YAMAMA: Yet Another Multi-Dialect Arabic Morphological Analyzer

Salam Khalifa, Nasser Zalmout and Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi, UAE
{salamkhalifa,nasser.zalmout,nizar.habash}@nyu.edu

Abstract

In this paper, we present YAMAMA, a multi-dialect Arabic morphological analyzer and disambiguator. Our system is almost five times faster than the state-of-the-art MADAMIRA system with a slightly lower quality. In addition to speed, YAMAMA outputs a rich representation which allows for a wider spectrum of use. In this regard, YAMAMA transcends other systems, such as FARASA, which is faster but provides specific outputs catering to specific applications.

1 Introduction

The Arabic language poses many challenges for Natural Language Processing (NLP). First, Arabic is morphologically rich, having a large number of inflections per lemma. Secondly, Arabic is orthographically ambiguous, having about 12 full morphological analyses per word on average. Finally, Arabic has a number of linguistic varieties among which Modern Standard Arabic (MSA) is the official primary written standard with numerous resources, while the other varieties are the unofficial primarily spoken Dialects of Arabic (DA). For more on Arabic NLP, see (Habash, 2010). Table 1 presents an example that showcases the aspect of morphological ambiguity which is shared across all varieties of Arabic.[1]

هل سينجح بين أفليك في دور باتمان ؟
hl synjH byn Âflyk fy dwr bAtmAn
Will Ben Affleck be a good Batman?

POS	Diac	Gloss
PV+PVSUFF_SUBJ:3MS	bay∼ana	He demonstrated
PV+PVSUFF_SUBJ:3FP	bay∼an∼a	They demonstrated
NOUN_PROP	biyn	Ben
ADJ	bay∼in	Clear
PREP	bayn	Between, among
NOUN_PROP	bi+yan	with a Yen
15 more analysis ...		

Figure 1: Possible analyses produced by the morphological analyzer of the word بين *byn*. The correct analysis is highlighted in gray.

Previous efforts on morphological analysis and disambiguation have led to the creation of a number of state-of-the-art tools with high accuracy, such as MADAMIRA (Pasha et al., 2014). MADAMIRA produces a rich output (diacritization, tokenization, part-of-speech (POS), lemmatization, gloss, and all inflected features), but it is slow. Other systems such as FARASA (Darwish and Mubarak, 2016) are very fast but focus on specific types of output with high quality performance (tokenization). Clearly, there is always a tradeoff between speed, quality and richness. Our system, YAMAMA (Yet Another Multi-Dialect Arabic Morphological Analyzer; Arabic يمامة 'Barbary Dove'), is an alternative to MADAMIRA and FARASA: it offers a faster performance than MADAMIRA but with all of MADAMIRA's rich output at a reasonable tradeoff of quality that varies depending on the specific feature.[2]

2 Related Work

There has been a considerable amount of work on MSA and DA morphological analysis, disambiguation, POS tagging, tokenization, lemmatization and diacritization. One of the most notable efforts is MADAMIRA (Pasha et al., 2014). MADAMIRA produces a rich feature set for each word, containing

[1] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).
[2] To obtain YAMAMA (Version 1.0), go to http://camel.abudhabi.nyu.edu/resources/.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 223–227, Osaka, Japan, December 11-17 2016.

more than 14 morphological and lexical features. It also provides different tokenization schemes. Additionally, MADAMIRA has two modes for MSA and Egyptian Arabic (EGY). The speed of MADAMIRA however is relatively slow (420 words/sec in stand-alone mode, and 1,013 words/sec in server mode) especially for NLP tasks where speed may be critical. Recently, Darwish and Mubarak (2016) and Abdelali et al. (2016) presented a new Arabic segmenter, FARASA. They reported much faster running times than MADAMIRA with similar accuracy on toekniztion. FARASA produces word segmentations only as opposed to MADAMIRA's richer output; and it currently does not handle any Arabic dialect. Our system, YAMAMA, uses some components from MADAMIRA, in particular the morphological analyzers (out-of-context readings) but has its own disambiguation models. This allows YAMAMA to maintain the richness of MADAMIRA, but increase the speed. The disambiguation modeling components are inspired by FARASA's design. In this paper, we compare to both systems in terms of quality and speed.

3 YAMAMA

Motivation We were motivated by the FARASA approach (Darwish and Mubarak, 2016; Abdelali et al., 2016). FARASA achieves very high tokenization accuracy at a very high speed by not using any context. It relies on simple probabilistic models of stems, prefixes, suffixes and their combinations. While this approach will be limiting for complex tasks such as POS tagging, it is sufficient for tokenization, particularly when it comes to specific applications such as machine translation (MT) and information retrieval (IR) (Abdelali et al., 2016). Our goal for YAMAMA is to create a system that combines the rich output of MADAMIRA with fast and simple out-of-context analysis selection comparable to FARASA's approach. For in-vocabulary words, YAMAMA uses a pre-computed maximum likelihood model to assign an analysis to every word. For out-of-vocabulary words, YAMAMA ranks all of the analyses for such words using two unigram language models of the lemma and the Buckwalter POS tag. In both cases, YAMAMA reduces the text to types and makes decisions in type space, thus benefiting from the low type to token ratio.[3]

Datasets For the training and development of our system, we used the same settings as those used for MADAMIRA. For MSA, we used the Penn Arabic Treebank (PATB parts 1,2 and 3) (Maamouri et al., 2004), and for EGY, the ARZ Treebank (Maamouri et al., 2014) . We followed the data splits recommend by Diab et al. (2013) for both treebanks.

Maximum Likelihood Model We created the maximum likelihood model based on the ATB *Train* dataset by selecting the most frequent analysis for each word token in the dataset. The selected analyses are then stored in a dictionary that is loaded once the system starts running. The analyses include all the morphological and lexical features as in MADAMIRA.

Analysis and Disambiguation For the OOV words, we run a morphological analyzer (same analyzer used in MADAMIRA). For MSA we used the SAMA database (Graff et al., 2009), and for EGY we used the CALIMA ARZ database (Habash et al., 2012). The analyses of each word are ranked using the multiplication of their lemma probability and their semi-lexicalized Buckwalter tag probability. Both probabilities are estimated using the training data. The highest ranking analysis is selected; and the word and analysis are added to the loaded analysis dictionary.

Tokenization YAMAMA currently produces a detailed segmentation consisting of the undiacritized morphemes from the Buckwalter tag analysis (BWTagTok). For the analysis dictionary the BWTagTok segmentation is generated for each word ahead of time. Whereas for OOV words, the segmentation is generated after disambiguation.

Output Generation Although all analyses are determined in type space, the output has to be generated in token space. YAMAMA's output is in the same format as MADAMIRA's.

[3]In a text of 80 words, the type to token ratio is 89%, whereas in a text of 8M words, the type to token ratio is only 3.7%.

4 Evaluation

We present next two sets of experiments. The first set targets accuracy and speed and the second set targets machine translation quality. In both sets, we try to compare YAMAMA[4] to MADAMIRA[5] and FARASA[6] both, when possible.

4.1 Accuracy and Speed Evaluation

Experimental Setup While MADAMIRA and YAMAMA share similar output, they are different from FARASA. To allow us to compare them, we conducted three experiments. First, we compared MADAMIRA and YAMAMA in terms of accuracy of their rich output. Second, we compared all systems in terms of accuracy of the specific tokenization output of FARASA. Finally, we compared all three systems in terms of speed on a very large corpus. We also report speeds in the first two experiments, although the test sets are relatively small. For the accuracy evaluation we used the Test sets recommended by Diab et al. (2013) of the Penn Arabic Treebank (PATB parts 1,2 and 3) (Maamouri et al., 2004) (for MSA) and the ARZ Treebank (Maamouri et al., 2014) (for EGY).

Results First, in Table 1, we compare YAMAMA to MADAMIRA in terms of accuracy over the (a) Buckwalter POS tag segmentation, which is an undiacritized segmentation based on the morphemes in the Buckwalter analysis, (b) Lemma, (c) POS, (d) Diacritization, and (e) ALL features. We also report the time the systems took to complete the task. To give an example of the various features evaluated, the word المجموعة *Almjmwςħ* 'collection/group' may have a correct analysis with the BWTagTok `Al+mjmwE+p`, the lemma `majomuwEap`, the POS `noun`, and the diacritization `AlmajomuwEapi`. The ALL condition would include all of these in addition to `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`. For MSA, YAMAMA performs very closely to MADAMIRA except for DIAC, which explains the drop in ALL. However, in EGY, YAMAMA beats MADAMIRA in almost all aspects, except for POS. YAMAMA is four times faster than MADAMIRA in the MSA setting, and two times faster in EGY; due to the large size of the CALIMA ARZ database which is three times larger than SAMA, hence more loading time. Also, the speed of YAMAMA is sensitive to the ratio of OOV types, where it uses the morphological analyzer.

	MDMR		**YMM**	
	MSA	**EGY**	**MSA**	**EGY**
BWTagTok	98.5	93.8	98.4	94.0
LEX	96.8	87.5	96.1	87.8
POS	96.8	92.5	96.1	91.9
DIAC	88.0	83.6	81.0	85.3
ALL	86.0	78.4	78.8	79.3
Time (s)	57.7	51.2	15.4	31.1

Table 1: Evaluation results for MADAMIRA (**MDMR**) and YAMAMA (**YMM**) on the two tests **MSA** (ATB) and **EGY** (ARZ-ALL) using a number of morphological features: Buckwalter POS Tag tokenization (BWTagTok), Lemma (LEX), Part-of-Speech (POS), Diacritization (DIAC) and all features together (ALL). We also report running time.

Second, in Table 2 we compare to MADAMIRA and YAMAMA to FARASA in terms of FARASA's tokenization scheme (FarasaTok), which is similar but not exactly the same as the BWTagTok. We automatically converted the MADAMIRA and YAMAMA outputs as well as the MSA and EGY test sets to FarasaTok to be able to compare in the same tokenization space. We also report on an Alif, Ya and Ta-Marbuta normalized version of FarasaTok (FarasaTokNorm) for all test conditions. In addition to the test sets reported on earlier, we add the MSA WikiNews test set that was reported on by Darwish and Mubarak (2016). Across all conditions, YAMAMA and MADAMIRA behave very similarly. In MSA WikiNews, all three systems behave similarly. However, as would be expected, YAMAMA and MADAMIRA beat FARASA on the EGY set by a large margin. YAMAMA and MADAMIRA also have higher performance than FARASA on the MSA set. In terms of speed, YAMAMA outperforms in all modes except for EGY. The speeds of YAMAMA are competitive with FARASA except for EGY for the reasons mentioned earlier.

[4]YAMAMA:Version: 1.0.

[5]MADAMIRA: Released on May 16, 2016, version 2.1.

[6]FARASA: Downloaded on May 27, 2016.

	MSA			EGY			MSA-Wiki		
	MDMR	**YMM**	**FRS**	**MDMR**	**YMM**	**FRS**	**MDMR**	**YMM**	**FRS**
FarasaTok	98.7	98.7	89.6	94.3	94.4	73.3	98.5	98.0	98.7
FarasaTok Norm	99.2	99.2	98.4	96.5	96.6	86.6	98.9	98.8	98.7
Time (s)	58.1	15.7	17.7	51.4	31.2	16.8	43.8	9.9	14.8

Table 2: Evaluation results for MADAMIRA (**MDMR**), YAMAMA (**YMM**) and FARASA (**FRS**) on the three tests **MSA** (ATB), **EGY** (ARZ-ALL) and **MSA-Wiki** (WikiNews) using FARASA tokenization scheme in basic (**FarasaTok**) and normalized forms (**FarasaTok Norm**). We also report running time.

Finally, we ran all systems through a large dataset of 7.5 million words from Gigaword (Parker et al., 2009). The reported running times for MADAMIRA (standalone mode), YAMAMA and FARASA are 2,305s, 398s and 99s, respectively. YAMAMA is five times faster than MADAMIRA and FARASA is four times faster than YAMAMA.

4.2 Machine Translation Evaluation

Experimental Setup We used the Moses toolkit (Koehn et al., 2007) with default parameters to develop the Statistical Machine Translation (SMT) systems. For alignment, we used GIZA++ (Och and Ney, 2003). And for language modeling, we used KenLM (Heafield et al., 2013) to build a 5-gram language model. We evaluate using BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005). We apply statistical significance tests using the paired bootstrapped resampling method (Koehn, 2004). We used the Arabic-English parallel component of the UN Corpus (Eisele and Chen, 2010), with about 9 million lines for the English language model (∼286 million words), 200 thousand parallel lines for training (∼5 million words), 2000 lines for tuning, and 3000 lines for testing. The English content was tokenized using the default English tokenizer at Moses, and the Arabic texts were tokenized through YAMAMA, MADAMIRA and FARASA into the same Arabic Treebank tokenization scheme. For YAMAMA, we used the TOKAN tool (Habash et al., 2009) to do the tokenization. The Arabic dataset we used had English text segments covering UN resolutions numbers and named entities; so we applied Moses' English whitespace tokenization scripts on the Arabic files in advance of the Arabic tokenization for the three systems to be of a better match to the English reference.

	With OOV		Without OOV	
	BLEU	**METEOR**	**BLEU**	**METEOR**
YAMAMA	39.49	0.3618	38.00	0.3448
MADAMIRA	39.52	0.3627	37.65	0.3435
FARASA	37.73	0.3301	37.76	0.3436

Table 3: Machine translation results

Results and Analysis The results of the SMT experiments are presented in Table 3, with YAMAMA and MADAMIRA showing a statistically significant performance improvement relative to FARASA. For a better understanding of the results, we analyzed the output files and observed that FARASA transliterates English words with Arabic letters and deletes the vowels, most likely the result of an internal minor transliteration error. This behavior is problematic for SMT, as Moses would pass such English Out-of-Vocabulary (OOV) words in Arabic letters. To facilitate a better comparison ignoring the effect of different OOV handling, we performed additional SMT experiments that drop the OOV words from all three systems' output. Results are also in Table 3, with YAMAMA outperforming the other two systems slightly but with statistical significance. MADAMIRA and FARASA performed closely, with a statistically insignificant difference. As a general observation, we conclude that the variations among the different systems don't have a profound impact on the SMT quality.[7]

5 Conclusions and Future Work

We presented YAMAMA, a multi-dialect Arabic morphological analyzer and disambiguator. YAMAMA is almost five times faster than MADAMIRA, with slightly lower quality. YAMAMA outputs a rich representation which allows for a wider spectrum of use, transcending other systems, such as FARASA,

[7]We would like to thank the Farasa team, specifically, Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali for helpful conversations. We have provided them with feedback and they have since released an updated version of Farasa.

which is faster but provides specific outputs catering to specific applications. There is yet much room for enhancing the speed and the quality of YAMAMA, which we plan to investigate.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaulation Measures for MT and/or Summarization*.

Kareem Darwish and Hamdy Mubarak. 2016. Farasa: A new fast and accurate Arabic word segmenter. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

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

Andreas Eisele and Yu Chen. 2010. Multiun: A multilingual corpus from united nation documents. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 2868–2872.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

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

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. The MEDAR Consortium, April.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

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.

Mohamed Maamouri, Ann Bies, Seth Kulick, Michael Ciul, Nizar Habash, and Ramy Eskander. 2014. Developing an Egyptian Arabic Treebank: Impact of Dialectal Morphology on Annotation and Tool Development. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2009. Arabic Gigaword Fourth Edition. LDC catalog number No. LDC2009T30, ISBN 1-58563-532-4.

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.