Automatic Morphological Rule Induction for Arabic

Ahmad Hany Hossny
Faculty of Computers and Information
Cairo University
ahmad.hossny@gmail.com

Khaled Shaalan
Faculty of Informatics
The British University in Dubai, PO Box 502216, Dubai, UAE
khaled.shaalan@buid.ac.ae

Aly Fahmy
Faculty of Computers and Information
Cairo University
A.fahmy@fci-cu.edu.eg

Abstract

In this paper, we introduce an algorithm for morphological rule induction using meta-rules for Arabic morphology based on inductive logic programming. The processing resources are a set of example pairs (stem and inflected form) with their feature vectors, either positive or negative, and the linguistic background knowledge from the Arabic morphological analysis domain. Each example pair has two words to be analyzed vocally into consonants and vowels. The algorithm applies two levels of mapping: between the vocal representation of the two words (stem, morphed) and between their feature vector. It differentiates between both mappings in order to accurately deduce which changes in the word structure led to which changes in its features. The paper also addresses the irregularity, productivity and model consistency issues. We have developed an Arabic morphological rule induction system (AMRIS). Successful evaluation has been performed and showed that the system performance results achieved were satisfactory.

1. Introduction

Many researchers have attacked the morphological rule induction problem in a variety of languages but only a few limited researches have focused on Arabic language. This is due to the characteristics and peculiarities of Arabic language, lack of resources, and the limited amount of progress made in Arabic natural language processing in general.

The following example gives a general idea about how rules are mapped by our morphological rule induction system. Consider the weak verb "وقى" (to-protect) which has the vocal representation "VCV". In present tense the word has the form "يقى" which is produced by removing the radical character "و" and adding the present tense prefix character "ي". The mapping rule for these two words is as follows:

V2C1 ي (tense:present) → V2C1 و (tense:past)

Applying this rule to the past tense weak verb "وعى"(to-be conscious, past tense), which has the vocal representation (VCو), yields the present tense form "يعى". The rule induction algorithm applies two levels of mapping: between the vocal representation of the two words (stem, morphed) and between their feature vector. It differentiates between both mappings in order to accurately deduce which changes in the word structure led to which changes in its features.

This paper advances the state of the art in the Arabic morphological rule induction by: 1) utilizing the vocal properties in the analysis of Arabic words (Beesley, 1996) for handling irregularities, 2) adopting morphological rule induction techniques in order to handle productivity and consistency issues, and 3) adding some semantic features to the feature vector of Arabic words in order to be convenient for a wide spectrum of natural language processing systems such as machine translation and search engines.

The rest of this paper is organized as follows. Section 2, presents different linguistic acquisition techniques. In Section 3, we introduce the proposed example-based Arabic morphological rules acquisition. Experimental results are discussed in Section 4. In Section 5, concluding remarks and directions for future work are derived.

2. Related Work

Grammar acquisition or grammar induction is usually done at the syntactic level for most Latin-based languages as they usually do not have great deal of irregularities (Daille, 2000). On the other hand Semitic languages such as Arabic and Hebrew need a significant handling at the morphological level and sometimes at the morphosyntactic level. In the following, we briefly present the major induction techniques.

2.1 State Space Generation

It is based on trial and error where all possible grammar rules are generated using all possible combination of features (Miller & Fox, 1994). This gives a full coverage of all rules but gives also a massive amount of ambiguities which reduces the accuracy of the parsing process and its efficiency as well.

2.2 Slots and Filler Technique

It is based on template matching of the sentence structure highly needed in lots of NLP applications such as machine translation (Edelman, Solan, Horn, Ruppin, Rich, 2003).

2.3 Stochastic Techniques

2.3.1 Genetic Programming (Pappa & Freitas, 2004) utilizes prior knowledge of rule induction domain to build asset of functions and terminals used to evolve a rule and then compute fitness value of candidate rule induction algorithms of genetic programming population.

2.3.2 Structural Zeros technique uses statistical or machine learning techniques to cover also any an unobserved combination that may arise due to sparse data or hard syntactic constraints (Mohri & Roark, 2006).

2.4 Case-Based Reasoning

It employs relevance weighting to access similarities between cases, making use of rule induction results to assign weights to each attribute-value pair of the query case (Cercone, An, Chan, 1999). Cases in the case base can then be ranked according to their probability of relevance to the new case.

2.5 Neural Networks

It treats the rule induction process as a classification problem aims to classify the sample to some rules, so it propose an activation function that simulates the behavior of logic induction, such neural system is to be trained using input data and output classification of the data to build the rule on the neurons basis which is kind of supervised learning for when it gets the relevant weights as basis for neurons it constructs the rules (Silva & Ludermir, 1999).

Most of the mentioned techniques need lots of data samples to build the language model especially in the statistical and stochastic domains, which is not the case in the problem we handle where we need to build logical language model with few examples in a decreasing growing rate.

3. Example-Based Arabic Morphological Rules Acquisition

In this section an automated morphological acquisition algorithm is presented along with its influence on the computational morphology module, it applies the inductive logic programming (Muggleton, 1999) concepts by dependency on example pairs and logic behind the acquisition process which is similar to automatic word guessing (Mikheev, 1997) that is used in French rule induction (Daille, 2000). In our discussion, we start with giving some related definitions of different sets, ordered pairs and operators. This is followed by the specification of the proposed algorithm of the automated rule generation. Finally, the parsing algorithm is presented.

The proposed learning technique depends on a set of example pairs and their corresponding feature vectors that are used for the training process.

Definition 3.1: Let $\Sigma = \{ا, \dots, ي\}$ be the set of Arabic literals. Then a set Σ^+ is the set of Arabic words of at least one literal.

Definition 3.2: Let Σ^+ be the set of Arabic words of at least one literal. Then the binary operator $+:\Sigma^+ \times \Sigma^+ \to \Sigma^+$ is called the suffix and prefix concatenation operator.

Definition 3.3: Let Σ^+ be the set of Arabic words of at least one literal. Then the binary operator $*: \Sigma^+ \times \Sigma^+ \to \Sigma^+$ is called the infix concatenation operator.

Definition 3.4: Let Σ^+ be the set of Arabic words of at least one literal. Then the binary operator $\div: \Sigma^+ \times \Sigma^+ \to \Sigma^{+4}$ is called the affix separation operator. It compares two Arabic words and maps the differences into a prefix, a suffix, an infix, and a stem word. The set Σ^{+4} is then called the set of uninflected words.

Definition 3.5: Let $\Sigma_V = \{\mathrm{c}, \mathrm{v}\}$ be the set of vocal literals, where v denotes to vowel literal and c is the notation for consonant literals. Then the set Σ_V^+ is the set of vocal words of at least one literal.

Definition 3.6: Let Σ_V^+ be the set of vocal words and Σ^+ be the set of Arabic words of at least on literal. Then a unary function $\Upsilon: \Sigma^+ \to \Sigma_V^+$ be a mapping function from Arabic words to vocal words.

Definition 3.7: Let $F = \{f_i | 1 \le i \le n\}$ be the feature set of an Arabic word. Let V_i be the value set of the word feature f_i. Then the ordered pair $(f_i, v) \in \{f_i\} \times V_i$ is called a feature-value pair and the set $P_{f_i}^{V_i}\Big|_{i=1,\dots,n}$ is the set of feature-value pairs.

Definition 3.8: Let $P_{f_i}^{V_i}\Big|_{i=1,\dots,n}$ be the set of feature-value pairs. An ordered tuple

$$\left(p_{f_1:v_1}, \dots, p_{f_n:v_n}\right) \in \prod_{i\in\{1,\dots,n\}} P_{f_i}^{V_i}$$

is called a featured vector.

Definition 3.9: Let $P_{f_i}^{V_i}\Big|_{i=1,\dots,n}$ be the set of feature-value pairs. Let Σ^{+4} be the set of uninflected words. A unary function $\psi: \Sigma^{+4} \to \prod_{i\in\{1,\dots,n\}} P_{f_i}^{V_i}$ is called a featured vector mapping function.

A featured word is composed of sequence of vocal characters and a feature vector. It works both ways. It is a data structure while generating the rules and a primary key in the parsing process.

Definition 3.10: Let $\prod_{i\in\{1,\dots,n\}} P_{f_i}^{V_i}$ be the set of feature vectors. Let Σ_V^+ be the set of vocal words. Then a set

$$W^F = \Sigma_V^+ \times \prod_{i\in\{1,\dots,n\}} P_{f_i}^{V_i}$$

is called the set of featured words and an ordered pair $w^F \in W^F$ is called a featured word.

The distance will be measured by considering the sequence of vocal word. There are three cases when measuring a distance depending on the examined vocals whether they are similar vocal words, semi different vocal words or totally different vocal words. In similar vocal words, both words being examined have the same sequence of consonants and vocals.

Consider the word 'يلعبون' (they-are-playing [masculine]) having the vocal representation 'ونC2C1C0ي'. It vocally matches the word 'يشربون' (they-are-drinking [masculine]) and hence has the same vocal representation. Semi-different vocal words are having different vocal representation. They only differ in consonants.

Algorithm: derive_rule
Inputs: stem x and morphed word $\mathbf{x}$
Ouputs: R

1. Extract the stem x from the morphed word $\mathbf{x}$ to get the uninflected word $\tilde{x}$ and the affixes that form the prefix(es) $\mathbf{x}^p$, infix(es) $\mathbf{x}^i$ and suffix(es) $\mathbf{x}^s$

$$(\tilde{x}, \mathbf{x}^p, \mathbf{x}^i, \mathbf{x}^s) \leftarrow \mathbf{x} \div x$$

2. Derive the feature vectors for both $\mathbf{x}$ and $\tilde{x}$

$$(\mathbf{x}_{f_1:v_1}, \dots, \mathbf{x}_{f_n:v_n}) \leftarrow \psi(\mathbf{x})$$
$$(\tilde{x}_{f_1:v_1}, \dots, \tilde{x}_{f_n:v_n}) \leftarrow \psi(\tilde{x})$$

3. Differentiate and compare the feature vectors of both of $\psi(\mathbf{x})$ and $\psi(\tilde{x})$ word via unification;

$$(\mathbf{x}_{f_1:v_1}, \dots, \mathbf{x}_{f_n:v_n}) \doteq (\tilde{x}_{f_1:v_1}, \dots, \tilde{x}_{f_n:v_n})$$

If $\mathbf{x}_{f_i:v_i} = \_$ and $\tilde{x}_{f_i:v_i} = v_i$ then
$r: r^L_{f_i:\_} \rightarrow \mathbf{x}^p + r^R_{f_i:v_i} + \mathbf{x}^s$
If $\mathbf{x}_{f_i:v_i} = \mathrm{v_i}$ and $\tilde{x}_{f_i:v_i} = \_$ then
$r: r^L_{f_i:v_i} \rightarrow \mathbf{x}^p + r^R_{f_i:\_} + \mathbf{x}^s$
If $\mathbf{x}_{f_i:v_i} = \mathrm{v_i}$ and $\tilde{x}_{f_i:v_i} = \mathrm{v_i}$ then
$r: r^L_{f_i:V_i} \rightarrow \mathbf{x}^p + r^R_{f_i:V_i} + \mathbf{x}^s$
If $\mathbf{x}_{f_i:v_i} = \mathrm{v_i}$ and $\tilde{x}_{f_i:v_i} = \tilde{\mathrm{v}}_\mathrm{i}$ then
$r: r^L_{f_i:v_i} \rightarrow \mathbf{x}^p + r^R_{f_i:\tilde{\mathrm{v}}_\mathrm{i}} + \mathbf{x}^s$

4. Generate vocal representations of $\mathbf{x}$.

$$\mathbf{x}^\Upsilon \leftarrow \mathbf{x}^p + \Upsilon(\mathbf{x}) + \mathbf{x}^s$$

5. Generate the j^{th} rule $\mathrm{r_j}$

$$R: (\mathbf{x}^\Upsilon, r^L_{f_i:V_i}) \rightarrow \mathbf{x}^p + r^R_{f_i:\tilde{V}_\mathrm{i}} + \mathbf{x}^s$$

where $1 \le i \le n$, and n is the number of features.

6. Return R.

Figure 1: Pseudo code Algorithm for learning Arabic morphological rules from examples

Consider the word 'يشددون' (they-are-stressing-on [masculine]) with a vocal representation 'ونC2C1C0ي'. It has one more consonant than 'يشدون' (they-are-pulling [masculine]) with a vocal representation 'ونC1C0ي'. The last cases are totally different vocal words where vowels are misplaced or differ in number such as 'ينوون' (they-intending [masculine]) which has the vocal representation 'ونV1C0ي' which having past 'نوى' which has the vocal representation 'ىV1C0' and 'يهوون' (they-interested in [masculine]) which has the vocal representation 'ونV1C0ي' which having past 'هوى' which has the vocal representation 'ىV1C0'.

The rule generation algorithm listed below in Figure 1 describes the steps followed to get the literal and feature vectors mapping between a stem and its inflected form. Throughout this algorithm, feature vectors are represented as lists. The algorithm uses a unification-based assignment. The underscore symbol '_' represents a 'do not care' case.

The algorithm takes as input a stem $\tilde{x} \in \Sigma^+$ and an inflected word form $\mathbf{x} \in \Sigma^+$. Then, it applies the n-gram algorithm for computing the dissimilarity between the two input words to derive the breakdowns of the inflected word into affixes and a stem $\tilde{x} \in \Sigma^+$. Then it compares the stem $\tilde{x}$ with the inflected word $\mathbf{x}$ to derive the literal mapping rules. The feature vectors of both the input word $(\mathbf{x}_{f_1:v_1}, \dots, \mathbf{x}_{f_n:v_n})$ and the stem $(\tilde{x}_{f_1:v_1}, \dots, \tilde{x}_{f_n:v_n})$ are also compared to derive the features mapping rules. Finally, a rule $\mathrm{r_j}$ that represents both the literal and features mapping is generated.

Every generated rule R is verified against an indexed rule base for redundancy, ambiguity and transitivity before it is accepted to be added to the rule base.

The following techniques are applied in order to maintain an efficient rule base:

- If the RHS part of a newly generated rule has included an expression that occur as a whole RHS of another existing in the rule base then replace this expression by the LHS of the existing rule. For example, give the following;

$$R_i: (\mathbf{x}_i^\Upsilon, r^L_{f_i:v_i}) \rightarrow \mathbf{x}^p + r^R_{f_i:\tilde{\mathrm{v}}_\mathrm{i}} + \mathbf{x}^s$$

$$R_j: (\mathbf{x}_j^\Upsilon, r^L_{f_i:v_i}) \rightarrow \mathbf{x}_j^p + (\mathbf{x}_i^p + r^R_{f_i:\tilde{\mathrm{v}}_\mathrm{i}} + \mathbf{x}_i^s) + \mathbf{x}_j^s$$

The new rule is modified to;

$$R_j: (\mathbf{x}_j^\Upsilon, r^L_{f_i:v_i}) \rightarrow \mathbf{x}_j^p + (\mathbf{x}_i^\Upsilon, r^L_{f_i:v_i}) + \mathbf{x}_j^s$$

- Merge or combine rules if they have common feature-value pairs that will construct a more generic rule with respect to their LHSs and RHSs. This will results in reducing the pattern matching time and the complexity of the generated rules.
  For example the following two grammar rules have different LHS but similar (f_i, v) pair at their RHS. The two rules can be resolved by replacing them with a that includes the value of this pair as unbound, e.g. $(f_i, \_)$.

$$R_i: \left(\mathbf{x}_i^Y, r_{f_i:v_i}^L\right) \rightarrow \mathbf{x}^p + r_{f_i:v_i}^R + \mathbf{x}^s$$

$$\tilde{R}_i: \left(x_i^Y, r_{f_i:\tilde{v}_i}^L\right) \rightarrow \mathbf{x}^p + r_{f_i:v_i}^{\tilde{R}} + \mathbf{x}^s$$

The combined rule is

$$R_i: \left(\mathbf{x}_i^Y, r_{f_i:\_}^L\right) \rightarrow \mathbf{x}^p + r_{f_i:v_i}^R + \mathbf{x}^s$$

Using the rule base for morphological analysis has become quite simple:

- Convert the inflected Arabic word into its vocal representation in an abstract form that consists of consonants and vowels.
- Index each letter in the input word its position.
- Search the rule base for the rule that better matches the vocal representation.
  - If no rule matches found for the vocal representation, the parser looks for the most similar rule using Levenshtein's weighted distance (1966) estimated by Wagner and Fisher algorithm (Wagner & Fisher, 1974).
- Retrieve this rule from the rule base.
- Fire (execute) the rule to build a new word feature-value pairs according to the fired rule and produce the stem.
- Lookup the stem in the lexicon to return the stem and its features.

Example for the analysis process:

Let the tested inflected word be 'يهتدون' (they discover [masculine]).

Convert it to vocal characters so it becomes 'V0C1C2C3V4C5'

Search for the vocal string in the LHS of the generated grammar rules

It matches with the LHS = 'ونC3C2C1ي' , So the matching rule is:

ونC3C2C1ي
[type:verb..tense:present..sex:male..count:plural..person:Third]→ ون + ى C3 C2 C1 ا
[type:verb..tense:past..sex:male..count:singular..person:Third] + ي

by this rule we deduce that the root is defined by removing *yaa* 'ي' character from the first of the word and adding *alef* 'أ' character at the first of the word and *alef layena* 'ى' at the end of the word , so we can search for it in dictionary by word 'إهتدى'

The changed features of the analyzed word according to the rule are the tense from past to present , and the count from singular to plural , and the other features either having constant values or the same as the stem like the person *feature* which is third person for stem and morphed words.

4. Experimental Results

In this section we discuss the results of an experiment that shows the convergent rate of the rule generation algorithm as the sample size increases. The experiment simply runs the proposed leaning algorithm on a sample set of 2500 words separated into several subsets of 500 each, the data is gathered from a raw corpus captured from news site (www.alarabiya.net) and tagged as part of speech (POS) using Arabic tagset described in (Khoja, Garside, Knowles, 2001).

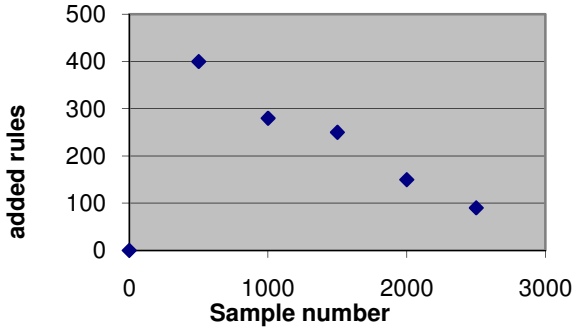

Figure 2: Number of generated rules against the sample size

The learning results of running algorithm that acquires Arabic morphological rules using a set of example pairs that forms a sample set of size 2500 word has produced 1160 morphological rules. They are depicted in Figure 2 using the results presented in Table 1. The results show that the changes in the number of rules as the data set increases.

No. Of Tokens	No. Of Rules
0 – 500	400
500 – 1000	280
1000 – 1500	250
1500 – 2000	140
2000 – 2500	90
Total	1160

Table 1: Number of generated rules against the change in the sample size

To check the effectiveness of the generated grammar, two sets of data are checked against induced rules. First set is the data used to train the system and is called positive data. Second set is raw data collected randomly from a news website.

Analyzing the results of the first set, 98% of the training words correctly matched the induced rules and 2% of them have wrong matching or no matching rule at all. Almost 75% of the training words matched one rule only and 25% matched more than one rule, which leads to ambiguity that should be resolved. Analyzing the results

of the second set, 80% of the raw morphed words correctly matched the induced rules and 20% of them have wrong matching or no matching rule at all. Almost 70% of the raw morphed words matched one rule only and 30% of them matched more than one rule, which leads to ambiguity.

5. Conclusions and Future Work

Arabic is morphologically rich language. Automated acquisition of Arabic morphology is a challenge task as Arabic has many peculiarities, which make the traditional manual acquisition problematic. This paper has discussed the development of a learning system for Arabic morphological rules acquisition from Arabic examples. We provided an algorithm which acquires the morphological rules in an efficient representation. We showed how morphological analysis and hence generation could be easily benefit from the repository of the rule base. We conducted an experiment on a sample of 2500 words which shows the learning curve of rules decreases as the number of words increases.

There are many advances to be considered in future to enhance the results of the learning algorithm. These advances include increasing the level of details for the speech vocal classification, integration with statistical based model in a hybrid model and ambiguity resolution.

References

Yona, S., Wintner, S. (2007). A finite-state morphological grammar of Hebrew, *Natural Language Engineering*, 14(2), pp. 173--190.

Mohri, M., Roark, B. (2006). Probabilistic Context-Free Grammar Induction Based on Structural Zeros, In *Proceedings of the Seventh Meeting of the Human Language Technology conference*.

Edelman, S., Solan, Z., Horn, D., Ruppin, E. Rich (2003). Syntax from a raw Corpus: Unsupervised Does it, Presented *at NIPS workshop on Syntax, Semantics and Statistics*.

Pappa, G. L., Freitas, A. (2004). Towards a genetic programming algorithm for automatically evolving rule induction algorithms. In *Proceedings of the Workshop W8 on Advances in Inductive Learning*, pp. 93--108,

Khoja, S., Garside, R., Knowles, G. (2001). A tagset for the morphosyntactic tagging of Arabic, In *Corpus Linguistics Conference*.

Daille, B. (2000). Morphological Rule Induction for Terminology Acquisition, In *Proceedings of the 18th International Conference on Computational Linguistic*.

Cercone, N., An, A., Chan, C. W. (1999). Rule-Induction and Case-Based Reasoning: Hybrid Architectures Appear Advantageous, *IEEE Trans. Knowl. Data Eng.*, 11(1), pp. 166--174.

Silva, R. B., Ludermir, T. B. (1999). Neural Network Methods for Rule Induction, In *Proceedings of International Joint Conference on Neural Networks*.

Mikheev A. (1997). Automatic rule induction for unknown-word guessing, *Computational Linguistics*, 23(3), pp.405--423.

Beesley, K. R. (1996), Arabic Finite-State Morphological Analysis and Generation, In *Proceedings of the 16th Internationa Conference on Computational Linguistics*, pp. 5--9.

Miller, S., Fox, H. J. (1994). Automatic Grammar Acquisition, In *Proceedings of Human Language Technology Conference*.

Uszkoreit, H. (1986). Categorial Unification Grammars, In *Proceedings of International Conference on Computational Linguistics*, pp. 187--194.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals, *Soviet Physics Doklady*, 10(8), pp. 707--710.

Wagner, R. A., Fisher, M. J. (1974). The String-to-String Correction Problem, *Journal of ACM*, 21(1), pp. 168--173.

Muggleton, S. (1999). Inductive Logic Programming: Issues, Results and the Challenge of Learning Language in Logic, *Artificial Intelligence*, 114(1-2), pp. 283--296.