Arabic Parsing Using Grammar Transforms

Lamia Tounsi and Josef van Genabith

NCLT, School of Computing, Dublin City University, Ireland
{ltounsi, josef}@computing.dcu.ie

Abstract

We investigate Arabic Context Free Grammar parsing with dependency annotation comparing lexicalised and unlexicalised parsers. We study how morphosyntactic as well as function tag information percolation in the form of grammar transforms (Johnson, 1998, Kulick et al., 2006) affects the performance of a parser and helps dependency assignment. We focus on the three most frequent functional tags in the Arabic Penn Treebank: subjects, direct objects and predicates . We merge these functional tags with their phrasal categories and (where appropriate) percolate case information to the non-terminal (POS) category to train the parsers. We then automatically enrich the output of these parsers with full dependency information in order to annotate trees with Lexical Functional Grammar (LFG) f-structure equations with produce f-structures, i.e. attribute-value matrices approximating to basic predicate-argument-adjunct structure representations. We present a series of experiments evaluating how well lexicalized, history-based, generative (Bikel) as well as latent variable PCFG (Berkeley) parsers cope with the enriched Arabic data. We measure quality and coverage of both the output trees and the generated LFG f-structures. We show that joint functional and morphological information percolation improves both the recovery of trees as well as dependency results in the form of LFG f-structures.

1. Introduction

Arabic is a Semitic language well-known for its morphological richness and syntactic complexity. Parsing Arabic sentences is a difficult task for several reasons including the relatively free word order of Arabic, the length of sentences, the omission of diacritics (vowels) in written Arabic and the frequency of pro-drop phenomena. The main objective of our research is to automatically enrich Arabic Penn Treebank (ATB) trees and ATB-trained parser output with dependency information such as ATB functional tags. Then, based on these and categorical and configurational information, we automatically annotate trees with LFG f-structure information representing predicate-argument structure relations to produce ATB-based LFG resources for parsing and generation similar to previous work on English by (Cahill and van Genabith, 2006; Cahill et al., 2008). We investigate Arabic parsing comparing lexicalised and unlexicalised parsers. We focus on ATB grammatical function tag assignment (Habash et al., 2009) using grammar transforms for different configurations, making use of morpho-syntactic information to detect subjects, direct objects and predicates. The paper is structured as follows: Section 2 describes the general background. Section 3 presents the parsers and grammar transforms focusing on the three most frequent Functional Tags in the ATB: subject, direct object and predicate. Section 4 discusses the results from a series of experiments conducted on parsers trained on the tranformed ATB, measuring quality and coverage of the output trees and the generated LFG f-structures.

2. General Background

Deep probabilistic constraint-based grammars such as LFGs can be acquired from treebanks. In fact, many treebanks (such as the Penn Treebank) contain deep linguistic information such as: traces, coindexation and functional tags to support the computation of meaning representations in the form of predicate-argument-adjunct structures or deep dependency representations.

2.1. Lexical Functional Grammar (LFG)

Lexical functional Grammar belongs to the family of constraint-based grammars (Kaplan and Bresnan, 1982; Bresnan, 2001) and features two levels of representation: c(onstituent)-structure and f(unctional)-structure.
C-structure describes surface grammatical configurations using phrase structure trees while f-structure encodes more abstract functional information as a matrix of attribute-value pairs to reduce the impact of surface configuration in the grammar.

2.2. Penn Arabic Treebank (ATB)

The Penn Arabic Treebank (Maamouri and Bies 2004) is a corpus of 23,611 parse-annotated sentences from Arabic newswire text in Modern Standard Arabic (MSA). The ATB is a fine-grained corpus, its annotation includes 22 phrasal tags, 20 individual functional tags and 24 basic POS-tags[1] (with a total of 497 different POS tags with morphological information). In addition, the ATB involves empty nodes to capture pro-drop as well as non-local dependencies (NLDs). The full POS tagset with morphological information indicates case, mood, gender, definiteness, etc.

2.3. LFG Arabic Annotation Algorithm (A^3)

The Arabic LFG f-structure annotation algorithm exploits syntactic, categorical, configurational and ATB functional information to automatically annotate the ATB with abstract LFG functional information (basic predicate-argument structures) (Tounsi et al., 2009). The methodology has originally been developed for English (Cahill et al., 2004) and extended to other languages including German, Chinese, Spanish and French. For ATB gold trees the A^3 achieves an f-score of 95% against a gold standard f-structure bank (Al-Raheb et al,. 2006).

[1] ATB annotation uses Tim Buckwalter's morphological analyser to generate POS values for each word/token of a sentence.

3. Arabic Parsing Using Grammar Transforms

A number of parsers for Arabic use handcrafted grammars (Ditters 2001, Zabokrtsky and Smrz 2003, Othman et al. 2003, Attia 2008). However none of these result in a large-scale parsing application for Arabic. Consequently, we decided to apply the Bikel (implementation of Collins Model 3 (Bikel, 2004)) and Berkeley[2] (automatic latent variable PCFG grammar induction (Petrov and Klein, 2007)) probabilistic parsing engines to Arabic CFG parsing, using the automatic LFG annotations methodologies and gold POS parsing. We conduct our experiments on the ATB. We split the data into three disjoint sets, 80% for training, 10% for development and 10% for testing. 33% of the test set sentences are sentences of length $\succ$ 40 tokens.

3.1. Grammar Transforms

Treebank-based parsing consists on "learning" a model given a treebank and using the model for "computing" the best parse on unseen text. (Johnson, 1998) improved the performance of PCFG parsing by enriching treebank nonterminal categories with information about the context in which they occur and (Klein and Manning, 2003) explored manual and automatic extension of the traditional linguistic categories into a richer category set to enhance PCFG parsing results. Traditionally, treebank-based (P)CFG parsing abstracts away from much of the information encoded in the treebank (traces, coindexation, functional tags, etc) concentrating exclusively on the CFG skeleton of the treebank trees. However, in particular for morphologically rich languages, morphological and functional information is important to optimally recover syntactic structure. What is more, the LFG A^3 crucially relies on morphological information and functional tags in ATB trees. In order to capture the 20 functional tags present in the ATB in parser output, our methodology applies "masking" and "percolation" of information through the grammar and training data used by the parsers. In addition we use grammar transformation-based case percolation.

3.2. Function Tag Masking and Unmasking

Function tag masking is a data transformation. We collapse functional and phrasal tags by fusing them into a single symbol: e.g. NP-SBJ becomes NPSBJ, effectively inflating the CFG category set. In the ATB, nonterminals can receive up to three functional tags. Consequently, the size of the phrasal tag set used by the parser increases from 20 tags to 150 tags. The parser is then updated and trained on this new data. After parsing, we unmask the function tags to make them available to the LFG A^3.

[2]Originally, the ATB sentences follows Buckwalter transliteration scheme, which is a 1-to-1 transliteration of MSA orthographic symbols using ASCII characters including special characters such as: *,$\succ$, $\approx$, upper and lower case (Buckwalter, 2004). However, the default package used by the Berkeley parser is case-sensitive. Consequently, we provide a new encoding based on lower case characters only. In addition, the current implementation of the parser requires CONLL format for the test set.

3.3. Case Percolation

As Arabic is morphologically rich, a lot of information is present at the leaves of the trees in the ATB. We percolate morphological information bottom-up in the trees to help grammatical function assignment. We focus on the three most frequent functional tags in the ATB: -SBJ, -OBJ, -PRD.[3] Case percolation aims to improve the determination of subject, object and predicate constituent(s) among the syntactic structures identified in the parse tree. Arabic has three grammatical cases: nominative, genitive and accusative. Except when they are governed by an overt copula or a subordinating conjunction, -SBJ and -PRD are nominative and -OBJ is accusative (Habash et al. 2005). Adding case information to POS tag increases the size of the POS tag set to 40 tags. e.g. the POS NN is expended to NN, NN_nom and NN_acc).

Figure 1 shows the (unmaked) output tree provided by the parser, trained on a version of the ATB which has undergone ATB function tag masking and case percolation, for sentence (1). Each node in the tree is assigned an f-structure equation using A^3. The subject NP receives '$\uparrow$ SUBJ = $\downarrow$' and the predicate which subcategorises for a copula complement receives '$\uparrow$ PRED = 'null_be', $\uparrow$ XCOMP = $\downarrow$, $\uparrow$ SUBJ= $\downarrow$ SUBJ'. The resolution of the equations produces the f-structure shown in Figure 2. Note that the subject of the matrix clause is coindexed with the subject of the embedded clause .

(1) الشمس مشرقة
Al$amosu mu$riqapu
the-sun bright.
The sun is bright.

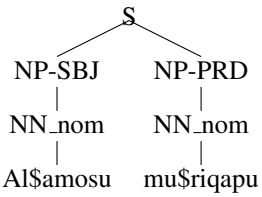

Figure 1: Parser output tree for sentence (1).

$$\begin{bmatrix} \text{PRED} & \text{null\_be}\langle \text{SUBJ , XCOMP} \rangle \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'Al\$amosu'} \\ \text{CASE} & \text{nom} \\ \text{DEF} & + \end{bmatrix} \boxed{1} \\ \text{XCOMP} & \begin{bmatrix} \text{PRED} & \text{'mu\$riqapu'} \\ \text{CASE} & \text{nom} \\ \text{DEF} & - \\ \text{SUBJ} & [\ ]\ \boxed{1} \end{bmatrix} \end{bmatrix}$$

Figure 2: F-structure for sentence (1).

[3]-SBJ, -OBJ and -PRD represent 68% of ATB functional tags tokens.

4. Experimental Results

Table 1 shows the results of a series of experiments conducted on parsers trained on the ATB for different transformations as follows:
- Column 1: parsers trained on the original ATB.
- Column 2: functional and phrasal tags merged for nonterminals and parsers trained on converted ATB.
- Column 3: functional and phrasal tags merged for nonterminals and nominative case percolated in the trees for the functional tag -SBJ.
- Column 4: functional and phrasal tags merged for nonterminals and nominative case percolated in the trees for the functional tags -SBJ, -PRD and accusative case is percolated for the functional tag -OBJ. Grammar 1-6 refers to the number of split-merge cycles involved in the Berkeley parser training procedure.

We measure quality and coverage of the output trees using the standard EVALB (Sekine and Collins 1997). Note that we ignore punctuation and we evaluate unmasked trees (functional information is not considered for the evaluation). We achieve best labeled bracketing f-scores of 73.58 for Bikel and 71.56 for Berkeley CFG trees and an LFG f-structure dependency f-score of 88.95 for Bikel and 88.5 for Berkely. As expected, grammar transforms have a positive effect on both parsers. The effect on both is quite similar, even if the absolute f-score improvements are not very large. In fact, the methodology presented in this paper does not confuse the parsers and improves the recovery of functional information (for the A^3 to produce f-structures).

5. Error Analysis

For both parsers, the noise comes from the data set itself. In fact, the ATB suffers from accusative-genitive case underspecification/ambiguity for feminine and masculine plurals. Both cases are given to nouns where the inflectional morphology is the same for both accusative and genitive cases. This issue leads the parsers to overuse the functional tag -OBJ. More precisely, the tag -OBJ is often duplicated in flat trees for both accusative and genitive nouns. We have also found a confusion between -SBJ and -PRD in nominal sentences.[4] In fact, the parsers never assign the tag -PRD before the tag -SBJ. In addition, when a sentence starts with an overt copula or a subordinating conjunction, the parsers analyse the sentence as a verbal sentence and mix-up the functional tags -PRD and -OBJ. On the other hand, Bikel output trees are better than Berkeley output trees, perhaps because Bikel's probability model is enriched with lexical information.

6. Conclusion and Future Directions

In this paper, we have shown that morpho-syntactic information helps to detect grammatical functions in probabilistic approaches to parsing Arabic. We used (i) function tag masking to include functional information in the phrasal tagset and (ii) case percolation to better identify subject, object and predicate constituent(s) among the syntactic structures identified in the parse tree. The work presented in this paper is a first step on function labeling methodologies using grammar transforms. In the future, we aim to use machine-learning-based ATB function labelers to provide input to the annotation algorithm.

[4] A nominal sentence in Arabic has no verb and is composed of a subject and predicate phrase. Usually, the subject precedes the predicate but a swap is also possible.

7. References

Y.Al-Raheb, A.Akrout, J.van Genabith, J.Dichy. 2006. *DCU 250 Arabic Dependency Bank: An LFG Gold Standard Resource for the Arabic Penn Treebank* The Challenge of Arabic for NLP/MT at the British Computer Society.

M.Attia. 2008. *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation*. Ph.D. Thesis. University of Manchester, UK.

D.Bikel. 2004. *Intricacies of Collins' parsing model* Computational Linguistics, 30(4), 2004.

J.Bresnan. 2001. *Lexical-Functional Syntax* Blackwell, Oxford.

A.Cahill, M.Burke, R.ODonovan, J.van Genabith, A.Way. 2004. *Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations*. In Proceedings of ACL-04, pages 320-327, Barcelona, Spain.

A.Cahill, J.van Genabith. 2006. *Robust PCFG-Based Generation using Automatically Acquired LFG Approximations*. In Proceedings of ACL-06, pages 1033-1040, Sydney, Australia.

A.Cahill, M.Burke, R.O'Donovan, S.Riezler, J.van Genabith, A.Way. 2008. *Wide-Coverage Deep Statistical Parsing using Automatic Dependency Structure Annotation*. Computational Linguistics, Vol. 34, No. 1.

E.Ditters. 2001. *A Formal Grammar for the Description of Sentence Structure in Modern Standard Arabic*. Workshop on Arabic Processing, ACL/EACL'01.

N.Habash, G.Ryan, R.Owen, K.Seth, M.Marcus. 2007. *Determining Case in Arabic: Learning Complex Linguistic Behaviour Requires Complex Linguistic Features*. In Proceedings of EMNLP'07, Prague, Czech Republic.

N.Habash, F.Reem and R.Ryan. 2009. *Syntactic Annotation in the Columbia Arabic Treebank*. In Proceedings of MEDAR'09.

M.Johnson 1998. *PCFG models of linguistic tree representations*. Computational Linguistics, 24(4), pages 613-632.

R.Kaplan , J.Bresnan. 1982. *Lexical Functional Grammar, a Formal System for Grammatical Representation*. The Mental Representation of Grammatical Relations.

D.Klein, C. Manning. 2003. *Accurate Unlexicalized Parsing*. In Proceedings of ACL03, Sapporo, Japan.

S.Kulick S, G.Ryan, M.Mitchell. 2006. *Parsing the Arabic Treebank: Analysis and improvements*. In Proceedings of the Treebanks and Linguistic Theories Conference, Prague.

M.Maamouri, A.Bies. 2004. *Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools* Proceedings of COLING 2004, Geneva, Switzerland.

E.Othman, K.Shaalan, A.Rafea. 2003. *A Chart Parser for Analyzing Modern Standard Arabic Sentence* In proceedings of the MT Summit IX Workshop on Machine Translation for Semitic Languages: Issues and Approaches, Louisiana, USA.

S.Petrov and D.Klein. 2007. *Improved inference for unlexicalized parsing* In Proceedings of HLT-NAACL'07,pages 404-411, New York, USA.

S.Sekine and M.J. Collins. 1997. *EVALB bracket scoring program* web page, http://nlp.cs.nyu.edu/evalb/.

L.Tounsi, M.Attia, J.van Genabith. 2009. *Automatic Treebank-Based Acquisition of Arabic LFG Dependency Structures* In Proceedings of EACL'2009, Athenes, Greece.

Z.Zabokrtsky, O.Smrz. 2003. *Arabic syntactic trees: from constituency to dependency* In Proceedings of EACL'03, pages 183-186, Budapest, Hungary.

F-score	**basic**	**masked**	**masked+ percol -SBJ**	**masked+ percol -SBJ -OBJ -PRD**
Berkeley				
Grammar 1 C-structure	64.10	64.10	64.48	64.84
Grammar 2 C-structure	67.75	67.75	68.43	68.83
Grammar 3 C-structure	69.63	69.63	70.23	70.65
Grammar 4 C-structure	70.86	70.86	71.19	71.48
Grammar 5 C-structure	71.31	71.31	71.78	**71.82**
Grammar 5 F-structure	n/a	87.98	88.50	**87.60**
Grammar 6 C-structure	70.99	70.99	71.34	71.56
Bikel				
C-structure	72.40	72.70	73.51	**73.58**
F-structure	n/a	87.09	88.53	**88.95**

Table 1: C-structure and F-structure evaluation for all sentence lengths.

F-score	**basic**	**masked**	**masked+ percol -SBJ**	**masked+ percol -SBJ -OBJ -PRD**
Berkeley				
Grammar 1 C-structure	67.97	67.97	68.10	69.01
Grammar 2 C-structure	71.18	71.18	72.05	72.72
Grammar 3 C-structure	73.02	73.02	73.94	74.46
Grammar 4 C-structure	74.37	74.37	74.69	75.26
Grammar 5 C-structure	75.07	75.07	75.37	**75.43**
Grammar 5 F-structure	n/a	87.50	88.68	**88.12**
Grammar 6 C-structure	74.61	74.61	74.68	74.98
Bikel				
C-structure	75.61	75.85	76.53	**76.53**
F-structure	n/a	87.12	89.20	**88.63**

Table 2: C-structure and F-structure evaluation for sentences $\leq$ 40 tokens in length.