a pre-annotated corpus which is used for training. The unsupervised POS tagging models do not require a pre-annotated corpus [2][4][8][13].

There has been different work done in POS tagging for Arabic. A recent survey for Arabic POS taggers is found in [7]. There are some comparative studies found in the literature on POS tagging for languages other than Arabic and can be found in [3][4][5]. Upon our knowledge there are no such studies are attempted for Arabic. In this paper we compared the performance of different tagging techniques such as N-Gram tagger, Brill's tagger, HMM, and TnT tagger for Arabic.

III. THE PROPOSED APPROACH

We used the Natural Language Toolkit, NLTK [9]. NLTK is open source Python modules, linguistic data and documentation for research and development in natural language processing and text analytics, with distributions for Windows, Mac OSX and Linux. We have experimented N-Gram, HMM, Brill and TnT, tagging modules from NLTK for Arabic Quran corpus. Some preprocessing work is done on the corpus, and some Python modules have been implemented and integrated into NLTK to process the corpus.

A. N-Gram tagging

Unigram taggers are based on a simple statistical algorithm. For each token, assign the tag that is most likely for that particular token. A unigram tagger only uses a single word as its context for determining the part-of-speech tag. Unigram Tagger can be trained by giving it a list of tagged sentences at initialization. The training process involves inspecting the tag of each word and storing the most likely tag for any word in a dictionary, stored inside the tagger. In addition to unigram tagger, there is bigram tagger which uses the previous tag as part of its context, and trigram tagger which uses the previous two tags. An N-Gram is a subsequence of n items, so the bigram tagger looks at two items (the previous tag and word), and the trigram tagger looks at three items. An N-Gram tagger is a generalization of a unigram tagger whose context is the current word together with the part-of-speech tags of the n-1 preceding tokens. An N-Gram tagger picks the tag that is most likely in the given context.

B. Backoff tagging

Backoff tagging is one of the core features of automated NLTK taggers. The purpose of it is to chain taggers together so that if one tagger doesn't know how to tag a word, it can pass the word on to the next backoff tagger. If that one can't do it, it can pass the word on to the next backoff tagger, and so on until there are no backoff taggers left to check. The only problem with the unigram tagger occurs if it encounters a token which has not been seen yet. In that case it assigns the default tag "NONE" to the token. Thus a default backoff tagger is used in unigram taggers to assign the tag "N" to untagged words. So after the tagging process every token has a valid and meaningful tag. By themselves, bigram tagger and trigram tagger perform quite poorly. This is partly because they cannot learn context from the first word(s) in a sentence. And as *n* gets larger, the specificity of the contexts increases, as does the chance that the data we wish to tag contains contexts that were not present in the training data. This is known as the *sparse data* problem, and is quite pervasive in NLP. They can make a contribution when we combine them with backoff tagging. For example, we could combine the results of a bigram tagger, a unigram tagger, and a default tagger, as follows:

1. Try tagging the token with the bigram tagger.
2. If the bigram tagger is unable to find a tag for the token, try the unigram tagger.
3. If the unigram tagger is also unable to find a tag, use a default tagger.

C. Brill tagger

The Brill Tagger is a transformation-based tagger. The Brill tagger uses a series of rules to correct the results of an initial tagger. These rules are scored based on how many errors they correct minus the number of new errors they produce. The Brill tagger guesses the tag of each word, then goes back and fixes the mistakes. In this way, a Brill tagger successively transforms a bad tagging of a text into a good one. As with N-Gram tagging this is a supervised learning method, since we need annotated training data. However, unlike N-Gram tagging, it does not count observations but compiles a list of transformational correction rules.

D. HMM tagger

The HMM is a probabilistic tagger that uses a Hidden Markov Model to find the most likely tag sequence for each sentence. The unigram tagger only considers the probability of a word for a given tag t; the surrounding context of that word is not considered. On the other hand, for a given sentence or word sequence, HMM taggers choose the tag sequence that maximizes the following formula:

P (word | tag) * P (tag | previous n tags).

E. TnT tagger

TnT stands for Trigrams'n'Tags. It is a statistical tagger based on second order Markov models, thus looking two words into the past [14]. The states of the model represent tags, outputs represent the words. Transition probabilities depend on the states, thus pairs of tags. Output probabilities only depend on the most recent category.

Transition and output probabilities are estimated from a tagged corpus. The TnT tagger is a trigram tagger where the probability of a tag depends on the previous two tags.

In our experiment we have used the taggers: Unigram, HMM, Brill's transformation based tagger, and TnT, described above. More detailed descriptions of these taggers and how to use them in NLTK are available in [6][12].

IV. The Used Corpus and Tagset

We used Quranic Arabic Corpus [11] named as: "quranic-corpus-text-0.2". The language form of the corpus is classical Arabic (CA). The Arabic alphabet of the corpus consists of the following letters: (ا ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي ء ى ة). The phonetic transcription for letters is shown in Table 1. The diacritics that used in the corpus are (بً بٌ بِ بَ بُ بّ بْ) [10], which are represented by the character (ب), and the phonetic transcription for these diacritics are shown in Table 2.

TABLE 1. Phonetic transcription for Arabic letters used in Quran corpus

Letter	Arabic	Transcription	Letter	Arabic	Transcription
Alif	ا	ā	ẓā	ظ	ẓ
Bā	ب	b	ʿayn	ع	ʿ
Tā	ت	t	ghayn	غ	gh
Thā	ث	th	fā	ف	f
Jīm	ج	j	qāf	ق	q
ḥā	ح	ḥ	kāf	ك	k
Khā	خ	kh	lām	ل	l
Dāl	د	d	mīm	م	m
Dhāl	ذ	dh	nūn	ن	n
Rā	ر	r	hā	ه	h
Zāy	ز	z	wāw	و	w
sīn	س	s	yā	ي	y
shīn	ش	sh	hamza	ء	'
ṣād	ص	ṣ	alif maksura	ى	ā
ḍād	ض	ḍ	ta marbūta	ة	t
ṭā	ط	ṭ			

The Quran corpus consists of 6236 sentences (verses) with total of 77430 tokens (words). The corpus used 33-tag tagset which consists of the following tags: Noun (N), Verb (V), Preposition (P), Proper Noun (PN), Relative Pronoun (REL), Personal Pronoun (PRON), Negative Particle (NEG), Accusative Particle (ACC), Adjective (ADJ), Time Adverb (T), Demonstrative Pronoun (DEM), Conditional Particle (COND), Coordinating Conjunction (CONJ), Subordinating conjunction (SUB), Location adverb (LOC), Restriction particle (RES), Particle of certainty (CERT), Interogative particle (INTG), Number (NUM), Prohibition particle (PRO), Inceptive particle (INC), Vocative particle (VOC), Retraction particle (RET), Amendment particle (AMD), Future particle (FUT), Exhortation particle (EXH), Exceptive particle (EXP), Explanation particle (EXL), Surprise particle (SUR), Aversion particle (AVR), Answer particle (ANS), Quranic initials (INL), Imperative verbal noun (IMPN) [10].

TABLE 2. Phonetic transcription for Arabic diacritics used in Quran corpus

Diacritic	Arabic	Transcription
fathatan	بً	an
dammatan	بٌ	un
kasratan	بٍ	in
fatha	بَ	a
damma	بُ	u
kasra	بِ	i
shadda	بّ	(double)
sukūn	بْ	'

A. Corpus preprocessing

In order to make the corpus available and easily processed by the NLTK automated taggers, some preprocessing steps were done on the original corpus. The original corpus is stored in a plain text file. When viewing the content of this file, each word appears in an individual line. A sample view of some words is depicted in Figure 1.

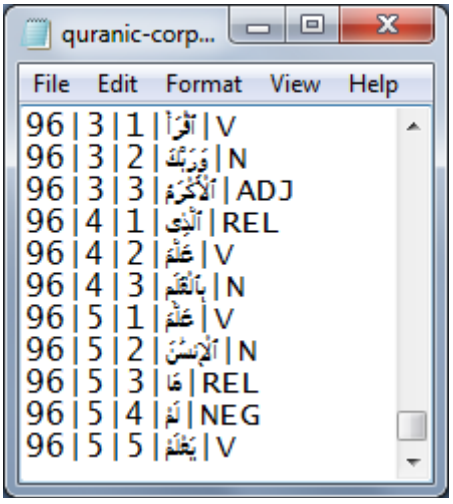

Figure 1. A sample view of some words from the original Quran corpus

As shown in Figure 1, each line includes the following information: the number of chapter (soursa), separator, the number of sentence (verse), separator, the number of word (token), separator, the word itself, separator, and finally the tag of the word.

Some processing is done on the content of the corpus file so that all individual words that belong to a single verse will occupy only one line in the file. This will let the NLTK's built-in tagging functions access the corpus file correctly. A sample view of some verses from the new created file is depicted in Figure 2. As shown in Figure 2, each verse occupies only one line.

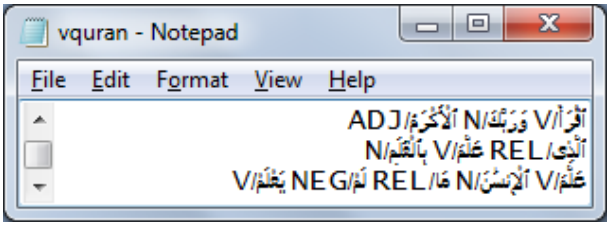

Figure 2. A Sample view of some diacritized verses from the Quran corpus

The undiacritized form of the corpus is stored in an additional new file by removing the diacritics from all the words of the corpus. A sample view of some undiacritized verses is depicted in Figure 3. It is important to mention that the undiacritized form of the Quran corpus will be used only for our study experiments. Experimental results on undiacritized Arabic are useful because Arabic script is mostly written without diacritics.

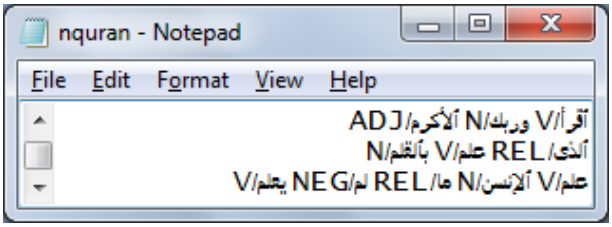

Figure 3. A Sample view of some undiacritized verses from the Quran corpus

Some work is also done into the NLTK tool in order to process the Quran corpus files using simplified tag set. The simplified tagset includes only 9 tags which are: Noun(N),Verb (V), Derived nominal (ADJ), Pronoun (PRON), Adverb(ADV), Preposition(PREP), Conjunction (CONJ), Particle (PART) and Disconnected Letter (INL). We followed the mapping criteria shown in Table 3 that are used to convert from the comprehensive tagset (33-tag) of the original corpus to the simplified one (9-tag).

TABLE 3. MAPPING FROM 33-TAG TAGSET TO 9-TAG TAGSET FOR THE QURAN CORPUS

Comprehensive tagset (33 tags)	Simplified tagset (9 tags)
N, PN, NUM	N
V	V
ADJ, IMPN	ADJ
PRON,REL, DEM	PRON
T, LOC	ADV
P	PREP
CONJ, SUB	CONJ
INL	INL
PRO, NEG, ACC, COND, RES, CERT, INTG, INC, VOC, RET, AMD, FUT, EXH, EXP, EXL, SUR, AVR, ANS	PART

We implemented new Python modules to integrate the new created files of the Quran corpus with the NLTK tool in order to perform our experiments. The experiments are based on classical Arabic for both diacritized and undiacritized forms. For the two forms, we experimented in both 33-tag tagset and 9-tag tagset. This will produce 4 cases of experiments as shown in Table 4. In all cases, our test set is separated from the training corpus.

V. EXPERIMENTAL RESULTS

We implemented a Python program to execute and evaluate N-Gram, Brill, HMM and TnT taggers on classical Arabic using Quran corpus. The training set includes 5612 sentences with total of 74859 tokens, and the testing set includes 624 sentences with total of 2571 tokens. The accuracy evaluations of the taggers are shown in Table 4 and they are represented graphically as shown in Figure 4. The accuracy represents the percentage of words the taggers have tagged correctly. Thus if a tagger has correctly tagged 7 out of 8 words, its accuracy would be equal to: Accuracy = 7/8 = 87.5%

TABLE 4. ACCURACY OF VARIOUS POS TAGGERS FOR ARABIC ON QURAN CORPUS

Experiment type	Accuracy of Taggers					
	Unigram	Bigram	Trigram	Brill	HMM	TnT
Diacritized Arabic: tagset 33	80.0%	80.1%	80.0%	36.4%	72.5%	64.9%
Undiacritized Arabic: tagset:33	80.4%	80.5%	80.3%	80.9%	75.2%	69.2%
Diacritized Arabic: tagset 9	81.9%	82.0%	81.8%	38.6%	75.4%	50.9%
Undiacritized Arabic: tagset:9	82.5%	82.3%	82.4%	83.2%	77.5%	59.0%

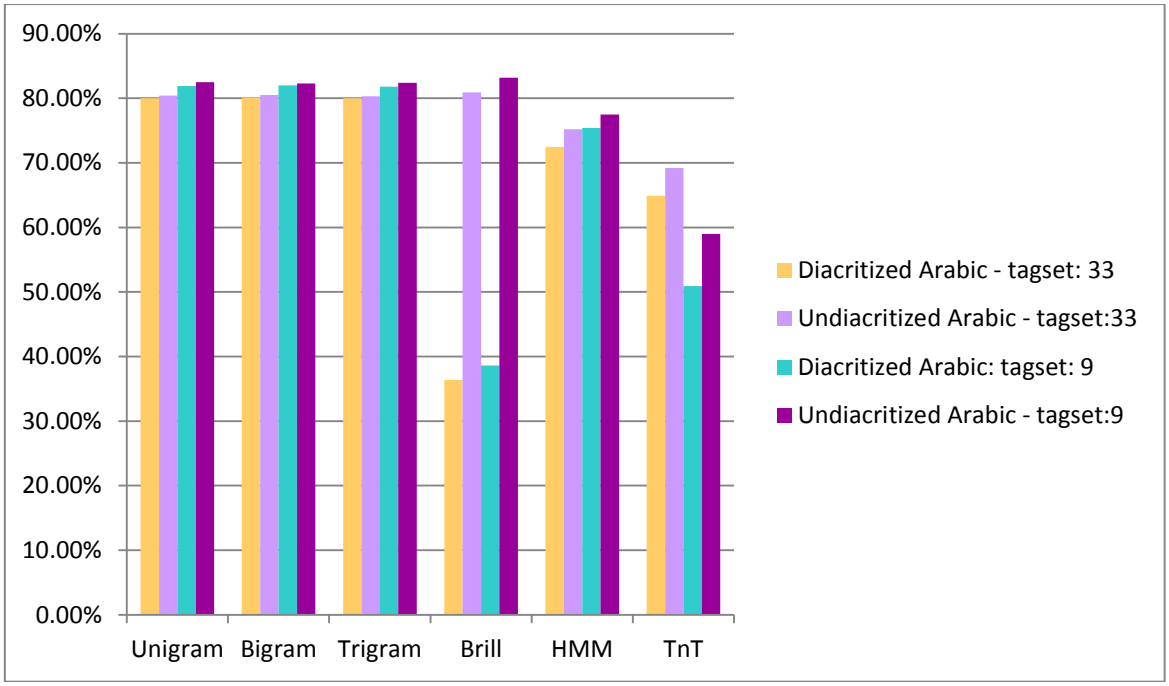

Figure 4. Accuracy of various POS taggers for Arabic on Quran corpus

VI. Analysis of Results

For Quran corpus (77430 tokens), our experiments suggest that for undiacritized Arabic (both with 33-tag tagset and 9-tag tagset), Brill's tagger performed slightly better than N-Gram (Unigram, Bigram and Trigarm) and HMM taggers, and significantly better than TnT tagger. However, for diacritized Arabic (both with 33-tag tagset and 9-tag tagset), N-Gram taggers outperform other taggers. By comparing the performances of N-Gram taggers to each other, we found that Bigram tagger outperformed Unigram and Trigram taggers when using diacritized Arabic for both 33-tag tageset and 9-tag tagset, and also when using diacritized Arabic with 9-tag tagset. While Unigram tagger outperformed Bigram and Trigram taggers when using undiacritized Arabic for 9-tag tagset.

One interesting observation is that each tagger gives better accuracy in tagging undiacritized Arabic than tagging diacritized Arabic (both with 33-tag tagset and 9-tag tagset). This is because that, diacritics are considered additional characters which are added to the word and hence will increase the ambiguity for the automatic taggers. Moreover diacritics add more morphological and linguistic complexity on the sentence being tagged. While N-Gram, HMM and TnT taggers are slightly affected with this complexity, the Brill tagger is affected relatively high. With other words, the Brill tagger has not performed well for diacritized Arabic.

It is noticed from the results that N-Gram, Brill, and HMM taggers performed better when simplifying the tags from 33 tags to 9 tags. However, for TnT tagger the reverse is correct.

VII. Conclusion and Future Work

We compared the performance of N-Gram, Brill (transformation based), HMM , and TnT, POS Taggers on Classical Arabic (CA) using Quran Corpus, and found that these taggers performed better for undiacritized Arabic than diacritized Arabic. N-Gram taggers are compared to each other. We also found that the Brill tagger performed the best for undiacritized Arabic (both 33-tag tagset and 9-tag tagset), among the taggers being compared. However, for diacritized Arabic, (both 33-tag tagset and 9-tag tagset), the transformation based Brill's approach did not perform well. We also proposed a reason behind the better performance of taggers when using undiacritized Arabic text. So researchers working on developing POS tagging techniques for diacritized Arabic text, will try to propose improvements in these techniques.

We also want to try similar comparisons using other corpora, and perform them for MSA Arabic. In parallel with this we also want to try a few other state of the art POS tagging techniques for Arabic.

References

[1] Arabic language - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Arabic_language

[2] F. Al Shamsi, and A. Guessoum. A Hidden Markov Model-Based POS Tagger for Arabic. JADT 2006:

8es Journées internationales d'Analyse statistique des Données Textuelles, 2006.

[3] F. M. Hasan, N. UzZaman and M. Khan. Comparison of different POS Tagging Techniques (N-Gram, HMM and Brill's tagger) for Bangla. Proc. of International Conference on Systems, Computing Sciences and Software Engineering (SCSS 06) of International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE 06), December 4 - 14, 2006.

[4] F. M. Hasan, N. UzZaman and M. Khan. Comparison of Unigram, Bigram, HMM and Brill's POS Tagging Approaches for some South Asian Languages. Proc. Conference on Language and Technology (CLT07), Pakistan, August 7 - 11, 2007.

[5] I. Boehm. Unigram Backoff vs. TnT. Evaluating Part of Speech Taggers, Introduction to Computational Linguistics. ByteLABS, 2005.

[6] J. Perkins. *Python Text Processing with NLTK 2.0 Cookbooks*. Packt Publishing, 2010.

[7] M. Albared, N. Omar, and M. J. Ab Aziz. Arabic Part of Speech Disambiguation: A Survey. International Review on Computers and Software, Vol. 4. n. 5, pp. 517-532, 2009.

[8] M. Albared, N. Omar, M. J. Ab Aziz, Mohd Zakree Ahmad Nazri. Automatic Part of Speech Tagging for Arabic: An Experiment Using Bigram Hidden Markov Model. In Proceedings of RSKT' 2010. pp.361-370

[9] NLTK, Natural Language Toolkit. http://www.nltk.org/Home

[10] Quran Tagset. http://corpus.quran.com/documentation/tagset.jsp

[11] Quranic Arabic Corpus. http://corpus.quran.com/

[12] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, http://www.nltk.org/book , 2009.

[13] S. Khoja. APT : Arabic Part-of-speech Tagger. In proc. of NAACL'2001 (the Student Workshop at the Second Meeting of the North American Chapter of the Association for Computational Linguistics) : 20-26, 2001.

[14] T. Brants. TnT:A statistical Part-of-speech tagger. Proceeding ANLC '2000 Proceedings of the sixth conference on Applied natural language processing, 2000.