Joint Arabic Segmentation and Part-Of-Speech Tagging

Shabib AlGahtani
Research and Development
Ministry of Interior
Riyadh, Saudi Arabia
shabib@moi.gov.sa

John McNaught
National Centre for Text Mining
University of Manchester
Manchester, United Kingdom
John.McNaught@manchester.ac.uk

Abstract

Arabic has a very complex morphological system, though a very structured one. Character patterns are often indicative of word class and word segmentation. In this paper, we explore a novel approach to Arabic word segmentation and part-of-speech tagging relying on character information. The approach is lexicon-free and does not require any morphological analysis, eliminating the factor of dictionary coverage. Using character-based analysis, the developed system yielded state-of-the-art accuracy comparing favourably with other taggers that involve external resources.

1 Background

Part-of-speech (POS) tagging is the process of assigning a morphosyntactic role to each word in a text and hence is considered to be a crucial step that highly affects subsequent NLP tasks. The POS tagging task differs in complexity from one language to another. For instance, in languages that lack space delimitation, word boundaries must be found before tagging. With respect to Modern Standard Arabic (MSA), the importance of POS tagging is even larger due to MSA characteristics that impose a number of processing challenges. For example, POS tagging is vital for Arabic named entity recognition, due to the absence of capitalization in proper nouns. In Semitic languages including Arabic, the phenomenon of clitic attachment is another challenge adding to POS tagging complexity. The process of finding the boundaries between the lemma and the clitics attached to it is called word tokenization or segmentation. Ambiguity can arise both in segmentation and in tagging for each segment. The two tasks are closely bound in a sense that finding the correct tagging requires the correct segmentation in advance.

In this paper, we introduce a novel approach to joint Arabic tagging and segmentation relying on character patterns, adopting a character-based method. Our work is inspired mainly by Asian language processing especially Chinese character-based processing (Qian and Liu 2012). In Chinese, text is a stream of characters (symbols) that could be interpreted differently based on their context where one symbol could be an independent word or part of a word. However, there is no space delimiting feature in Chinese while it exists in Arabic between words that are a combination of segments. Arabic examples given in this paper will be transliterated using the Buckwalter transliteration scheme[1].

2 Arabic Language

The main feature of MSA that affects processing is the total or partial absence of diacritical marks that historically represented vowels, adding more complexity to both syntactic and semantic analysis. This is due to the fact that diacritics reduce the number of possible classes of the word. This feature is not present in English, but can be imagined by dropping vowels from words. For example, dropping the vowel from *is* would result in three possible interpretations: *us*, *is* and *as*. Still, vowels would have to be restored by the context to decide on the correct word.

One critical aspect of Arabic writing today is spelling errors. Common sources of spelling error were studied in (Shaalan et al. 2003), and categorized as errors of hearing, writing, morphology, etc. More details of Arabic characteristics are demonstrated in (AlGahtani 2012).

3 Arabic Morphology

Arabic derivational morphology is based on the principle of roots and patterns to generate open-

[1] http://www.qamus.org/transliteration.htm

Proceedings of the Second Workshop on Arabic Natural Language Processing, pages 108–117,
Beijing, China, July 26-31, 2015.

class stems. A root (called radical) is a sequence of consonants, commonly triliteral (Beesley 2001).

Arabic has a complex morphological system that makes it a highly inflected language, with the presence of prefixation, suffixation, inflectional and derivational processes. Although it has a complex system, it is strongly structured (Kiraz 2002). Arabic also has a rich morphological system, where words are explicitly marked for case, gender, number, definiteness, mood, person, voice, tense and other morphological features (Maamouri et al. 2006).

An Arabic word is composed of stem plus affixation to indicate tense, gender and number. In addition to affixes, clitics are attached to the beginning, the end or to both. Clitics are segments that represent an independent syntactic role: mainly conjunctions, prepositions and pronouns. Prepositions and conjunctions are attached to the beginning of the word and pronouns at the end (Diab et al. 2004). Clitics are composed of general Arabic characters that could be part of the stem, and hence pose problems for tokenization. To appreciate the problem of clitic attachment in English, we use the example illustrated in (AlGahtani et al. 2009). Consider passing English text through a noisy channel with the possibility of dropping the space delimiter between words, resulting in word concatenation. Assume the following (noisy) sentence is received:

Those cars useless fuel.

The wordform *useless* has two interpretations as it is a candidate that might have been formed by concatenation due to noise; one interpretation is correct and the other is the result of concatenating the words *use* and *less*. If we use the POS tagging information of the previous word *cars*, it would be more sensible to choose the interpretation *use less*, since verbs are more likely to follow nouns than adjectives.

Bar-Haim et al. (2005) refer to each unit of the word that represents an independent tag as a segment. In Arabic, the word [ولدك ,wldk, your child] has three valid segmentations; wld+k, w+ld+k and w+l+dk. Each of these corresponds to a number of POS tagging annotations; for example, the segmentation w+ld+k, might have the POS tagging sequence of CC+NN+PRP$. Combining both the segmentation with the tagging information constitutes a full analysis; w/CC+ld/VBD+k/PRP. These two tasks are bound together in such a way that the correct tagging analysis always encodes the correct segmentation. Multiple analysis of the following sentence is illustrated in Table 1:

[قرأ ولدك الكتاب , qr> wldk AlktAb,
Your child read the book]

word	Translation	Full Analysis
qr>	read	qr>/VBD
wldk	your boy	wld/NN +k/PRP$
	and diverted you	w/CC+ld/VBD +k/PRP
	and to demolish	w/CC+l/IN +dk/NN
AlktAb	the book	AlktAb /NN

Table 1: Sample sentence analysis

Given the clitic attachment feature in Arabic, the POS tag of a word could be compound in nature, leading to tagset extension which, in turn, adds more complexity to this task. Also, this adds the problem of data sparseness (fewer forms with specific compound tags).

4 Previous Work

Recent advances in POS tagging have introduced the concept of bidirectional learning, which has resulted in the now state-of-the-art accuracy of above 97% for English. Bidirectional learning uses previous and successive context explicitly to find the tag of the current word. One instance of bidirectional learning is the bidirectional dependency network proposed and discussed in (Toutanova et al. 2003), which yielded 97.20% on the WSJ corpus. Moreover, the same concept was also adopted to develop a biomedical text tagger, discussed in (Tsuruoka et al. 2005). Their results showed the robustness of the tagger when tested on different genres. Another instance of bidirectional learning is the perceptron-like guided learning explained in (Shen et al. 2007), which also yielded comparable results.

In Arabic POS tagging, Khoja (2001) used a hybrid technique of statistical and rule-based analysis with a morphosyntactic tagset. Later, Support Vector Machines were used to separately implement a character based word-tokenizer and a POS tagger with a collapsed tagset of the Arabic Tree Bank, achieving scores of 99.7% and 95.5% on word-tokenization and tagging respectively (Diab et al. 2004). An enhancement of this system is discussed in (Diab 2009). With the help of the rich morphological features of Arabic, Habash and Rambow were able to tackle both tokenization and tagging in one step, achieving an accuracy of 97.5% (Habash and

Rambow 2005). Later, their system was extended in (Habash et al. 2009). An HMM Hebrew tagger was ported to Arabic, yielding an accuracy of 96.1% (Mansour et al. 2007). Transformation-based Learning has been investigated in (AlGahtani et al. 2009) yielding comparable results.

A recent morphological analyzer and POS tagger was implemented and discussed in (Sawalha and Atwell 2010). With 22 morphological features, their tool produces all possible analyses of Arabic words including lemma, root, pattern and vowelization (adding diacritical marks).

MADAMIRA discussed in (Pasha et al. 2014) is a hybrid system that combines aspects of MADA (Habash and Rambow 2005) and AMIRA (Diab 2009). Their system was blind-tested on part of the standard split highlighted in (Diab et al. 2013).

The performance of the systems discussed in this brief review are given if the tool has been tested on a standard dataset although not standard settings.

Selecting the most appropriate tagger for an application is quite difficult given these taggers have not been benchmarked due to the lack of standard test data which was not defined until recently. However, we expect character-based approaches to be more portable to different text genres. Most taggers developed for Arabic employ lexicons either directly or implicitly by using morphological analyzers. To our knowledge, there is no state of the art accuracy tagger that is lexicon free.

5 Joint segmentation and tagging Approach

Most taggers attempt to find the correct segmentation of a word before tagging, i.e., sequential processing. Sequential processing limits transfer and sharing of knowledge between different levels of analysis. Moreover, errors committed at any level of analysis will propagate to the subsequent levels. Word-based and segment-based techniques are highly affected by noise such as degraded text in the Web where people do not follow standard writing. Character-based approaches are very robust techniques and more efficient with unknown words due to their capability of capturing internal word patterns. Spelling errors are thus more tolerated in such approaches.

Given the rich morphology of Arabic that is encoded in a very structured system, a character-based approach would be appropriate to capture external and internal patterns. Also, given that clitics are always at the boundary and tagging of the word is governed by patterns, a joint approach will be used for both tasks.

To achieve the target of this study, we focus on the character as the unit of analysis. The aim is to find the word boundaries and find the correct POS of the word jointly. We model the problem as a sequence tagging problem, using machine learning. The learning algorithm's goal is to build a probability model; the model's goal in the decoding phase is to find the best sequence of character tags given raw text characters.

In the segmentation task, the appropriate representation is the same as used for boundary detection: IOB representation discussed in (Kudo and Matsumoto 2001). The IOB scheme was successfully applied to Arabic segmentation by (Diab et al. 2004). We investigate both IOB (Inside, Outside, Begin) and the more comprehensive IOBES (Inside, Outside, Begin, End, Single). The only modification is that the O indicator will not be used as each character should have a tag. In POS tagging, the tagset is the Arabic collapsed tagset[2].

segment (s)	t(s)	char (c)	t(c)
w	CC	w	S-CC
b	CC	b	S-CC
ktAbAt	NNS	k	B-NNS
		t	I-NNS
		A	I-NNS
		b	I-NNS
		A	I-NNS
		t	E-NNS
hm	PRP$	h	B-PRP$
		m	E-PRP$

Table 2: Arabic segment vs. character tagging of "wbktAbAthm"

Since the two tasks are bound, the joint segmentation and tagging is done by merging the tagsets of both tasks. Extending the POS tagset with character position indicators used in IOBES adds 4 subcategories to each tag. For instance, the NN tag will be extended to S-NN, B-NN, I-NN and E-NN. We also use a special character tag for the space delimiter. Table 2 illustrates the tagging of segments vs. characters, where s, c

[2] http://www.ircs.upenn.edu/arabic/Jan03release/arabic-POStags-collapse-to-PennPOStags.txt

and t represent segment, character and tag, respectively.

We use maximum entropy modeling to build our tagging model. Maximum entropy modeling has been widely used in various NLP tasks including POS tagging. It is known for its ability to combine features from diverse knowledge sources successfully. Given a sequence of words:

$$w = \{w_1, w_2, w_3, \ldots, w_{n-1}, w_n\}$$

we try to find the best POS sequence:

$$t = \{t_1, t_2, t_3, \ldots, t_{n-1}, t_n\}$$

by splitting the words into their characters:

$$c = \{c_1, c_2, c_3, \ldots, c_{n-1}, c_n\}$$

then finding the best sequence of tags for characters:

$$tc = \{tc_1, tc_2, tc_3, \ldots, tc_{n-1}, tc_n\}$$

Finally, we reconstruct the word POS tags from character tags.

In the decoding phase, character tags will be evaluated using gold standard (GS) annotations. The elementary decision of the tagger is finding the tag of each character from a tagset of (4 x 27) possible tags: t(c) in Table 2.

Beam search with window size 5 is used to find the best sequence of character tags for the whole sentence and to assist in ignoring inadmissible sequences i.e. 'Inside' tag following 'End' (E-NN, I-NN).

After decoding, the bare tags of the segments are constructed from the character tagging sequence. The position indicator is stripped of the tag and if a middle character has the position indicator "B" or "S" that means it is a start of a new segment and a plus sign "+" is inserted.

As per Arabic writing, some letters might change based on their position in the word. Feminine indicator 'p' is changed to 't' when connected to a pronoun. Such a case does not receive special processing in our approach since the tagger tries to find the POS tag of a sequence of characters without attempting to find the standard form of the word. If the tagger has never seen this word form in training, it still has the chance of correctly tagging it using its features and the features of surrounding characters. The only preprocessing we do is in the case of omission such as omitting 'A' from determiner 'Al' when connected to a preposition 'l' producing 'll'. We replace any 'll' in the input text with 'lAl'. This rule of transformation will fail in very rare cases i.e. when a word starts with letter 'l'.

We use a Java implementation available in the openNLP project[3] which has been used extensively in NLP tasks.

The feature set used is a combination of lexical and contextual features of the stream of characters, focusing on the current character. For instance, consider character (*H*) in the word w*b>bHAvhm* underlined in the following sentence:

"$ArkwA bHwrhm wb>bHAvhm fy Alm&tmr"

"شاركوا بحضورهم وبأبحاثهم في المؤتمر"

which translates as: "participated with their attendance and with their researches in the conference".

feature	description	value
c_i	curr char	H
c_{-1}	prev char	b
c_{-2}	prev char	>
$c_{i-1}c_i$	prev, curr char	bH
$c_{i-2}c_{i-1}$	prev prev, prev char	>b
$c_{i-2}c_{i-1}c_i$	prev prev, prev, curr	>bH
$c_{i-1}c_ic_{i+1}$	prev,curr,next	bHA
c_{+1}	next char	A
c_ic_{i+1}	curr,next char	HA
$c_{i+1}c_{i+2}$	next, next next char	Av
$c_ic_{i+1}c_{i+2}$	curr,next,next next char	HAv
c_0	leading char	w
c_0c_1	leading bigram	wb
$c_0c_1c_2$	leading trigram	wb>
c_n	trailing char	m
c_nc_{n-1}	trailing bigram	hm
$c_nc_{n-1}c_{n-2}$	trailing trigram	vhm
tc_{i-1}	tag of prev char	I-NN
tc_{i-2}	tag of prev prev char	I-NN
tb_{i-1}	bare tag of prev char	NN
tb_{i-2}	bare tag of prev prev char	NN
w_i	curr word	wb>bHAvhm
w_{i-1}	prev word	bHDwrhm
w_{i+1}	next word	fy
tw_{n-1}	tag of prev word	IN+VBN
tw_{n-2}	tag of prev prev word	VBD

Table 3: Feature set example

Table 3 gives a list of the features generated for that character, assuming previous context preceding this character has already been processed. Here, w, c, tw, tc, tb, i, n represent word,

[3] http://maxent.sourceforge.net/

character, word tag, character tag, bare tag, index of character and last character.

6 Experiments

6.1 Corpus

The corpus used in our experiments is the Arabic Tree Bank (ATB) which is a standard data set developed by the Linguistic Data Consortium (LDC)[4]. It is manually annotated with morphology syntactic features. The Treebank has gone through a number of revisions. Although previous studies involved the same corpus, different splits were used. The most common parts used in previous studies are ATB 1,2,3 as noted by (Diab et al. 2013) where a standard split was also defined. Table 4 shows details of parts used in this experiment following Diab et al. (2013) guidelines.

Part	Version	LDC Catalog	Source
ATB1	4.1	LDC2010T13	AFP
ATB2	3.1	LDC2011T09	Ummah
ATB3	3.2	LDC2010T08	Annahar

Table 4: Corpus ATB parts

The total number of words is some 738k. The annotations include morphological analysis and syntactic trees of sentences. For our task, only the morphological analysis is needed. We first mapped the morphological analysis annotation to the Arabic collapsed tagset distributed with ATB, which comprises 24 tags. We maintained two versions of the same corpus: **uns**egmented **c**orpus (UNSC) and **seg**mented **c**orpus (SEGC). The format in the unsegmented version is a full word level one (compound tag) whereas, in the segmented version, single tags are produced. Assigning extended tags to word characters occurs in the training phase where each word is split into its characters then tags assigned as described.

Table 5 shows the number of words in both segmented and unsegmented format. A word list was built from each version. The format of the word list is simply each word with the possible tags and their frequencies. The word list size varies in both given the generative behaviour of Arabic. Tag-per-word measure is given to appreciate the complexity of the task, showing 1.8 in the segmented corpus. Almost half of the corpus was ambiguous in the sense that a word was tagged with at least two different tags. We note also that a word could be formed by up to 6 segments although very rarely.

	SEGC	UNSC
corpus size (k)	738.89	637.02
tag-per-word	1.862	1.539
ambiguous token (%)	49.33	36.85
word list size (k)	46.529	68.031
ambiguous tokens (%)	11.13	8.64
Word count per number of tags	1=41348 2=4456 3=600 4=103 5=19 6=3	1=62148 2=5176 3=593 4=94 5=16 6=4

Table 5: Corpus ambiguity analysis

6.2 Settings

In order to evaluate the performance of our approach, experiments were conducted on each version of the corpus. Each experiment was performed on segmented text (GS segmentation provided in corpus) and on the unsegmented version. The unsegmented text is the primary goal of this approach, i.e, performing segmentation and tagging simultaneously.

Division	Doc	Document_Range
ATB1_TRAIN	568	20000715_AFP_ARB.0074 20001115_AFP_ARB.0128
ATB1_TEST	95	20001115_AFP_ARB.0129 20001115_AFP_ARB.0236
ATB2_TRAIN	400	UMAAH_UM.ARB_20020224-a.0005 - UMAAH_UM.ARB_backissue_34-a.0013
ATB2_TEST	51	UMAAH_UM.ARB_backissue_34-a.0014 - UMAAH_UM.ARB_backissue_40-e.0025
ATB3_TRAIN	480	ANN20020215.0001 ANN20021115.0033
ATB3_TEST	61	ANN20021115.0034 ANN20021215.0045

Table 6: Standard split (Diab et al. 2013)

The split used is the same setting detailed in (Diab et al. 2013). The training set was a combination of the three training set parts. The test set was formed likewise, as in Table 6. We followed the exact setting, excluding the development part as it was not required for our model.

We firstly constructed baselines for the two corpus versions, by assigning the most frequent

[4] https://www.ldc.upenn.edu

tag in the training set to corresponding test set tokens and tagging OOV tokens as NN. Most frequent tags were extracted from a word list built from the training set to produce the analysis.

In the first experiment, a statistical tagging model was produced using the joint segmentation and tagging approach detailed in section 5. In this experiment, we evaluate our approach on the two versions of the corpus. The segmented corpus is already segmented using the GS segmentation provided in the corpus, thus only testing of POS tagging accuracy is actually performed. The full evaluation of our joint approach is carried out by testing on the unsegmented corpus.

As the ATB was generated from different sources and annotated at different times, presumably by different annotators, in our second experiment we measured the performance on different parts only with respect to the unsegmented corpus. The performance is measured on each ATB part independently with its corresponding split.

Finally, a confusion matrix and error analysis was produced.

6.3 Results and discussion

The baseline tagger, which tags each token with the most frequent tag, achieved 91.02% and 88.34% on segmented and unsegmented corpora, respectively.

The results of the joint approach are shown in Table 7, which provides details of results obtained at each stage of the first experiment on segmented and unsegmented versions of the corpus. The performance is comparable to the state of the art, achieving an accuracy of 95.54% and 94.29%, on segmented and unsegmented corpora, respectively, yielding 4.5% and 6.95% improvements over the baseline.

	SEGC	UNSC
Size	664.95k	573.11k
Train set	590.82k	509.23k
Test set	74.13k	63.87k
OOV	4.14%	8.49%
Baseline POS	91.02%	88.34%
OOV Baseline acc.	30.09%	13.74%
Joint POS acc.	95.54%	94.29%
Joint SEG acc.	100 GS	99.36%
Joint OOV acc.	75.89%	73.811%
Joint POS acc. no lex	-----	93.00%
Joint SEG acc. no lex	-----	99.13
tag set count	27	186

Table 7: Experimental results

The difference in unknown words percentage between the two versions demonstrates the higher data sparseness in the unsegmented text, which is consistent with the fact that sparseness is increased due to clitic attachment.

The number of OOV items in the unsegmented corpus was double that of the segmented corpus, interestingly; guessing accuracy of unknown words in both experiments is almost equal, above 70%. The OOV guessing as NN in the baseline on the segmented corpus was double the accuracy of that on the unsegmented one. This was probably the cause of degradation by 3% in performance of the baseline between the two versions.

Original tagging inconsistency of the ATB dataset is present in some tokens, e.g., month names are tagged as either NN or NNP, which is also a cause of degradation.

The segmentation module achieved an accuracy of 99.4% on the unsegmented corpus, while segmented corpus evaluation used the gold standard segmentation. Segmentation accuracy was calculated as number of words correctly segmented over the total number of words. The result is comparable to what has been achieved by other systems. The superior accuracy of the segmentation was achieved due to the low number of words having multiple segmentations in the corpus.

Disabling lexical features (word, previous word, next word) had higher effect on tagging than segmentation performance. The accuracy degradation was 1.29% in tagging and 0.23% in segmentation.

Applying IOBES representation performed slightly better than IOB, with 0.2% difference in tagging accuracy. Table 7 results are achieved using the IOBES scheme.

The results of testing the model on each part independently are shown in Table 8. The model trained on the whole training set is tested on the test set of each part. Then a single model is built from each training set of each part and tested on the test set of the given part. The highest scores are in bold showing the best tagging was achieved on ATB1 and best segmentation on ATB2.

Train/Test	Task	ATB1	ATB2	ATB3
per-part	**POS**	**94.37**	93.75	93.03
	SEG	99.24	**99.29**	99.12
all parts	**POS**	**95.45**	95.09	93.44
	SEG	99.53	**99.64**	99.23

Table 8: Testing results per part

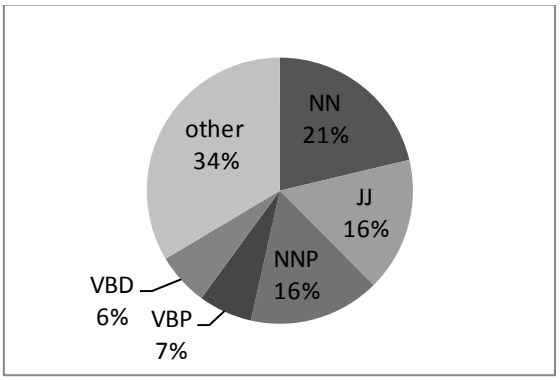

Figure 1: Error distribution – SEGC

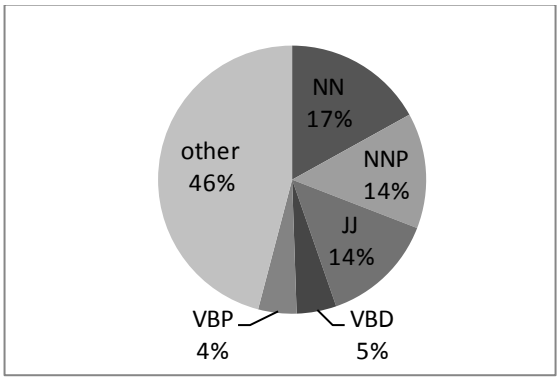

Figure 2: Error distribution – UNSC

	total error	largest target	total count	relative
NN	685	JJ	20867	3.28
JJ	524	NN	6106	8.58
NNP	513	NN	5967	**8.60**
VBP	211	NN	2663	7.92
VBD	206	NN	3047	6.76

Table 9: Most errorneous classes – SEGC

	total error	largest target	total count	relative
NN	612	JJ	16342	3.74
NNP	503	NN	5421	9.28
JJ	498	NN	5854	8.51
VBD	173	NN	1772	**9.76**
VBP	167	NN	2018	8.28

Table 10: Most errorneous classes – UNSC

To determine the highest ambiguous classes, we generated the confusion matrix of our tagger errors. The pie charts in Figure 1 and Figure 2 show the largest classes of the errors committed by the tagger in the two experiments. The three largest classes NN, JJ and NNP constitute almost half of the errors. The NN error rate is affected by the frequency of occurrence of that class in the corpus. Also, nouns share most of adjective and some verb forms.

Given that this measure is affected by the frequency of specific tags, we calculated the relative error where the number of errors is divided by the total number of occurrences of the given class (last columns of Table 9 and Table 10). On the segmented corpus, the NNP class has the highest relative error followed by JJ. This was due to the general case of Arabic proper nouns that are in the form of general nouns or adjectives. Arabic proper noun characteristics are highlighted in (AlGahtani 2012). Adjectives share most morphological features with nouns, such as gender and number indicators.

On the unsegmented version, the highest relative error was VBD and NNP. Errors in tagging VBD are attributed to verbs sharing the exact form of writing with nouns apart from a different vowelization, which is not present in written MSA. The largest error target class was tagging NN as JJ, followed by the remaining five classes tagged as NN affected by the dominating number of NN in the corpus.

The other analysis we carried out was to find the most erroneous tokens in our experiments. The list of the highest 10 tokens are in Table 11 and Table 12. These tokens were highly ambiguous in terms of the number of tags they could be assigned. The tables show each token with possible tags and frequency. The ones having unique tags but that were mistagged are due to the use of "_" as token/tag separator by our training algorithm implementation, which will be reconsidered in a future experiment.

The token "hA" was in the list due to being used as possessive pronoun or personal pronoun based on its preceding token. If the preceding token is mistagged, it will also be mistagged as a result. The rule is when the preceding token is a verb then the following pronoun is a personal pronoun and if the preceding token is a noun then it is possessive.

We have not been able to compare this work with previous work due to different settings used. The only published work that applied the splits highlighted in (Diab et al. 2013) was (Pasha et al. 2014). However, another tagset was used and their test was only on part of the test set, 25k blindly selected from the test set. Mapping from the ATB tagset to their tagset was not feasible.

token	error/all	tag	frequency
f	56/226	CC	1094
		NN	199
		RP	688
		IN	17
An	53/867	IN	7124
		VBP	693
		NN	36
		NNP	3
lA	45/274	RP	1921
		VBP	350
		CC	77
		UH	15
		NNP	5
mA	45/335	WP	2047
		IN	801
		RP	145
		NN	37
		VBP	15
l_	33/33	IN	223
mn	32/1356	IN	9978
		WP	303
		RP	23
h	27/1234	PRP$	5721
		PRP	4630
		RP	1
		NN	1
AlvAny	27/76	ADJNUM	158
		NNP	152
		JJ	9
hA	23/1160	PRP$	4466
		PRP	4122
		DT	4
		VBP	2
		UH	1
w	22/4644	CC	35983
		IN	196
		NN	77

Table 11: Most erroneous tokens – SEGC

token	error /all	tag	frequency
An	36/643	IN	5586
		VBP	303
		NN	36
		NNP	3
l_	33/33	IN	33
mA	32/268	WP	1672
		IN	763
		RP	93
		NN	37
		VBP	12
mn	27/1209	IN	8972
		WP	192
		RP	22
lA	27/188	RP	1363
		VBP	246
		CC	58
		UH	15
AlvAny	26/72	ADJNUM	150
		NNP	136
		JJ	9
Al_	18/18	DT	171
<n	16/49	IN	2098
		NN	9
wlA	16/63	CC+RP	419
		CC+VBP	88
		CC+CC	19
		IN+RP	2
b_	15/15	IN	15

Table 12: Most erroneous tokens – UNSC

7 Future Work

The study has showed that our approach succeeded in performing segmentation and tagging jointly. The tagger designed performs comparably to state of the art taggers for Arabic POS tagging, without knowledge-deep features, as well as being lexicon-free. This approach is applicable to any concatenating language such as the Semitic family languages.

To improve our tagger, we plan to have a wider context of features. Also, we plan to apply it in other tasks such as morphological analysis and named entity recognition.

Acknowledgments

We would like to acknowledge the support of Ilya Ahtaridis, from LDC, for providing upgrades of ATB versions.

References

AlGahtani, S., W. Black, and J. McNaught. 2009. 'Arabic Part-of-Speech Tagging Using Transformation-Based Learning'. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. Cairo, Egypt.: The MEDAR Consortium. <http://www.elda.org/medar-conference/pdf/43.pdf>.

AlGahtani, S. 2012. 'Arabic Named Entity Recognition: A Corpus-Based Study'. University of Man-

chester. <http://www.manchester.ac.uk/escholar/uk-ac-man-scw:158690>.

Beesley, K. 2001. 'Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001'. In *ACL Workshop on Arabic Language Processing: Status and Perspective*, 1:1–8. Toulouse, France. <http://www.xrce.xerox.com/Research-Development/Publications/2001-0094/%28language%29/eng-GB>.

Diab, M. 2009. 'Second Generation AMIRA Tools for Arabic Processing: Fast and Robust Tokenization, POS Tagging, and Base Phrase Chunking'. In *Proceedings of the MEDAR'09 Conference*. Cairo, Egypt. <www.elda.org/medar-conference/pdf/56.pdf>.

Diab, M., K. Hacioglu, and D. Jurafsky. 2004. 'Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks'. In *Proceedings of HLT-NAACL 2004: Short Papers*, 149–152. <http://dl.acm.org/citation.cfm?id=1614022>

Diab, Mona, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. 'LDC Arabic Treebanks and Associated Corpora: Data Divisions Manual'. *arXiv Preprint arXiv:1309.5652*. <http://arxiv.org/abs/1309.5652>.

Habash, N., and O. Rambow. 2005. 'Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop'. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 573–580doi:10.3115/1219840.1219911. . <http://dl.acm.org/citation.cfm?id=1219911>

Habash, N., O. Rambow, and R. Roth. 2009. 'Mada+ Tokan: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, Pos Tagging, Stemming and Lemmatization'. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR)*. Cairo, Egypt. <www.elda.org/medar-conference/pdf/24.pdf>.

Bar-Haim, R., K. Sima'an, and Y. Winter. 2005. 'Choosing an Optimal Architecture for Segmentation and POS-Tagging of Modern Hebrew'. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, 39–46. <http://dl.acm.org/citation.cfm?id=1621796>

Khoja, S. 2001. 'APT: Arabic Part-of-Speech Tagger'. In *Proceedings of the Student Workshop at NAACL-2001*, 20–25. <https://nats-www.informatik.uni-hamburg.de/intern/proceedings/2001/naacl/srw/pdf/srw-06.pdf>.

Kiraz, G. 2002. 'Computational Nonlinear Morphology: With Emphasis on Semitic Languages'. *Computational Linguistics* 28 (4): 576–58doi:10.1162/coli.2002.28.4.576. . <http://dx.doi.org/10.1162/coli.2002.28.4.576>.

Kudo, T., and Y. Matsumoto. 2001. 'Chunking with Support Vector Machines'. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, 1–8. <http://dx.doi.org/10.3115/1073336.1073361>.

Maamouri, M., A. Bies, and S. Kulick. 2006. 'Diacritization: A Challenge to Arabic Treebank Annotation and Parsing'. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*. <http://papers.ldc.upenn.edu/NLTSG/DiacritizationArabicTreebank.pdf>.

Mansour, S., K. Sima'an, and Y. Winter. 2007. 'Smoothing a Lexicon-Based Pos Tagger for Arabic and Hebrew'. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, 97–103. <http://dl.acm.org/ft_gateway.cfm?id=1654593&type=pdf&CFID=95957381&CFTOKEN=73775663>.

Pasha, A, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth. 2014. 'Madamira: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic'. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*. <http://www.lrec-conf.org/proceedings/lrec2014/pdf/593_Paper.pdf>

Qian, X, and Y. Liu. 2012. 'Joint Chinese Word Segmentation, POS Tagging and Parsing'. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 501–511. <http://dl.acm.org/citation.cfm?id=2391007>.

Sawalha, M., and E. S. Atwell. 2010. 'Fine-Grain Morphological Analyzer and Part-of-Speech Tagger for Arabic Text'. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation ((LREC'10))*, 1258–1265. Valleta, Malta. <http://www.lrec-conf.org/proceedings/lrec2010/pdf/282_Paper.pdf>

Shaalan, K., A. Allam, and A. Gomah. 2003. 'Towards Automatic Spell Checking for Arabic'. In *Proceedings of the 4th Conference on Language Engineering ((ELSE '03))*, 240–247. Cairo, Egypt: Egyptian Society of Language Engineering. <http://www.claes.sci.eg/NARIMS_upload/CLAESFILES/3847.pdf>.

Shen, L., G. Satta, and A. Joshi. 2007. 'Guided Learning for Bidirectional Sequence Classification'. In

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 45:760–767. Prague, Czech Republic. <http://acl.ldc.upenn.edu/P/P07/P07-1096.pdf>.

Toutanova, K., D. Klein, C.D. Manning, and Y. Singer. 2003. 'Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network'. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, 1:173–180. Edmonton, Canada. <http://dx.doi.org/10.3115/1073445.1073478>.

Tsuruoka, Y., Y. Tateishi, J.D. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii. 2005. 'Developing a Robust Part-of-Speech Tagger for Biomedical Text'. *Advances in Informatics* 3746. Lecture Notes in Computer Science: 382–392. <http://dx.doi.org/10.1007/11573036_36>.