One-Step Statistical Parsing of Hybrid Dependency-Constituency Syntactic Representations

Kais Dukes
Institute for Artificial Intelligence
University of Leeds, United Kingdom
sckd@leeds.ac.uk

Nizar Habash
Center for Computational Learning Systems
Columbia University, New York, USA
habash@ccls.columbia.edu

Abstract

In this paper, we describe and compare two statistical parsing approaches for the hybrid dependency-constituency syntactic representation used in the Quranic Arabic Treebank (Dukes and Buckwalter, 2010). In our first approach, we apply a multi-step process in which we use a shift-reduce algorithm trained on a pure dependency preprocessed version of the treebank. After parsing, the dependency output is converted into the hybrid representation. This is compared to a novel one-step parser that is able to learn the hybrid representation without preprocessing. We define an extended labelled attachment score (ELAS) as our performance metric for hybrid parsing, and report 87.47% (F1 score) for the multi-step approach, and 89.03% (F1 score) for the one-step integrated algorithm. We also consider the effect of using different sets of morphological features for parsing the Quran, comparing our results to recent work on Modern Standard Arabic.

1 Introduction

Research into statistical parsing for English has resulted in over a decade of high-performance results (Collins, 1999; Charniak, 2000), with most work focusing on the Penn English Treebank (Marcus et al., 1993). However, adapting these parsing models to other languages has been less successful. Results from the CoNLL-X shared task on multilingual dependency parsing showed that Arabic is one of the most challenging languages to parse, due to its rich morphology and relatively free word order (Nivre et al., 2007b).

In this paper we consider statistical parsing for Classical Arabic, the direct ancestor of Modern Standard Arabic (MSA). As a training and test data source, we use the online Quranic Arabic Corpus (http://corpus.quran.com). This website is a useful study aid for understanding the Quran through grammatical annotation, and is used by 100,000 people monthly. In contrast to other recent annotation efforts for MSA (Maamouri et al., 2004; Habash and Roth, 2009), Quranic Treebank annotators have cross-checked their analyses against historical works based on the traditional Arabic grammar known as *i'rāb* (إعراب) (Salih, 2007; Dukes et al., 2011; Dukes and Buckwalter, 2010). To provide annotation that closely follows traditional grammar, the treebank creators used a hybrid representation that supports relations between morphemes, as well as between phrases, clauses and sentences. The treebank also includes empty nodes for pro-drop and for semantic elision.

This representation presents several challenges to statistical parsing. Possible pipeline approaches include converting to pure constituency or to pure dependency as a preprocessing step. The alternative we pursue is using an integrated model to parse the hybrid representation. For other parsing tasks, integrated models have been shown to out-perform pipeline approaches, e.g., Goldberg and Tsarfaty (2008) integrate morphological and syntactic disambiguation for Hebrew, and report improved parsing performance.

In the next section, we review the Treebank. In Section 3, we survey related work. Section 4 describes our parsing algorithm. We present our parsing approaches and evaluate in Sections 5 and 6, respectively.

2 The Quranic Arabic Treebank

For our parsing experiments we use version 0.5 of the Quranic Treebank, containing 37,578 word-forms (~ 49% of the full Quranic text), segmented into 47,220 tokens (Dukes and Buckwalter, 2010). A common scheme for en-

Proceedings of the 12th International Conference on Parsing Technologies, pages 92–103,
October 5-7, 2011, Dublin City University.

Node	Type	Extent	Form	Tag	Head	Dep	Features
1	T	_	qaAla	V	_	_	PERF|LEM:qaAla|ROOT:qwl|3MS
2	E	_	Huwa	PRON	1	subj	_
3	T	_	ha`*aA	DEM	_	_	LEM:ha`*aA|MS
4	T	_	rab~i	N	3	pred	LEM:rab~|ROOT:rbb|M|NOM
5	T	_	Y	PRON	4	poss	PRON:1S|SUFFIX
6	P	3-5	_	NS	1	obj	_

Figure 1: Example hybrid graph in extended CoNLL-X format. Node types: T = Terminal node, E = Elided word / Empty category, P = Phrase node; Tags: V = Verb, PRON = Pronoun, DEM = Demonstrative, N = Noun, NS = Nominal Sentence; Dependencies: subj = subject, pred = predicate, poss = possessive construction, obj = object.

coding dependency treebanks is the CoNLL-X format (Nivre et al., 2007b). The Quranic Treebank uses an extension of this to support phrases and elided words. Figure 1 shows an example sentence illustrating common tags used in the treebank. The extended format adds two new columns: TYPE indicates the different types of nodes, and EXTENT defines a phrase by specifying start and end terminal nodes. Head nodes, dependency labels, and morphological features are shown in separate columns. On the treebank's website, this information is represented visually using dependency graphs (Figure 2). Read from right-to-left, this graph uses a convention that dependent nodes point towards their heads, and edges are labelled with roles in Arabic. The second Arabic word in parenthesis is an implied elided subject pronoun that has been syntactically annotated, and is not in the original Quranic verse. The last word on the left has been segmented into two tokens, resulting in a total of five terminal nodes in the graph.

This hybrid representation is based on the traditional *i'rāb* analysis found in Salih (2007), a 12-volume work collating previous historical analyses, e.g., the analysis for Figure 2 states:

> In this verse, 'said' is a perfective verb, whose subject is an elided pronoun of the form 'he'. The noun 'lord' is in the nominative case and is the predicate of the demonstrative pronoun 'this'. The suffixed pronoun 'my' attached to the noun is a possessive clitic. The nominal sentence, headed by the demonstrative pronoun, acts as the object of the verb 'said'.

The example sentence shown in Figures 1 and 2 illustrates several linguistic aspects of the Quranic Treebank which make it challenging for statistical parsing: rich morphology, phrase structure, and elision. We discuss various aspects of the Treebank next.

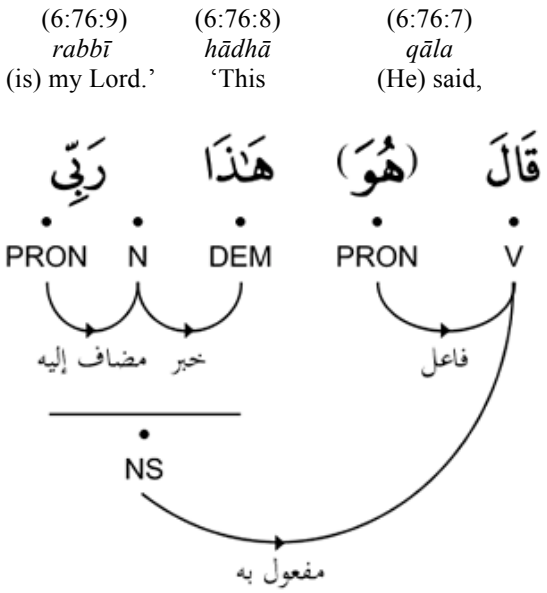

Figure 2: Hybrid graph in presentation form for the annotated example: *He said, 'This is my Lord.'*

Dependency: Although traditional Arabic grammar developed independently from modern European linguistics, it uses the concepts of heads (*āmil*) and dependents (*ma'mūl fī-hi*). Along with Panini's grammar for Classical Sanskrit, it is considered to be one of the origins of modern dependency grammar (Owens, 1988; Bohas et al., 1990).

Morphemes: The basic unit of analysis in traditional Arabic grammar is the morphological segment. Compound word-forms in Classical Arabic are tokenized into independent grammatical units. This agrees with other recent treebanking efforts for MSA such as the Columbia Arabic Treebank (Habash and Roth, 2009), the Prague Arabic Dependency Treebank (Smrž, et al., 2008), and the Penn Arabic Treebank (Maamouri et al., 2004). This also compares with recent parsing work for other morphological rich languages. For example, Eryiğit et al. (2008) show that for Turkish using groups of morphemes as

the basic unit instead of words improves parsing accuracy.

However, in contrast to other Arabic treebanks that define their own segmentation schemes, morphological annotation in the Quranic Treebank closely follows segmentation rules from *i'rāb* (Dukes and Habash, 2010). In addition to part-of-speech, the grammar describes multiple features for each morpheme, including person, gender, number, verb mood, noun case and state. The treebank also includes roots and lemmas. A root is a sequence of three or four radicals that are used to group words with related derivational morphology. The lemma is a further subdivision that groups forms varying only in inflectional morphology (Habash, 2010). In our experiments, we consider how different combinations of these rich morphological features affect the accuracy of statistical parsing.

Phrases: Most dependencies in the treebank relate morphological segments. The treebank also includes dependencies between these units and phrases, and between pairs of phrases. In accordance with traditional Arabic grammar, the Quranic Treebank annotates five phrase types: nominal sentences (جملة اسمية), verbal sentences (جملة فعلية), conditionals (جملة شرطية), preposition phrases (جار ومجرور) and subordinate clauses (تأويل مصدر). There are simple rules for distinguishing between these phrase types in the grammar.

Phrase nodes are used to model constituency structure. In a pure dependency representation, the grammatical relationship between a pair of phrases is implicit in the edge that connects the head words of the two phrases. In the traditional Arabic grammar of the Quran, phrase-level relations such as conjunction and apposition are made explicit in syntactic analysis. Since the grammatical rules that determine these phrase structures allow recursion, the Quranic Treebank includes hybrid graphs that contain multiple levels of nested consistency structure.

Elision: In the Penn English Treebank, empty categories are used for constructions such as null complementizers: 'The man (0) I saw', and for *wh*-movement: 'What$_1$ do you want (NP *T*-1)?'. The Quranic Treebank distinguishes between morphological, syntactic and semantic elision. Like MSA, Classical Arabic is a pro-drop language. A verb's dropped subject pronoun is implied through its morphology. Syntactic elision arises in order to satisfy constraints in the grammar. For example, in certain cases preposition phrases are attached to nouns via an elided adjective. Semantic elision involves elided words, used to explain the reason for case markers in certain verses of the Quran. Dukes, Atwell and Sharaf (2010) provide a more detailed description of elision in the Quranic Treebank. In our parsing experiments in this paper, we handle the three types of elision separately as they form different edge patterns in dependency graphs.

3 Previous Related Work

Related computational work includes statistical parsing for other morphologically rich languages, as well as recent parsing work for hybrid representations, and for recovering elision.

Hebrew, another Semitic language, faces a similar set of challenges in comparison to parsing Arabic. Both feature relatively free word order and require morphological disambiguation for syntactic parsing. For dependency grammar, Goldberg and Elhadad (2010), apply a pipeline approach by disambiguating morphology and syntax in two separate steps. They report a 84.2% labelled attachment score using gold morphological disambiguation, and 76.2% when using automatic morphological analysis.

For Arabic, Kulick et al. (2006) discuss parsing the Penn Arabic Treebank using phrase structure grammar. One conclusion that can be drawn from their results is that parsing using a constituency representation leads to lower accuracy for Arabic in comparison to English. They report a Parseval F1-score of 74% for version 1 of the Penn Arabic treebank, and 88% for English using a similar sized corpus, trained using Bikel's parser (Bikel, 2004).

More recent work for Modern Arabic has focused on dependency grammar. Marton et al. (2010) use MaltParser for parsing the CATiB treebank, and experiment with different combinations of rich morphological features. Like the Quranic Arabic Treebank, CATiB is also based on traditional Arabic grammar, although it uses a subset of the full traditional syntactic roles and only six POS tags. Another related Arabic treebank that uses a dependency representation is the Prague Arabic Dependency Treebank (Smrž, et al., 2008). Hall et al. (2007) use MaltParser to parse ten different languages, including data from the Prague Arabic Treebank. They compare this to an ensemble system that combines six different strategies to boost parsing performance.

In addition to parsing Arabic, MaltParser is ideally suited to parsing morphologically rich languages, due to its integration of flexible feature sets during training (Nivre, et al., 2007a).

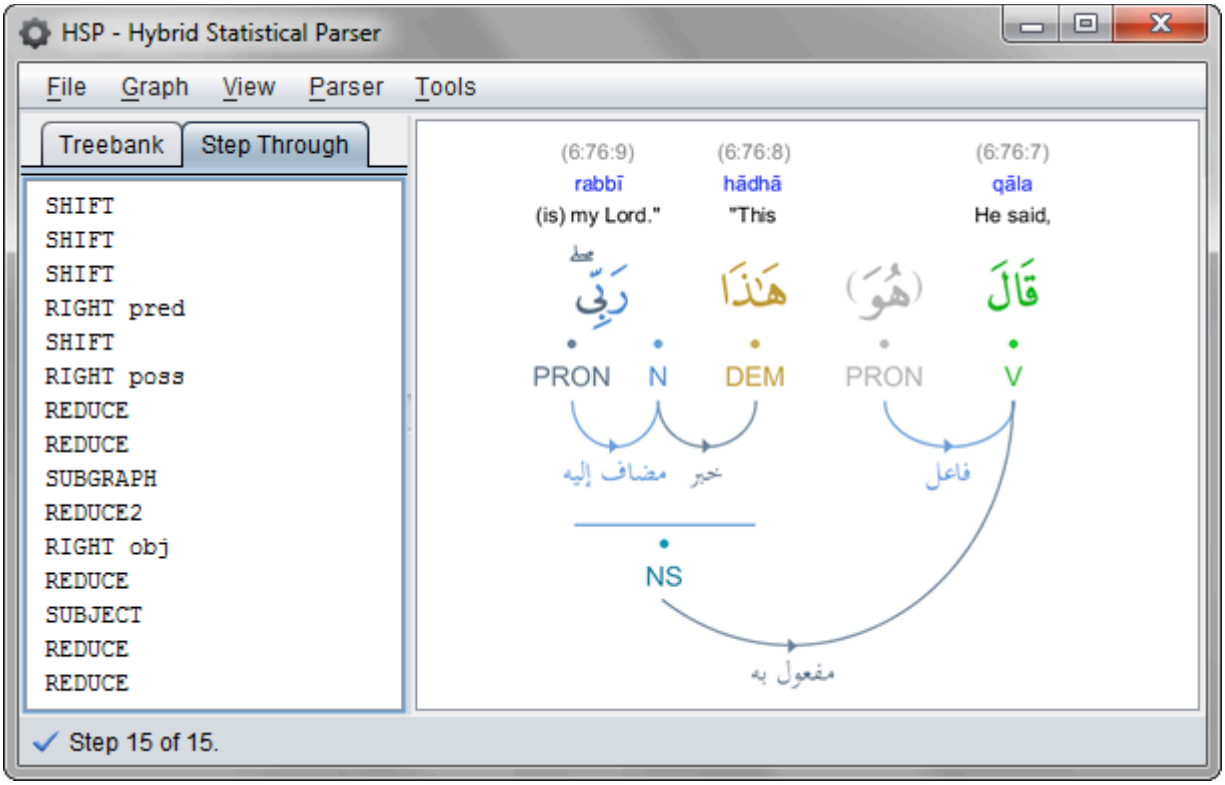

Figure 3: Custom Java application for HSP showing the steps in an example parsing program for Arabic. The various operations on the left panel are described in Section 4.1.

For example, Bengoetxea and Gojenola (2010) use MaltParser for version 2 of Basque Dependency Treebank, although to simplify the syntactic representation this latest version no longer includes empty nodes for ellipsis and coordination. Most statistical parsers do not handle elision. However, Gabbard et al. (2006) show that it is possible to fully recover Penn Treebank-style trees for English including empty categories, by training a cascade of statistical classifiers.

For parsing hybrid representations, Hall and Nivre (2008) adapt MaltParser to the German TIGER and TüBa-D/Z treebanks, reporting a labelled attachment score close to 90%. Similarly, Hall et al. (2007) adapt MaltParser for hybrid constituency-dependency parsing of the Swedish Talbanken05 treebank. These treebanks are hybrid in a different sense to the Quranic Treebank. For each sentence, they include dual annotation in both constituency and dependency grammar, in contrast to combining these into a single representation. A related example is the Hindi/Urdu multi-representational treebank (Bhatt et al., 2009). For the baseline experiment that we describe in section 5, we use a dependency-based encoding similar to the scheme Hall et al. (2007) use for parsing the hybrid German and Swedish treebanks.

4 Hybrid Statistical Parser

In this section we describe the Hybrid Statistical Parser (HSP) that we use for our parsing experiments. Instead of using MaltParser, we implemented a new parser in Java using a similar algorithm, along with a new graphical user interface (Figure 3). We made this decision to allow us to extend the parsing architecture, and created the user interface to help debug the parser. As with previous work that uses MaltParser for other morphologically rich languages, we assume that input sentences to HSP have been tokenized and already annotated with part-of-speech tags and features. We use gold-standard tokenization and POS tags.

HSP outputs both pure and hybrid dependency graphs. Each of these have a formal definition. Let $(t_1, ..., t_n)$ be an input sentence that has been morphologically tokenized, and let R denote the set of dependency relations. A pure dependency graph is defined as $G = (V, E, L)$, where $V = \{t_1, ..., t_n\}$ are the vertices formed from the input tokens, $E \subseteq V \times V$ are the graph's edges, and $L : E \rightarrow R$ are the edge labels. For hybrid graphs, we extend the set of vertices to include phrase nodes. Let $p_{ij} = (t_i, t_j)$ denote the phrase that spans the tokens from t_i to t_j inclusively, and let P denote the set of all possible phrases. We de-

fine a hybrid dependency graph as $G' = (V', E', L')$ where $V' = \{t_1, ..., t_n\} \cup P'$ and $P' \subseteq P$. As before, the edges are $E' \subseteq V' \times V'$ with labels $L' : E' \rightarrow R$. For elision, we further extend the set of vertices to include empty categories as additional terminal nodes.

4.1 Parsing Algorithm

HSP uses a shift-reduce algorithm similar to Nivre's. Two data structures are used for parsing: a stack S for temporary storage and a queue Q to buffer input. In its initial configuration, the parser has all input tokens placed onto the queue with the stack empty: $Q = (t_1, \ldots, t_n)$ and $S = \emptyset$. The parser reads from the queue and finishes when the queue and stack are both empty ($Q = \emptyset \wedge S = \emptyset$). To construct a dependency graph, a sequence of operations are executed by the parser, analogous to the instructions in a computer program. Figure 3 shows the operations used to parse the example sentence used previously in Figures 1 and 2.

In contrast to MaltParser, HSP uses an extended instruction set. To define these operations, let $Q = (q_1, \ldots, q_A)$ and $S = (s_1, \ldots, s_B)$ be the state of the parser at an intermediate stage of the program, and Q' and S' be the state after executing the next instruction. The operations are:

1. SHIFT reads the next token from the queue and moves this to the top of the stack: $Q' = (q_2, \ldots, q_A)$ and $S' = (q_1, s_1, \ldots, s_B)$.

2. REDUCE pops the stack: $S' = (s_2, \ldots, s_B)$.

3. LEFT adds an edge to the graph, with s_1 as the head node and s_2 as the dependent node.

4. RIGHT adds an edge to the graph, with s_2 as the head node and s_1 as the dependent node.

5. REDUCE2 pops the second node on the stack: $S' = (s_1, s_3, \ldots, s_B)$.

6. EMPTY adds an empty node e to the graph after s_1. The elided node e is pushed onto the stack: $S' = (e, s_1, \ldots, s_B)$.

7. SUBJECT is only applicable if s_1 is a verb. An elided pronoun e is inserted after s_1, and a *subj* edge is added with s_1 as the head node, and e as the dependent node. e is pushed onto the stack: $S' = (e, s_1, \ldots, s_B)$.

8. SUBGRAPH adds a phrase node p to the graph spanning the terminal nodes from s_1 to the end of the subgraph with root s_1. p is pushed onto the stack: $S' = (p, s_1, \ldots, s_B)$.

Three of these instructions are parameterized. LEFT and RIGHT take an edge relation $r \in R$, and EMPTY takes a part-of-speech as a parameter. The last four instructions are extensions compared to MaltParser. EMPTY is used to add elided nodes with a specific part-of-speech. SUBJECT is similar to the combination EMPTY then LEFT , but takes into consideration the morphology of the verb to produce a correctly inflected subject pronoun. REDUCE2 is useful in the situation where an edge should be formed between the first and third elements of the stack, so that the second element can be easily discarded. After a SUBGRAPH operation, it is possible to use REDUCE2 to discard the head of the subgraph, which would now be at the second element of the stack. See the parsing run in Figure 3 for an example of this. Only the first four instructions listed above are used for pure dependency graphs. In hybrid mode, the parser uses all eight instructions.

4.2 Machine Learning

Like MaltParser, HSP uses supervised learning during training. For each graph in the training data, an oracle driven by a small set of rules is used to deduce the sequence of actions required to construct the graph. For machine learning, we use support vector machines, implemented by the Java version of LIBSVM (Chang and Lin, 2001). For each step in the parsing programs, a collection of SVM classifiers learn to predict the next operation, given the feature vector associated with the first few nodes at the top of the queue and stack. Feature selection is described in more detail in the next section.

We apply the standard technique of binarization of input features in the training data, so that a single symbolic feature is represented using many binary predicates (Yamada and Matsumoto, 2003). To reduce learning time, the training set is partitioned using the part-of-speech at the top of the stack, and one statistical classifier is trained for each part-of-speech. We use the same LIBSVM settings that Hall and Nivre (2008) use for parsing the German TIGER and TüBa-D/Z treebanks: $\gamma = 0.2$ and $r = 0$ for kernel parameters, $C = 0.5$ for penalty and $\varepsilon = 1$ for termination. We also use the same quadratic kernel:

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^2.$$

5 Parsing Experiments

We compare two approaches to parsing the treebank. First, we use HSP in pure dependency mode and recover the hybrid representation through a post-processing step. The second experiment uses an integrated approach that builds phrase structure and elided nodes during parsing. Both of these experiments are repeated using different sets of morphological features.

5.1 Multi-step Parsing

In our first experiment, we perform the following steps:

1. The training data is converted to pure dependency by encoding additional information using new complex edge labels.

2. In the learning phase, we restrict HSP to using only the four operations that are required for pure dependency parsing: SHIFT, REDUCE, LEFT and RIGHT.

3. The parser's output is pure dependency. We recover the hybrid representation by reversing the transformations in step 1.

The size of the unconverted dataset is 50,955 tokens, including 3,775 empty categories. The dependency graphs in the treebank contain 9,847 phrase nodes and 38,642 edges. After conversion, all phrase nodes and empty categories were removed, resulting in 47,220 tokens and a total of 34,849 edges. The number of edges dropped due to collapsing edges between empty categories.

For conversion, we use a similar process to Hall et al. (2007, 2008)'s approach for German and Swedish, but adapt this to the representation used for traditional Arabic grammar. During the conversion process, we apply graph transformations to encode information about phrase structure and elision:

Phrases: Let $p = (t_i, t_j)$ be a phrase node in the hybrid graph covering the terminal nodes from t_i to t_j inclusively. The conversion for the phrase node p is based on the observation that the phrase covers a subgraph with root ω_0. If p is a dependent node with edge E, head h, and dependency relation r, we remove E and p and add a new edge E' with dependent ω_0, head h, and label $+r$. Similarly, if p is a head node, we add a new edge with label $r+$. For the inverse transformation, $+r$ and $r+$ denote expanding the edge's dependent or head into a subgraph respectively. The label $+r+$ indicates that both head and dependent nodes for that edge should be expanded, to produce an edge between a pair of phrases. Figure 4 illustrates this conversion process. As with the dependency graphs displayed on the treebank's website, the convention in these diagrams is that dependents point towards heads.

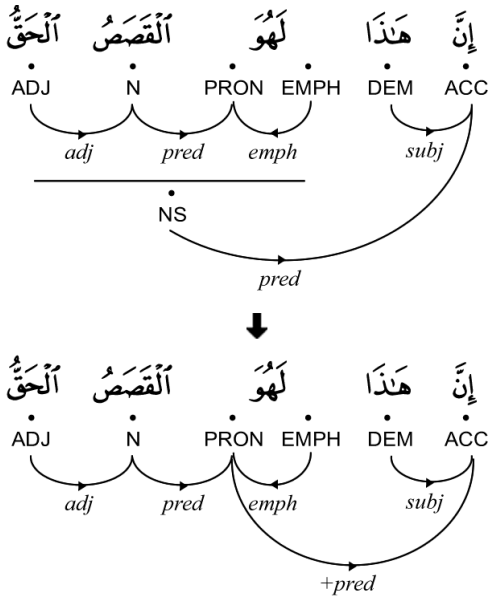

Figure 4: Conversion of phrase structure.

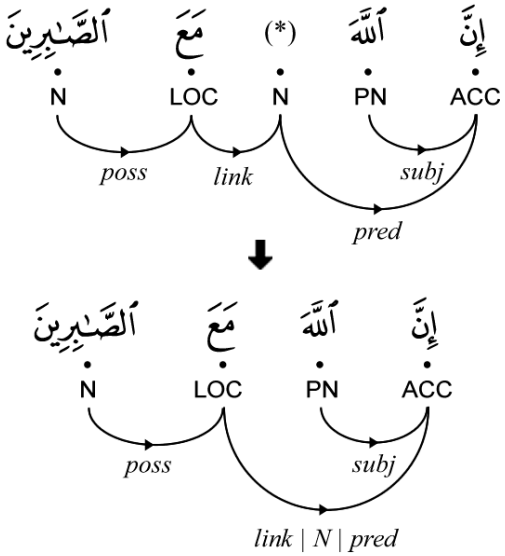

Figure 5: Conversion of syntactic elision.

Elision: For verbs with elided subject pronouns, we simply remove these from the converted graph as they are easily recovered through the verb's rich morphology. To keep the transformation rules simple, for syntactic elision we consider only the most common case where two

tokens are connected via an empty category (Figure 5). If *a* depends on an elided node *e* with part-of-speech *pos* and relation r_1, and *e* depends on *b* with relation r_2, we remove *e* and the two edges. We add a new edge with dependent *a*, head *b* and complex label $r_1 \mid pos \mid r_2$.

As we discuss in the evaluation section, the performance of the baseline approach to parsing is affected by the coverage of the conversion process. However, the small set of rules above for phrases and elision allow us to correctly recover nearly all edges in the hybrid graphs through this process.

5.2 Integrated Parsing

The integrated approach is simpler because there are no conversion steps. We train HSP using the treebank's full hybrid representation without preprocessing. In this experiment, we add to the instruction set the four parser actions that are required to build hybrid graphs: REDUCE2, EMPTY, SUBJECT and SUBGRAPH. Although in both cases the same set of features are available during the training phase, the two approaches lead to different machine learning problems. In the first experiment, the parser has to learn more complex edge labels. In the second experiment, there are fewer classes for classification, and phrase structure and elision are integrated directly into the parsing process.

5.3 Feature Selection

The parser uses graph features as well as morphological features, taken from the top three nodes on the stack and the top from the queue. The graph features are DEPREL, IS ROOT and IS EDGE. The first of these is a compound feature: For each relation $r \in R$, a binary predicate is set if the node has a dependent with that relation. IS ROOT is set if the node is the root of a well-formed subgraph, and IS EDGE is set if s_1 and s_2 form a previously parsed edge.

After initial work using a subset of the data, we decided to use five different sets of morphological features, which we grouped together to simplify the number of parsing experiments (Figure 6). A more detailed description of these features is given in the treebank's annotation guidelines (Dukes, Atwell and Sharaf, 2010). Each feature set also includes the same graph features. For our parsing experiments, we use gold-standard morphological data for the parser's input.

Features	POS	MORPH6	MORPH9	LEMMA	PHI
POS	Y	Y	Y	Y	Y
PHRASE	Y	Y	Y	Y	Y
VOICE	-	Y	Y	Y	Y
MOOD	-	Y	Y	Y	Y
CASE	-	Y	Y	Y	Y
STATE	-	Y	Y	Y	Y
PRONTYPE	-	-	Y	Y	Y
SEGTYPE	-	-	Y	Y	Y
COPULA	-	-	Y	Y	Y
LEMMA	-	-	-	Y	Y
PERSON	-	-	-	-	Y
GENDER	-	-	-	-	Y
NUMBER	-	-	-	-	Y

Figure 6: Morphological features used for parsing.

In comparison, Marton et al. (2010) show that for modern Arabic using predicted features or gold-standard morphological features for parsing achieves similar results. Our different feature sets are described below:

POS: This baseline feature set includes the part-of-speech and phrase tags for the selected nodes.

MORPH6: This set adds the core morphological features that might help with parsing, based on domain knowledge of traditional Arabic grammar: VOICE, MOOD, CASE and STATE. State is either not-specified, definite (for the Arabic definite article *al-* prefix) or indefinite (for *tanween*).

MORPH9: Adds a further three morphological features. PRONTYPE marks a pronoun clitic as either an object pronoun or subject pronoun. Due to Arabic's rich morphology, these different types of clitics are common, and they form either subject or object dependency relations when attached to verbs. The feature SEGTYPE indicates if a token is a prefix, stem or suffix. The COPULA feature is used for a subset of copular verbs known as *kāna wa akhwātaha* (كان واخواتها). Although assigned the same part-of-speech tag as normal verbs, in traditional Arabic grammar these words form subject and predicate relations instead of subject and object.

LEMMA: To test the effect of lexicalization on the parser, this feature set adds lemmas.

PHI: This feature set includes the so-called phi-features of person, gender and number.

6 Evaluation

6.1 Metrics and Methodology

Two standard metrics for evaluating parsing performance are LAS (labelled attachment score) for pure dependency parsing, and Parseval for constituency parsing. LAS is a single measure, while Parseval defines three measures: precision, recall, and F1-score, where F1-score is the harmonic mean of precision and recall. For hybrid parsing, we combine both LAS and Parseval into a new metric which we call ELAS (extended labelled attachment score). We first define the two existing metrics in set-theoretic terms, and then show how they can be combined.

In the CoNLL-X shared task on multilingual dependency parsing (Nivre, et al. 2007b), LAS was used an official accuracy metric. Let (t_1, ..., t_n) be an input sentence that has been morphologically tokenized, $G = (V, E, L)$ be an expected graph from the reference data, and $G' = (V', E', L')$ be the corresponding pure dependency graph output by the parser. Let $H(t)$ be the expected head of the token $t \in \{t_1, ..., t_n\}$, or ϕ if t is headless. Similarly, if $H(t) \neq \phi$, let $l(t) \in L$ denote the expected label of the edge $e \in E$ from t to $H(t)$. The LAS metric for the dependency parse pair (G, G') is then defined as the cardinality ratio:

$$\frac{\left|\left\{t : H(t) \neq \phi \wedge H(t) = H'(t) \wedge l(t) = l'(t)\right\}\right|}{\left|\left\{t : H(t) \neq \phi\right\}\right|}$$

For a pure dependency graph, this is the fraction of tokens that are assigned the correct head node and dependency label. This token-based definition does not easily generalize to hybrid parsing since hybrid graphs can contain edges between phrase nodes. Therefore, we provide a second definition of LAS by shifting focus from tokens to edges. For a well-formed pure dependency graph, the number of tokens with heads is the same as the number of edges. We define the edge equivalence relation $e \equiv e'$ to be true if and only if e and e' both connect t to $H(t)$ and if $l(e) = l(e')$. We then have the following edge-based definition:

$$\text{LAS} = \frac{\left|\left\{e' \in E' : \exists e \in E \ \left(e \equiv e'\right)\right\}\right|}{\left|E\right|}$$

For constituency phrase structure, the Parseval metric (Black et al., 1991) can also be defined using a similar equivalence relation. Let C denote the set of constituency labels. Given a sentence (t_1, ..., t_n), we let $p_{ij} = (t_i, t_j)$ be the phrase that spans the tokens from t_i to t_j inclusively with label $c(p) \in C$. Let P denote the set of non-terminal phrases in a parse tree from the reference data, and P' be the corresponding set of phrases output by a pure constituency parser. A phrase $p' \in P'$ is considered to be correct if there exists an equivalent phrase $p \in P$ with the same label that spans the same tokens. We define the phrase equivalence relation $p \equiv p' \Leftrightarrow \exists i, j : p = p_{ij} \wedge p' = p'_{ij} \wedge c(p) = c(p')$. For the constituency parse pair (P, P') we define Parseval precision and recall scores as:

$$\text{Precision} = \frac{\left|\left\{p' \in P' : \exists p \in P \ \left(p \equiv p'\right)\right\}\right|}{\left|P'\right|}$$

$$\text{Recall} = \frac{\left|\left\{p' \in P' : \exists p \in P \ \left(p \equiv p'\right)\right\}\right|}{\left|P\right|}$$

For hybrid parsing, we consider an edge in a parsed graph $G' = (V', E', L')$ to be correct if it has an equivalent edge in the reference graph $G = (V, E, L)$. Two edges are equivalent if they have the same edge label, and connect equivalent vertices. A vertex $v \in V$ may represent a token, a phrase node or elision. We define the vertex equivalence relation $v \equiv v'$ to be true when v and v' are both the same token. For two vertices that are phrases ($v = p \wedge v' = p'$), we use the same phrase equivalence relation $p \equiv p'$ in the Parseval metric. For elision, two vertices are equivalent if they have the same POS tag and surface form. For two edges, e from v to $H(v)$, and e' from v' to $H'(v')$, we define the edge equivalence relation as $e \equiv e' \Leftrightarrow v \equiv v' \wedge H(v) \equiv H'(v') \wedge l(e) = l(e')$. We then define ELAS precision and recall scores as:

$$\text{Precision} = \frac{\left|\left\{e' \in E' : \exists e \in E \ \left(e \equiv e'\right)\right\}\right|}{\left|E'\right|}$$

$$\text{Recall} = \frac{\left|\left\{e' \in E' : \exists e \in E \ \left(e \equiv e'\right)\right\}\right|}{\left|E\right|}$$

and the F1-score as the harmonic mean of precision and recall. This metric combines LAS and Parseval. For pure dependency graphs, ELAS recall is the same as vanilla labelled attachment score. For an edge between two phrase nodes in a hybrid graph, the metric uses a Parseval-like measure of correctness for the two phrases.

Feature Set	Multi-step Parser			Integrated Parser			F1-Diff.
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
POS	76.73	74.38	75.54	78.28	75.01	76.61	+1.07
MORPH6	82.52	79.74	81.10	84.62	80.64	82.58	+1.48
MORPH9	86.98	85.32	86.14	89.42	86.35	87.86	+1.72
LEMMA	**88.42**	**86.54**	**87.47**	**90.98**	**87.16**	**89.03**	**+1.56**
PHI	88.23	86.35	87.28	90.87	87.02	88.90	+1.62

Figure 7: Extended labelled attachment scores (ELAS) for parsing the treebank using different feature sets.

6.2 Results

We use ELAS as our evaluation metric for measuring the performance of HSP in both the one-step and two-step parsing experiments. To reduce sample bias, we use 10-fold cross-validation. Our F1-scores are calculated by aggregating the total number of true positives and false positives across the ten folds, as per method three in Forman and Scholz (2009).

Figure 7 shows the results for the two parsing approaches. Using the best performing feature set, HSP achieves an F1-score of 87.47% for the multi-step approach, and 89.03% for the integrated approach. This high performance may not only be due to the treebank being annotated with rich morphological features or our choice of algorithm. The Quranic text contains many examples of syntactic and stylistic repetition (Salih, 2007). Repetition leads to an easier machine learning problem, as fewer non-standard cases are encountered during training.

For statistical parsing, the five feature sets above each give different results. It is surprising that the POS feature set is already a good baseline. Using no morphological features and only part-of-speech tags, this feature set produces scores of 75.54% and 76.61% for the two approaches respectively. Our explanation for this is the fact that the treebank uses a detailed part-of-speech tagset, with 45 tags. However, we note that all five feature sets use the same graph features defined in the previous section. Without these graph features, accuracy for the baseline POS feature set drops to only 21.64%. The graph features provide constraints on possible dependencies. For example, the DEPREL features stop additional edges being formed where these would not make sense based on examples in the training data, such as multiple subjects for the same verb.

The next feature set MORPH6 adds voice, mood, case and state. The improvement over the POS feature set is 5.56% for the multi-step approach and 5.97% for the integrated approach. This is consistent with recent work for parsing Modern Standard Arabic. Marton et al. (2010) use a similar set of morphological features to improve parsing accuracy for CATiB (Habash and Roth, 2009). The next set MORPH9 similarly improves performance using further morphological features.

In comparison to parsing Modern Standard Arabic, the best feature set is LEMMA, which boosts performance by a further 1.33% and 1.17% respectively over MORPH9. However, the feature set PHI that adds person, gender and number, surprisingly degrades performance by 0.19% and 0.13% for the two approaches. This contrasts with recent work for parsing CATiB, where the phi-features have been shown to be helpful. We conclude that adding these features may not be statistically significant for parsing the Quran using 10-fold cross-validation, or that this last feature set possibly includes too many features for our SVM model, given the relatively small size of the current version of the treebank.

6.3 Effect of the Conversion Process

The results above show that the integrated parser outperforms the multi-step parser for all of the five feature sets. However, it is interesting that the absolute difference between the two F1-scores consistently lies in the narrow band 1.4 ± 0.32. This suggests that the two parsers have similar sensitivities to feature selection.

Another factor affecting the performance of the multi-step parser is the accuracy of the conversion process from the hybrid representation to pure dependency, and then back to hybrid. The rule-based conversion algorithm outlined in section 5.1 correctly recovers 94.81% of edges. Alt-

hough it might have been possible to improve the accuracy of the conversion process, this would have required a larger set of more complex rules for uncommon structures, such as the few cases of non-projective edges in the treebank, or for semantic elision.

To measure the effect of the conversion process, we performed a further experiment. We excluded from the treebank all dependency graphs that did not have a perfect reversible conversion to pure dependency (~ 8% of all graphs). We then repeated the 10-fold cross-validation tests using the best performing configuration for both approaches, the LEMMA feature set. On this subset of the data, the multi-step parser achieved an F1-score of 88.89% (89.33 precision, 88.45 recall), and the integrated parser's F1-score was 90.24% (91.48 precision, 89.03 recall). The difference between the two F1-scores was +1.35, which lies in the same narrow band of 1.4 ± 0.32.

These results suggest that the absence of a conversion process is not the largest contributing factor to integrated parser's improved performance. Although additional investigation into optimizing the two-step parsing algorithm could be further pursued, we choose not to. Instead, we argue that the integrated approach is not only simpler as there is no conversion, but is also better suited to the hybrid representation in the treebank.

7 Conclusion and Future Work

In this paper we presented HSP, a Hybrid Statistical Parser, trained using data from the Quranic Treebank. This treebank is a resource for studying the Quran online, and uses a hybrid representation that closely follows the traditional Arabic grammar known as *i'rāb* (إعراب). The treebank's syntactic representation includes phrase nodes and elided words, and presents a special challenge to statistical parsing.

We described two approaches to parsing using different sets of rich morphological features, and compared this to recent work for Modern Standard Arabic. Our shift-reduce algorithm is able to parse hybrid syntactic representations using a one-step process. We concluded that our novel integrated architecture is not only more elegant, but that encoding information this way also improves performance, resulting in a 1.6% ELAS absolute increase over the multi-step baseline for the integrated approach. To the best of our knowledge, this is the first work on statistical parsing for the Classical Arabic language of the Quran.

In the future, we plan to continue our work on hybrid parsing by focusing on three key areas: integrating morphological disambiguation into the parser, comparing HSP to other statistical parsers, and extending the parser to other related languages.

Morphological disambiguation is an important component of our proposed architecture. In this paper, we focused on parsing using only gold standard morphological input. However, Marton et al. (2010) show that parsing Arabic using predicted instead of gold morphological input gives similar results for different feature sets. For Hebrew, Goldberg and Tsarfaty (2008) show that joint morphological and syntactic disambiguation outperforms a pipeline approach. We plan to determine if the same applies to parsing the Quran.

Another area of future work is to compare HSP to other statistical parsers. Since our two-step approach converts the hybrid representation to pure dependency, we could in principle parse the Quranic Treebank using any pure dependency parser. For example, MSTParser (McDonald, et al., 2006) could be used to compare one-step hybrid parsing to two-step pure dependency parsing using an alternative graph-based parsing algorithm.

We also plan to extend HSP to parse other languages and treebanks. Classical languages such as Quranic Arabic are sometimes easier to parse statistically compared to modern languages, since vocabulary size and the number of linguistic constructions in such languages is smaller. We are interested to determine if our approach generalizes to other classical languages such as Biblical Hebrew, as well as modern texts, beyond this particular dataset.

Acknowledgments

The authors would like to thank the three anonymous reviewers who provided invaluable feedback to improve the quality of this paper. We thank Eric Atwell at Institute for Artificial Intelligence, University of Leeds for reviewing this paper and providing numerous useful suggestions. We also acknowledge the hard work of the volunteer collaborators involved in online annotation of the Quranic Treebank.

References

Kepa Bengoetxea and Koldo Gojenola. 2010. Application of Different Techniques to Dependency Parsing of Basque. In *Proceedings of the NAACL/HLT Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, California.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma and Fei Xia. 2009. Multi-Representational and Multi-Layered Treebank for Hindi/Urdu. In *Proceedings of the Third Linguistic Annotation Workshop at the conference of the Association for Computational Linguistics (ACL-IJCNLP),* Suntec, Singapore.

Daniel Bikel. 2004. On the Parameter Space of Lexicalized Statistical Parsing Models. *PhD thesis, Department of Computer and Information Sciences*. University of Pennsylvania.

Ezra Black et al. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*.

Georges Bohas, Jean-Patrick Guillaume, and Djamel Eddin Kouloughli. 1990. The Arabic linguistic tradition. *Arabic Thought and Culture*. Routledge.

Chih-Chung Chang and Chih-Jen Lin. 2001. LIBSVM: A Library for Support Vector Machines. *Technical Report, Department of Computer Science and Information Engineering*, National Taiwan University.

Eugene Charniak. 2000. A Maximum-entropy-inspired Parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, Seattle.

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Kais Dukes, Eric Atwell and Nizar Habash. 2011. Supervised Collaboration for Syntactic Annotation of Quranic Arabic. To appear in *Language Resources and Evaluation Journal (LREJ): Special Issue on Collaboratively Constructed Language Resources*.

Kais Dukes, Eric Atwell and Abdul-Baquee Sharaf. 2010. Syntactic annotation guidelines for the Quranic Arabic Dependency Treebank. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*. Valletta, Malta.

Kais Dukes and Timothy Buckwalter. 2010. A Dependency Treebank of the Quran using Traditional Arabic Grammar. In *Proceedings of the 7th international conference on Informatics and Systems (INFOS)*. Cairo, Egypt.

Kais Dukes and Nizar Habash. 2010. Morphological Annotation of Quranic Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*. Valletta, Malta.

Gülsen Eryiğit, Joakim Nivre and Kemal Oflazer. 2008. Dependency Parsing of Turkish. *Computational Linguistics*.

George Forman and Martin Scholz. 2009. Apples-to-apples in Cross-validation Studies: Pitfalls in Classifier Performance Measurement. *HP technical Reports*, HPL-2009-359.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL*, New York.

Yoav Goldberg and Michael Elhadad. 2010. Easy-First Dependency Parsing of Modern Hebrew. In *Proceedings of the NAACL/HLT Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, California.

Yoav Goldberg and Reut Tsarfaty. 2008. A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing. In *Proceedings of ACL-HLT*. Columbus, Ohio.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Suntec, Singapore, August.

Nizar Habash. 2010. Introduction to Arabic Natural Language Processing. Morgan & Claypool Publishers.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single Malt or Blended? A Study in Multilingual Parser Optimization. In *Proceedings of EMNLP-CoNLL*.

Johan Hall and Joakim Nivre. 2008. A Dependency-driven Parser for German Dependency and Constituency Representations. In *Proceedings of the ACL Workshop on Parsing German (PaGe08)*, Columbus, Ohio.

Johan Hall, Joakim Nivre and Jens Nilsson. 2007. A Hybrid Constituency-dependency Parser for Swedish. In *Proceedings of NODALIDA*, Tartu, Estonia.

Seth Kulick, Ryan Gabbard, and Mitchell Marcus. 2006. Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of Treebanks and Linguistic Theories Conference*. Prague, Czech Republic.

Mohamed Maamouri, Ann Bies, Timothy Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-scale Annotated Arabic Corpus. In *Proceedings of the NEMLAR Confer-*

ence on Arabic Language Resources and Tools. Cairo, Egypt.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated corpus of English: The Penn Treebank. *Computational Linguistics*.

Yuval Marton, Nizar Habash, and Owen Rambow. 2010. Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features. In *Proceedings of the NAACL/HLT Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, California.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual Dependency Analysis with a Two-stage Discriminative Parser. In *Proceedings of CoNLL*. New York.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007a. MaltParser: A Language Independent System for Data-driven Dependency Parsing. *Natural Language Engineering*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel and Deniz Yuret. 2007b. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of EMNLP-CoNLL*.

Jonathan Owens. 1988. The Foundations of Grammar: An Introduction to Medieval Arabic Grammatical Theory. *Amsterdam Studies in the Theory and History of Linguistic Science*. John Benjamins.

Bahjat Salih. 2007. *al-i'rāb al-mufassal li-kitāb allāh al-murattal* (‘A Detailed Grammatical Analysis of the Recited Quran using *i'rāb*’). Dar Al-Fikr, Beirut.

Otakar Smrž, Viktor Bielický, Iveta Kourilová, Jakub Kráčmar, Jan Hajic, 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)*. Marrakech, Morocco.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of International Conference on Parsing Technologies (IWPT 2003)*.