# CamelParser: A System for Arabic Syntactic Analysis and Morphological Disambiguation

**Anas Shahrour, Salam Khalifa, Dima Taji, Nizar Habash**
Computational Approaches to Modeling Language (CAMeL) Lab
New York University Abu Dhabi, UAE
{anas.shahrour,salamkhalifa,dima.taji,nizar.habash}@nyu.edu

## Abstract

In this paper, we present CamelParser, a state-of-the-art system for Arabic syntactic dependency analysis aligned with contextually disambiguated morphological features. CamelParser uses a state-of-the-art morphological disambiguator and improves its results using syntactically driven features. The system offers a number of output formats that include basic dependency with morphological features, two tree visualization modes, and traditional Arabic grammatical analysis.

## 1 Introduction

Automatic processing of Arabic is challenging for several reasons (Habash, 2010). Arabic is morphologically rich and highly ambiguous. The morphological analyzer we use represents Arabic words with 15 features, such as gender, number, person, state, case, etc. (Pasha et al., 2014). And due to Arabic's optional diacritization orthography, words have an average of 12 analyses per word (Pasha et al., 2014). Furthermore, Arabic morphology and syntax have complex agreement rules. For example, a noun may get its case by being the subject of a verb and its state by being the head of an Idafa (possessive) construction; while adjectives modifying this noun agree with it in its case, their state is determined by the state of the last element in the Idafa construction chain the noun heads.

In this paper, we present CamelParser, a system for Arabic syntactic dependency analysis aligned with contextually disambiguated morphological features. CamelParser uses a state-of-the-art morphological disambiguator, MADAMIRA (Pasha et al., 2014), and improves its results using syntactically driven features. The system offers a number of output formats that include basic Columbia Arabic Treebank dependency (Habash and Roth, 2009) with morphological features, two tree visualization modes, and traditional Arabic grammatical analysis. CamelParser is publicly available for research purposes.[1]

## 2 Related Work

In related work on modeling Arabic syntax and morphology, Habash et al. (2007a) demonstrated that given good syntactic representations, case prediction can be done with a high degree of accuracy. Alkuhlani et al. (2013) later extended this work to cover all morphological features. Marton et al. (2013) explored the contribution of different POS tag sets and several lexical and inflectional morphology features to dependency parsing of Arabic. Building on all of these previous efforts, Shahrour et al. (2015) presented an approach for improving the quality of several morphological features using syntax. They demonstrated that predicted syntax can significantly improve the full morphological analysis choice, particularly for case and state. Our system, CamelParser, further builds on their approach and improves on it by optimizing the syntactic parsing and by offering output in several formats including aligned syntax and morphology, traditional Arabic grammatical analysis (إعراب *Ǎiςrab*),[2] tree visualization, and tree annotation output compatible with the dependency annotation tool TrEd (Pajas, 2008).

[1]CamelParser can be downloaded from `http://camel.abudhabi.nyu.edu/resources/`.
[2]Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007b).

## 3 System Description

### 3.1 Overview

CamelParser is based on the system proposed by Shahrour et al. (2015) and uses the same data splits (*Train*, *Dev* and *Test*) described by Diab et al. (2013) for the Penn Arabic Treebank (PATB, parts 1, 2 and 3) (Maamouri et al., 2004), and the same morphological feature representations derived from the PATB analyses following the approach used in the MADAMIRA Arabic morphological analysis and disambiguation system (Pasha et al., 2014). We trained an Arabic dependency parser using MaltParser (Nivre et al., 2007) on the Columbia Arabic Treebank (CATiB) version of the PATB (Habash and Roth, 2009). We used an enriched data format that includes all our morphological features and three POS tag sets with different degrees of granularity: CATiBex (Marton et al., 2013), Kulick (Kulick et al., 2006), and Buckwalter (Buckwalter, 2002).

CamelParser improves on the morphological disambiguation done in MADAMIRA using the syntactic analysis information to predict the case and state features with an unlexicalized machine learning model. The predicted case and state are used to select a new morphological analysis. The final output consists of the syntax in CATiB style aligned with the MADAMIRA morphology features for each token (see the list of features exemplified in Table 1). The output formats are discussed in section 3.4.

### 3.2 Syntactic Parser Optimization

For parser optimization, we divided the *Dev* set into two parts with nearly equal number of tokens: *DevTune* (35,759 words) and *DevTest* (35,750 words). During the optimization, the parser was trained on *Train* (with gold morphological features) and evaluated against *DevTune* (with predicted morphological features). *DevTest* is reserved for future development.

The optimization was performed in two stages. The first stage is using MaltOptimizer (Ballesteros and Nivre, 2012), a tool for optimizing MaltParser for a specific dataset. MaltOptimizer does not allow the user to use separate train and test sets, a feature that we need to train the parser on gold features and evaluate against predicted features. For this reason, we used a modified version of MaltOptimizer that allows the user to provide separate train and test sets.[3]

While MaltOptimizer optimizes for the parsing algorithm, the learning algorithm, and the feature model, it doesn't optimize for the set of morphological features used. That is, it cannot produce an optimal subset of morphological features, it can either conclude that it helps to include all features or that it helps to exclude them all. This is why a second stage of optimization is performed to optimize for the morphological feature set. The lemma is the exception as it is represented in a separate column in the CoNLL-X format, and so MaltOptimizer can optimize for it separately. In the second stage of optimization, we use MaltParser in the optimal settings from the first stage. The optimization is done in a greedy way starting with an empty morphological feature set or with the lemma if it was selected in the first stage. At each round of optimization, we decide which of the remaining morphological features score is the highest when added to the current feature set, and if the added feature gives an improvement to the labeled attachment score, a new feature set is determined and the optimization continues. Otherwise, the new feature is dropped and the optimization round is concluded.

### 3.3 Arabic Grammatical Analysis

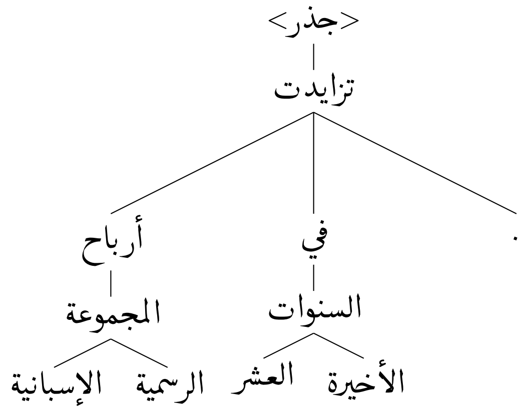
To facilitate the use of this system in the context of Arabic grammar education, the CamelParser output is mapped to the traditional way of expressing Arabic grammatical analyses (إعراب *Ǎiςrab*). The system traverses the syntactic tree and uses a collection of grammatical analysis and morphology rules to translate the syntactic and morphological information to the traditional grammatical analysis representation. The output is provided in plain text or tabular HTML format. Figure 1.(iii) shows an example grammatical analysis output. The grammatical analysis output for the noun أرباح *ÂrbAH* 'profits', which is the subject in the example translates to the following: *the subject with a nominative case indicated by the 'Damma' ending, and it is the head of an Idafa construct.*

[3]We are thankful to Miguel Ballesteros and Joakim Nivre for providing us with the modified version of MaltOptimizer.

**1.(i) CATiB Dependency Column Format**

| | | | | |
|---|---|---|---|---|
| 1 | tzAydt | VRB | 0 | — |
| 2 | >rbAH | NOM | 1 | SBJ |
| 3 | AlmjmwEp | NOM | 2 | IDF |
| 4 | Al<sbAnyp | PROP | 3 | MOD |
| 5 | Alrsmyp | NOM | 3 | MOD |
| 6 | fy | PRT | 1 | MOD |
| 7 | AlsnwAt | NOM | 6 | OBJ |
| 8 | AlE$r | NOM | 7 | MOD |
| 9 | Al>xyrp | NOM | 7 | MOD |
| 10 | . | PNX | 1 | MOD |

**1.(ii) CATiB Dependency Tree Format**



**1.(iii) Traditional Arabic Grammatical Analysis (إعراب)**

| | | |
|---|---|---|
| تزايدت | : | فعل ماضي مبني على الفتح، والتاء تاء التأنيث الساكنة لا محل لها من الاعراب |
| أرباح | : | فاعل مرفوع وعلامة رفعه الضمة وهو مضاف |
| المجموعة | : | مضاف اليه مجرور وعلامة جره الكسرة |
| الإسبانية | : | نعت [المجموعة] مجرور وعلامة جره الكسرة |
| الرسمية | : | نعت [المجموعة] مجرور وعلامة جره الكسرة |
| في | : | حرف جر |
| السنوات | : | اسم مجرور بحرف الجر [في] وعلامة جره الكسرة (وهو جمع مؤنث سالم) |
| العشر | : | نعت [السنوات] مجرور وعلامة جره الكسرة |
| الأخيرة | : | نعت [السنوات] مجرور وعلامة جره الكسرة |
| . | : | علامة ترقيم |

Figure 1: CamelParser Output Formats. Morphological features are not shown due to space limitations. The example sentence is تزايدت أرباح المجموعة الإسبانية الرسمية في السنوات العشر الأخيرة. *tzAydt ÂrbAH AlmjmwEħ AlǍsbAnyħ Alrsmyħ fy AlsnwAt Alςšr AlÂxyrħ* (Lit. *increased profits the-group the-Spanish the-official in the-years the-ten the-last.*) 'The profits of the official Spanish group increased in the last ten years.' The output in **1.(i)** is in the Buckwalter Transliteration scheme (Habash et al., 2007b).

## 3.4 Output Formats

The CamelParser produces the following output formats:

- Syntactic analysis in the CATiB dependency representation (Habash and Roth, 2009) aligned with the morphology of each token in the feature-value pair format used in MADAMIRA (Pasha et al., 2014). Table 1 shows the features and their values for one example word. For more details on Arabic morphological features, see (Habash, 2010).
- Traditional Arabic grammatical analysis format.
- Tree visualization in PDF format.
- Tree file in `.fs` format for annotation in the TrEd tree editor (Pajas, 2008).

Additionally, as a side product, CamelParser generates an improved morphological disambiguation file in `.mada` format (MADAMIRA tool output), and a corresponding `ATB4MT` tokenization file (Arabic Treebank tokenization scheme). Figure 1 shows the same analysis in the first three output formats discussed above.

|  | Feature | Value | Description of Feature and Value |
|---|---|---|---|
| 1 | tok | AlmjmwEp | Arabic Treebank token form (Buckwalter Transliteration) |
| 2 | toknorm | AlmjmwEp | Alif/Ya normalized token (Buckwalter Transliteration) |
| 3 | toklex | majomuwEap | Lemma of token (Buckwalter Transliteration) |
| 4 | tok_utf8 | المجموعة | Arabic Treebank token (UTF8) |
| 5 | toknorm_utf8 | المجموعة | Alif/Ya normalized token form (UTF8) |
| 6 | toklex_utf8 | مَجْمُوعَة | Lemma of token (UTF8) |
| 7 | catib4 | NOM | A coarser version of CATiB POS (PROP is mapped to NOM, and VRB-PASS is mapped to VRB) |
| 8 | catibex | Al-NOM-p | CATiBEX POS tag (Marton et al., 2013) |
| 9 | pos | noun | MADAMIRA's POS tag (Pasha et al., 2014) |
| 10 | gloss | collection;group;bloc | English Gloss |
| 11 | prc3 | 0 | Proclitic 3 (question proclitic), with value *0* |
| 12 | prc2 | 0 | Proclitic 2 (conjunction proclitic), with value *0* |
| 13 | prc1 | 0 | Proclitic 1 (preposition proclitic), with value *0* |
| 14 | prc0 | Al_det | Proclitic 0 (article proclitic), with value *Al_det* for the determiner ال *Al* 'the' |
| 15 | per | na | Person, with value *N/A* |
| 16 | asp | na | Aspect, with value *N/A* |
| 17 | vox | na | Voice, with value *N/A* |
| 18 | mod | na | Mood, with value *N/A* |
| 19 | gen | f | Gender, with value *f* for *feminine* |
| 20 | num | s | Number, with value *s* for *single* |
| 21 | stt | d | State, with value *d* for *definite* |
| 22 | cas | g | Case, with value *g* for *genitive* |
| 23 | enc0 | 0 | Enclitic 0 (pronominals), with value *0* |
| 24 | rat | y | Rationality (currently under development; defaulting to *y*) |
| 25 | catib6 | NOM | CATiB POS tag (Habash and Roth, 2009) |
| 26 | penpos | DT+NN | Penn POS tag (Habash, 2010) |
| 27 | bw | DET+NOUN+NSUFF_FEM_SG +CASE_DEF_GEN | Buckwalter POS tag (Buckwalter, 2002) |

Table 1: The features generated by CamelParser for the word المجموعة *Almjmwςħ* '*the group*' from the example shown in Figure 1, and their values and descriptions. The system's transliterated Arabic values are in the Buckwalter scheme (Habash et al., 2007b). The exact output line from CamelParser is:

```
3 AlmjmwEp NOM 2 IDF tok:AlmjmwEp toknorm:AlmjmwEp toklex:majomuwEap tok_utf8:المجموعة
toknorm_utf8:المجموعة toklex_utf8:مَجْمُوعَة catib4:NOM catibex:Al-NOM-p pos:noun
gloss:collection;group;bloc prc3:0 prc2:0 prc1:0 prc0:Al_det per:na asp:na
vox:na mod:na gen:f num:s stt:d cas:g enc0:0 rat:y catib6:NOM pennpos:DT+NN
bw:DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN.
```

## 4 Evaluation

**Parsing Accuracy** We compare the performance of CamelParser to the parser described in Shahrour et al. (2015), henceforth, Baseline Parser. The two systems use the same training data (*Train*) and report on the same testing data (*Test*) in predicted morphology setting. CamelParser, however, uses more features and has been optimized as discussed above. In terms of labeled attachment, unlabeled attachment, and label accuracy, CamelParser achieves 83.8%, 86.4%, and 93.2%, respectively. These are significant improvements over the Baseline Parser's respective scores of 81.6%, 84.6%, and 92.0%. Previously reported state-of-the-art results by Marton et al. (2013) are 81.7%, 84.6%, and 92.8% (again, for labeled attachment, unlabeled attachment, and label accuracy, respectively). We present their numbers here although it is hard to conduct a fair comparison because of differences in training and testing data sets.

**Morphological Disambiguation Accuracy** We compare the performance of the morphological disambiguation produced by CamelParser to that of the MADAMIRA system which CamelParser uses. All the results are reported on the same data set (*Test*). There are many metrics that can be used, but we focus here on two harsh metrics: full word diacritization accuracy and all morphological feature selection. The MADAMIRA system produces 88.1% and 86.1% for these two metrics respectively. CamelParser's use of syntax to improve morphological analysis raises the scores to 90.8% and 88.7%.

## 5 Conclusion

We presented CamelParser, a system for improved syntactic analysis and morphological disambiguation of Arabic. The system uses an optimized syntactic parser and produces morphologically-enriched syntactic dependencies, along with tree visualizations, TrEd file outputs for annotation, and traditional grammatical analyses.

## References

Sarah Alkuhlani, Nizar Habash, and Ryan Roth. 2013. Automatic morphological enrichment of a morphologically underspecified treebank. In *HLT-NAACL*, pages 460–470.

Miguel Ballesteros and Joakim Nivre. 2012. Maltoptimizer: an optimization tool for maltparser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–62. Association for Computational Linguistics.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Nizar Habash and Ryan M Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224. Association for Computational Linguistics.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitchell P Marcus. 2007a. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *EMNLP-CoNLL*, pages 1084–1092.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007b. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

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

Seth Kulick, Ryan Gabbard, and Mitch Marcus. 2006. Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference*, pages 31–42, Prague, Czech Republic.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Petr Pajas. 2008. Tred: Tree editor. http://ufal.mff.cuni.cz/ pajas/tred.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*, Reykjavik, Iceland.

Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving Arabic diacritization through syntactic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1309–1315.