Arabic Morphological Analysis Techniques: A Comprehensive Survey

Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi
Computer and Electronics Research Institute, King Abdul Aziz City for Science and Technology, P.O. Box 6086, Riyadh, Saudi Arabia 11442. E-mail: {Kharashi, Imad}@kacst.edu.sa

After several decades of heavy research activity on English stemmers, Arabic morphological analysis techniques have become a popular area of research. The Arabic language is one of the Semitic languages; it exhibits a very systematic but complex morphological structure based on root-pattern schemes. As a consequence, survey of such techniques proves to be more necessary. The aim of this paper is to summarize and organize the information available in the literature in an attempt to motivate researchers to look into these techniques and try to develop more advanced ones. This paper introduces, classifies, and surveys Arabic morphological analysis techniques. Furthermore, conclusions, open areas, and future directions are provided at the end.

Introduction

Researchers in many fields of computer science are interested in developing systems to improve the interaction between humans and computers. Natural language-processing systems represent one of the most important fields of investigation to serve this interest. Morphological analysis techniques form the basis of most natural language processing systems. Such techniques are very useful for many applications, such as information retrieval, text categorization, dictionary automation, text compression, data encryption, vowelization and spelling aids, automatic translation, and computer-aided instruction.

Due to their nonconcatenative nature, processing Semitic languages such as Arabic is not an easy task. For example, though Arabic words may be formed from concatenating morphemes, they are in fact normally formed using root-pattern schemes. Morphologically, the Arabic language is a most complicated and rich language. Tens or hundreds of words can be formed using one root, a few patterns, and a few affixes. Arabic also has a high degree of ambiguity for many reasons, such as the omission of vowels and the similarity of affixed letters to stem or root letters. Morphological analysis usually affects other higher levels of analysis such as syntactical and semantic analyses.

Studying and evaluating Arabic morphological analysis techniques is difficult for many reasons that will be clear by the end of this article. The article provides an introduction to Arabic morphological analysis techniques, including basic definitions, effectiveness measures, and a short review of English stemming algorithms. Throughout this paper, Romanization of Arabic letters and symbols is adopted from Buckwalter (2002). The main objective of this paper is to survey Arabic morphological analysis techniques, including two-level and finite-state morphology.

Basic Definitions

The literature on Arabic morphological systems provides no standard definitions of certain terms. In the literature, many terms have been misused or used interchangeably with other terms. Table 1 lists normalized translations of some Arabic terms frequently used by Arabic linguists (Al-Khuli, 1991; Bokhaddood, 1994; El-Dahdah, 1988; Metri & George, 1990). Following is a list of some basic terms used in this work and their definitions. Terms and their definitions are selected on the basis of the frequency of their use by researchers. Table 2 lists these terms, summarizes their definitions and functions, and provides some examples.

A *word* is the single and isolated lexeme that represents a certain meaning (Al-Hamalawee, 2000). Some researchers define a word orthographically as a unit that is surrounded by two spaces (Saliba & Al-Dannan, 1989). Invariable, solid or inert words, as opposed to normal words, are those words with a single morphological shape. Defective words, as opposed to intact words, contain defective or weak letters (hamza and gemination) (Al-Khuli, 1991; El-Dahdah, 1988; Metri & George, 1990). Words are classified by Arab linguists as nouns, verbs, or particles (Al-Saeedi, 1999).

A *morpheme* is the smallest element that has a meaning. Some morphemes exist as words at the same time. Morphemes cannot be split into smaller ones, and they should

Received October 7, 2002; revised June 9, 2003; accepted June 9, 2003

TABLE 1. Arabic linguistic terms and their translation.

Arabic Term	Translation	Example
الإبدال	Mutation	Mutation[word[صبر Sbr, افتعل AfvEl] = اصتبر ASvbr] = اصطبر ASTbr
الإعلال	Vocalization, defection	Vocalization derives [قم qm] from [قام qAm]
التنوين	Nunation	Adding [ً N] to [رجل rjl] to yield [رجلً rjlN]
جمع التكسير	Broken plural	Broken Plural [حجر Hjr, أفعال >fEAl] = أحجار >HjAr
إدغام – مماثلة صوتية	Assimilation, incorporation	Assimilation [مدد mdd] = مدّ md~
الشدّة	Gemination mark or tashdeed	عدّة Ed~p
الأسماء الجامدة أو السماعية	Inert / aplastic nouns / nonstandard / solid	قلم qlm, إنسان <nsAn
الأفعال الجامدة أو السماعية	Inert / aplastic verbs/ nonstandard / solid	ليس lys, نعم nEm
الجذور المعتلة	Defective roots, weak roots	أكل >kl, قول qwl, وصل wSl
الميزان الصرفي	Morphologic balance, pattern, surface level, or surface pattern	فاعل fAEl, فعلاء fElA', فال fAl
الصيغة الصرفية	Pattern, lexical level, or deep pattern	فعل fEl, فعول fEwl, فعيل fEyl
حروف الزيادة	Access or augmented letters	س أ ل ت م و ن ي هـ ا s > l v m w n y h A
علامات التشكيل	Short vowels	[ُ o, َ a, ُ u, ِ i, ّ ~, ٌ N, ٍ K, ً F]
حروف المد	Long vowels	[ا A, و w, ي y]
الحالة الإعرابية	Syntax case or grammatic case	رفع rfE, نصب nSb, جر jr, ...
التشكيل	Vowelization, diacritization	مُجَاهِدٌ mujaAohidN
الجذر	Root	[ح م د Hmd] from [المحمدون AlmHmdwn]
الجذع	Stem	[محمد mHmd] from [المحمدون AlmHmdwn]
الضمائر المتصلة	Connected pronouns	Stem [سلم slm] + Pronoun [وا wA] = سلموا slmwA
ظرف المكان	Noun of place	تحت vHv, فوق fwq
ظرف الزمان	Noun of time	ظهراً ZhrAN
إسم الآلة	Noun of instrument	مقص mqS
الماضي	Perfect	ذهب *hb
المظارع	Imperfect	يذهب y*hb
الأمر	Imperative	اذهب A*hb
المتكلم	First person	أذهب >*hb
المخاطب	Second person	تذهب v*hb
الغائب	Third person	يذهب y*hb
أدوات/حروف /كلمات التوقف والبناء	Particles / function or stop words	من mn, تحت vHv, و w, عليهما ElyhmA
الضمائر	Pronouns	ها- -hA, كم- -km
أدوات العطف	Conjunctions	و w, ف f, ثم tm, حتى HvY
القواعد الصرفية – الصوتية	Morphophonetic rules	الإدغام Al<dgAm

impart a function or a meaning to the word which they are part of (Spencer, 1991).

The *root* is a single morpheme that provides the basic meaning of a word (Spencer, 1991). Generally speaking, in English, the root is sometimes called the *word base* or stem; it is the part of the word that remains after the removal of affixes (Al-Khuli, 1991). In Arabic, however, the base or stem is different from the root (Al-Atram, 1990). In Arabic,

TABLE 2. Basic terms used in morphologic analysis techniques.

Term	Definition	Function	Examples
Word	The single and isolated lexeme or unit that is surrounded by two spaces	To represent certain meaning	Peacefully, organic. السلام AlslAm, المعتزون AlmEvzwn
Morpheme	The smallest element that has meaning	To impart a function or meaning to the word	dis-,-ment, agree ون -wn, أكل >kl
Root	A single morpheme	To provide the basic meaning of the word	agree, ح م د H m d
Affix	A single morpheme	To serve as prefix, suffix, or infix	Un-, الـ Al -ness, ـون -wn ـا- -A-
Stem	A single or concatenated morphemes	Refer to some central idea or meaning	بنت bnv use
Morphology	A branch of linguistics	Studies the word formation	
Conflation	Gathering process of some non-identical words	Used in many natural language processing applications	Group and collection, total and complete
Stemming	Word standardization	Reducing the word to its stem	Believable to believe الوالدان AlwAldAn, إلى to والد wAld
Stemming algorithm	A computational process that conflates words with the same root/stem to a common form	Reducing the word to its stem	Swimming to swim
Morphologic analysis technique	A computational process that analyzes words by considering their internal structure	Compression, Encryption, Full-text searching …	Word = الضاربون AlDArbwn Prefix = ال Al Suffix = ون wn Root = ض ر ب D r b Stem = ضارب DArb Pattern = فاعل fAEl
Basic Arabic pattern letters	The three Arabic letters ف f, ع E, ل l	Used in constructing the morphologic balance or pattern	فاعل fAEl, فعل faEila
Morphologic balance or pattern	A model or measure	To model the internal structure of Arabic words using the basic Arabic pattern letters	Word = رحم raHima Pattern = فعل faEila
Morphologic computational balances	Balances suggested and created by some researchers	To ease the processing of some Arabic words from a computational viewpoint	افدعل AfdEl
Access or augmented letters	A small set of Arabic letters that includes ا، س، أ، ل، ت، م، و، ن، ي، هـ s, >, l, v, m, w, n, y, h, A	Used to construct a pattern	
Vowelization or diacritization	The process of putting diacritical marks or short vowels above or under letters of Arabic words	To correctly analyze and pronounce Arabic words	تَجَمَّلَ vajam~ala
Kasheeda	The symbol "ـ"	Used to horizontally and orthographically stretch some Arabic characters	ســـــلام s_ _ _ _lAm
Stop word	A word that does not represent the document content	To build natural text	من mn, على ElY, لماذا lmA*A And, the, while
Linguistic test data set	A set of documents, words, and/or their decomposition	Used for testing the performance of automated systems and related algorithms that deal with natural language	
Test collection	A set of documents, queries, and associated relevance judgments	To test the performance of information retrieval systems	ADI, Cranfield, TREC
Finite State Automaton (FSA)	A mathematical model of a system with discrete inputs and outputs	Modeling and evaluating systems	
Two-level morphology	A system of word recognition that involves two-level finite state phonology or graphology	Natural language processing	

the root is the original form of the word before any transformation process, and it plays an important role in language studies (Metri & George, 1990). Defective or weak roots are the roots with one or more long vowels.

A *stem* is a morpheme or a set of concatenated morphemes that can accept an affix (Al-Khuli, 1991). The stem expresses some central idea or meaning (Paice, 1994).

An *affix* is a morpheme that can be added before or after, or inserted inside, a root or a stem as a *prefix*, *suffix* or *infix*, respectively, to form new words or meanings (Al-Khuli, 1991; Thalouth & Al-Dannan, 1987). Arabic prefixes are sets of letters and articles attached to the beginning of the lexical word and written as part of it, while suffixes are sets of letters, articles, and pronouns attached to the end of the

word and written as part of it (Al-Atram, 1990). English has 75 prefixes and about 250 suffixes (Salton, 1989). Arabic has fewer affixes. Arabic affixes have the feature of concatenating with each other in predefined linguistic rules. This feature increases the overall number of affixes (Ali, 1988). The removal of prefixes in English is usually harmful because it can reverse or otherwise alter the meaning or grammatical function of the word. This is not so in Arabic, since the removal of prefixes does not usually reverse the meaning of words.

Morphology is the branch of linguistics that deals with the internal structure of words. It studies word formation, including affixation behavior, roots, and pattern properties (Al-Khuli, 1991; Hull & Grefenstette, 1996; Krovetz, 1993). Morphology can be classified as either inflectional or derivational (Aref, 1997; Hull & Grefenstette, 1996; Krovetz, 1993; Spencer, 1991). Inflectional morphology is applied to a given stem with predictable formation. It does not affect the word's grammatical category, such as noun, verb, etc. Case, gender, number, tense, person, mood, and voice are some examples of characteristics that might be affected by inflection. Derivational morphology, on the other hand, concatenates to a given word a set of morphemes that may affect the syntactic category of the word. The distinction between these two classes is not an easy one to make, and it differs from one language to another.

Word morphology usually refers to the different forms of individual words. These forms express the type and function of the individual words. Word morphology is very helpful in the process of learning to use a dictionary efficiently and acquiring linguistic information. It also has an important role to play in the disambiguation of word sense.

A *morphological analysis technique* is a computational process that analyzes natural words by considering their internal structures. The internal structure of a word may include stem, root, affixes, and patterns. Morphological analysis techniques can be viewed as clustering mechanisms and usually help in resolving lexical ambiguity. Clustering is a very useful process in many natural language applications (Krovetz, 1993) such as information retrieval (El-Affendi, 1998), text classification, and text compression. In the case of Arabic, the main purpose of any morphological analysis technique is to obtain the root of a given word (El-Affendi, 1991). This is true of many applications, but not of all.

Conflation is the process of gathering together nonidentical words that share the same principal concept; that is, they are semantically equivalent. The process of conflating words with the same root may utilize tables or orthographical similarity, or both. In the latter case, stemming algorithms are the most commonly used conflation technique (Foxley & Feddag, 1990; Frakes & Baeza-Yates, 1992; Lennon, Pierece, Tarry, & Willett, 1981; Paice, 1990, 1994, 1996; Salton, 1989; Xu, & Croft, 1998).

Stemming is a method of word standardization used to match some morphologically related words. The *stemming algorithm* is a computational process that gathers all words that share the same stem and have some semantic relation (Paice, 1996). The main objective of the stemming process is to remove all possible affixes and thus reduce the word to its stem (Dawson, 1974; Paice, 1990, 1994; Lennon et al, 1981; Lovins, 1968).

Stemming, as a term, is widely used by researchers dealing with languages with simple morphological systems such as English and similar languages. Morphological analysis is a term widely used by researchers in languages with complex morphological systems such as Arabic and Hebrew.

Morphological balance or *pattern* is a model used to study the internal structure of Arabic words. It consists of the three basic Arabic pattern letters that corresponds to the first, second, and third letters of the Arabic triliteral root, respectively. For quadriliteral roots, the third letter is duplicated to represent the fourth root letter. Furthermore, zero or more augmented letters or one or more short vowels are inserted. Access or augmented letters are used to construct a pattern based on the basic triliteral or quadriliteral patterns. A pattern is defined by some researchers as morphological balance but with no letter deleted.

Morphological computational balances are those balances suggested by some researchers and created to ease the processing of some Arabic words from a computational viewpoint (Thalouth & Al-Dannan, 1987).

Vowelization or *diacritization* is the process of putting diacritical marks or short vowels (o, a, u, _, i, ~, N, K, F) above or under letters of Arabic words. **Nunation** is the process of putting one of the set of vowels (N, K, F) at the end of the word to produce a phonetic effect that adds the sound of the letter ([ن n]). *Gemination* or *tashdeed* is the process of putting the vowel (~) above a letter to duplicate it phonetically. *Syntax case*, presented at the end of the word, is the parsing of its grammatical state (Metri & George, 1990). **Kasheeda (-)** is the symbol used to horizontally and orthographically stretch some Arabic characters.

Mutation is the process in certain Arabic words whereby a certain letter is replaced by another, while *Vocalization* is the process of modification, in certain Arabic words, of one of the defective or weak letters (Al-Khuli, 1991; Metri & George, 1990). *Assimilation* is the process of replacing two Arabic letters with a single one and adding a gemination mark (Metri & George, 1990).

A **stop list** is a list of words that do not represent a document's contents and may include prepositions, pronouns, and conjunctions. Such a list is also referred to as a functional or structural word list (Salton, 1989).

A *linguistic test data set* is a set of documents, words, and perhaps their decomposition, that is used for testing the performance of automated systems and related algorithms that deal with natural language (Popovic & Willet, 1992).

A *test collection* is a set of documents, queries, and associated judgments of relevance used for testing the performance of information retrieval systems.

A *finite-state automaton* (FSA) or **finite automaton** (FA), for short, is a mathematical model of a system with discrete inputs and outputs. An FA consists of a finite set of

states and a set of transitions from state to state that occur on input symbols chosen from a certain alphabet (Hopcroft & Ullman, 1979).

A *finite-state transducer* (FST) is a device that recognizes some sequences of inputs and associates them with some outputs. Typically, an input sequence is a string of characters or a list of ordered words written in a natural language, and the output is a list of linguistic information (Al-Shalabi, 1996; Beesley, 1998). Advantages of FSTs include bidirectionality, where the same transducer can be used for analysis as well as generation, compactness, speed, and simplicity (Karttunen, 1983, 1994).

Two-level morphology is a system of word recognition that involves two-level finite-state phonology or graphology (Beesley, 1990; Kay, 1987). The two levels are closely related, character-by-character, with two-level rules. The *two-level finite-state morphology* is based on two main parts, namely, the two level rules and the lexicon. *Two-level rules* are simply descriptions of the correspondence between lexical and surface forms (Karttunen, 1983). The lexical form represents a collection of morphemes, each consisting of a string of characters, as they appear in the lexicon. The surface form is a string of characters representing the word as it actually appears in some surface orthography (Beesley, 1991; Kiraz, 1995). In many languages, especially Arabic, many realizations exist and are controlled by phonological and orthographical rules (Beesley, 1991). For efficient processing, these rules are represented as finite-state transducers. Koskenniemi's Finnish system required about 40 rules; seven were needed for Aymara and none for Esperanto (Beesley, 1991).

The *lexicon* is the set of valid lexical forms of a language (Karttunen, 1994). It can be thought of as a dictionary that contains lexical roots and affixes where each root is defined only once. A lexical entry may contain prefixes and infixes as well as an information field expressing syntactic and semantic properties (Beesley, 1991; Karttunen, 1983). Rules and lexicons work together and represent the main entities of the finite-state and two-level models (Beesley, 1991; Karttunen, 1983; Koskenniemi, 1983; Narayanan & Hashem, 1993).

Measures of Effectiveness of Morphological Analysis Techniques

The effectiveness of morphological analysis techniques is measured using factors including efficiency, compactness, bidirectionality, success rate, correctness, and retrieval performance.

Efficiency is one of the most important factors usually used to rank morphological analysis algorithms. The processing complexity and size of the tables used are the commonest sources of slowness. Some researchers have suggested using the *redundancy* factor as a measure of efficiency (El-Affendi, 1998). This factor results from the division of the number of acceptable decompositions by the number of suggested decompositions, which is strongly correlated with speed.

Compactness is a factor representing the size optimization of the algorithm. The *bidirectionality* factor indicates the ability of the algorithm to work for both generation and analysis. Some researchers have used morphological algorithms for compressing text (Ali, 1988). This results in the ability to use *success rates* as a measure of effectiveness.

Correctness, accuracy, and degree of coverage are equivalent terms given to the same factor (El-Affendi, 1998). In general, correctness refers to the ability to analyze all given words and produce one or more correct decompositions. Correctness can be affected by the ability to process stop words, foreign words, proper names, and multiword expressions. In Arabic, correctness is greatly affected by the ability to deal with many complicated phenomena such as vowelization, assimilation, vocalization, mutation, the existence of inert nouns and verbs, as well as the ability to deal with triliteral, quadriliteral, and quinqueliteral roots. In English, both *overstemming* and *understemming* usually degrade the correctness of stemming algorithms (Frakes & Baeza-Yates, 1992).

Retrieval performance is the most common factor used in evaluating morphological analysis techniques. It indicates the ability of the algorithm to retrieve the required documents. This factor is usually measured by the two well-known measures, namely, *recall* and *precision* (Salton, 1989).

Languages and Morphology

Languages vary considerably in morphological complexity. English, for example, has a simple morphology compared with languages such as Arabic and Hebrew. European languages involve more complex morphologies than does English (Savoy, 1999). For example, Slovenian is, like English, concatenative in its nature, but with a more complex morphology (Popovic & Willet, 1992). The affixation process in English and similar languages is simpler than that in other languages such as Arabic. In more complex languages, removal of suffixes alone will not be sufficient for many natural language applications (Ahmad et al., 1996; Krovetz, 1993; Paice, 1996). **Popovic** and **Willet** (1992) showed that the effectiveness of a stemming algorithm of a given language is determined by its morphological complexity. As an answer to a question raised by Harman on the effectiveness of suffixing (Harman, 1991), Popovic observed that suffixing can be very effective for languages with sufficient degree of morphological complexity (Popovic & Willet, 1992).

The direction of the writing of the script is not the only difference between Arabic and many other languages. The major difference is that Arabic is mainly derivational while others are concatenative. Figures 1 and 2 show a schematic diagram and an example of the Arabic derivational system (Al-Sughaiyer & Al-Kharashi, 2000; 2002c). Table 3 summarizes the major morphological differences between the Arabic and English languages (Ali, 1988).

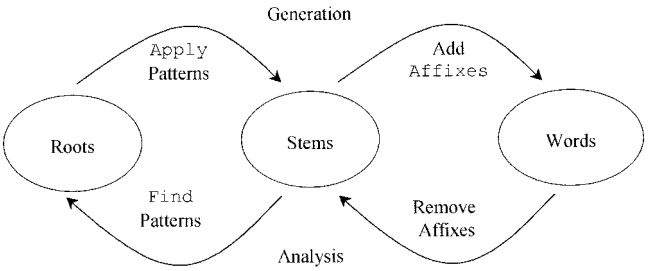

FIG. 1. Arabic derivational system.

Morphological Analysis Techniques and Information Retrieval

Considerable research on stemming and morphological analysis is building up for the Arabic language, but no standard information retrieval–oriented algorithm has yet emerged; meanwhile, available approaches have limited scope for use in information retrieval (De Roeck & Al-Fares, 2000; Larkey, Ballesteros, & Connell, 2002).

Researchers concluded that Arabic information retrieval can be enhanced when the roots or stems are used in indexing and searching. However, Arabic morphology is complex, and root identification may degrade the efficiency when used for information retrieval systems. Al-Suwaynea (1994) explained theoretically that using the root as the desired level of analysis might be useful for dictionaries and other natural language applications but not for information retrieval tasks. The reason behind this is that using the root causes conflation of a lot more terms and creation of invalid conflation classes, which degrades the performance by introducing extra noise.

Published comparison studies of using stems against using roots for information retrieval are discrepant. Older studies revealed that words sharing a root are semantically related, and root indexing is reported to outperform stem and word indexing on retrieval performance (Abu-Salem, 1999; Darwish, 2002a; Hmeidi, 1997). However, later works on the TREC collection showed two different results. Darwish et al. (2001, as cited by Larkey et al., 2002) found no consistent difference between root and stem while Al-Jlayl and Frieder (2002) showed that stem-based retrieval is more effective than root-based retrieval. The older studies showing the superiority of roots over stems are based on small and nonstandard test collections, making results nonjustifiable.[1]

The main problem of the root-based algorithm in information retrieval is that many surface word variants do not have similar semantic interpretations. Although these surface words are different in meaning, they originate from the same root. Thus, using the root-based algorithms in retrieval increases word ambiguities. Word-sense disambiguation is an essential to improve any Arabic information retrieval system (Al-Fares, 2000; Al-Jlayl & Frieder, 2002; Darwish, 2002a; De Roeck & Al-Fares, 2000; Larkey et al., 2002; Xu & Weischedel, 2002).

In the recall-precision graph, the low recall region is of great interest. Since users in any information retrieval medium (like the Web) are unlikely to read many retrieved documents, the higher precision values at lower recall are more significant.

English Language Stemming Algorithms

The morphological structure of English is very simple and straightforward. English stemming algorithms can be classified into *strong* and *weak* stemmers. Strong stemmers tend to remove a wide range of suffixes and bring about a large amount of conflation. They cause overstemming errors because of removal of all suspected endings even if some are not actual suffixes. Furthermore, they tend to relate all words with the same stem even if results are not linguistically correct or they refer to distinct concepts. They sometimes produce strings that are not correct English words due to the fact that word has an ending that is similar to a suffix. Weak stemmers, on the other hand, tend to remove only shorter suffixes and cause many understemming errors. Understemming errors cause words that refer to the same concept to be reduced to different stems (Lennon et al., 1981; Lovins, 1968; Paice, 1994; Popovic & Willet, 1992; Salton, 1989).

Stemming algorithms have their own limitations (Dawson, 1974; Lovins, 1968; Salton, 1989). In English, for example, removing some prefixes is not recommended because such removal effectively reverses the meaning of analyzed words. Furthermore, the similarity between some prefixes and the beginnings of some words (e.g., "re-" and "reason") may cause the removal of false prefixes. Most of the available stemmers are not fully accurate. Researchers attribute that to the fact that most stemmers handle words in an isolated manner, rather than dealing with them on a contextual basis. It is believed that most of the early developed stemmers do not pay attention to a word's different meanings (Hull & Grefenstette, 1996; Krovetz, 1993). Producing linguistically incorrect stems and considering a set of words conflated to the same root as synonyms are other commonly encountered limitations.

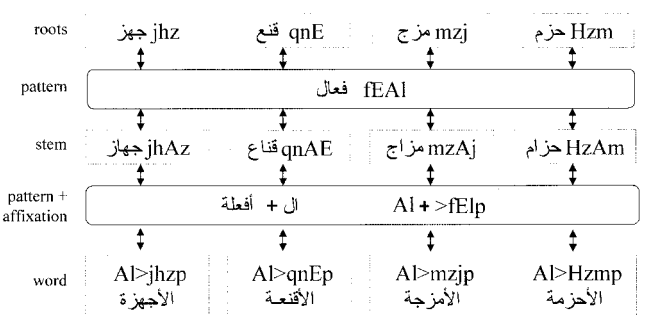

FIG. 2. Examples of the Arabic derivational system.

[1]Although TREC is the only standard Arabic test collection available, this test is small and has some problems (Larkey et al., 2002).

TABLE 3. Comparisons of morphologic properties: Arabic versus English.

Field of comparison	Arabic	English
Derivation	Based on patterns with few augmented letters	Mainly concatenative with large number of affixes
Inflection	Systematic with some exceptions	Too many exceptions
Root	The basic entry for dictionaries that provide the base of meaning	English language does not deal with roots; instead it uses stems and words as keys for dictionaries
Attaching articles and pronouns to words	Natural phenomena	Very limited
Morphophonemic modifications	Exist [common] like mutation and vocalization	Very limited

Overview of English Stemming Algorithms

In this section, a brief description of common English stemming algorithms is given. Extensive surveys of English stemming algorithms can be found in related literature (Dawson, 1974; Frakes & Baeza-Yates, 1992; Lennon et al., 1981; Lovins, 1968; Harman, 1991; Hull, 1996; Hull & Grefenstette, 1996). As shown in Figure 3, English stemmers can be classified into table lookup, linguistic, and computational approaches (Frakes & Baeza-Yates, 1992). Computational approaches are subclassified into successor-variety and n-gram approaches. Finite-state automata and affix removal represent linguistic algorithms where the latter can be subclassified into simple removal and longest match approaches.

Table lookup, computational, and linguistic affix removal approaches are referred to as traditional, cut-and-paste (Beesley, 1991), generative, or rule-based approaches (Narayanan & Hashem, 1994).

In the *table lookup* approach, words and their stems or roots, or both, are stored in tables. Utilizing a lookup process using these tables performs stemming. Although such a process might be very fast, it has very serious limitations, including storage overhead and the difficulties in constructing such tables (Frakes & Baeza-Yates, 1992).

In the *successor-variety* approach, the text itself determines the appropriate word segmentation, consequently allowing the algorithm to strip off affixes to produce stems. This approach**,** proposed by Hafer and Weiss (1974), is based on the linguistic concept of letter successor variety. The intention is to develop a segmentation process that requires minimal human intervention and prior decisions. Hafer and Weiss concluded that this process achieves accurate word segmentation, and information retrieval results were virtually identical to those produced by more manually oriented forms of stemming.

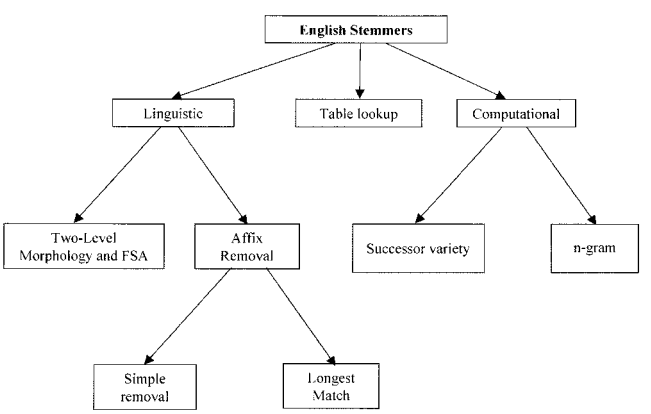

FIG. 3. Classification of English stemmers.

In the *n-gram* stemmer, proposed by Adamson and Boreham (1974), words are conflated by calculating some similarity coefficients between words. The similarity coefficient measures the number of shared consecutive n-characters found in different words. Although this approach does not produce stems, it actually conflates words so that it can be used as alternative search terms for information retrieval.

Linguistic approaches are the most commonly used, and can be classified as *affix removal* and *finite-state automata*. *Affix removal* approaches strip off suffixes and prefixes from words to produce the stems. Many researchers have followed an affix removal approach that can be divided into two main categories, namely, "simple removal," or "S" stemmer, and "longest match". The "S" stemmer is a simple algorithm that is easy to implement. It is based on removing the suffix letter "s" to conflate words from plural to singular forms (Frakes & Baeza-Yates, 1992; Harman, 1991). Lovins (1968) and Porter (1980), among others, proposed longest match algorithms. Lovins proposed a two-phase stemming algorithm. In the first phase, the longest possible ending that matches one on a predefined suffix list will be removed. The second phase covers spelling exceptions. Porter based his work on stripping off suffixes, using a list of suffixes and some condition codes to identify both the suffix to be removed and how to produce a valid stem.

Finally, *finite-state automata* (FSA) and *two-level morphological analysis techniques* are the most rapidly emerging and most promising techniques. They are based on the extensive use of lexicons to remove affixes and two-level rules to consider spelling exceptions (Karttunen, 1983; Koskenniemi, 1983). More details about FSA and two-level morphological analysis techniques will be given later, when Arabic morphological analysis techniques are considered.

Evaluation Studies

Suggested stemming algorithms for English were evaluated on different criteria. Lennon et al. (1981) evaluated the algorithms of Lovins (1968), Porter (1980), Dawson (1974), Hafer and Weiss (1974), Adamson and Boreham (1974), and other stemming algorithms, considering their effective-

ness in decreasing the size of dictionaries and improving retrieval performance. They concluded that retrieval effectiveness is sometimes improved but never worsened, and that storage utilization and the improvements in effectiveness are about the same for all stemmers.

Harman (1991) studied the effect of stemming on retrieval performance using S, Porter and Lovins stemmers. She concluded that using these algorithms did not improve retrieval performance in the test collections that were used, nor did her attempts to improve performance succeed.

Frakes and Baeza-Yates (1992), on the other hand, gave more detailed experimental evaluations of stemming algorithms. They presented the studies mentioned above, and also other studies that considered many parameters, such as which test collection was used, the use of statistical testing, and weak and strong points.

Hull and Grefenstette (1996) studied the use of an English lexical database produced by Xerox labs as a stemming algorithm. They compared the performance of the Xerox lexical database against some of the very well known English stemmers. Hull concluded that the average performance of linguistically based stemming is not significantly better than other algorithms, but with proper modifications, linguistic tools based on morphological analysis should work very successfully as stemming algorithms.

In conclusion, the process of conflation can affect retrieval performance. Most of the studies showed a beneficial effect, with minor differences noticed between different proposed algorithms. Studying the effectiveness of stemming algorithms showed a noticeable variation between different test collections. Researchers in the field of English stemming can benefit from many good features including clearly explained works, provided survey and comparative studies, used standard test collections for performance evaluation, and, finally, availability of many algorithms, mostly with source code.

Except for finite-state two-level morphology, the main limitation of these algorithms is that they deal with languages with simple morphological systems. As a result, most are not applicable to more morphologically complicated languages without modification.

Classification of Arabic Morphological Analysis Techniques

In this section, a review of the proposed classifications of Arabic morphological analysis techniques found in the literature is given, along with brief judgments, and followed by a description of each technique. The suggested classification is given at the end of this section.

Proposed Classifications

Ali (1988) classified techniques into technical morphological analyzers, two-level models, the Martin model, and syntactically based morphological analyzers. In technical morphological analyzers, words are treated as a sequence of character codes that should be automatically manipulated. Such analyzers are not expandable, and are hard to integrate with other systems. In two-level model techniques, tables are needed to guide the automaton that relates each character at the surface and lexical levels. The lexical level may contain nonalphanumeric symbols. This technique is bidirectional and hence can be used in both analysis and generation processes. It works well with concatenative languages but has severe limitations with Semitic languages. The Martin model technique, which is based on McCarthy's theorem, is a complicated analyzing method with limited usage. A syntactically based morphological analyzer is built by separating morphological rules from the morphological analyzer itself. It has a strong linguistic base, but presents some difficulties in modeling the process of derivation by patterns.

The classification proposed by Ali is reasonable, but his belief that technical analyzers are not expandable is questionable. As can be seen later, some, such as those proposed by Al-Fedaghi and Al-Anzi (1989) and El-Affendi (1998, 1999) are very promising. The judgment by Ali that two-level models are not applicable to Semitic languages was not so accurate. Researchers in the field adapted two-level models to suit Semitic languages.

In another scheme, Hlal (1989) proposed two classes: pattern compatibility and augmented transition networks (ATN). The pattern-compatibility approach, invented by Hlal, is based mainly on stripping out prefixes and suffixes and comparing what remains against pattern lists. Hlal has listed augmented transition networks without further information.

This classification ignored combinatorial approaches and included the augmented transition network approach, which is actually a tool rather than a separate approach.

Al-Fedaghi and Al-Anzi (1989) classified algorithms into linguistic and combinatorial approaches. The linguistic approach requires a large number of lists and tables. The result is an elaborate Arabic morphological expert system. In order to develop a set of rules to find proper decompositions, this approach is based on an extensive morphological analysis of the Arabic language. The combinatorial approach, on the other hand, generates all combinations of letters of a tested word. The resulting combinations will be compared against lists of roots and patterns and, on match, valid roots and patterns are extracted.

The classification suggested by Al-Fedaghi is reasonable and reflects the main types of analyzers.

El-Affendi (1991) provided yet another classification. He classified techniques into the subjective linguistic approach, the combination approach, and the exhaustive General Standard form (GS-form) approach. GS-form stands for the form of any pattern with proper prefixes and suffixes. The subjective linguistic approach simulates the process used by a linguistic expert. It consists of removing affixes through comparisons against predefined lists and transforming what remains, the stem, to a root, after possible alteration through adding, deleting, or modifying some of its letters. Crucial factors in this approach are the length of the stem and the possibility of matching against one of the lexicon entries.

The combination approach is a trial-and-error process. It takes all possible combinations of three and four letters of the word and compares them against lexicon entries. Matching roots will be accepted as possible analyses. The exhaustive GS-form approach compares the analyzed word with all stored GS-forms of the same length. The process of comparison generates a set of roots that will be searched for in the root list.

El-Affendi's proposed classification is excellent, given that the exhaustive GS-form approach is categorized as a subjective linguistic approach.

Xu, Fraser, and Weischedel (2002) explained that Arabic stemming algorithms can be classified, according to the desired level of analysis, as either stem-based or root-based algorithms. Stem-based algorithms, such as Buckwalter (2002) stemmer, remove prefixes and suffixes from Arabic words, while root-based algorithms, such as Beesley (1996) and Khoja (Khoja & Garside 1999) stemmers, reduce stems to roots. This classification is reasonable considering the desired level of analysis.

Larkey et al. (2002) proposed classifying Arabic stemmers into four different classes, namely, manually constructed dictionaries, algorithmic light stemmers, morphological analyzers, and statistical stemmers. Buckwalter (2002), which represents one example of the manually constructed dictionaries, developed a set of dictionaries of Arabic stems, prefixes, and suffixes, with their compatibility tables. *Light stemming* refers to the process of stripping off a small set of prefixes and/or suffixes without trying to deal with infixes or recognize patterns and find roots. Morphological analyzers attempt to find roots while statistical stemmers group word variants using clustering techniques. This classification is a mix of two different classification approaches.

Darwish (2002a) suggested classifying approaches into the symbolic approach, the statistical approach, and the hybrid approach. In the first approach, morphotactic and orthographic rules are programmed into a finite-state transducer (FST). This approach was criticized by Ahmed (2000, cited by Darwish, 2002a) for requiring too much manual processing to state rules in an FST and for the failure to analyze words that do not appear in Arabic dictionaries. The second approach learns prefixes, suffixes, and patterns from a corpus or word list in the target language without any need for human involvement. However, such a system would not be effective in Arabic morphology, because it doesn't tackle the issues of affixation. A third approach uses rules in conjunction with some language statistics. This approach uses a list of prefixes, suffixes, and patterns to transform from stem to root. Possible prefix-pattern-suffix combinations are constructed for a word to derive the possible roots. The advantage of this approach is the ability to achieve broader morphological coverage of the Arabic language. Disadvantages, however, include a lengthy manual derivation of rules, time consumption, and requirement of a good knowledge of Arabic orthographic and morphotactic system.

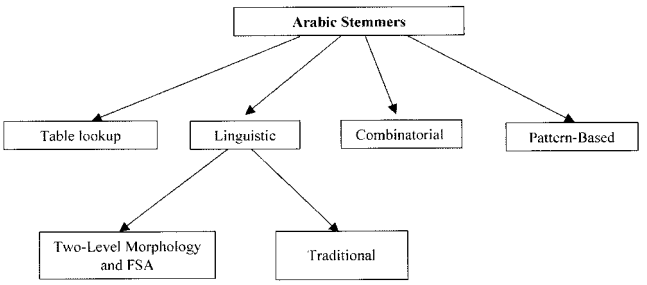

FIG. 4. Classification of Arabic stemmers.

This classification has the flaw of neglecting table lookup and combinatorial approaches. It also represented a shorthand linguistic approach.

Suggested Classification

As shown in Figure 4, it is believed that Arabic morphological analysis techniques can be placed into four main categories, namely, table lookup, linguistic, combinatorial, and pattern-based approaches.

In the *table lookup* approach, all valid natural Arabic words along with their morphological decompositions are stored in a huge table. A given word is analyzed simply by accessing the table and retrieving information associated with that entry. *Linguistic* approaches, on the other hand, utilize linguistic rules that have been derived through deep analysis of Arabic morphological systems. This process simulates the behavior of a linguist during the analysis of a given Arabic word. *Combinatorial* approaches are processes of trial and error, where all combinations of letters of a given word are tested and compared against a list of roots. Finally, the *pattern-based* approach utilizes the apparent symmetry of generated natural Arabic words.

All proposed algorithms could be easily placed in one of these four categories. In addition, the proposed classification according to the desired level of analysis is very legitimate. It should be noted, however, that ambiguity is a common symptom of all of the mentioned approaches. Context analysis and higher-level analysis is the solution to this problem.

Survey of Arabic Morphological Analysis Techniques

In this section, a survey of Arabic morphological analysis techniques is conducted that includes a description of the algorithm, the method of its implementation, and its advantages and weaknesses. In this survey, a few tasks were performed. First, a large amount of literature was collected. Then, every proposed algorithm was studied and summarized. Finally, a standard framework for evaluating different algorithms was developed. This framework includes a summary of each algorithm, a collection of other researchers' judgments of it, and a conclusion giving our judgment.

TABLE 4. Summary of surveyed algorithms.

Author(s) and date	Algorithm name and type	Implementation	Test data	Success rate	Language coverage	Required lists	Presentation
Hegazi & El-Sharkawi (1985, 1986)	CAMH Linguistic	NA	NA	NA	Vowelized only	Roots, patterns, tools, affixes	3
Hlal (1985, 1987, 1989, 1990)	Linguistic	PC, Pascal	NA	NA	Vowelized & non-vowelized	Roots, patterns, defective roots, affixes, primary articles	5
Gheith et al. (1985, 1987)	Linguistic	NA	NA	NA	Non-vowelized, triliteral roots only	NA	3
Thalouth, Saliba, Al-Dannan (1987)	Linguistic	NA	NA	NA	Non-vowelized only	Affixes, patterns, foreign words, function words, solid word roots	9
Ali & Al-Shami (1988, 1992); Sakhr Software	MMMP Linguistic	NA	NA	NA	Vowelized, non-vowelized, partially vowelized	Roots, patterns, solid nouns, rules	8
Foxley & Feddag (1990)	Linguistic	NA	Holy Quran, Arabic Poetry	Algorithm1: 99.8% Algorithm2: 84.8%	Vowelized	Affixes, stems	6
Al-Fedaghi & Al-Anzi (1989)	Combinatorial	C language	Holy Quran, Tawfeeq, Aqad, info Arabic text	72% [on average]	Triliteral roots only	Roots & Patterns with affixes	8
Al-Uthman (1990)	Linguistic	PC, Prolog	NA	NA	NA	Patterns	7
Beesley (1990, 1991)	ALPNET, Linguistic	Macintosh, Lisp & IBM 4361 mainframe	NA	NA	Vowelized, non-vowelized, partially vowelized	Patterns, stems, and affixes	9
El-Affendi (1991, 1992)	Linguistic	Pascal	NA	NA	Vowelized, non-vowelized, partially vowelized	Patterns with affixes, roots	9
Al-Bawab et al. (1994, 1998)	Linguistic	Prolog in PC and VAX11-780	NA	NA	Vowelized, non-vowelized, partially vowelized	Affixes, patterns, noun derivatives, solid nouns	6
Al-Shalabi (1996)	Combinatorial	PC, C language	7 abstracts from NCC	19%	Triliteral & quadriliteral roots	Prefixes, roots, stop words, weak verbs	7
Beesley (1996, 2001)	Xerox, Linguistic	Xerox FST format	NA	NA	Vowelized, non-vowelized, partially vowelized	Patterns, stems, and affixes	9
Aref (1997)	Linguistic	NA	NA	NA	NA	Stems, affixes	6
Khoja & Garside (1999)	Linguistic	Java 1.2	Classical Arabic text	96%	Vowelized & non-vowelized	Stop words, patterns and roots	8
El-Affendi (1998, 1999)	SWAM Combinatorial	Jscript	Classical Arabic book	97%	NA	Roots, patterns, particles, affixes, invariable nouns	9
De Roeck & Al-Fares (2000)	Combinatorial	NA	Five data sets	94.06%	NA	Affixes	8
Al-Kharashi & Al-Sughaiyer (2000, 2002a,b,c,d,e)	Rule based	PC, C++	Classical Arabic text	80%	Non-vowelized	Not required	9
Darwish (2002a)	Sebawai, Linguistic	Perl	Xerox analyzer	84%	Non-vowelized	Root, stop word, suffix, prefix and pattern lists	9
Darwish (2002b)	Linguistic	Perl	NA	NA	Non-vowelized	Not required	7
Buckwalter (2002)	Linguistic	Perl	NA	NA	Non-vowelized	Prefix, stem, suffix and compatibility tables	9
Al-Jlayl & Frieder (2002)	Linguistic	NA	TREC benchmark data	87.4% increase in average precision	NA	Affixes, foreign words and function words	8
Larkey et al. (2002)	Linguistic	NA	TREC benchmark data	NA	Vowelized & non-vowelized	Affixes, function words	8
Abuleil et al. (2002)	Linguistic	NA	500 nouns from newspaper	90.2% with user-feedback	NA	Suffixes, patterns, nouns database	8

Applying an evaluation framework involves some difficulties. First of all, most of the suggested analyzers are not available either as binaries or as source codes. This makes it impossible to consider many important evaluation parameters such as speed, correctness, bidirectionality, and modularity. Second, many of the algorithms did not take into account many important parameters such as language coverage, implementation issues, the size of the algorithm, and the lists used.

Table 4 gives an overall summary of the algorithms studied. The last column, namely presentation, reflects our opinion about the clarity and completeness of the algorithms, where 10 is the highest rating and 1 is the lowest. "NA" stands for "no available information." Despite the difficulties mentioned above, the table was created using all available information. Some algorithms and research activities were not included because they did not represent a complete morphological analyzer, or there was not enough information.

There follows a description of the four major approaches used in Arabic morphological analysis where techniques are presented in chronological order according to the date of publication and the type of approach.

Table Lookup Approach

The table lookup approach depends mainly on very large tables storing Arabic words found in natural texts with their

Natural word	Stem	Root	Prefix	Suffix
AlvHlyl التحليل	vHlyl تحليل	Hll ح ل ل	Al ال	
...				
vHlyl تحليل	vHlyl تحليل	Hll ح ل ل		
...				
vHlylh تحليله	vHlyl تحليل	Hll ح ل ل		h هـ
...				
wAlvHlyl والتحليل	vHlyl تحليل	Hll ح ل ل	wAl وال	
...				
wbvHlylAv وبتحليلات	vHlyl تحليل	Hll ح ل ل	wb وب	Av ات
...				

FIG. 5. Word entries in the table lookup.

corresponding morphological parts. Such parts include stem, root, and affixation. Words may include functional words, foreign words, and proper names where each word uses a unique entry in the table as shown in Figure 5. Multiple entries may exist for words spelled the same to reflect the possibility of multiple analyses.

Table entries are listed in alphabetical order. A hash table or binary search list can be used to optimize the search. Furthermore, a compression technique can be used to reduce storage requirements. Morphological analysis is then as simple as accessing the hash table or binary search tree.

Even though such approaches are accurate, the problems associated with them include the lack of such comprehensive data; the need for too much work, including linguistic effort to develop them; storage overhead; and retrieval time needed for such data (Frakes & Baeza-Yates, 1992; Kalamboukis, 1995).

Combinatorial Approach

Combinatorial approaches compare tested words against prepared lists for roots, patterns, particles, and affixes. The comparison is based on a combinatorial algorithm that tests all combinations of three or four letters of a given word in order to extract the root (see Figure 6). In general, such approaches are simple but take a long time to process and require very large lists.

Al-Fedaghi and Al-Anzi (1989) enhanced an algorithm proposed previously by Al-Fedaghi and Al-Sadoun. The algorithm processes triliteral root words only, and is very simple and straightforward but very slow (it is of the order O [w^3] where w is the word length). The enhancement is achieved by solving some important phonological and orthographical problems, such as assimilation, mutation, vocalization, and gemination. The modified algorithm uses lists of triliteral roots and patterns with all combinations of affixes. It starts by comparing the input word against the pattern list to extract the root. The algorithm then runs in four modes to cover all possible orthographical and phonological cases. Cases cover the deletion of any single or two letters and dealing with intact words. The results of testing the algorithm were given, which includes the total processing time for different modes. Except when dealing with the deletion of two characters, which produces less than 0.1% of the reduced words, the algorithm is reasonably fast. Al-Fedaghi and Al-Anzi concluded that some linguistic rules could be integrated with combinatorial algorithms in order to produce a heterogeneous algorithm that combines the advantages of both.

The main advantages of this work are the use of test data sets and the provision of the success rate. The main disadvantages include an insufficient success rate (Al-Atram, 1990), restriction to triliteral roots (Al-Shalabi, 1996), slowness (Al-Kharashi, 1991), and storage overhead for patterns.

Al-Shalabi (1996) proposed a slightly modified version of Al-Fedaghi's algorithm (Al-Fedaghi & Al-Anzi, 1989) for analyzing Arabic words. The modification is made to handle words with quadriliteral roots. Furthermore, Al-Shalabi proposed a new algorithm for finding triliteral and quadriliteral roots and their patterns. In this algorithm, the longest prefix preceding the first root letter in the analyzed word is removed. The algorithm assumes that the root letters are contained in the first four or five letters of the remaining part of the word. Once a root is found by testing different combinations, a pattern is then constructed by replacing the root letters from the remaining part of the word with the letters of the basic pattern. To test the system, Al-Shalabi used 242 Arabic abstracts where seven are used for detailed analyses. In the testing set, all stop words are manually identified and removed. The system uses a few tables, including a list of weak verbs with their corresponding roots, stems, and patterns, a list of twelve pronouns for generating different forms of the word, a list of prefixes, and a list of roots taken solely from the test set. Al-Shalabi provided a study comparing his algorithm with Al-Fedaghi's. Al-Shalabi believes that this system is faster, re-

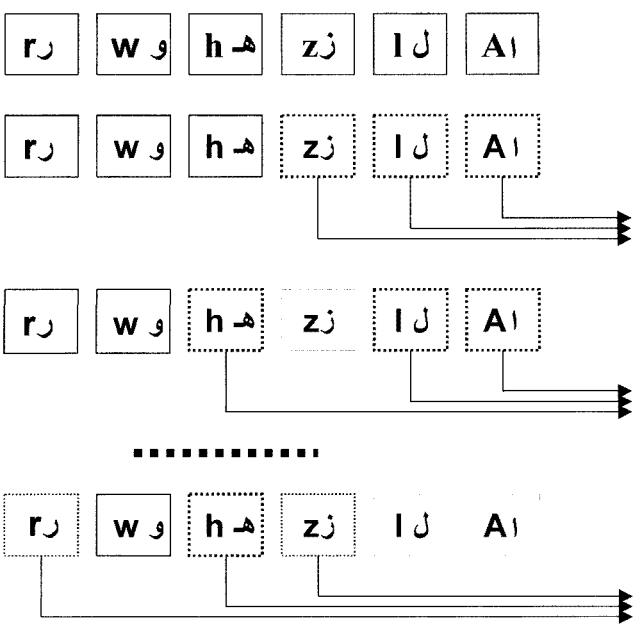

FIG. 6. Combinatorial-based approach.

quires much less space, and is capable of dealing with triliteral and quadriliteral roots. The problem of mutation and vocalization has been partially resolved using a hash table.

Abuleil, Alsamara, and Evens (2002) explained that Al-Shalabi reduced the processing, but he discussed this from the verbs point of view. The success rate was given but a comparison of the correctness of the two algorithms was not provided. The system gave a very low success rate (19%) in recognition of derived nouns. The system requires much data preparation that includes lists of roots, prefixes, function words, and weak verbs.

El-Affendi (1998, 1999) proposed a sliding window approximate matching (SWAM) algorithm based on an approximate matching algorithm for morphological analysis. While the algorithm deals with derivational Arabic words as well as combinations of particles and pronouns, it excludes inert words. The algorithm requires a list of roots, patterns, particles, prefixes, and suffixes. It also requires a list of all invariable nouns. El-Affendi used the term "stored pattern" to represent a pattern, particle, or an invariable noun. The algorithm attempts to find some stored patterns that achieve a reasonable match score against the input word. The match score is calculated by sliding a window containing the stored pattern against the analyzed word in an attempt to find the correct decomposition. By using lists for defective Arabic words and roots, current implementation of the algorithm deals with the problem of character transformation and deletion, including mutation and vocalization.

El-Affendi evaluated the performance of this algorithm using two indicators, namely, the degree of redundancy and the degree of coverage. The achieved redundancy is below 10%, while the degree of coverage is above 97%. A text of classical Arabic book was used for testing the algorithm. El-Affendi concluded that the algorithm, when applied to a very large set of data, required a long time to search for correct solutions. According to El-Affendi, advantages of the algorithm include effectiveness in analysis and generation, avoidance of backtracking problems, and high susceptibility to parallelism. In general, the performance of the algorithm is quite satisfactory, but further work is needed to make it perfect. Another modification (El-Affendi, 1999) suggested the use of some heuristics and rules to decide the root of a given word. This algorithm uses a sliding window fuzzy algorithm. However, no more details were given.

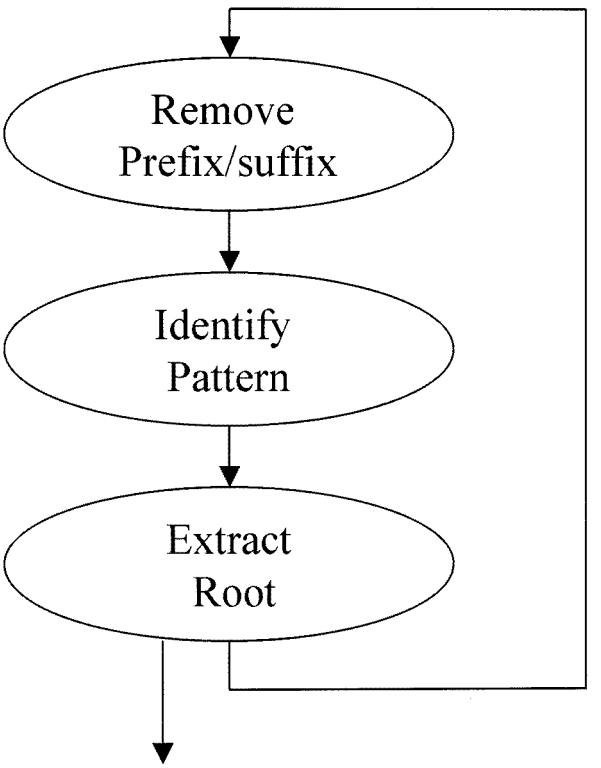

FIG. 7. Linguistic-based approach.

Pattern bitmap

Root 1	1	0					1	0
Root 2	0	1					1	1
	⋮		.....					
Root n-1	1	1					0	1
Root n	0	0					1	0

FIG. 8. Compatibility vectors.

Linguistic Approach

Linguistic-based approaches require a deep analysis of the Arabic morphological system. They simulate the behavior of a linguist during the analysis of a given Arabic word. In this approach, tested words are compared against lists of affixes or function words to derive stems and then compared against lists of patterns and roots in order to derive roots (see Figure 7).

In general, such linguistic approaches are more accurate but require too many lists that need to be prepared and checked linguistically. Such lists incur a time overhead for accessing them. In addition, the mechanism for removing affixes is almost a process of trial and error, in which results are not guaranteed to be accurate.

El-Affendi (1999) gave a framework for a linguistic approach based on compatibility vectors. In this framework, and since there is no obvious rule to relate roots to patterns, the compatibility vectors shown in Figure 8 might be used to aid the analysis process. Note that "1" indicates that the corresponding pattern is compatible with the current root and "0" indicates a noncompatible pair.

In Arabic, there are about 10,000 roots and 900 patterns. Not every root is compatible with every pattern. Linguistic experts must be consulted when creating comprehensive

compatibility vectors. If compatibility vectors for all roots are available, then the general linguistic algorithm can be used. Once found, roots should be checked in the vectors to verify the compatibility between the roots and the suggested patterns.

Linguistic approaches can be subdivided into traditional or cut-and-paste approaches, and finite-state automata and two-level morphology approaches. In the following, traditional linguistic approaches will be discussed, followed by finite-state and two-level morphology models.

Traditional Approaches

Hegazi and El-Sharkawi (1985, 1986) proposed a computer-aided morphological hierarchy (CAMH) algorithm for deriving the root of a given vowelized Arabic word, its morphological pattern, and morphological attributes. Hegazi and El-Sharkawi explained the approaches that Arabic linguists used to deal with Arabic morphology. The first uses some morphological rules, which requires memory and mental skills and is dependent on human accumulated knowledge. The second uses some phonetic rules, which may produce some incorrect results. Hegazi and El-Sharkawi believe that using both approaches will produce better results. They used their algorithm to build an automated lexical analyzer that requires a list giving Arabic roots and affixes, morphological patterns, and function words.

The flowchart of the analyzer was provided with almost no explanation. The analyzer is based on morphological and phonetic rules and takes into account foreign words, spelling mistakes, and the required vowelization.

The main advantages of the system are that it is linguistically based and that it provides a good tool for automatic translation (Al-Uthman, 1990). Even though a modified version of this analyzer was claimed to be suitable for dealing with nonvowelized Arabic texts, the main disadvantage is that the proposed system processes vowelized Arabic text only. The analyzer has not been explained in detail, and no experimental evaluations or results have been provided.

Hlal (1985, 1987, 1989, 1990) proposed a two-stage cascaded system that uses a set of elementary instructions and a set of rules. The system makes extensive use of multiple lists that include prefixes, suffixes, roots, patterns, defective roots, and primary articles. The system processes functional words in the first stage and regular words in the second stage.

In the first stage, the system depends on a list of primary functional words, a decomposition algorithm, and a set of rules for the validation of the decomposition of functional words into primary functional words. *Functional word* is defined as the concatenation of more than one primary functional word. The algorithm in this stage terminates if the word is found in the primary functional word list. If not, either the word is processed as a complex functional word and analyzed or it is treated as a regular word and passed to the next stage.

In the second stage, the processing of regular words is based on determining the longest prefix and suffix, then finding the length of the remaining part. On the basis of the extracted information, a series of actions is performed to find the root and verify the correctness of the decomposition. If the process does not succeed, then smaller prefixes and suffixes are selected, and the same actions are repeated.

According to Hlal, advantages include the ease with which the system may be checked since the rules are separate from the elementary instructions, the independence of the algorithm's design, and the ability to process other languages by simply writing a set of rules.

The system proposed by Hlal was one of the earliest published works. It gave a comprehensive theoretical approach for analyzing and generating Arabic words that deals with both vowelized and nonvowelized words (Al-Kharashi, 1991; Al-Uthman, 1990). More details about the implementation of rules and compilation of lists were provided. The main disadvantages of the system are its heavy dependence on lists and the difficulty of tracing and producing implemented rules. No experimental evaluations and results were given.

Gheith (Gheith & El-Sadany, 1987; Gheith & Mashour, 1985) proposed a linguistic model that works for nonvowelized text. It was composed of a list of basic words and a morphological analyzer. The list contains all vocabulary used but not all inflections of words (the analyzer handles these inflections). Morphological rules, in the morphological analyzer, were implemented using the augmented transition network (ATN) originally used in the syntactical analysis. The proposed model is based on two cascaded stages. In the first stage, the model tries to find an identical word in the list of basic words. If found, then the analysis process is terminated. Otherwise, the second stage is activated and the ATN will be traversed to seek an analysis of the input word. During this process, the list of basic words is consulted in each traversed state of the ATN. As output, the ATN should give the root and the morphological features.

In this work, Gheith gave some sample output that contains valuable information. The limitation to Arabic words with triliteral roots represents the main disadvantage of this work. The system also lacks a proper description of how it works. Neither the success rate nor experimental evaluations were provided.

Thalouth and Al-Dannan (1987) and Saliba and Al-Dannan (1989) proposed an algorithm based on solid linguistic rules using an extensive pattern-matching system. For that, lists of prefixes, suffixes, verb roots, solid word roots, patterns, foreign words, and function words were created using statistical studies. These lists are accompanied by a large number of flags such as compatibility between roots and patterns, compatibility between patterns on the one hand and suffixes and prefixes on the other, and compatibility between suffixes and prefixes. The aim of this work was to build a morphological analyzer and generator for nonvowelized Arabic words. The proposed algorithm modified some predefined patterns created by Arab linguists. It used morphological computational balances, which overcomes problems such as mutation and vocaliza-

tion. In this algorithm, Arabic words are placed in two main categories, namely, function and content words. Function words consist mainly of about 200 prepositions and pronouns. With suffixes, the number will increase to about 600 words. Content words are all other Arabic words, including foreign words. The algorithm is based on three cascaded processing stages, namely, function-, content-, and foreign-word stages.

In the function-word processing stage, the algorithm removes the longest prefix, then checks if the remainder is a function word. In this case, many flags are checked to identify the compatibility between various word types and prefixes. In the second stage, the largest suffix is removed and its compatibility with the prefix is checked. If they are compatible, the remaining string is checked against stored patterns. Fewer patterns will be suggested as valid by checking compatibility flags. Valid roots are extracted using pattern matching, checking against the root list, and validating their derivability. The process of this stage is repeated using all possible suffixes and prefixes. Saliba and Al-Dannan (1989) gave more details about content word analysis. Finally, in the foreign-word stage, the compatibility of the tested words and affixes is checked against the foreign-word database. If not compatible, other possible combinations of prefixes and suffixes should be used. The authors concluded that this algorithm has proved to be successful in treating derived and inflected words.

The main advantages of this work include providing a more practical approach to the analysis of an unvowelized Arabic text (Al-Kharashi, 1991) and building a very clear and isolated list (Al-Uthman, 1990). However, the main disadvantages include inefficiency and the use of too many lists (Al-Uthman, 1990), the use of computational balances instead of linguistic balances, the need for a lengthy backtracking process for affix removal (El-Affendi, 1998), and the absence of experimental evaluations and success rates.

Ali and Al-Shami (Sakhr Software; Ali, 1988, 1992) developed a multi-mode morphological processor (MMMP) that can deal with Arabic words with full, partial, or no vowels. The algorithm is based on the analysis-by-synthesis model that was initially developed for phonetics. This processor contains four main modules. The morphosyntactical module removes affixes using the ATN and deals with phonological processes such as assimilation. The output of this module is a set of possible solutions that will be fed into the next module. The derivational module extracts the root and the patterns, using a pattern-matching algorithm. If one or more roots were found, the process had succeeded; otherwise some morphophonetic rules should be applied. The output of this module is a list of roots and patterns that will be fed into the next module. The third module, the parsing module, basically identifies the parsing or syntax case. The main objective of the last module, the vowelization module, is to validate the analyses. It reconstructs the word and compares it with the input word. If they are identical, then the process succeeds; otherwise more combinations should be tested.

The main advantages of the system are the ability to process Arabic words with full, partial, or no vowels and its bidirectionality and modularity. Furthermore, the system has been implemented as part of many commercial products (Sakhr Software). The main disadvantage is that neither experimental evaluation using free natural text nor success rate has been shown in the literature.

Foxley and Feddag (1990) proposed a syntactic and morphological analyzer that decomposes Arabic words into roots, morphological patterns, and affixes. The proposed analyzer requires a list of affixes and stems. Foxley and Feddag developed and tested two algorithms to examine a given vowelized Arabic word. The difference between the two algorithms is the method of storing affixes. In the first method, three different lists of affixes were created in which a group of all possible combinations were stored. In the second method, only isolated affixes were stored in three different lists. Both algorithms were provided with experimental tests. The total number of tested distinct words was 20,000. Two tables were given to show the results of processing these words using the two proposed algorithms. The experiments show that the first gave a 99.8% success rate of stripping affixes, while the second gave a success rate of only 84.8%.

According to the authors, inclusion of the English meanings of Arabic stems makes the system of practical value in translation by machine from Arabic to English and in teaching Arabic grammar to non-Arabic speakers.

The main advantage is that their work was one of the first implementations to include test data and to give a success rate. Disadvantages of this work, however, are that it deals with vowelized Arabic words only and lacks an adequate explanation of algorithms and their implementation.

Al-Uthman (1990) developed a morphological analysis system to classify an input word and provide its root and pattern. The system consists of two main modules, namely, a user interface and a morphological module. The morphological module consists of control, verb, noun, and article submodules. The control submodule attempts to classify the input word as verb, noun, or article and activates the corresponding submodule. In both the verb and noun submodules, the corresponding patterns are used to extract the roots and classify them according to their lengths, ranging from 1 to 12 and 2 to 14 letters, respectively. The article submodule, on the other hand, classifies articles according to their lengths, ranging from 2 to 7 letters. Al-Uthman gave a short explanation about the system with some examples.

The main advantages of this work include provision of a short survey of some morphological analysis techniques and a full explanation of the morphological characteristics of the Arabic language. Disadvantages, however, include a lack of explanation of how to identify the word type in the control submodule, no explanation of how to discriminate between different cases inside submodules, and that neither success rate nor experimental evaluations were given.

El-Affendi (1991) developed an algebraic algorithm to analyze Arabic words. This algorithm is based on the numerical representation method discussed in El-Affendi

(1992) and requires a bitmap lexicon of Arabic roots. It uses the root standard form (RS-form), stem standard form (SS-form), and generalized standard form (GS-form). The RS-Form is a model word for the root formed using basic Arabic pattern letters (e.g., فَعَلَ faEala). The SS-form is a form obtained from the RS-form by adding some nonoriginal infixes and/or modifying one or more of the original letters (e.g., فَعُولٌ faEuwolN). The GS-form is obtained from the SS-form by attaching some prefixes and/or suffixes (e.g., فَعَلْتُمُوهَا faEalovumuwohaA). The GS-form encapsulates most of the information on natural Arabic words, including mutation and vocalization. El-Affendi (1992) showed that any Arabic word could be mapped to a unique pentadecimal integer (PI) value. The mapping process is performed by representing each letter as a unique integer ranging from 1 to 31 using five binary digits. The PI value is extended to code binary representation and presence of vowels using the B-form and NV-form, respectively. The morphological analysis is performed by identifying the GS-form that matches the input word using its PI value. Once the GS-form is identified, then root, stem, pattern, and affixes can be easily extracted. To perform morphological analysis, El-Affendi developed two theorems with their proofs. These theorems are the base of the suggested morphological analysis algorithm. The first theorem is used to identify the GS-forms that match the tested word. The second, however, is used to identify the root of a given PI value.

According to El-Affendi, the method as an illustration greatly simplifies the process of morphological analysis and improves its efficiency. It can be easily extended for morphological synthesis. Furthermore, the storage capacity required by the algorithm is far less than that required by conventional algorithms. In this work, neither samples nor success rate for the algorithm has been given.

Al-Bawab et al. (Al-Bawab & Al-Tayyan, 1998; Al-Bawab, Mrayati, Alam, & Al-Tayyan, 1994) proposed a morphosyntactic system for generating and analyzing Arabic words. The analyzer guesses the type of input word as verb, noun, or article. A little information can be used as a hint as to word type, such as a nunation mark. Verb analysis starts by removing affixes and, in the case of false partitioning, it will be reattached completely or partially. After partial or full removal of affixes, the remainder will be compared with patterns that are equal in length. Entries in the pattern list are sorted according to pattern lengths and vowelization and consist of patterns, roots, and morphological properties. Many test processes are carried out to assure correct analysis including compatibility between prefix and patterns, the availability of the root in the list, and checking pronouns and syntax case of verbs.

For noun analysis, a large number of lists need to be investigated, such as noun derivatives (e.g., nouns of place, of time, of instrument), nonstandard or solid nouns and different forms (e.g., plural, dual, and feminine). The authors state that the system covers the Arabic dictionary and can analyze vowelized and nonvowelized Arabic words. They also state that their implementation of pattern matching requires almost no time and lists do not require vast memory space.

Although this work neither included the use of test collection nor reported success rate, it has the advantage of being clearly built on a very strong linguistic base.

Aref (1997) presented an object-oriented algorithm of morphological analysis that is based on an object-oriented representation of the lexicon. This algorithm accesses the lexicon to check the existence of word stems and to obtain linguistic information about them. The lexicon is built as a hierarchy of classes, subclasses, and superclasses to represent lexemes, hyponyms of a lexeme, and hypernyms of several lexemes, respectively. An object-oriented representation is used to accommodate the grammatical properties of words and the semantic relations between them. In this representation, linguistic information and processes are distributed over several levels of classes, superclasses, and subclasses.

This algorithm provides the stem of the input word along with relevant linguistic information and all affixes attached to it. The algorithm is implemented using a message-handling approach.

In this work, test collection was not used nor was success rate reported; hence it is difficult to assess the algorithm.

Khoja and Garside (1999) presented a linguistic algorithm that analyzes vowelized and nonvowelized Arabic words. In this algorithm, a list of function words is used to detect and filter them. The algorithm also uses backtracking as a remedy for the erroneous removal of affixes. Straightforward pattern matching is used to extract roots, where a prepared list of roots is used to check the validity of extracted roots. The algorithm tries to solve special cases, such as words containing weak letters or gemination mark and hamza or glottal stop. The output of the system gives some useful statistics. The work concludes with a statement of the advantages and disadvantages of the system. Advantages include high accuracy and high processing speed.

Khoja's algorithm showed superiority over previous works in root detection algorithms (Al-Jlayl & Frieder, 2002). Larkey et al. (2002) explained that, unlike the Buckwalter (2002) approach, the Khoja scheme has no table restricting the patterns and affixes applicable to particular stems and roots. Preliminary work with the Khoja stemmer revealed problems with proper names.

The main disadvantages are the inability to stem inert names and foreign words, or to analyze them correctly. Furthermore, the algorithm produces roots, instead of stems or patterns. Without modifications, this makes it imperfect for the purposes of information retrieval.

However, in TREC10 cross-language track, the algorithm proposed by Khoja has been used with some modifications to test the performance of an information retrieval system. The authors showed that the algorithm made many mistakes but greatly improved the performance (Larkey & Connell, 2001).

De Roeck and Al-Fares (2000) presented a clustering algorithm for Arabic words sharing the same root. They adopted Adamson and Boreham (1974) algorithm to de-

velop and implement a two-stage algorithm. The two-stage algorithm applies light stemming before calculating word pair similarity coefficients using techniques sensitive to Arabic morphology. The objective of this work is to avoid morphological and syntactical analyses of Arabic words by using clustering as technique for grouping words sharing a root. Root-based clusters can substitute root dictionaries for indexing in information retrieval and provide an alternative search terms. The authors explained that clusters grow dynamically without maintenance and accommodate regional spelling conventions and even some spelling errors.

Authors performed few experiments using five data sets. The first set is a controlled one containing selected roots with derived words chosen for their problematic structure, while the rest are natural text sets. Following Adamson, function words have been removed. Sets one to three are used in refining the algorithm while sets four and five are used for evaluation purposes.

Adamson's algorithm was successful for English but it did not perform equally well on Arabic. This is due to the fact that Arabic words tend to be short, and the chance of words derived from different roots sharing a significant proportion of characters is high. Authors introduced and tested a number of successive enhancements based on the morphological knowledge. Refinements include using light stemming, reducing the effect of weak letters, determining the best setting for n-gram and character overlap size, and using blank insertion.

The authors explained that algorithms designed for relatively uninflected languages can be adapted for highly inflected languages by using morphological knowledge. Furthermore, the two-stage algorithm gave a significant improvement over Adamson's algorithm for used data sets. Tests show a successful treatment of infixes and accurate clustering to up to 94.06% for unedited Arabic text samples without the use of dictionaries. The modified algorithm dealt successfully with infixes in multiword clustering, an area where Adamson's algorithm failed. It matched the strength of Adamson in identifying single-word clusters, and sometimes did better. Finally, weak letters and the overlap between root and affix consonants continue to cause interference. However, the results are promising and suggest that the approach may scale up. Authors explained that evaluations are very encouraging, and though the samples are small, they give a strong indication that this kind of approach may transfer well to text from different domains on a large scale. Although the current results are promising, evaluation was hampered by the lack of a sizable data set to verify whether the solution would scale up.

Advantages of this work include the efforts of adjusting techniques to be suitable for Arabic language and trying to deal with difficult phenomena like mutation and vocalization. However, main disadvantages include the requirement of intense calculations for finding the similarity coefficient factor, not to evaluate the performance in an information retrieval context (Larkey et al., 2002) and concentrating on Arabic words sharing the same verbal root (Abuleil et al., 2002).

Darwish (2002a) designed a light Arabic stemmer that removes common prefixes and suffixes. The stemmer is very compact in size[2] and has been significantly improved by a modification done by Leah Larkey. The stemmer accepts input in either CP-1256 or UTF-8 encoding. Internally, it transliterates input string into Roman characters, removes diacritics, and makes letter normalization. This stemmer is available publicly for research purposes.

Simplicity and compactness are the main advantages of this stemmer. It should be noticed that this is a light stemmer that may produce too many erroneous analyses. Many errors can be encountered for words having start or end characters that are similar to affixes. Neither success rate nor retrieval performance was reported in the literature.

In other work, Darwish (2002a) presented a quick method for performing shallow morphological analysis for use in information retrieval. This analyzer, named Sebawai, is possibly the first freely available analyzer for Arabic. It does not require a manually constructed lists of rules and affixes. It should work with any three or four letter root and requires root, function word, suffix, prefix, and pattern lists. It has a command line interface that displays output on the screen.

Darwish developed two main modules to train and build the analyzer. The first module utilizes a list of Arabic word-root pairs to derive a list of prefixes and suffixes, construct stem patterns,[3] and compute the likelihood that a prefix, a suffix, or a pattern would appear. The second module accepts Arabic words as input, attempts to construct possible prefix-suffix pattern combinations, and outputs a ranked list of possible roots.

The list of word-root pairs was automatically constructed by utilizing a preexisting morphological analyzer (ALPNET) using two different lists of words. Words that the ALPNET was unable to analyze were removed from the list.

The first module is used to train the system by analyzing every word-root pair to determine the prefix, suffix, and stem pattern. It also calculates the number of occurrences of prefix, suffix, and stem patterns to assign, at the end of the training, conditional probabilities for different decompositions. The second module, the root detection module, accepts an Arabic word and attempts to generate prefix-pattern-suffix combinations. The combinations are produced by progressively removing prefixes and suffixes and then trying to match all the produced stems to a pattern. Produced roots are ordered according to their probabilities and compared against a list of 10,000 roots to verify their existence.

To compensate for mutation, vocalization, and gemination problems, two-letter stems were corrected by introducing new stems that were generated by doubling the last letter and by adding weak letters before or after the stem. As for stems with weak middle letter, new stems were introduced by substituting the middle letter with the other weak letters.

[2]$\sim$ 2 KB.

[3]Named Arabic orthographic templates by Darwish.

Probabilities of letter substitution and addition will be calculated to account for such changes. To account for particles, a list of Arabic particles was constructed. The system employs a letter normalization strategy in order to account for spelling variations and to ease in the deduction of roots from words.

The author performed three experiments to evaluate the stemmer. In the first and second experiments, Sebawai is trained on large and small lists of word-root pairs, respectively. After the training, a list of words is fed into Sebawai and ALPNET for analysis. The correctness of analysis and coverage of both systems are compared. Both experiments showed ALPNET failure and low accuracy of Sebawai. A quick review of the list shows a high frequency of named entities, misspelled words, and obscure words. In the third experiment, a document collection is indexed using roots produced by both systems. Retrieval effectiveness of indexing using roots produced from each system is examined. In this experiment, Sebawai significantly outperformed ALPNET.

The author explained that the coverage of the morphological analyzer is not perfect. Usually, the first two roots are correct. Beyond the first two roots, the chance that a root is correct is low, but possible. Lately, the analyzer was improved by backing off to light stemming when the analyzer failed, adding extra checks to insure the stop words were detected correctly, and weighting the roots. Sebawai cannot fully deal with named entities, one-letter words, and words constituting complete sentences. It also lacks the ability to identify which prefix-suffix combinations are legal. Sebawai is faster than ALPNET on the same machine. The method used to develop Sebawai can be used to develop analyzers for other similar languages.

The assumption that the root is better than stem for information retrieval is very questionable. Since this analyzer is built using ALPNET results, its performance will be greatly affected by the correctness of ALPNET analyzer. It will perform well on words that the ALPNET can analyze, but poorly on most other words (Darwish & Oard, 2002). Anyhow, advantages include testing the correctness against Xerox analyzer, checking function words, backing off to light stemming in case of failure, and trying to deal with mutation, vocalization, and gemination.

Buckwalter (2002) designed an Arabic morphological analyzer that is used for part-of-speech (POS) tagging Arabic text. The code for morphological analysis is implemented using Perl script.[4] It uses three Arabic-English lists, including prefixes with 299 entries, suffixes with 618 entries, and stems with 82,158 entries. The lists are supplemented by three morphological compatibility tables used for controlling prefix-stem combinations with 1,648 entries, stem-suffix combinations with 1,285 entries, and prefix-suffix combinations with 598 entries.

[4]About 17KB for the Perl code and about 4MB for used lists.

The analyzer performs the following functions: tokenization, word segmentation, dictionary lookup, compatibility check, analysis report, and second lookup (orthographic variants). Arabic words are segmented according to some developed rules. Given these rules, the input word is segmented into a finite number of 3-part segments: prefix, stem, and suffix. Dictionary lookup function consists of checking, for each segment, the existence of prefix, stem, and suffix. If all three components are found in their respective tables, the next step is to determine whether their respective morphological categories are compatible. For each of the three components, the compatibility is checked. If all three pairs are found in their respective tables, the three components are compatible and the word is valid, then the analysis will be reported. When a word returns no analysis, the orthography of the input string will be checked and a list of alternative spellings based on some hypotheses will be created.

Xu and Weischedel (2002) commented that this analyzer, although having quite a large list of stems, is still incomplete, and some Arabic orthographical errors were encountered in their experiments. This analyzer has the advantage of being built on a linguistic base. It is a free analyzer where the Linguistic Data Consortium is releasing it under the GNU General Public License. In addition to its large size and needed lists, the main disadvantages include that no success rate was seen in the literature.

Al-Jlayl and Frieder (2002) focuses on the improvement of Arabic information retrieval systems. They were motivated in developing a new stemmer to minimize the sense ambiguity associated with the root-based stemmers and to conflate the various semantically related words into the same conflation class. To resolve ambiguity, they proposed a light-stemming algorithm for Arabic text. This algorithm is not an aggressive as the root-based algorithm. The aim of this technique is not to produce the linguistic root of a given Arabic surface form; rather, it is to remove the most frequent suffixes and prefixes. They based their work on the hypothesis that developing a stemming algorithm that retains the word meaning improves the retrieval performance of an Arabic information retrieval system.

Al-Jlayl and Frieder empirically investigated the effectiveness of the retrieval without stemming. This approach degrades retrieval precision since Arabic is highly inflected language. After that, they tested the retrieval using the root-based stemmer proposed by Khoja and Garside (1999). They showed a statistically significant improvement over the retrieval without stemming. In addition, they modified the algorithm of Khoja to make it more suitable for information retrieval. Furthermore, they added one more pattern and modified the algorithm to deal with phenomena of gemination, mutation, and vocalization.

Later, Al-Jlayl and Frieder developed the light stemmer. This stemmer is mainly based on diacritics removal, normalization, and attempting to locate and strip out the most frequent prefixes and suffixes. The stemmer checks against a list of foreign words. If it is found, then the foreign word

is returned; otherwise, the technique proceeds in the stemming process.

Al-Jlayl and Frieder tested the effectiveness of the retrieval using TREC benchmark data in three cases: no stemming, root-based (modified Khoja stemmer), and their light stemmer with the elimination of function words. Their results showed that all stemmers significantly perform better than no stemming at all, and light stemmer significantly outperforms the root-based algorithm. They found an average 87.4% and 24.1% increase in average precision over no stemming retrieval and root-based retrieval, respectively. This improvement has been statistically proved. At the low recall levels, the difference between the light stemmer and the other approaches is even more noticeable.

Testing the performance of the stemmer in information retrieval and on the Web is the main advantage of this work. However, disadvantages include no clear distinction between prefixes and definite articles and too many errors originating from the similarity between start/end strings with affixes.

Larkey et al. (2002) developed four light stemmers based on heuristics for Arabic retrieval. In addition, they developed a statistical stemmer based on word cooccurrence. Developed light stemmers removed a small number of prefixes and suffixes while the cooccurrence-based statistical stemmer created large stem classes by vowel removal and then refined these classes using cooccurrence. The difference between their developed four stemmers is the number of removed prefixes and suffixes. In addition, the authors developed a few simple stemmers. The main goal was to develop a simple and effective Arabic stemmer without a considerable amount of linguistic knowledge.

In this work, authors evaluated their approaches against no stemming and a morphological analyzer developed by Khoja and Garside (1999) for monolingual and cross-language retrieval. Khoja's analyzer suffers from the problem of dealing with proper names, so their implementation included a list of country and major city names translated into Arabic. Such names are eliminated from further stemming. Morphological analyzer was evaluated with and without the list of proper names. Their results of comparison using the TREC-2001 data reveals that their light stemmer, which removes 6 prefixes, 10 suffixes, and function words, gives the best performance in the standard recall-precision graph followed by Khoja's analyzer. The results have been verified using the statistical significant test. The authors tested the effect of removing function words where results were statistically significant for some of their light stemmers, but not for all.

The best stemmer was very simple and did not try to find roots or take into account most of Arabic morphology. It is properly not essential for the stemmer to yield the correct forms, whether roots or stems. It is sufficient for it to group most of the forms that belongs together.

In addition to the monolingual information retrieval, the authors studied the cross-language English-Arabic retrieval. The major result was that their best light stemmer with function word removal outperforms Khoja stemmer.

Advantages of this work include the high effectiveness, the usage of test collections, and the evaluation of the cross-language retrieval. The main disadvantage, however, is that such developed stemmers are suitable mainly for information retrieval.

Abuleil et al. (2002) described and implemented a learning system that can analyze some types of Arabic nouns to produce their morphological information using a rule base that uses suffix analysis as well as pattern analysis. The system utilizes user feedback to classify the noun and identify the group that it belongs to.

The authors classified the nouns into 84 groups according to their patterns for singular, plural, masculine, and feminine. For each group, a function was implemented to find the morphological information for a given noun. Very few groups have a unique pattern for plural and singular, and most of them share the same pattern with other groups.

The system consist of the following modules: interface module, type-finder module, database module, noun morphology-analyzer module, suffix-analyzer module, pattern-generator module, database-checker module, and user-feedback module. It reads a noun from the text, isolates suffixes, and generates its pattern. It uses the classified noun table, the suffix/pattern analysis, or the user-feedback module to find the group to which the noun belongs and updates the database.

The main function of type-finder module is to find the part of speech of the word by running several tests. The database-checker module identifies any already classified noun or any noun derived from it. It includes a classified noun table that contains each root noun and the number of the group to which the noun belongs. The table is updated every time the system identifies a new noun.

The noun morphology analyzer module is the core of the system that calls different modules to analyze a given noun and updates the database. The noun is analyzed by removing its suffix, producing some lexical information and finding the pattern and its groups. If the system failed to identify the group, then it calls the user-feedback module to help in identifying a unique group.

The authors tested the system using 500 nouns from newspaper text. The system identified 90.2% of them, 7.1% by just analyzing the suffix and the pattern of the noun, 28.8% by using the database-checker module and the classified noun table, and 64.1% by using user-feedback module. The system failed on 9.8% of the tested nouns.

This work represents a good effort on the direction of deep understanding of the Arabic language. Building the work on a strong linguistic base is another advantage. However, disadvantages include the low language coverage and the need for user feedback to analyze 64% of input nouns.

Finite-state Automata and Two-Level Morphology Approaches

Historical view. In terms of conceptual appropriateness and computational efficiency, researchers who work in fi-

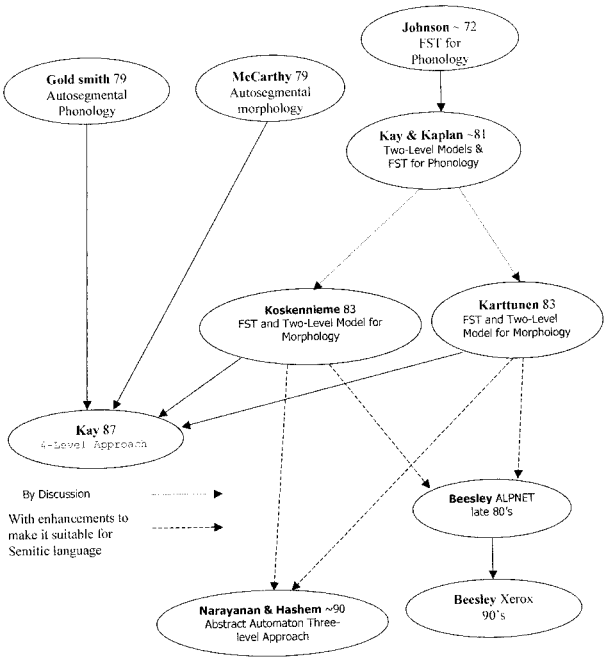

FIG. 9. Historical frame of two-level models and finite automata research activities.

nite-state and two-level morphology approaches believe that two-level models are more appropriate for processing morphologically rich inflectional languages (Narayanan & Hashem, 1994). Beesley (1991) indicated that cut-and-paste approaches suffer from difficulties during processing words that follow the pattern [افتعل AfvEl] because of the infix [ت v] Assimilation and weak roots are other challenges to these approaches. Beesley and others believe that such cases are easily handled by two-level systems using the phonological and orthographical rules.

Figure 9 shows the historical frame of research activities in the field of finite-state and two-level morphology. It illustrates the contributions and dependence of main researchers. This figure is the result of our review of different papers and their interrelations. Note that the number after the name represents the publication date.

Kay (1987) explains that finite-state and two-level morphology have been built over the concepts of autosegmental phonology proposed by Goldsmith and autosegmental morphology proposed by McCarthy.

McCarthy, as cited by Kay (1987), proposed in 1979 the auto-egmental theory for Arabic morphology. This work has been supported and defended by Haile and Mtenje, but Hudson raised serious theoretical challenges (Beesley, 1998). Beesley (1991) believes that it is not clear how McCarthy's Arabic morphology can be implemented to do practical morphological analysis or generation. Ali (1988) commented that the theory developed by McCarthy might be suitable for phonology, but that applying it to morphology might be very complicated. McCarthy updated his proposal so that moraic analysis can be applied to Arabic nouns and affixational analysis can be applied to Arabic verbs (Beesley, 1991, 1998; Kiraz, 1995).

Kay (1987) utilized his modified version of McCarthy's approach by proposing a four-level finite-state model as a solution to Arabic morphology, yielding a complicated and limited system (Ali, 1988). The system consists of a set of FSTs similar to Koskenniemi's set, but on four tapes rather than just two. Beesley (1991) commented that Kay's approach gave hope that the finite-state approach is computationally feasible and tractable. However, the practical problems of carrying out four-level system derivation and handling the nonconcatenative aspects of the Arabic language are still to be resolved.

The early stages of finite-state and two-level Arabic morphology were started by Koskenniemi and Karttunen in 1983. Koskenniemi (1983) developed a linguistic, computationally implemented model for morphological analysis and synthesis. The model is based on the lexicon and a set of parallel rules. Koskenniemi's two-level model is essentially a simplified version of the early Kay-Kaplan two-level models and FST for phonology (Karttunen, 1983). Karttunen (1983) developed a very similar model, in that the most important technical feature of both models is that rules are represented as finite-state transducers with one-to-one correspondence between the rules and the automata.

Koskenniemi's and similar systems were originally developed for the Finnish language and then applied to English, French, and other languages. Many drawbacks caused by implementation methods make generation less efficient than analysis (Beesley, 1996; Karttunen, Kaplan, & Zaenen, 1992). Some proposed improvements to resolve drawbacks include the use of rule compilers to automatically convert rules to finite-state transducers, mapping inflected forms of the same word to the same canonical dictionary form, and, finally, representing information and morphological categories at leaves of trees as part of the lexical form (Beesley, 1996; Karttunen, 1983).

ALPNET and Xerox projects. In non-Semitic languages, morphemes are concatenated with roots and stems, while in Semitic language inflectional patterns are not completely concatenative but interdigitate or intercalate (Narayanan & Hashem, 1993). Stems are generated through interdigitation of roots and patterns. The standard system of Arabic orthography perhaps represents the extreme of complexity because of the great irregularities between the lexical and surface strings caused by phenomena including weak roots, hamza orthography, and the zero realization of short vowels and gemination mark (Beesley, 1991). Even though Arabic morphology is an extreme test for any theory, a two-level model will be very suitable for the analysis of Arabic (Beesley, 1990). Kiraz (1995) believes that nobody has dealt computationally with the challenging problem of the Arabic broken plural. Broken plurals are not handled by most of current Arabic stemmers (Xu & Weischedel, 2002).

At the early stages, Ali (1988) mentioned that, even though two-level finite-state systems are capable of analysis and generation, they are more suitable for languages with concatenative morphology than for Semitic languages. This is true in that traditional two-level systems were not able to

deal with Semitic languages without some modifications. Some researchers, as will be shown soon, tried to adapt the two-level morphology and finite-state automata systems in order to deal with nonconcatenative languages.

In the ALPNET project, Beesley (1991) implemented the two-level theory whereby lexicons drive the search process, while rules act as filters. In traditional two-level systems, lexicons contain morphemes, and the target is to find the correct lexicon entries that comprise the correct word. This mechanism, without modification, is not suitable for Arabic. The ALPNET project is based on the traditional two-level implementation proposed by Karttunen (1983), but with two main adjustments to make the system suitable for Arabic. The two adjustments are "detouring" and "feature unification." Beesley proposed having two separate lexicons: one for patterns, the other for roots. The concept of detouring is that the system can be made capable of scanning and tracing the two lexicons simultaneously to find patterns and roots of investigated stems.

Arabic words could be built using affixes, roots, patterns, and vowelization, where each has a few meanings. Considering word parts independently gives a high degree of ambiguity. The concept of feature unification is that morphemes can be checked collectively, so that very many meanings can be discarded. Meanings of these parts might assist each other, reduce the meanings, or contradict each other (which will cause that path of analysis to be blocked).

For the ALPNET project, 112 rules have been developed. The lexicons included 4,940 roots and 408 patterns. On average, the system finds five morphological decomposition solutions for each input word, where many differ only in case ending. It is claimed that the analysis is robust, accurate, and almost distressingly thorough, finding many valid but highly unlikely solutions (Beesley, 1990).

Many researchers have investigated detouring, proposed by Beesley (1999). Narayanan and Hashem (1993) indicated that another approach for detouring might be preferred if it is more consistent with the two-level approaches. They also indicated that the implications of detouring for parallel evaluation are not clear. However, they stated that detouring has the advantage of forcing inflectional patterns to be kept together in the dictionary, rather than be split up. Narayanan and Hashem proposed an alternative approach for nonconcatenative languages, namely, an abstract automaton, three-level approach.

Later, Beesley has discovered that detouring operations in real time were inherently inefficient, since the resulting system was rather slow, analyzing about two words per second on a small IBM mainframe (Beesley, 1996). Beesley, Karttunen, and other researchers (Beesley, 1996; Karttunen et al., 1992) noticed that morphological information at leaves of trees could be encoded as symbols within strings; hence, lexicons could be compiled into finite-state transducers. Beesley (1996, 1998) reported that the ALPNET lexicon has been converted into the format of the lexicon compiler "Lexc," a language developed by Karttunen and Beesley in 1992, initially developed for concatenative languages. Karttunen and Beesley, however, noticed that interdigitation could be performed using the intersection process between roots and patterns. This process is supported by Xerox's finite-state calculus, and lexicons are written using the Lexc language and compiled into finite-state transducers. In this case, detouring is no longer needed, and the matching of roots and patterns is the responsibility of a rule compiler developed by Karttunen and Beesley. The compiler automatically compiles rules and intersects them into a single-rule, finite-state transducer. This will be combined with the lexicon's finite-state transducer to produce a single lexical transducer.

Beesley and others began the work of converting ALPNET to the Xerox project in 1995. In 1996, the Arabic system was produced and put on the Internet. In 1997, the rules were rewritten to support more reliable generation, and a Java user interface was added to allow users to interact with the system through the Internet in standard Arabic orthography. The Internet-based system accepts Arabic words and returns morphological analyses that identify affixes, roots, and patterns. Input words may include full or partial vowels, or none. The presence of vowels certainly reduces ambiguity. The system has wide dictionary coverage and is intended as a pedagogical dictionary-lookup aid, a comprehension-assistance tool, and a component in larger natural-language processing systems (Beesley, 1998). In the expandable lexicons, 4,930 roots have been manually encoded to identify the compatible patterns. On average, each root produces 18 stems. The total number of stems produced is 90,000; 20,000 have been cancelled for phonological reasons. The pattern lexicon contains 400 patterns. Other lexicons have been built for prefixes, suffixes, and stems that are not root based. The compilation of all these lexicons has produced 72,000,000 abstract words. Ambiguity at surface level causes the system to test too many paths and gives a set of analyses. This causes the Arabic system to be larger and slower than other systems. According to Beesley (1998), the Xerox system is still under development, but it will be continued and commercialized if motivation and interest are shown. The dictionary still lacks proper names and multiword expressions. Finally, the analyzer can be redesigned using some new finite-state algorithms, such as merge, that make the system more efficient in building stems using intersection operations (Karttunen, 1983; Karttunen et al., 1992). Later, after a few years' suspension, a new project was begun in April 2001 to test, update, and redevelop the Xerox Arabic system for commercial use (Beesley, 2001).

Pattern-Based Approach

Al-Kharashi and Al-Sughaiyer (2000, 2002a-e) mentioned that researchers proposing different morphological analysis techniques were seeking a high degree of accuracy. This caused proposed systems to be based on heavy computational processes or the use of large amounts of associated information. Consequently, they proposed a pattern-based algorithm that did not require either complex computations or associated tables. The proposed technique was

based on fast-surface morphological analysis of Arabic words.

The work started with a proposal of a framework to be used for testing and evaluating Arabic morphological and stemming techniques. A new Arabic stemmer was proposed whereby the generated data set was used to construct, test, and evaluate the stemmer. In this stemmer, a unique regular expression-based rule was generated for groups of similar Arabic words. Rules were used to describe the internal morphological structure of Arabic words and guide the decomposition process of a given word to its basic units, i.e., stem, prefix, and suffix. A very simple rule parser was developed to perform the analysis and to process and extract the morphological components of words. The parser is used to perform matching between input rule and a given Arabic word.[5]

A few experiments have been conducted, including the study of rule growth in a natural text, the time needed to analyze collections of words, firing policies, accuracy, and rule merging. An Arabic data set has been used to test the accuracy of the stemmer. Straightforward experimentation showed 80% accuracy in identifying stems.

Al-Kharashi and Al-Sughaiyer concluded their work as follows: Reducing the number of rules and increasing language coverage while maintaining the same level of performance and functionality can achieve enhancement. Merging rules is one method that can be used for enhancing the pattern-based stemmer. While keeping time efficiency almost the same, merging rules reduces the rule list dramatically. Merged rules also produce a better performance. The concept of merging opens up the possibility of increasing the coverage of the system by expanding most of the existing rules. Further improvement can be achieved by introducing better clustering mechanisms such as cascading and back referencing.

The accuracy of the stemmer can be improved by testing different approaches for firing policies. Selecting the best approach is a trade-off between performance and cost. Taking the generalization issue into consideration leads to the selection of the triple approach among others proposed. Firing policies should be thoroughly studied to improve the accuracy of the proposed system. Although it is impractical to achieve an ideal state, it is possible to have a certain ordering of rules that produces the best performance for such a rule set. Furthermore, introducing more rules should increase the scope of the system.

Although linguistically not 100% accurate, the proposed stemmer is believed to be accurate enough for general-purpose applications such as information retrieval systems. The argument that information retrieval systems may not need a high accuracy has been proved by some researchers for Arabic and other languages. For example, Larkey et al. concluded that it is properly not essential for the Arabic stemmer to produce the correct root or stem forms. It is adequate for it to conflate most of the forms that belongs together (Larkey et al., 2002). Also, Kalamboukis (1995) concluded that his tests on the Modern Greek language show that a simple stemming process is very effective in information retrieval.

Related Works

Many related works have been conducted to utilize morphological analysis techniques, to enhance natural language processing applications, or to assist and support such techniques. This section surveys and comments on some such works.

Youssef (1987) demonstrated a linguistic system that can be used as a base for morphological and syntactical analyzers. Adi and Ewell (1987) introduced a new discovery that has its origins in deep morphological analysis as well as introspection and philosophical thought. Without the availability of more information about these two references, it is very hard to judge them.

El-Dessouki et al. (1988, 1989) explained the implementation of an expert system to analyze and understand Arabic text. The system consists of three subsystems, namely, lexical, syntactic, and semantic analyzers. The lexical analyzer analyzes the words according to the morphological patterns of the Arabic language, the syntax analyzer analyzes the complete sentence according to its grammatical attributes, and the semantic analyzer verifies the meaning of the whole sentence. The lexical analyzer is based on the process of matching patterns against well-known morphological Arabic patterns in order to determine the root of the word. The purpose of the lexical analyzer is to assist in other stages such as syntactical and semantical analyses. It is constructed using a rule-based technique and uses database files to extract affixes. Although some sample rules were given, the main disadvantage of this work is that it has been introduced without any detailed explanations or experimental evaluations.

El-Sadany and Hashish (1988, 1989) proposed a semi-automatic vowelization system based on the morphological analyzer proposed by Gheith and El-Sadany (1987). The system covers the whole Arabic language dictionary and is capable of analyzing and generating words with different cases of vowelization. A lexicon with about 5,700 roots was used, assisted by a minimal hashing function and indexing technique to increase efficiency of access. The analyzer consists of a series of five modules. The first and second remove vowels, suffixes, and prefixes to produce the stem, extract possible roots from the stem using pattern matching, and, finally, compare extracted roots using the root lexicon. The output of this module is a set of roots with some morphological characteristics. In the third module, the stems are reconstructed from the extracted roots, and then suffixes and prefixes are added to produce words. This module is built using a rule-based system to produce a set of vowelized words that are not in their final orthographical shape. Module four, the morphophonemic and morphographemic module, puts the words generated into their final

[5]Parser size = 4.5KB, rule list size = 14KB.

TABLE 5. Summary of related work.

Author(s) and date	Algorithm name and type	Implementation	Test data	Success rate	Language coverage	Required lists	Presentation
El-Dessouki et al. (1988, 1989)	Linguistic	NA	NA	NA	NA	Affixes, patterns	3
El sadany & Hashish (1988, 1989)	Linguistic	PC, Logic programming language	NA	NA	Vowelized, non-vowelized, partially vowelized	Roots, affixes	5
Al-Atram (1990)	Linguistic	NA	NA	NA	NA	NA	6
Ben-Hamadou (1992)	Linguistic	Unix, C language	NA	NA	NA	Affixes, roots	6
Mahgoub & Abdelazim (1992)	Linguistic	PC, Mainframe, OS/2	NA	NA	NA	Stems, patterns	6
Ouersighni (2001)	Linguistic	SAMIA model	DIINAR.1	NA	Non-vowelized	Stem, affixes	7

forms. The fifth module compares words produced with those analyzed to select a smaller set of correct analyses.

El-Sadany mentioned that this system is very fast in processing verbs but very slow in processing nouns. Owing to the use of artificial patterns or computational balances, the system might not be very efficient for teaching the Arabic morphology. The author suggested the removal of artificial patterns for nouns and using a rule-based approach instead. In this work, neither experimental evaluations nor success rates were given.

Al-Atram (1990) studied the effectiveness of natural language in indexing and retrieving Arabic documents. In this study, he sketched an algorithm for stripping prefixes and suffixes from a given word. He also mentioned that a unified algorithm had been created and tested, with good results. He believes that this algorithm still needs more investigations. Since it was not the purpose of his study to develop a morphological analyzer, Al-Atram did not discuss other important issues, such as infixes, mutation, and vocalization. Furthermore, Al-Atram did not give the success rate of the proposed algorithm in stripping affixes.

Ben-Hamadou (1992) proposed a recovery strategy for lexical errors using an affixal analysis of the affected words. This strategy restricts the correction process to the morpheme level and suggests a short list of correction scenarios. The basic steps of the strategy include identifying affixes and roots, identifying those with a higher possibility of error, and filtering them using the congruence test that checks the compatibility of affixes. Finally, the words are filtered again, using a single-error test based on some statistical adjacency studies. In this technique, there is most likely to be an error in only one of the affixes of the tested word. In the filtration process, correction possibilities or scenarios will be selected. Ben-Hamadou showed that on average it takes 1.4 seconds of CPU time to analyze a given word. No success rate was given as to what proportion of erroneous words was detected and corrected.

Mahgoub and Abdelazim (1992) proposed a spelling verification and assistance system. This system analyzes a given Arabic word and checks the consistency between different morphemes of the word, including Hamza shapes and affixation compatibility. It uses a lexicon of about 28,000 Arabic stems, and consists of two main parts: spelling verification and spelling aid. The first part isolates affixes and assures their compatibility. The pattern of the remaining part of the word is compared against a list of patterns. Each entry in the list contains the pattern, together with the stem pattern and its compatibility with affixes. If compatible, the pattern will be accepted; otherwise, other patterns should be tested. Finally, the resulting stem is compared against a list of stems to check its compatibility with affixes. If compatible, the word is accepted as a correctly spelled word; otherwise, it has to be corrected by the spelling aid. The stem lexicon is based on the matrix representation of Arabic inflected forms built by El-Sadany and Hashish (1989). The second part is the spelling aid, which the authors explained and provided some examples of. They concluded that the system has been thoroughly tested, and explained that it provides a fast and efficient spell checking. The system processes 1,400 words a minute. The authors, however, gave neither success rates nor further explanation of how to deal with hamza, mutation, vocalization, and gemination marks. Furthermore, definitions of some basic terms are somehow not clear. The system requires very large lists that include stems, roots, and patterns with their affix compatibility.

Ouersighni (2001) presented an algorithm as part of the "AraParse" Arabic morphosyntactic analyzer developed under the DIINAR-MBC project. The algorithm, which is an extended version of a spell checker, considers written words as a suite of morphemes and decomposes them into stems and affixes. It also allocates these compositions to their grammatical categories and associated features. The algorithm accesses a lexicon of stems and affixes while decomposing the tested word. This reference is not enough to allow an assessment of this algorithm. Neither its success rate nor the use of test data or collection has been reported. A large list with much linguistic effort seems necessary in the use of this algorithm. As a conclusion, Table 5 gives a summary of most of the related algorithms.

Conclusion

Most of the surveyed Arabic morphological analysis techniques are mainly linguistically based. Most researchers provided general descriptions of their approaches with almost no measures of effectiveness or efficiency and pro-

vided little in-depth survey of any of the available techniques. As far as we know, only very few stemmers are available for the testing and evaluation needed for comparison studies. Such stemmers may include Khoja, Darwish, Buckwalter, and the systems developed at Xerox and SAKHR laboratories (Sakhr, 2000). Even though the SAKHR morphological analyzer is implemented inside some of their software, this system is not available as a stand-alone product; hence, it cannot be evaluated. The following points may be noted as a summary:

At many levels, there are no standards. There are none for basic Arabic linguistic terms and their definitions, none for terms and their translation into English, and none for test collections and performance evaluations.

Many old research efforts have been weak in the technical and editorial presentation.

Very few researchers have acknowledged the efforts of others, and fewer still compared their work with that of others.

Most researchers did not use systematic and scientific procedures in evaluating their algorithms.

Except for SAKHR, Xerox, and IBM, the efforts were not supported commercially, and many researchers did not cooperate with others in constructing algorithms and evaluating them. As far as we know, there are very few analyzers available in the public domain.

At present, designing an Arabic morphological analyzer suitable for all applications might not be possible. Instead, analyzers should be application oriented. Furthermore, some applications tend to require a complete and comprehensive analysis of words, while others do not.

Xerox, SAKHR, and IBM projects are purely commercial works, which means source codes are not available for research activities.

Writing rules and lexicons in Roman rather than Arabic characters may not be advisable, especially in the long term. The reason is that there is no universally accepted standard for representing Arabic characters and symbols in English.

Some areas are open for investigation, and future developments can be envisaged. Some important issues can be summarized as follows:

Developing standards for linguistic terms, their definitions and translations, comprehensive function lists, and test collections

Developing application-oriented analyzers

Spending more effort on collecting analyzers and finding criteria for their testing and evaluation

Some linguistic issues need to be investigated; for example, selecting a unique definition for an Arabic word or accepting or rejecting Arabic computational balances

Developing some analyzers and making them open to research activities

Some straightforward algorithms may soon be developed that might be an effective solution for many applications

Integrating morphological and higher level analyzers seems to be necessary, especially when seeking a high degree of accuracy

Using nonlinguistic–based approaches to assist linguistic approaches

Building and disseminating morphological databases to accelerate the development process in linguistic-based approaches

Establishing special interest groups in Arabic computational linguistics to oversee efforts in the field and promote and establish links between researchers

Establishing a framework for evaluating and judging work on morphological analyzers

References

Abuleil, S., Alsamara, K., & Evens, M. (2002, July). Acquisition system for Arabic noun morphology. In Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages. Association for Computational Linguistics 40th Anniversary Meeting (pp. 19–26). University of Pennsylvania, Philadelphia.

Abu-Salem, H., Al-omari, M., & Evens, M. (1999). Stemming methodologies over individual query words for an Arabic information retrieval system. Journal of the American Society for Information Science, 50(6), 524–529.

Adamson, G., & Boreham, J. (1974). The use of an association measure based on character structure to identify semantically related words and document titles. Information Storage and Retrieval, 10, 253–260.

Adi, T., & Ewell, O.K. (1987). Letter semantics in arabic morphology: A discovery about human languages (pp. 21–52). Arabic Morphology Workshop. Stanford, CA: Linguistic Institute, Stanford University.

Ahmad, F., Yusoff, M., & Sembok, T.M.T. (1996). Experiments with a stemming algorithm for malay words. Journal of the American Society for Information Sciences, 47, 909–918.

Al-Atram, M.A. (1990). Effectiveness of natural language in indexing and retrieving arabic documents [in Arabic] (King Abdulaziz City for Science and Technology Project number AR-8-47). Riyadh, Saudi Arabia.

Al-Bawab, M., & Al-Tayyan, M. (1998). Computerized processing of Arabic morphology [in Arabic]. Arabian Magazine of Sciences, 32, 6–13, Arab League Educational, Cultural and Scientific Organization, ALECSO.

Al-Bawab, M., Mrayati, M., Alam, Y.M., & Al-Tayyan, M.H. (1994). A computerized morpho-syntactic system of Arabic. The Arabian Journal of Science and Engineering, 19, 461–480. Published by KFUPM, Dhahran, Saudi Arabia.

Al-Fedaghi, S.S., & Al-Anzi, F.S. (1989). A new algorithm to generate root-pattern forms. In Proceedings of the 11th National Computer Conference (pp. 391–400). Published by KFUPM, Dhahran, Saudi Arabia.

Al-Hamalawee, A. (2000). Shatha alarf fe fn alsarf. Dar Alkotob Al-ilmiyah [in Arabic]. Beirut, Lebanon.

Ali, N. (1988). Arabic language and computer [In Arabic]. Ta'reeb.

Ali, N. (1992). Parsing and automatic diacritization of written Arabic: A breakthrough. In Proceedings of the 13th National Computer Conference (pp. 794–812). Riyadh, Saudi Arabia: King Abdulaziz City for Science & Technology.

Al-Jlayl, M., & Frieder, O. (2002). On Arabic search: Improving the retrieval effectiveness via light stemming approach. In Proceedings of the 11th ACM International Conference on Information and Knowledge Management, Illinois Institute of Technology (pp. 340–347). New York: ACM Press.

Al-Kharashi, I.A. (1991). MICRO-AIRS: A microcomputer-based Arabic information retrieval system comparing words, stems, and roots as index terms. Doctoral dissertation, Illinois Institute of Technology, Illinois.

Al-Kharashi, I.A., & Al-Sughaiyer, I.A. (2002a). Data set for designing and testing an Arabic stemmer. In Proceedings of Arabic Language Resources and Evaluation: Status and Prospects, Spain. Organized by ELRA-European Language Resources Association.

Al-Kharashi, I.A., & Al-Sughaiyer, I.A. (2002b). Rule merging in a rule-based Arabic stemmer. In Proceedings of the 19th International Confer-

ence on Computational Linguistics (COLING-02), Volume 1 (pp. 432–438), Taipei, Taiwan.

Al-Kharashi, I.A., & Al-Sughaiyer, I.A. (2002e). Pattern-based Arabic stemmer. In Proceedings of the 2nd Saudi Technical Conference and Exhibition (STCEX2002), Volume II (pp. 238–244), Riyadh, Saudi Arabia.

Al-Khuli, M. (1991). A dictionary of theoretical linguistics: English-Arabic with an Arabic-English glossary. Published by Library of Lebanon.

Al-Saeedi, M. (1999). Awdah Almasalik ila Alfiyat Ibn Malek. Published by Dar ihyaa al oloom [In Arabic]. Beruit, Saudi Arabia.

Al-Shalabi, R. (1996). Design and implementation of an Arabic morphological system to support natural language processing. Doctoral dissertation, Illinois Institute of Technology, Chicago.

Al-Sughaiyer, I.A., & Al-Kharashi, I.A. (2000, March). An efficient Arabic morphological analysis technique for information retrieval systems. In ACIDCA'2000 International Conference. Monastir, Tunisia.

Al-Sughaiyer, I.A., & Al-Kharashi, I.A. (2002c, September). Rule parser for Arabic stemmer. In Proceedings of the Text, Speech and Dialogue 5th International Conference. TSD2002 (pp. 11–18), Brno, Czech Republic.

Al-Sughaiyer, I.A., & Al-Kharashi, I.A. (2002d). Firing policies for an Arabic rule-based stemmer. In Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002) (pp. 98–103), Lisbon, Portugal.

Al-Suwaynea, A.S. (1994). Information retrieval in Arabic language [In Arabic]. Riyadh, Saudi Arabia: Published by King Fahad National Library.

Al-Uthman, A. (1990). A morphological analyzer for Arabic. Master's thesis, KFUPM, Dhahran, Saudi Arabia.

Aref, M.M. (1997). Object-oriented approach for morphological analysis. In Proceedings of the 15th National Computer Conference (pp. 5–11). Published by KFUPM, Dhahran, Saudi Arabia.

Beesley, K.R. (1990, September). Finite-state descriptions of Arabic morphology. In Proceedings of the 2nd Cambridge Conference on Bilingual Computing in Arabic and English. Cambridge University, UK: Literary and Linguistic Computing Centre.

Beesley, K.R. (1991). Computer analysis of Arabic morphology: A two-level approach with detours. In B. Comrie & M. Eid (Eds.), Perspectives on Arabic Linguistics, III: Papers from the 3rd Annual Symposium on Arabic Linguistics, Volume 12 (pp. 155–172). Benjamins, Amsterdam.

Beesley, K.R. (1996). Arabic finite-state morphological analysis and generation. In Proceedings of the 16th International Conference on Computational Linguistics (COLING-96), Volume 1 (pp. 89–94), University of Copenhagen, Denmark.

Beesley, K.R. (1998). Arabic morphological analysis on the Internet. In Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing, Cambridge, UK.

Beesley, K.R. (2001, July). Finite-state morphological analysis and generation of Arabic at Xerox Research: Status and plans in 2001. In Proceedings of the Arabic Language Processing: Status and Prospect—39th Annual Meeting of the Association for Computational Linguistics (pp. 1–8), Toulouse, France.

Ben-Hamadou, A. (1992). An affixational treatment of the errors in Arabic natural language dialogue between man and machine. In Proceedings of the 13th National Computer Conference. (pp. 830–842). Published by King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia.

Bokhaddood, A. (1994). Morphological introductory: Application and training on Arabic morphology. Published by Academic establishment for studies, publication and distribution [In Arabic]. Beirut, Lebanon.

Buckwalter, T. (2002). Buckwalter Arabic Morphological Analyzer Version 1.0, Linguistic Data Consortium (LDC) catalog number LDC2002L49 and ISBN 1-58563-257-0.

Darwish, K. (2002a). Building a shallow Arabic morphological analyzer in one day. In Proceedings of the Association for Computational Linguistics (ACL-02), 40th Anniversary Meeting (pp. 47–54), University of Pennsylvania, Philadelphia.

Darwish, K. (2002b). Al-stem: A light Arabic stemmer [Online]. Available: http://www.glue.umd.edu/~kareem/research.

Darwish, K., & Oard, D.W. (2002). Term selection for searching printed Arabic. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002) (pp. 261–268), Tampere, Finland.

Dawson, J.L. (1974). Suffix removal and word conflation. ALLC (Association for Literacy &Linguistic Computing) Bulletin, 2(3), 33–46.

De Roeck, A.N., & Al-Fares, W. (2000). A morphologically sensitive clustering algorithm for identifying Arabic roots. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000), Hong Kong, China.

El-Affendi, M.A. (1991). An algebraic algorithm for Arabic morphological analysis. The Arabian Journal for Science and Engineering, 16(4B), 605–611. Published by KFUPM, Dhahran, Saudi Arabia.

El-Affendi, M.A. (1992). Arabic dictionary compression using an invertible integer transformation and a bitmap representation. Journal of King Saud University, Engineering Sciences, 4(1), 105–125. Riyadh, Saudi Arabia.

El-Affendi, M.A. (1998). Performing Arabic morphological search on the Internet: A sliding window approximate matching (SWAM) algorithm and its performance. (Personal contact). King Saud University, Riyadh, Saudi Arabia.

El-Affendi, M.A. (1999). Building an Arabic distributed collaboration environment. The final report. Research project: AR-16-094, Supported by King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia.

El-Dahdah, A. (1988). Dictionary of terms of declension and structure in universal Arabic grammar: Arabic-English, English-Arabic. Beirut, Lebanon: Librairie Du Liban.

El-Dessouki, A., Nazif, A., El-Dessouki, O., & Ahmed, M. (1988). An expert system for understanding Arabic sentences. In Proceedings of the 10th National Computer Conference (pp. 745–759). Jeddah, Saudi Arabia: King Abdulaziz University.

El-Dessouki, A., El-Dessouki, O., Nazif, A., & Ahmed, M. (1989). An ATN approach for understanding Arabic sentence. In Proceedings of the 11th National Computer Conference and Exhibition (pp. 762–773), Dhahran, Saudi Arabia.

El-Sadany, T.A., & Hashish, M.A. (1988). Semi-automatic vowelization of Arabic verbs. In Proceedings of the 10th National Computer Conference. Jeddah, Saudi Arabia: King Abdulaziz University.

El-Sadany, T.A., & Hashish, M.A. (1989). An Arabic morphological system. IBM Systems Journal, 28(4), 600–612.

Foxley, E., & Feddag, A. (1990). A syntactic and morphological analyser for Arabic words. In Proceedings of the 2nd International Conference on Bi-lingual Computing, Cambridge University, UK.

Frakes, W.B., & Baeza-Yates, R. (Eds.). (1992). Information retrieval: Data structures & algorithms. Upper Saddle River, NJ: Prentice Hall.

Gheith, M., & El-Sadany, T. (1987, April). Arabic morphological analyzer on a personal computer. In Proceedings of the 1st King Saud University (KSU) Symposium on Computer Arabization (pp. 55–65), Riyadh, Saudi Arabia.

Gheith, M., & Mashour, M. (1985). A computer based system for understanding Arabic language. Computer processing of the Arabic language, Workshop Papers, Volume I, Kuwait.

Hafer, M., & Weiss, S.F. (1974). Word segmentation by letter successor varieties. Information Storage and Retrieval, 10, 371–385.

Harman, D. (1991). How effective is suffixing? Journal of American Society for Information Science, 42, 7–15.

Hegazi, N.H., & El-Sharkawi, A.A. (1985). An approach to a computerized lexical analyzer for natural Arabic text. In Proceedings of the Arabic Language Conference, Kuwait.

Hegazi, N.H., & El-Sharkawi, A.A. (1986). Natural Arabic language processing. In Proceedings of the National Computer Conference, Volume 2 (pp. 10-5-1–10-5-17), Riyadh, Saudi Arabia.

Hlal, Y. (1985). Morphological analysis of Arabic speech. Computer Processing of the Arabic Language, Workshop Papers [In Arabic], Volume I. Kuwait.

Hlal, Y. (1987). Information systems and Arabic: The use of Arabic in information systems. In Proceedings of the Arab School of Science and

Technology. Applied Arabic Linguistics and signal and informatics (pp. 191–197).
Hlal, Y. (1989). Automatic processing of Arabic with applications [In Arabic]. In Proceedings of the 1st Kuwaiti Computer Conference (pp. 145–171). Kuwait.
Hlal, Y. (1990). Morphology and syntax of the Arabic language. In Proceedings of the Arab School of Science and Technology Applied Arabic Linguistics for Informatics (pp. 201).
Hmeidi, I., Kanaan, K., & Evens, M. (1997). Design and implementation of automatic indexing for information retrieval with Arabic documents. Journal of the American Society for Information Science, 48(10), 867–881.
Hopcroft, J.E., & Ullman, J.D. (1979). Introduction to automata theory, languages, and computation. Reading, MA: Addison-Wesley.
Hull, D.A. (1996). Stemming algorithms: A case study for detailed evaluation. Journal of the American Society for Information Science, 47, 70–84.
Hull, D.A., & Grefenstette, G. (1996). A detailed analysis of English stemming algorithms. Rank Xerox Research Centre.
Kalamboukis, T.Z. (1995). Suffix stripping with modern Greek. Program, 29(3), 313–321.
Karttunen, L. (1983). KIMMO: A general morphological processor. Texas Linguistic Forum, 22, 165–186.
Karttunen, L. (1994). Constructing lexical transducers. In Proceedings of COLING-94. Kyoto, Japan.
Karttunen, L., Kaplan, R.M., & Zaenen, A. (1992). Two level morphology with composition. In Proceedings of the International Conference on Computational Linguistics (COLING-92), Volume 1 (pp. 141–148). Nantes, France.
Kay, M. (1987). Nonconcatenative finite-state morphology. In Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics (pp. 2–10). Copenhagen, Denmark.
Khoja, S., & Garside, R. (1999). Stemming Arabic text. Computing Department, Lancaster University, UK. Available: http://www.comp.lancs.ac.uk/computing/users/khoja/index.htm
Kiraz, G.A. (1995). Computational analysis of Arabic morphology. Computer Laboratory, University of Cambridge (St. John's College). E-mail: George.kiraz@cl.cam.ac.uk
Koskenniemi, K. (1983). Two-level model for morphological analysis. In Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83) (pp. 683–685). Karlsruhe, Germany.
Krovetz, R. (1993, July). Viewing morphology as an inference process. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 191–202). New York: ACM.
Larkey, L.S., Ballesteros, L., & Connell, M.E. (2002). Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002) (pp. 275–282). Tampere, Finland.
Larkey, L.S., & Connell, M.E. (2001). Arabic information retrieval at UMass in TREC-10. In Proceedings of the 10th Text Retrieval Conference (TREC2001) (pp. 562–570). Gaithersburg, Maryland.
Lennon, M., Pierce, D.S., Tarry, B.D., & Willett, P. (1981). An evaluation of some conflation algorithms for information retrieval. Journal of Information Science, 3, 177–183.
Lovins, J.B. (1968). Development of a stemming algorithm. Mechanical Translation and Computational Linguistics, 11, 22–31.
Mahgoub, H., & Abdelazim, H. (1992). Arabic spelling verification and assistance. In Conference of Arabic Language Usage in Information Technology. Riyadh, Saudi Arabia: King Abdulaziz Public Library.
Metri, G., & George, H. (1990). Al Khaleel. A dictionary of Arabic syntax terms. Beirut: Library of Lebanon.
Narayanan, A., & Hashem, L. (1993). On abstract, finite-state morphology. In Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (pp. 297–304). Netherlands.
Narayanan, A., & Hashem, L. (1994). Finite-state abstractions on Arabic morphology. Artificial Intelligence Review, 7, 373–399.
Ouersighni, R. (2001, July). A major offshoot of the DIINAR-MBC project: AraParse, a morpho-syntactic analyzer for unvowelled Arabic texts. In the Arabic NLP Workshop at ACL/EACL 2001, Toulouse, France.
Paice, C.D. (1990). Another stemmer. SIGIR Forum, 24, 56–61.
Paice, C.D. (1994). An evaluation method for stemming algorithms. In W.B. Croft & C.J. van Rijsbergen (Eds.), Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (pp. 42–50). London: Springer-Verlag.
Paice, C.D. (1996). A method for the evaluation of stemming algorithms based on error counting. Journal of the American Society for Information Science, 47, 632–649.
Popovic, M., & Willet, P. (1992). The effectiveness of stemming for natural-language access to slovene textual data. Journal of the American Society for Information Science, 43, 384–390.
Porter, M.F. (1980). An algorithm for suffix stripping. Program, 14, 130–137.
Sakhr Software. Multi-mode morphological processor [Online]. Available: http://www.sakhr.com/Sakhr_e/technology/Morphology.htm?Index=5&Main=Technology&Sub=Morphology
Saliba, B., & Al-Dannan, A. (1989, March). Automatic morphological analysis of Arabic: A study of content word analysis. In Proceedings of the 1st Kuwait Computer Conference, Kuwait.
Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of information by computer. Reading, Massachusetts: Addison-Wesley.
Savoy, J. (1999). A stemming procedure and stop word list for general French corpora. Journal of the American Society for Information Science, 50, 944–952.
Spencer, A. (1991). Morphological theory. Oxford: Basil Blackwell.
Thalouth, B., & Al-Dannan, A. (1987). A comprehensive Arabic morphological analyzer/generator. IBM Kuwait Scientific Center. Kuwait.
Xu, J., & Croft, W.B. (1998). Corpus-based stemming using co-occurrence of word variants. ACM Transactions on Information Systems, 16(1), 61–81.
Xu, J., Fraser, A., & Weischedel, R.M. (2002). Empirical studies in strategies for Arabic retrieval. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002) (pp. 269–274). Tampere, Finland.
Youssef, C. (1987). Production system of morphological and syntactical analyzers: Application to the Arabic language. In Proceedings of the 1st KSU Symposium on Computer Arabization (pp. 45–54). Riyadh, Saudi Arabia.