable decision of identifying an NN category as the head of VP.

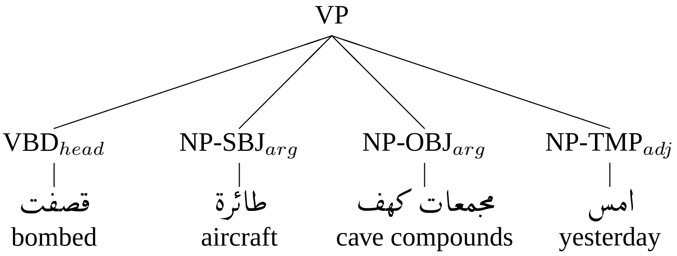

Figure 8: The non-head constituents are labeled as arguments or adjuncts.

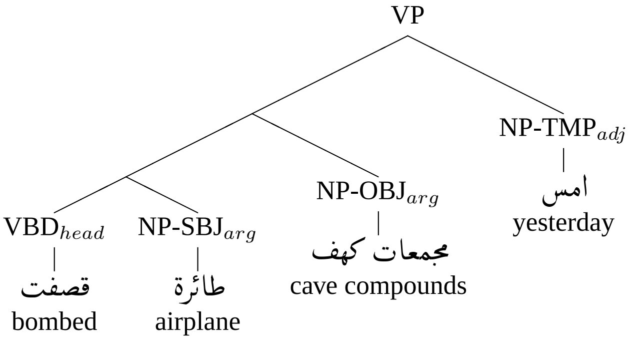

Figure 10: The phrase is binarized, starting from the head.

4.3. Binarization

The third step in the pipeline is to binarize the trees in the treebank. To accomplish this, we first attach each node to the left of the head in turn, then each node to the right, in a manner analogous to the English CCGbank. An example of this process is shown in figure 10. Notice that some of the nodes do not have grammatical categories – this is because they correspond to spans that are not annotated in the Penn Arabic Treebank. In the next step, these nodes will be assigned CCG categories along with all the others.

4.4. Conversion to CCG categories

The final step in the conversion process is to turn each node in the tree from a traditional grammatical category into a CCG category. To achieve this, we will assign a starting category to the root node of each sentence (usually S). We will then walk down the tree, undoing forward and backward applications on each node, informed by their relative position to their head and the nature of the daughter (argument or adjunct). We recursively walk down the tree until we reach the terminal nodes. An example of this process over a single constituent is shown in figure 11.

5. Arabic-Specific Challenges

Arabic, like English, is not so simple that the general algorithm is completely sufficient to maintain faithfulness to linguistic reality. Therefore, we must give special treatment to some exceptional cases.

5.1. Subject Topicalization

The standard word order for Arabic is VSO. In many cases, however, the subject is fronted to provide extra emphasis. Because of this, the basic algorithm will predict two categories for each verb: one for VSO sentences, and one for

Parent Category	Argument Categories	Adjunct Categories
ADJP		JJ PP
VP	NP-SBJ NP-OBJ NP-CLR -PRD	-ADV -TMP
NP		DT JJ PP
S	NP-SBJ NP-TPC	PP -TMP
PP	NP	

Figure 9: A sample of the heuristics used to sort arguments from adjuncts.

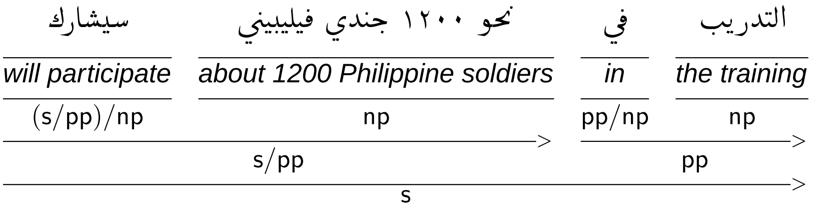

Figure 12: The prediction of the basic algorithm over a simple sentence in the basic VSO word order of Arabic meaning "about 1200 Philippine soldiers will participate in the training". Note the category for *will participate*, which subcategorizes for an np subject and a pp object.

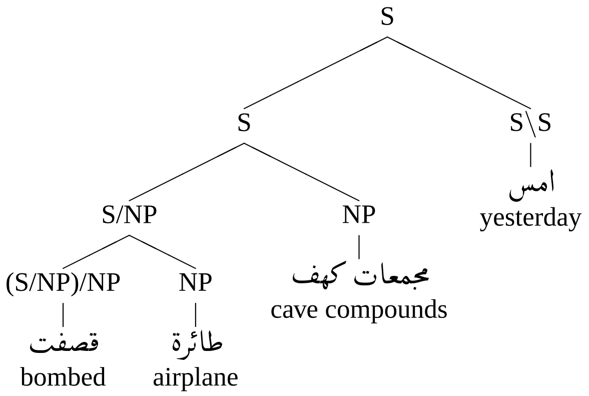

Figure 11: Starting with an s category, we recursively walk down the tree assigning CCG categories.

category	arg	from	to
(s/pp)/np	2	سيشارك	نحو ١٢٠٠ جندي فيليبيني
(s/pp)/np	1	سيشارك	في
pp/np	1	في	التدريب

Figure 13: The dependency graph generated by the analysis in figure 12.

category	arg	from	to
(s/pp)/np	2	سينضمون	نحو ٦٥٠ جنديا اميركيا
(s/pp)/np	1	سينضمون	الى
pp/np	1	الى	قوات فيليبينية

Figure 16: The dependency graph for an SVO sentence using a unary rule on the verb. The full analysis is shown in figure 15. Notice the similarities between this dependency graph and the dependency graph generated by the VSO sentence in figure 14.

SVO (figures 12 and 14), even though intuition tells us that these verbs have similar distribution. Multiplying the categories of every verb in the corpus could cause serious data sparsity issues. Compounding this problem is the unusual case of VOS word order (figure 17), which usually occurs when the pronominal object is attached to the verb.

Our solution is to introduce a unary rule that allows us to topicalize the subject, applied whenever the PATB indicates a subject trace. The rule would convert categories like (s/np$_1$)/np$_2$ into categories like (s\np$_2$)/np$_1$. This analysis is shown in figure 15. Modifying the category in this way (as opposed to generating a different category at the lexical level) handles the combinatorics of subject topicalization, preserves the constituent bracketing of the PATB, and eliminates the data sparsity issue by generating dependencies over the VSO word order (see the dependency graph in figure 16).

5.2. Noun Constructs

Noun constructs, or *iḍaafa* structures, are similar to genitive constructions in other languages. They are used to describe a variety of noun-noun relations, including identity (*the city of Jerusalem*), partitivity (*the best conditions*), possession (*the father of Hasan*) and agency (*the crowing of the rooster*) (Ryding, 2005). The second word of a two-word noun construct is always in the genitive case and, syntactically, behaves like a nominal modifier. The basic algorithm correctly captures this linguistic intuition.

Noun constructs can even appear recursively. Consider the phrase meaning "the knowledge of the cause of the accident" (shown in figure 18). The unlabeled dependencies yielded by this derivation are correct (*accident* modifies *cause* rather than *knowledge*), but we are in the somewhat awkward position of modeling the same phenomenon two different ways (one as a noun modifier, and the other as a noun modifier modifier). While this may be acceptable

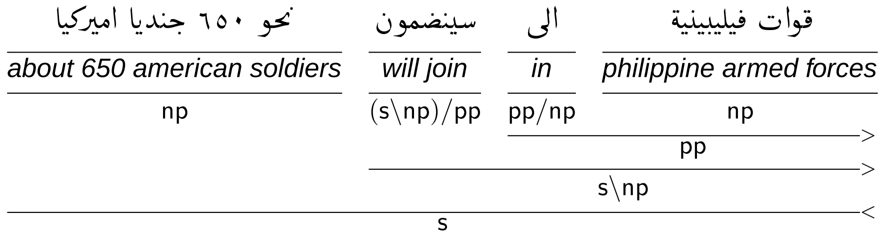

Figure 14: The prediction of the basic algorithm over a sentence meaning “about 650 American soldiers will join with the Philippine army”. Notice that the verbal category, like the sentence in figure 14, subcategorizes for an np subject and pp object, but the two verbal categories are not identical.

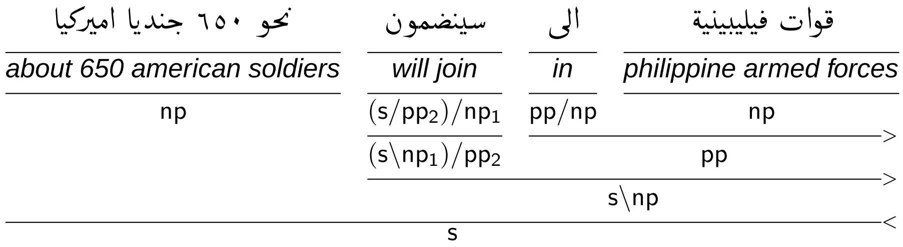

Figure 15: Incorporating a unary rule to account for SVO word order enables us to maintain consistent labeled dependencies across verbs in SVO and VSO sentences.

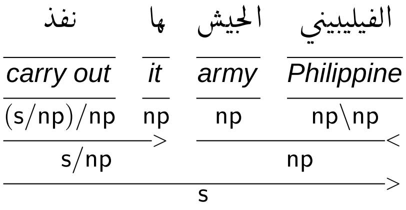

Figure 17: The prediction of the basic algorithm on the somewhat exceptional case of VOS word order, where the object is generally a pronominal suffix. The sentence means *the Philippine army carried it out.*

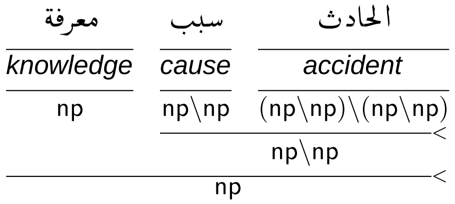

Figure 18: A nested construct phrase meaning “the knowledge of the cause of the accident”. Notice that *accident* modifies *reason*, not *knowledge* The Penn Arabic Treebank correctly represents this in the constituency of the phrase, and we must take care to represent it in the dependencies.

with three-part constructs, these constructions can grow to an arbitrary size: consider “the celebration of the planting of a cedar tree” (four parts) or “the application of all of the resolutions of the Security Council” (five parts) (Ryding, 2005). The problem of modifier category proliferation could quickly becomes an issue. For now, because four and five part constructs are relatively uncommon, we will allow the basic algorithm to predict larger categories, and revisit this decision before the final release if the modifier category proliferation becomes unmanageable.

5.3. Relative Clauses

One appealing feature of CCG in English is its ability to resolve the gaps in relative clauses to their corresponding nouns. In the relative clause example given earlier in figure 3, the subject dependency of *ate* is resolved to *the man* because the relativizer selects for an incomplete sentence s\np. This unfilled dependency in the relative clause can then be passed up to *the man* through a clever coindexation scheme. This dependency is important to us because the nouns that relative clauses modify almost always bear semantic roles, and our English semantic role labeler (Boxwell et al., 2009) relies heavily on these syntactic dependencies.

The handling of relative clauses in Arabic is complicated by resumptive pronouns. Consider the derivation of the sentence meaning “The group that Osama bin Laden leads” (figure 19). In English, the relativizer would be of the category (np\np)/(s\np), enabling it to pass the object dependency up to *group*. In Arabic, however, the resumptive pronoun appears instead of a gap, causing there to be no unfilled dependency to pass along. This is further complicated by the fact that subject pronouns are usually dropped, including resumptive ones.

This presents us with a dilemma: do we allow the basic algorithm to draw a dependency between the verb and the resumptive pronoun (figure 19), or do we treat the resumptive pronoun as semantically and syntactically null and force the dependency up to the noun that the relative clause modifies (figure 20)?

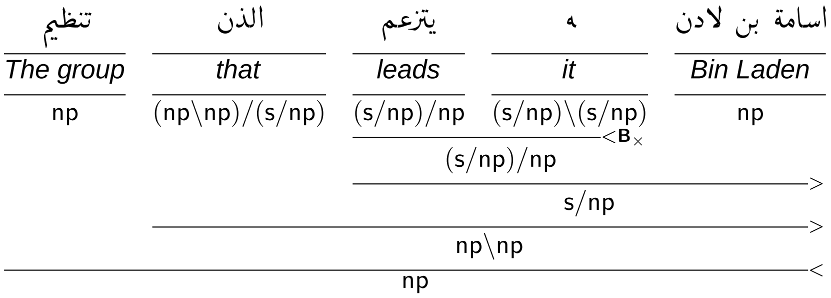

Figure 20: Another possible analysis of Arabic relative clauses. This is appealing because it enables a dependency to be drawn from *leads* to *network*, preserving a key feature of CCG.

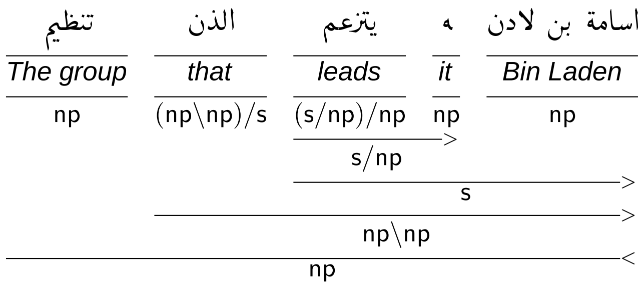

Figure 19: The prediction of the basic algorithm for an Arabic relative clause. Notice that the presence of the resumptive pronoun blocks the construction of a syntactic dependency between *leads* and *network*, as the subordinate clause has no gap.

The first option (ignoring the noun that the relative clause modifies) is certainly simpler. That simplicity, however, comes at the cost of one of CCG's most appealing features, making it unacceptable for our purposes. While it may make sense for the resumptive pronoun to be linked to the proper slot on the verb (as it is essentially a grammatical placeholder), that particular dependency may not be useful for NLP applications – certainly less useful than a dependency between the verb and the noun that the relative clause modifies.

The second option, which we choose, is treating the resumptive pronoun as a syntactic adjunct. This is appealing because it allows us to reserve a dependency for the noun that the relative clause modifies. It does, however, require us to hypothesize a rather unusual category for a pronoun. Claiming that this is a valid category for a pronoun could compound parsing problems caused by incorrect verbal tags. We feel, however, that the presence of a relativizer provides a sufficiently strong predictive mechanism for supertagging purposes, and the benefit of the useful dependency outweighs the cost of the special category.

5.4. Optional Pronominal Subjects

Like many languages, Arabic exhibits strong tendencies towards optional pronominal subjects, or "pro-drop". In cases where the the subject of the verb is obvious from context or verbal agreement features, the subject pronoun is usually dropped. This is potentially problematic for CCG, which requires verbs to explicitly state their combinatorics on their lexical category. Consider the examples in figures 21 and 22. If we allow the basic algorithm to predict the lexical categories without interference, the two instances of the same verb will have slightly different syntactic categories, even though intuition tells us that the verbs are identical and that their subcategorization frames are the same. Treating them as having different categories misses an obvious generalization and could lead to data sparsity errors downstream.

Another solution is to introduce a unary rule to remove a single argument from verbal categories, which could be applied optionally. This would allow us to use the same verbal category in both contexts, which is appealing because it would facilitate the automatic assignment of CCG categories for parsing. It would also keep our dependencies consistent, effectively addressing the issue of data sparsity for parsing applications that rely on CCG dependencies.

A potential weakness of this approach is the possibility of repeated application of the rule. At parse time, the pro-drop rule could essentially discharge the outermost argument for free (say, with (s/np)/np going to s/np). There is nothing, however, stopping the rule from applying again (since a pro-dropped transitive verb looks like a non-pro-dropped intransitive verb). This could cause grievous damage to parser accuracy and efficiency.

To block this, it is necessary to explicitly mark case on our pro-drop rule. For example, a transitive verb would have a category (s/np[acc])/np[nom], and the pro-drop rule would only remove an outermost nominative np. Although case is not always explicitly marked in written text, it is provided by the treebank annotators, which is enough to accurately predict verbal categories and constrain parsing, even if the case of specific nouns is at first uncertain at parse time.

6. Conclusions and Future Work

At the time of writing, only a handful of sentences fail to resolve into well-formed CCG derivations, usually as a result of exceptional annotation of argument cluster coordination or ambiguous PP attachments to coordinated phrases. Ongoing work involves resolving these cases and incorporating crossing-composition combinators to combat modifier category proliferation.

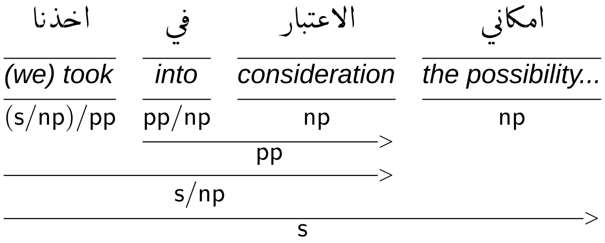

Figure 21: A fragment of an Arabic sentence meaning “we took into consideration the possibility of their suffering damage”. Because the pronominal subject is redundant given the verbal inflection, the subject is dropped. This sentence is based on ANN20020115.0001.10.

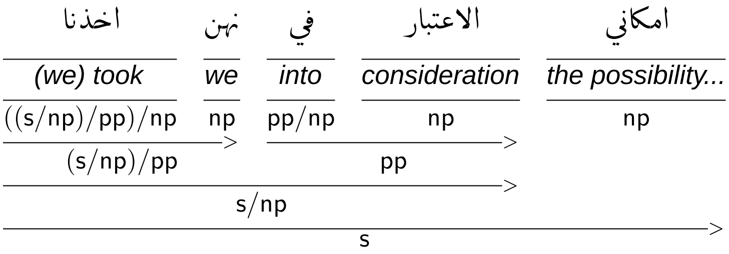

Figure 22: An Arabic sentence equivalent in meaning to the one in figure 21, but with the subject explicit (which adds emphasis). Notice that the verbs in the two sentences have different categories.

7. Acknowledgments

This work was funded by NSF grant IIS-0347799 and the Targeted Investment Initiative of the College of Humanities and the OSU linguistics department. We would like to thank Dennis Mehay, Markus Dickinson, Nizar Habash, Matthew Honnibal, Judith Tonhauser, and Mike White for their helpful comments and feedback. We would also like to thank Abderrahim Slaouihakim for his expert consultations on Arabic grammar – all mistakes are ours.

8. References

Stephen A. Boxwell, Dennis N. Mehay, and Chris Brew. 2009. Brutus: A semantic role labeling system incorporating CCG, CFG, and Dependency features. In *Proc. ACL-09*.

R. Çakıcı. 2005. Automatic induction of a CCG grammar for Turkish. In *ACL Student Research Workshop*, pages 73–78.

N. Habash and O. Rambow. 2004. Extracting a Tree Adjoining Grammar from the Penn Arabic Treebank. In *Proceedings of Traitement Automatique du Langage Naturel (TALN-04)*.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

J. Hockenmaier. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the ACL*, volume 44, page 505.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004a. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109. Citeseer.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004b. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2008. Enhancing the arabic treebank: a collaborative effort toward new annotation guidelines. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC’08)*, Marrakech, Morocco, May.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2009. Creating a Methodology for Large-Scale Correction of Treebank Annotation: The Case of the Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

K.C. Ryding. 2005. *A reference grammar of modern standard Arabic*. Cambridge University Press.

Otakar Smrž, Viktor Bielický, Iveta Kouřilová, Jakub Kráčmar, Jan Hajič, and Petr Zemánek. 2008. Prague arabic dependency treebank: A word on the million words. In *Proceedings of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Marrakech, Morocco.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Lamia Tounsi, Mohammed Attia, and Josef van Genabith. 2009. Automatic treebank-based acquisition of Arabic LFG dependency structures. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, pages 45–52, Athens, Greece.

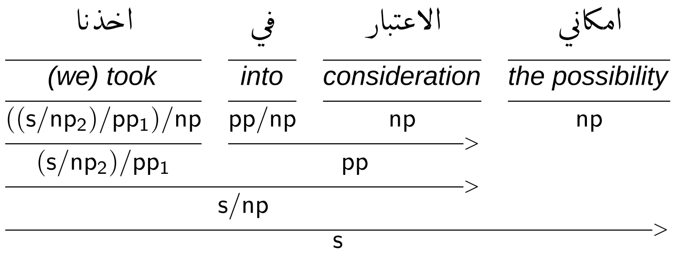

Figure 23: Introducing a unary rule allows us to use the same verbal category for both instances of a verb.