The Effect of Automatic Tokenization, Vocalization, Stemming, and POS Tagging on Arabic Dependency Parsing

Emad Mohamed
Suez Canal University
Suez, Egypt
emohamed@umail.iu.edu

Abstract

We use an automatic pipeline of word tokenization, stemming, POS tagging, and vocalization to perform real-world Arabic dependency parsing. In spite of the high accuracy on the modules, the very few errors in tokenization, which reaches an accuracy of 99.34%, lead to a drop of more than 10% in parsing, indicating that no high quality dependency parsing of Arabic, and possibly other morphologically rich languages, can be reached without (semi-)perfect tokenization. The other module components, stemming, vocalization, and part of speech tagging, do not have the same profound effect on the dependency parsing process.

1. Introduction

Arabic is a morphologically rich language in which words may be composed of several tokens and hold several syntactic relations. We define *word* to be a whitespace delimited unit and token to be (part of) a word that has a syntactic function. For example, the word *wsytzwjhA* (وسيتزوجها)(English: And he will marry her) consists of 4 tokens: a conjunction *w*, a future marker *s*, a verb inflected for the singular masculine in the perfective form *ytzwj*, and a feminine singular 3rd person object pronoun. Parsing such a word requires tokenization, and performing dependency parsing in the tradition of the CoNLL-X (Buchholz and Marsi, 2006) and CoNLL 2007 shared task (Nivre et al, 2007) also requires part of speech tagging, lemmatization, linguistic features, and vocalization, all of which were in the human annotated gold standard form in the shared task.

The current study aims at measuring the effect of a pipeline of non gold standard tokenization, lemmatization, vocalization, linguistic features and POS tagging on the quality of Arabic dependency parsing. We only assume that we have gold standard sentence boundaries since we do not agree with the sentence boundaries in the data, and introducing our own will have a complicating effect on evaluation. The CoNLL shared tasks of 2006 and 2007 used gold standard components in all fields, which is not realistic for Arabic, or for any other language. For Arabic and other morphologically rich languages, it may be more unrealistic than it is for English, for example, since the CoNLL 2007 Arabic dataset has tokens, rather than white space delimited words, as entries. A single word may have more than one syntactically functional token. Dependency parsing has been selected in belief that it is more suitable for Arabic than constituent-based parsing. All grammatical relations in Arabic are binary asymmetrical relations that exist between the tokens of a sentence. According to Jonathan Owens (1997: 52): “In general the Arabic notion of dependency and that defined in certain modern versions e.g. Tesniere (1959) rest on common principles”.

With a tokenization accuracy of 99.34%, a POS tagging accuracy of 96.39%, and with the absence of linguistic features and the use of word stems instead of lemmas, the Labeled Attachment Score drops from 74.75% in the gold standard experiment to 63.10% in the completely automatic experiment. Most errors are a direct result of tokenization errors, which indicates that despite the high accuracy on tokenization, it is still not enough to produce satisfactory parsing numbers.

2. Related Studies

The bulk of literature on Arabic Dependency Parsing stems from the two CoNLL shared tasks of 2006 and 2007. In CoNLL-X (Buchholz and

Marsi, 2006), the average Labeled Attachment Score on Arabic across all results presented by the 19 participating teams was 59.9% with a standard deviation of 6.5. The best results were obtained by McDonald et al (2006) with a score of 66.9% followed by Nivre et al (2006) with 66.7%.

The best results on Arabic in the CoNLL 2007 shared task were obtained by Hall et al (2007) as they obtained a Labeled Attachment Score of 76.52%, 9.6 percentage points above the highest score of the 2006 shared task. Hall et al used an ensemble system, based on the MaltParser dependency parser that extrapolates from a single MaltParser system. The settings with the Single MaltParser led to a Labeled Accuracy Score of 74.75% on Arabic. The Single MaltParser is the one used in the current paper. All the papers in both shared tasks used gold standard tokenization, vocalization, lemmatization, POS tags, and linguistic features.

A more recent study is that by Marton et al (2010). Although Marton et al varied the POS distribution and linguistic features, they still used gold standard tokenization. They also used the Columbia Arabic Treebank, which makes both the methods and data different from those presented here.

3. Data, Methods, and Evaluation

3.1.Data

The data used for the current study is the same data set used for the CoNLL (2007) shared task, with the same division into training set, and test set. This design helps in comparing results in a way that enables us to measure the effect of automatic pre-processing on parsing accuracy. The data is in the CoNLL column format. In this format, each token is represented through columns each of which has some specific information. The first column is the ID, the second the token, the third the lemma, the fourth the coarse-grained POS tag, the fifth the POS tag, and the sixth column is a list of linguistic features. The last two columns of the vector include the head of the token and the dependency relation between the token and its head. Linguistic features are an unordered set of syntactic and/or morphological features, separated by a vertical bar (|), or an underscore if not available. The features in the CoNLL 2007 Arabic dataset represent case, mood, definiteness, voice, number, gender and person.

The data used for training the stemmer/tokenizer is taken from the Arabic Treebank (Maamouri and Bies, 2004). Care has been taken not to use the parts of the ATB that are also used in the Prague Arabic Dependency Treebank (Haijc et al 2004) since the PADT and the ATB share material.

3.2. Methods

We implement a pipeline as follows

(1) We build a memory-based word segmenter using TIMBL (Daelemans et al, 2007) which treats segmentation as a per letter classification in which each word segment is delimited by a + sign whether it is syntactic or inflectional. A set of hand-written rules then produces tokens and stems based on this. Tokens are syntactically functional units, and stems are the tokens without the inflectional segments, For example, the word *wsytzwjhA* above is segmented as *w+s+y+tzwj+hA*. The tokenizer splits this into four tokens *w*, *s*, *ytzwj*, and *hA*, and the stemmer strips the inflectional prefix from *ytzwj* to produce *tzwj*. In the segmentation experiments, the best results were obtained with the IB1 algorithm with similarity computed as weighted overlap, relevance weights computed with gain ratio, and the number of *k* nearest distances equal to 1.

(2) The tokens are passed to the part of speech tagger. We use the Memory-based Tagger, MBT, (Daelemans et al: 2007). The MBT features for known words include the two context words to the left along with their disambiguated POS tags, the focus word itself, and one word to the right along with its ambitag (the set of all possible tags it can take). For unknown words, the features include the first five letters and the last three letters of the word, the, the left context tag, the right context

ambitag, one word to the left, the focus word itself, one ambitag to the right, and one word to the right.

(3) The column containing the linguistic features in the real world dependency experiment will have to remain vacant due to the fact that it is hard to produce these features automatically given only naturally occurring text.

(4) The dependency parser (MaltParser 1.3.1) takes all the information above and produces the data with head and dependency annotations.

Although the purpose of this experiment is to perform dependency parsing of Arabic without any assumptions, one assumption we cannot avoid is that the input text should be divided into sentences. For this purpose, we use the gold standard division of text into sentences without trying to detect the sentence boundaries, although this would be necessary in actual real-world use of dependency parsing. The reason for this is that it is not clear how sentence boundaries are marked in the data as there are sentences whose length exceeds 300 tokens. If we detected the boundaries automatically, then we would face the problem of aligning our sentences with those of the test set for evaluation, and many of the dependencies would not still hold.

In the parsing experiments below, we will use the dependency parser MaltParser (Nivre et al., 2006). We will use Single MaltParser, as used by Hall et al (2007), with the same settings for Arabic that were used in the CoNLL 2007 shared task on the same data to be as close as possible to the original results in order to be able to compare the effect of non gold standard elements in the parsing process.

3.3.Evaluation

The official evaluation metric in the CoNLL 2007 shared task on dependency parsing was the **labeled attachment score** (LAS), i.e., the percentage of tokens for which a system has predicted the correct HEAD and DEPREL, but results reported also included **unlabeled attachment score** (UAS), i.e., the percentage of tokens with correct HEAD, and the **label accuracy** (LA), i.e., the percentage of tokens with correct DEPREL. We will use the same metrics here.

One major difference between the parsing experiments which were performed in the 2007 shared task and the ones performed here is vocalization. The data set which was used in the shared task was completely vocalized with both word-internal short vowels and case markings. Since vocalization in such a perfect form is almost impossible to produce automatically, we have decided to primarily use unvocalized data instead. We have removed the word internal short vowels as well as the case markings from both the training set and the test set. This has the advantage of representing naturally occurring Arabic more closely, and the disadvantage of losing information that is only available through vocalization. We will, however, report on the effect of vocalization on dependency parsing in the discussion.

To give an estimate of the effects vocalization has on dependency parsing, we have replicated the original task with the vocalized data, and then re-run the experiment with the unvocalized version. Table 1 presents the results:

	Vocalized	**Unvocalized**
LAS	74.77%	74.16%
UAS	84.09%	83.53%
LA	85.68%	85.44%

Table 1: Vocalized versus unvocalized dependency parsing

The results of the experiment indicate that vocalization has a positive effect on the quality of the parsing output, which may be due to the fact that ambiguity decreases with vocalization. Labeled attachment score drops from 74.77% on the vocalized data to 74.16% on unvocalized data. Unlabeled attachment score drops from 84.09% to 83.53% and labeled accuracy score from 85.68% to 85.44%. The difference is minimal, and is expected to be even smaller with automatic vocalization

4. Results and discussion

4.1.Tokenization

We obtain an accuracy of 99.34%. Out of the 4550 words which the test set comprises, there are only 30 errors affecting 21 out of the 132 sentences in the test set. 17 of the errors can be characterized as over-tokenization while the other 13 are undertokenization. 13 of the over- tokenization cases are different tokens of the word *blywn* (Eng. billion) as the initial *b* in the words was treated as a preposition while it is an original part of the word.

A closer examination of the errors in the tokenization process reveals that most of the words which are incorrectly tokenized do not occur in the training set, or occur there only in the form produced by the tokenizer. For example, the word *blywn* does not occur in the training set, but the form *b+lywn+p* occurs in the training set, and this is the reason the word is tokenized erroneously. Another example is the word *bAsm*, which is ambiguous between a one-token word *bAsm* (Eng. smiling), and a two-token word, *b+Asm* (Eng. in the name of). Although the word should be tokenized as *b+Asm*, the word occurs in the training set as *bAsm*, which is a personal name.

In fact, only five words in the 30 mis-tokenized words are available in the training set, which means that the tokenizer has a very high accuracy on known words. There are yet two examples that are worthy of discussion. The first one involves suboptimal orthography. The word *r>smAl* (Eng. capital in the financial sense) is in the training set but is nonetheless incorrectly tokenized in our experiments because it is written as *brAsmAl* (with the preposition *b*) but with an *alif* instead of the *hamza*. The word was thus not tokenized correctly. The other example involves an error in the tokenization in the Prague Arabic Dependency Treebank. The word *>wjh* (Eng. I give/address) has been tokenized in the Prague Arabic dependency treebank as >wj+h (Eng. its utmost/prime), which is not the correct tokenization in this context as the *h* is part of the word and is not a different token. The classifier did nonetheless tokenize it correctly but it was counted as wrong in the evaluation since it does not agree with the PADT gold standard.

4.2.Stemming

Since stemming involves removing all the inflectional prefixes and suffixes from the words, and since inflectional affixes are not demarcated in the PADT data set used in the CoNLL shared tasks, there is no way to know the exact accuracy of the stemming process in that specific experiment, but since stemming is a by-product of segmentation, and since segmentation in general reaches an accuracy in excess of 98%, stemming should be trusted as an accurate process.

4.3.Part of speech tagging

The performance of the tagger on gold standard data with gold standard tokenization is shown in table 2. The experiment yields an accuracy of 96.39% on all tokens. Known tokens reach an accuracy of 97.49% while unknown tokens reach an accuracy of 81.48%. These numbers constitute the ceiling for accuracy since the real-world experiment makes use of automatic tokenization, which definitely leads to lower numbers.

Unknown	**Known**	**Total**
81.48%	97.49%	96.39%

Table 2: Part of speech tagging on gold standard tokenization

When we run the experiment using automatic tokenization we obtain an accuracy of 95.70% which is less than 1% lower than the gold standard accuracy. This indicates that part of speech tagging has been affected by tokenization quality. The drop in quality in part of speech tagging is almost identical to the drop in quality in tokenization.

While some of the errors made by the part of speech tagger are due to the fact that nouns, adjectives, and proper nouns cannot be distinguished by any formal features, a large number of the nominal class annotation in the gold standard data can hardly be justified. For example, the expression الاتحاد الأوروبي (Eng. the European Union) is annotated once in the training data as proper noun and adjective, and another time as a noun and adjective. A similar confusion holds for the names of the months and the weekdays, which are sometimes tagged as nouns and sometimes as proper nouns.

4.4. Dependency parsing

Now that we have stems, tokens, and part of speech tags, we can proceed with the parsing experiment, the final step and the ultimate goal of the preprocessing modules we have introduced so far. In order to prepare the training data, we have replaced the lemmas in the training and testing sets with the stems since we do not have access to lemmas in real-world experiments. While this introduces an automatic element in the training set, it guarantees the similarity between the features in the training set and those in the test set.

In order to discover whether the fine-grained POS tagset is necessary, we have run two parsing experiments using gold standard parts of speech with stems instead of lemmas, but without any of the linguistic features included in the gold standard: the first experiment has the two distinct part of speech tags and the other one has only the coarse-grained part of speech tags. Table 3 outlines the results.

	LAS	**UAS**	**LA**
CPOS+POS	72.54%	82.92%	84.04%
CPOS	73.11%	83.31%	84.39%
CoNLL2007	74.75%	84.21%	84.21%

Table 3: effect of fine-grained POS

As can be seen from table 3, using two part of speech tagsets harms the performance of the dependency parser. While the one-tag dependency parser obtains a Labeled Accuracy Score of 73.11%, the number goes down to 72.54% when we used the fine-grained part of speech set. In Unlabeled Attachment Score, the one tag parser achieves an accuracy of 83.31% compared to 82.92% on two tag parser. The same is also true for Label Accuracy Score as the numbers go down from 84.39% when using only one tagset compared to 84.04% when using two tagsets. This means that the fine-grained tagset is not needed to perform real world parsing. We have thus decided to use the coarse-grained tagset in the two positions of the part of speech tags. We can also see that this setting produces results that are 1.64% lower than those of the Single MaltParser results reported in the CoNLL 2007 shared task in terms of Labeled Accuracy Score. The difference can be attributed to the lack of linguistic features, vocalization, and the use of stems instead of lemmas. The LAS of 73.11% now constitutes the upper bound for real world experiments where also parts of speech and tokens have to be obtained automatically (since vocalization has been removed, linguistic features have been removed, and lemmas have been replaced with automatic stems). It should be noted that our experiments, with the complete set of gold standard features, achieve higher results than those reported in the CoNLL 2007 shared task: a LAS of 74.77 (here) versus a LAS of 74.75 (CoNLL, 2007). This may be attributed to the change of the parser since we use the 1.3.1 version whereas the parser used in the 2007 shared task was the 0.4 version.

Using the settings above, we have run an experiment to parse the test set, which is now automatic in terms of tokenization, lemmatization, and part of speech tags, and in the absence of the linguistic features that enrich the gold standard training and test sets. Table 4 presents the results of this experiment.

	Automatic	**Gold Standard**
LAS	63.10%	73.11%
UAS	72.19%	83.31%
LA	82.61%	84.39%

Table 4: Automatic dependency parsing experiment

The LAS drops more than 10 percentage points from 73.11 to 63.10. This considerable drop in accuracy is expected since there is a mismatch in the tokenization which leads to mismatch in the sentences. The 30 errors in tokenization affect 21 sentences out of a total of 129 in the test set. When we evaluate the dependency parsing output on the correctly tokenized sentences only, we obtain much better results (shown in Table 5). Labeled Attachment Score on correctly tokenized sentences is 71.56%, Unlabeled Attachment Score 81.91%, and Label Accuracy Score is 83.22%. This indicates that no good quality parsing can be obtained if there are problems in the tokenization. A drop of a half percent in the quality of tokenization causes a drop of ten percentage points in the quality of parsing, whereas automatic POS tags and stemming, and the lack of linguistic features do not cause the same negative effect.

	Correctly-tokenized Sentences	**Incorrectly-Tokenized Sentences**
LAS	71.56%	33.60%
UAS	81.91%	38.32%
LA	83.22%	80.49%

Table 5: Dependency parsing Evaluation on Correctly vs. Incorrectly Tokenized Sentences

While correctly tokenized sentences yield results that are not extremely different from those using gold standard information, and the drop in accuracy in them can be attributed to the differences introduced through stemming and automatic parts of speech as well as the absence of the linguistic features, incorrectly tokenized sentences show a completely different picture as the Labeled Attachment Score now plummets to 33.6%, which is 37.96 percentage points below that on correctly tokenized sentences. The Unlabeled Attachment Score also drops from 81.91% in correctly tokenized sentences to 38.32% on incorrectly tokenized sentences with a difference of 43.59 percentage points.

Error Analysis

Considering the total number of errors, out of the 5124 tokens in the test set, there are 1425 head errors (28%), and 891 dependency errors (17%). In addition, there are 8% of the tokens in which both the dependency and the head are incorrectly assigned by the parser. The POS tag with the largest percentage of head errors is the Adverb (D) with an error rate of 57%, followed by Preposition (P) at 34%, and Conjunctions at 34%. The preposition and conjunction errors are common among all experiments: those with gold standard and those with automatic information. These results also show that assigning the correct head is more difficult than assigning the correct dependency. This is reasonable since some tokens will have specific dependency types. Also, while there are a limited number of dependency relations, the number of potential heads is much larger.

If we look at the lexicon and examine the tokens in which most errors occur, we can see one conjunction and five prepositions. The conjunction *w* (Eng. and) tops the list, followed by the preposition *l* (Eng. for, to), followed by the preposition *fy* (Eng. in), then the preposition *b* (Eng. with), then the preposition *ElY* (Eng. on), and finally the preposition *mn* (Eng. from, of). We conclude this section by examining a very short sentence in which we can see the effect of tokenization on dependency parsing. Table 6 is a sentence that has an instance of incorrect tokenization.

Arabic	المساعدات الأمريكية الاستثنائية لمصر بليون دولار حتى مارس
English	The American exceptional aid to Egypt is a billion dollars until March.
Buckwalter (Gold Standard Tokenization)	AlmsAEdAt Al>mrykyp AlAstvnA}yp l mSr **blywn** dwlAr HtY |*Ar
Buckwalter (Automatic Tokenization)	AlmsAEdAt Al>mrykyp AlAstvnA}yp l mSr **b lywn** dwlAr HtY |*Ar

Table 6: A sentence showing the effect of tokenization

The sentence has 8 words one of which comprises two tokens. The word *lmSr* comprises a preposition *l*, and the proper noun *mSr* (Eng. Egypt). The tokenizer succeeds in splitting the word into two tokens, but it fails on the one-token word *blywn* (Eng. billion) and splits it into two tokens *b* and *lywn*. The word is ambiguous between *blywn* (Eng. one billion) and *b+lywn* (Eng. in the city of Lyon), and since the second solution is much more frequent in the training set, it is the one incorrectly selected by the tokenizer.

This tokenization decision leads to an ill-alignment between the gold standard sentence and the automatic one as the gold standard has 8 tokens while the automatically produced one has 9. This thus affects the POS tagging decisions as *blywn*, which in the gold standard is a NOUN, has been now tagged as b/PREPOSITION and lywn/PROPER_NOUN. This has also affected the assignment of heads and dependency relations. While *blywn* is a predicate dependent on the root of the sentence, it has been annotated as two tokens: *b* is a preposition dependent on the subject, and *lywn* is an attribute dependent on *b*.

Using the Penn Tags

So far, we have used only the POS tags of the PADT, and have not discussed the possibility of using the Penn Arabic Treebank. The difference is that the PADT tags are basic while the ATB ones have detailed representations of inflections. While the word *AlmtHdp* is given the tag ADJ in the PADT, it is tagged as DET+ADJ+FEMININE_SINGULAR_MARKER in the ATB. Table 7 shows the effect of using the Penn tagset with the gold standard full-featured dataset in three different experiments as compared with the PADT tagset:

(1) The original Unvocalized Experiment with the full set of features and gold standard components. The Penn tagset is not used in this experiment, and it is provided for reference purposes only.
(2) Unvocalized experiment with Penn tags as CPOS tags. In this experiment, the Penn tagset is used instead of the coarse grained POS tagset, while the fine-grained pos tagset remains unchanged.
(3) Using Penn tags as fine grained POS tags, while the CPOS tags remain unchanged.
(4) Using the Penn POS tags in both positions.

In the four experiments, the only features that change are the POS and CPOS features.

Experiment	LAS	UAS
Unvocalized Original	74.16%	83.53 %
Using Penn Tags as CPOS tags	74.12%	83.43 %
Using Penn tags as POS	72.40%	81.79 %
Using Penn tags in both positions	69.63%	79.33 %

Table 7: Using the ATB tagset with the PADT dataset

As can be seen from Table 7, in all three cases the Penn tagset produces lower results than the PADT tagset. The reason for this may be that the tagset is automatic in both cases, and the perfect accuracy of the PADT tags helps the classifier embedded in the MaltParser parser to choose the correct label and head. The results also show that when we use the Penn tagset as the CPOS tagset, the results are almost no different from the gold standard PADT tagset (74.12% vs. 74.16%). The fact that the Penn tagset does not harm the results encourages the inclusion of the Penn tags as CPOS tags in the automatic experiments that have been used throughout this chapter. The worst results are those obtained by using the Penn tags in both positions (POS and CPOS).

Using the Penn tagset with the reduced experiments, those without the linguistic features, gives a different picture from that in the full standard experiments, as detailed in table 8.

Experiment	LAS	UAS
Reduced with both PADT tags	72.54%	82.92 %
Reduced with Penn tags as CPOS	73.09%	83.16 %
Reduced with Penn tags as CPOS and automatic tokenization	63.11%	72.38 %

Table 8: Including the Penn full tagset in the reduced experiments

While the Penn tagset does not help improve parsing accuracy with the full-featured parsing experiments, it helps with the reduced experiments. While the experiment without the Penn tags score an LAS of 72.54%, replacing the CPOS tags in this experiment with the Penn tagset raises the accuracy to 73.09%, with an increase of 0.55%. This may be due to the fact that the full tagset gives more information that helps the parser. The increase is not as noticeable in the automatic tokenization experiment where the accuracy minimally changes from 63.10% to 63.11%.

Effect of Vocalization

We have stated in the methodology section that we use unvocalized data since naturally occurring Arabic is hardly vocalized. While this is a reasonable approach, it is worth checking the effect of vocalization on dependency parsing. Table 9 presents the results of vocalization effect in three experiments: (a) All the gold standard features with vocalization. This is the experiment reported in the literature on Arabic dependency parsing in CoNLL (2007), (b) All the gold standard features without the vocalization, (c) All gold standard features except for vocalization which is automatic, and (d) the automatic experiment with automatic vocalization. The vocalizer in the latter 2 experiments is trained on the PADT. The TIMBL memory-based learner is used in the experiment. The best results are obtained with the IB1 algorithm with similarity computed as weighted overlap,. Relevance weights are computed with gain ratio, and the number of *k* nearest neighbors is set to 1. The vocalizer has an accuracy of 93.8% on the PADT test set.

Experiment	**LAS**	**UAS**
Fully Gold Standard Vocalized	74.77%	84.09 %
Fully Gold Standard Unvocalized	74.16%	83.53 %
Full-featured with automatic vocalization	74.43%	83.88 %
Completely automatic (with automatic vocalization)	63.11%	72.19 %
Completely automatic without vocalization	63.11%	72.38 %

Table 9: Vocalization Effect on Dependency Parsing

As can be seen from Table 9, gold standard vocalization with gold standard features produces the best results (LAS: 74.77%) followed by the same settings, but with automatic vocalization with a LAS of 74.43%, then unvocalized gold standard with a LAS of 74.16%. The fact that even automatic vocalization produces better results than unvocalized text given the same conditions, in spite of a token error rate of 6.2%, may be attributed to the ability of vocalization to disambiguate text even when it is not perfect. We can also notice that the LAS for the Automatic experiment is the same whether or not vocalization is used. This indicates that vocalization, in spite of its imperfections, does not harm performance, although it also does not help the parser. Tokenization sets a ceiling for parsing accuracy.

5. Conclusion

We have presented an experiment in real world dependency parsing of Arabic using the same data, algorithm and settings used in the CoNLL (2007) shared task on dependency parsing. The real world experiment included performing tokenization, stemming, and part of speech tagging of the data before it was passed to MaltParser.

Tokenization was performed using the memory-based segmenter/tokenizer/stemmer and it reached an accuracy of 99.34% on the CoNLL 2007 test set. We performed stemming rather than lemmatization due to the many problems and difficulties involved in obtaining the lemmas.

Part of speech tagging scored 96.39% on all tokens on gold standard tokenization, but the accuracy dropped to 95.70% on automatic tokens. We also found that using the coarse grained POS tagset alone yielded better results than using it in combination with the fine-grained POS tagset.

The tokens, stems, and CPOS tags were then fed into the dependency parser, but the linguistic features were not since it was not feasible to obtain these automatically. The parser yielded a Labeled Accuracy Score of 63.10%, more than 10% below the accuracy obtained on when all the components are gold standard. The main reason behind the accuracy drop is the tokenization module, since tokenization is responsible for creating the nodes that carry syntactic functions. Since this process was not perfect, many nodes were wrong, and the right heads were missing. When we evaluated the parser on correctly tokenized sentences, we obtained a Labeled Accuracy Score of 71.56%. On incorrectly tokenized sentences, however, the LAS score drops to 33.60%.

We have also found that the full tagset of the Penn Arabic Treebank improves parsing results minimally in the automatic experiments, but not in the gold standard experiments.

Vocalization does not help in the real world experiment unlike in the gold standard one.

These results show that tokenization is the major hindrance to obtaining high quality parsing in Arabic. Arabic computational linguistics should thus focus on ways to perfect tokenization, or try to find ways to parsing without having to perform tokenization.

Acknowledgment

We would like to thank Joakim Nivre for his helpful answers to our questions concerning MaltParser.

References

Buchholz, Sabine and Marsi, Erwin (2006). *CoNLL-X shared task on multilingual dependency parsing*. In Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL), pages 149–164.

Daelemans, Walter; Zavrel, Jakub; van der Sloot, Ko and van den Bosch, Antal (2007). *TiMBL: Tilburg memory based learner – version 6.1 – reference guide*. Technical Report ILK 07-09, Induction of Linguistic Knowledge, Computational Linguistics, Tilburg University.

Daelemans, Walter,; Zavrel, Jakub; an den Bosch, Antal, and van der Sloot, Ko (2007). MBT: Memory-Based Tagger- Version 3.1. Reference Guide. Technical Report ILK 07-09, Induction of Linguistic Knowledge, Computational Linguistics, Tilburg University.

Hajič, Jan; Smrž, Otakar; Zemánek, Petr; Šnaidauf, Jan, and Beška, Emanuel (2004). *Prague Arabic Dependency Treebank: Development in Data and Tools*. In *Proceedings of the EMLAR International Conference on Arabic Language Resources and Tools*, pages 110-117, Cairo, Egypt, September 2004.

Hall, Johan; Nilsson, Jens; Nivre, Joakim; Eryigit, Gülsen; Megyesi, Beáta; Nilsson, Mattias and Saers, Markus (2007). Single Malt or Blended? A Study in Multilingual Parser Optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, 933-939.

Maamouri, Mohamed and Bies, Ann (2004) *Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools.* In Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, COLING 2004, Geneva, August 28, 2004.

Marton, Yuval; Habash, Nizar; and Rambow, Owen (2010). Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features. In *Proceddings of The FirstWorkshop on Statistical Parsing of Morphologically Rich Languages* (SPMRL 2010), LA, California.

McDonald, Ryan; Lerman, Kevin and Pereira, Fernando (2006). *Multilingual dependency analysis with a two-stage discriminative parser*. CoNLLX shared task on multilingual dependency parsing. In Proceedings of the 10th Conference on Computational Natural Language Learning

Nivre, Joakim; Hall, Jonathan; Nilsson, Jens; Eryigit, Gülsen and Marinov, Svetsolav (2006). Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*

Nivre, Joakim; Hall, Johan; Kübler, Sandra; McDonald, Ryan; Nilsson, Jens; Riedel, Sebastian, and Yuret, Deniz. (2007). *The CoNLL 2007 shared task on dependency parsing*. In Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007, pages 915–932

Owen, Jonathan. *The Arabic Grammatical Tradition*. In Hetzron, Robert (ed.) (1997). *The Semitic Languages*. Routledge, London.