ISSN 2007-9737

A Graph-based Word Segmentation Algorithm for Dialectal Arabic: Libyan Dialect as a Case Study

Husien Alhammi[1,*], Kais Haddar[2]

[1] University of Sfax ,
Faculty of Economics and Management,
Tunisia

[2] University of Sfax,
Laboratory MIRACL,
Tunisia

h1974hami@gmail.com, kais.haddar@fss.usf.tn

Abstract. Arabic language and its dialects both have a very rich and complex morphology, and they face the same challenge, which is called agglutination, where the words might be attached to one or more affixes. However, word segmentation has become a very important preprocessing procedure for many natural language processing tasks that deal with agglutinative languages to improve their performance. Besides Arabic, Arabic dialects are known for their complex agglutination system, which makes word segmentation challenging. To address this challenge, this paper presents an out-of-context full word segmentation algorithm that is based on weighted directed graph theory. The main purpose of this algorithm is to tackle the agglutination phenomena observed in dialectal Arabic. To illustrate the efficacy of the algorithm, the Libyan dialect is selected as a case study for testing its feasibility. A test dataset of 1,200 Libyan dialect words was used to manually evaluate the algorithm for accuracy. The experimental results show that the proposed algorithm achieves good outcomes on the test dataset.

Keywords. Libyan dialect, word segmentation algorithm, morphological segmentation, weighted directed graph, Arabic dialects.

1 Introduction

Arabic dialects are now widely used as informal languages in everyday life, and they have become the preferred language for Arabic users to communicate with each other on social media networks. In contrast, Modern Standard Arabic (MSA) is used as a formal language in the media, newspapers, education, and so on [1, 12].

However, compared to MSA, Arabic dialects are considered low-resourced languages. Researchers in Arabic Natural Language Processing (NLP) do not take them into account. Very little work has been done on Arabic dialects in the last decade, although the situation is changing.

The Arabic dialects are distinct from MSA and also differ from each other [6]. In fact, there are considerable lexical and morphological variations among Arabic dialects. Obviously, Maghrebi dialects are different from Middle-east dialects. For example, dual forms are generally generated in Maghrebi dialects, where the word "زَوج" in Libyan Dialect (LD) or "جُوج" in Moroccan Dialect (MD), which means "pair" or "two" in English, is placed before the nouns. For instance, the dual form of the singular noun "دِرهَم", which means "cent" in English, is generated by placing the word "زَوج" before its plural form "دِرَاهِم", which translates to "cents" in English.

Consequently, the resulting phrase "زَوج دِرَاهِم" represents the dual form of the singular noun "دِرهَم" and conveys the meaning of "two cents"

in English. In contrast, the suffix "ين", quite commonly used in Levantine dialects, is attached to the endings of singular nouns to create dual forms. For example, the suffix "ين" can be attached to the singular noun "دِرهَم" to generate its dual form "دَرهَمَين", which also means "two cents" in English. Additionally, vocabulary differs considerably among Arabic dialects.

For example, the Arabic dialect equivalents of the English word "bottle" can be "قَرعَه", "دبوزة", "شيشة", "بطل", "إِزازَة" and "دبة" in Morocco, Tunisia, Libya, Iraq, Egypt, and the UAE, respectively. In terms of morphology, Arabic dialect words can be generated by adding different affixes to the base word, depending on where they are spoken. For example, the future tense of the Arabic verb "كتب", which means "to write" in English, can be formed by adding the prefixes "هت" or "بت" to the verb. This gives the words "هتكتب" and "بتكتب", respectively, both of which mean "he will write" in English. The prefix "هت" is commonly used in the Egyptian Dialect (ED), while "بت" is widely used in the LD. The LD is one of the Arabic dialects that belongs to the Maghrebi dialect family.

LD is spoken by about six million people in north Africa. Generally, the LD can be divided by linguists into three main dialects, which are the eastern, southern, and western dialects. LD is influenced by other languages.

Therefore, a sentence in LD might contain words whose original roots come from other languages, such as Italian, Turkish, Berber, and Arabic. Actually, LD is an extremely agglutinative language, and it tends to have a high rate of affixes.

A single word in LD might be the equivalent of a whole sentence in other languages. For example, the LD word "منقَالهَالك" can be translated into a sentence in both English and MSA, which are "who told you that" and "من الذي قَال لك ذلك" respectively. Word segmentation, also known as morphological segmentation, is a crucial preprocessing step in various NLP tasks, including machine translation, information retrieval, parsing, and speech recognition [25]. The process of word segmentation involves dividing a word into linguistic units known as morphemes, which carry meaning. In other words, it is the process of separating a word into a base word and affixes.

For instance, the LD word "وبيديروهَالك" represents a case of agglutination that means "And they will do it to you" in English. It can also be segmented into many linguistic units, which are ("و"+"ب"++"ي"++"دير"+"و"+"هَا"+"ل"+"ك"). Where "دير" is a base word, "و" and "ب" are proclitics, "ي" is a prefix, "و" is a suffix, as well as "ك", "ل", and "هَا" are enclitics. It is clear that a high agglutination ratio makes word segmentation incredibly difficult.

In situations where languages are both low-resource and agglutinative, word segmentation becomes even more exacerbated. In general, low-resource languages face a shortage of electronically available data, such as training data, corpora, lexicons, and dictionaries. This data is required for machine learning and other NLP tasks. Arabic dialects, in particular, are considered low-resource due to limited resources, which makes research on dialect word segmentation challenging. To overcome this limitation, an algorithm is introduced to solve the agglutination phenomenon by leveraging the advantages of Weighted Directed Graph (WDG) techniques.

The main contribution of the paper lies in the application of the WDG model to tackle the agglutination problem in LD, which has not been previously investigated. Additionally, the approach presented in this paper provides an alternative way to overcome the scarcity of available training data or corpora. This approach is particularly advantageous for low-resource languages because it does not require additional data resources, such as training data.

The idea of the proposed algorithm is designed not only for LD but also to be applicable to all Arabic dialects because they share a common morphological structure. LD was used as a case study to identify the strengths and limitations of the algorithm. The reason for choosing LD as a case study is that it is the native language of the first author. This means that the author has

a deep understanding of the LD. This familiarity with the LD allowed the authors to easily collect data and conduct a linguistic study. In this script, it is important to note that the term "prefix" refers to any component that might be attached to the beginnings of the base words, while the term "suffix" refers to any component that can be attached to the endings of the base words. Furthermore, the term "word segmentation" refers to both morphological segmentation and compound word splitting.

2 Related Work

The work that has been done on Dialectal Arabic (DA) is limited compared to MSA. NLP for DA is still in its early stages of development, and there are many challenges that need to be overcome, such as the lack of language resources and tools. However, a limited amount of DA research has been devoted to morphological segmentation. A variety of approaches have been used in word segmentation, including rule-based, statistical, machine learning, and hybrid approaches. Significant work that has been done on segmenting words in DA is listed below.

In 2012, Mohamed et al. [20] developed a word segmenter for the ED that uses memory-based learning. A small corpus of 20,022 words in the ED that were collected from 320 reviews was used to train the segmenter. These words were divided into their morphemes and annotated by two native ED speakers. In terms of segmentation accuracy, they received a score of 91.90 percent. Pasha et al. introduced MADAMIRA [24] in 2014, a system for morphological analysis and disambiguation of Arabic that includes the best features of two earlier systems for Arabic, MADA [14] and AMIRA [10]. MADAMIRA has been developed with additional features to analyze ED and MSA. In 2014, Monroe et al. segmented MSA and several Arabic dialects using a single dialect-independent model [21]. Three different corpora were used to train and evaluate their Conditional Random Fields (CRF) model. They claimed that their segmenter exceeded the existing systems on newswire, broadcast news, and ED. They achieved an F1 segmentation score of 95.1% on a recently published ED corpus, compared to 90.8% for another segmenter developed specifically for ED.

In 2016, Farasa was presented [9], which is a fast and accurate Arabic segmenter. The approach is based on Support Vector Machine (SVM) to rank the possible legal segmentations of an Arabic word. Their experimental results show that the Farasa outperforms state-of-the-art Arabic segmenters, called QATARA [8] and MADAMIRA. Eldesouki et al. [11] presented Arabic multi-dialect segmentation using SVM-based ranking and bi-LSTM-CRF sequence labeling for segmenting Egyptian, Levantine, Gulf, and Maghrebi dialects, using only a few thousand training samples for each dialect. Both approaches yielded comparable results, with accuracies ranging from 91% to 95% for different dialects.

YAMAMA was presented in 2016 [18], a multi-dialect Arabic morphological analyzer and disambiguator for MSA and ED. YAMAMA was created by combining the benefits of two MSA morphological analyzers: MADAMIRA and FARASA. In fact, they leveraged MADAMIRA component analysis and FARASA disambiguation modeling. YAMAMA was compared to MADAMIRA and FARASA in the context of Statistical Machine Translation (SMT). Although YAMAMA is about five times faster than MADAMIRA, its quality is slightly lower. It provides a rich representation of outputs that can be used for a broader range of applications. FARASA, on the other hand, is faster but gives particular outputs specialized for certain applications.

In 2019, Tawfik et al. conducted several experiments on Egyptian Arabic (EA) [29], Levantine Arabic (LA), and Gulf Arabic (GA) to investigate the impact of the dialectical segmenter on the quality of the Machine Translation (MT) system. And they contrasted dialectal segmentation with other segmentation techniques such as Byte-Pair-Encoding (BPE) [28] and Sub-word Regularization (SR) [19]. Five word segmentation tests were used in their experiments: dialectal segmenter, BPE, SR, dialectal segmenter with BPE, and dialectal segmenter with SR. In their work, a retrained version of the Unified Dialectal Arabic Segmenter (UDAS) [27] was used as a dialectal segmenter. The experiments were

carried out using Marian v1.7.6 [17], which is a public neural machine translation framework. The results show that using a high-accuracy dialectal segmenter together with a language-independent word segmentation approach such as BPE or SR has some advantages.

Almuhareb et al. in 2019 presented an Arabic word segmentation method that is based on a bi-directional long short-term memory deep neural network [4]. Their method consists of two tasks: word segmentation only and word segmentation for nine cases of the rewrite. In Arabic orthography, when the base word is attached to another unit (affix), the letters of the base word might be dropped or changed. A rewrite process is required in this case to write these letters back to the base word when the units (a base word and affixes) are separated as a result of segmentation. In their work, they achieved an F1 score of 98.03% for word segmentation only and above 99% for word segmentation with the rewrite.

In 2022, Camelira was presented [23], a web-based Arabic multi-dialect morphological disambiguation tool that supports MSA as well as three major Arabic dialects: Egyptian, Gulf, and Levantine. And it provides linguistic information such as part-of-speech, tokenization, and lemmas to academics and language learners via a user-friendly web interface. The web interface allows users to explore the deep linguistic analysis of a given sentence by taking a sentence as input and offering automatically disambiguated interpretations for each word in both context and its out-of-context alternatives. Generally, Camelira combines Arabic morphological disambiguation and the Dialect Identification (DID) system. Actually, it integrates a state-of-the-art morphological disambiguator described by [16] with the most efficient fine-grained Arabic DID system presented by [26].

3 Linguistic Study

Linguistically, the LD has a complex agglutination, which makes it a challenge for word segmentation. Consequently, a comprehensive linguistic study of affixation is required. Affixation can be defined as the process of adding an affix to a base word to form a new word [7]. To conduct the linguistic study, a recent LD Twitter corpus of 5000 tweets was used as a study dataset [3]. Moreover, episodes from the Libya Al-Ahrar TV program on YouTube for LD, titled "مطر الصحَاح"[1], were utilized to gather valuable information regarding the LD. All the information discussed in Section 3 was collected during the linguistic study.

3.1 Orthographic Variations

Due to the oral nature of LD, words can be written in different forms with no orthographic standardization. For example, the English word "much" can be written either "هلبة" or "هلبَا" in LD. Furthermore, in some cases, the letter's order of the words can be changed to produce their synonyms. For instance, the order between letters three and two of the word "قعَمز", which means "seat" in English, can be exchanged to produce the other word with the same meaning, which is "قمعَز". Likewise, the LD words "عَمَاكم" and "معَاكم", which mean "with you" in English, have the same meaning but different spelling because of the change in order between letters one and two.

Occasionally, there is no rule for writing letters that have a similar sound. The letters "س" and "ص" have a similar sound. Therefore, they can be used alternatively to write the same word. For example, the words "سميطري" and "صميطري" are the same word, which both translate to "cold" in English, but they are written with different first letters: "س" and "ص". Also, the letters "ذ" and "د", and the letters "ج" and "ز" have a similar sound and can be used alternatively to write the same word. Furthermore, the letter "ا" can be optionally added to the beginning of some words. For example, the English word "trainers" might be written in LD in two different forms: "سبيدرو" and "اسبيدرو".

[1] https://www.youtube.com/playlist?app=desktop&list=PL1NMeupVTOlhL_F13HOQN-2e-pTegg20W

3.2 Lexical Variations

The LD can be mainly classified into eastern, western, and southern dialects based on where it is spoken. For example, the English word "much" has many equivalents in LD, which are respectively "هَلبة","وَاجد", and "يَاسر" in the west, east, and south of the country. LD is characterized by the usage of synonyms, where different words convey the same meaning. For example, the words "شن", "شنو", and "شني", which mean in English "what", have the same meaning but are written and pronounced differently.

Homophones are also used in LD when words have the same pronunciation and spelling but have different meaning. For example, the word "بَنة" has two meanings. It means "a smell" in the east and "a taste" in the west of the country. The word is pronounced and spelled the same but has two different meanings.

3.3 Morphological Variations

Since LD has a rich morphological system, a single word within the language can have multiple morphological forms within the same grammatical case. For example, the feminine plural pronouns "ن" and "و" are suffixes that can be attached to the verbs based on the areas or regions of speakers. The suffix "ن" is used in the eastern region, whereas the suffix "و" is used in the rest of the country. For example, the English word "they dance" can be written either "يرقصن" in the eastern region or "يرقصو" in the western region, which are both feminine plural forms of the verb "يرقص" that means "to dance" in English. In the eastern region, the feminine plural suffix is similar to Nūn-Niswa (نون النسوة) in classical Arabic.

The grammar of LD has been influenced by both the Arabic and Amazigh languages. The Amazigh language is also known as the Berber language. The LD numbers system uses singular, dual, and plural forms. The dual form in the LD is expressed in two different ways. In the first method, the suffix "ين" can be attached to the singular nouns in order to convert them to dual forms. For example, the suffix "ين" is attached to the singular noun "دَار", meaning "room", to change it to the dual form "دَارين", which means "two rooms". In the second method, the word "زوز", which means "pair" or "two" in English, must precede the nouns to convert them to dual forms. For example, the dual noun "زوز ديَار", which also means "two rooms", can be created by placing the word "زوز" before the plural noun "ديَار", which means "rooms".

The second method of dual form is commonly used in LD. These differences have resulted in some unique features in the LD grammar. Furthermore, some LD nouns have many plural forms. For example, the LD equivalent of the English word "a cup" is "طَاسة", which comes from the Italian word "tazza". The word "طَاسة" has three plural forms, which are "طيس","طوَاسي", and "طَاسَات", all of which mean "cups" in English.

3.4 Space Omission Problem

Due to the lack of standardized orthographic rules, LD words are usually spelled as they are pronounced. Therefore, spaces between words might be omitted. Compound words, which are combinations of two or more words, are created when spaces between words are omitted. Space omission poses a challenging task for word segmentation because the identification of word boundaries in merged or compound words is very complicated.

Consequentially, considerable work is needed to deal with the space omission issue.

Because of space omission, LD words were written in a variety of forms in the study dataset. For instance, the English question "What did you do to it?" can be expressed in LD in two forms. The first form is "شن درتلهَا", which consists of the two separate words "درتلهَا" and "شن", where "درتلهَا" means in English "You did something to it" and "شن" means in English "what". While the second form merges or combines two words together to

produce one word "شندرتلهَا", the space between the two words "درتلهَا" and "شن" is omitted in this case.

In the linguistic study, they found that some interrogative pronouns, such as "من" and "شن", which respectively mean in English "who" and "what", can be connected to other words to form compound words. Moreover, the most frequent combinations in the study dataset are preposition+pronoun, which can be added to the endings of words to create compound words. For example, the preposition+pronoun combination "لك", meaning in English "to you" can be attached to the ending of the word "قلتهَا", which means in English "I say it" to make up the compound word "قلتهَالك", meaning "I say it to you". In contrast, interjections are sometimes attached to the beginning of words to form compound words. For instance, the interjection "تي", which means in English "Psst", might be attached to the beginning of the word "بَاهي", meaning in English "ok" to form the compound word "تيبَاهي", which means in English "Psst, it's ok". Table 1 shows examples of the number of space omissions for LD words that can be translated to the English phrase "Psst, who will say it to you?". Notice that all forms of LD in Table 1 were found in the study dataset.

Table 1. Possible number of space omissions for the same LD words

Usage	Number of space omission	LD words
less commonly	0	تي من بيقولهَا لك
more commonly	1	تي من بيقولهَالك
commonly	2	تي منبيقولهَالك
rarely	3	تيمنبيقولهَالك

Here are other types of compound words that are created by joining two words. Firstly, subject personal pronouns might be attached to the endings of adverbs. For example, the LD word "كَانشهوَا", which means in English "maybe he". The word can be split into two words: "هوَا" + "كَانش", which are a personal pronoun and an adverb, respectively. Additionally, prepositions might be connected to the beginnings of nouns. For example, the LD word "عَالوطَا" which means in English "on the ground". It can be segmented into a pair of words: "الوطَا" + "عَ". Where "الوطَا" is a noun that means "the ground" and "عَ" is a preposition that means "on".

Furthermore, the beginnings of nouns might be attached to demonstrative pronouns. For example, the LD word "هَالجو" which means in English "this weather". It can be segmented into two separate words: "الجو" + "ه", where "الجو" is a noun and "ه" is a demonstrative pronoun. Also, personal pronouns can be negated by adding a negator word to their beginnings. For example, the LD word "مشحنَا", which means "not us" in English, can be split into two words: "حنَا" + "مش", where "حنَا" is a personal pronoun and "مش" is a negator word. Additionally, the word "مش" can also be attached to the beginnings of adjectives to create their negated forms. For example, the word "مشبَاهي", which means "not good" in English, is made up of two words: the negator word "مش" and the adjective word "بَاهي".

3.5 Affixation in Libyan Dialect

In linguistics, an affix is a morpheme that is attached to a word in order to change its meaning. Generally, affixes can be classified into several types, depending on their position and morphology. As for their position, affixes can be prefixes, suffixes, infixes, or circumfixes. While the morphological classifications are inflection, derivation, and enclitization. In this work, the definitions of the two terms "prefix" and "suffix" need to be modified. The term "prefix" is modified to indicate any part that can be attached to the beginnings of the base words, whereas the term "suffix" can be defined as any part that can be attached to the endings of the base words.

Thus, the work aims to segment all prefixes and suffixes that might be attached to the base word, including verbs, nouns, and function word affixes. Many cases of agglutination that

were encountered in the linguistic study required meticulous and thorough analysis in order to extract fundamental insights about how they are constructed. Table 2 shows some words that were selected from the study dataset to demonstrate different types of agglutination of affixes along with their segmentation.

Affixes of negation are commonly used in LD to make negated forms of the words. Affixes of negation usually appear as circumfixes. A prefix and a suffix must be added to a word in order to negate it. Some cases of negation are shown in Table 2. For example, the fifth row of Table 2 illustrates the negated form of the word "هذرز", which means "he talks" in English. The negation form is created by adding the prefix "م" to the beginning and the suffix "ش" to the end of the word "هذرز". This results in the negated word "مهذرزش", which means "he does not talk" in English.

The second row in Table 2 shows the long LD word "تيمنبيقولهَالك", which means "Psst, who will say it to you?" in English. This word is made up of seven affixes added to the base word "قول", which can be segmented into (تي+من+ب+ي+قول+هَا+ل+ك). Obviously, the examples in Table 2 illustrate the complex agglutinative morphology of LD, which makes the word segmentation process extremely difficult and poses one of the major challenges to NLP tasks for LD. The main objective of the linguistic study is to construct lists of LD affixes, including prefixes and suffixes, that can be segmented using the algorithm. Table 3 shows a total of 26 prefixes and 23 suffixes that were manually identified and collected during the linguistic study.

4 Word Segmentation in NLP

Arabic, along with its dialects, is characterized by a root-based morphology where words are formed by combining prefixes, suffixes, and infixes to a base root, resulting in the generation of many possible word forms [13]. Generally, the boundaries among word components are often unclear, which poses significant challenges in accurately dividing words into meaningful morphemes, including both affixes and roots. These morphemes are essential for many NLP tasks, such as information retrieval, machine translation, and sentiment analysis, to improve their performance. Here is an example to illustrate how word segmentation improves MT systems.

Consider translating the LD word "مكتبتَاش" into English. The MT system first tries to find a word in its data or dictionary. If the word does not exist, then the word segmentation is used to break down the word "مكتبتَاش" into its constituent morphemes: (م+كتب+تَا+ش). After that, the MT system can recognize each of these morphemes as individual units of meaning and maps each morpheme into its corresponding word in English. It converts the "كتب" to "write", "ش" to "not" and so on. Finally, the translation of the LD word "مكتبتَاش" is generated, which is "I didn't write it" in English.

5 Graphs in NLP

Graphs are powerful computational models for NLP. Graph-based models are able to represent complex relationships and dependencies that can be found in linguistic constructs [15]. Graphs can contain nodes for representing entities (people, places, concepts, and events) and edges for representing relationships (born in, worked in, and married to) among entities. Graphs are applied in various NLP tasks to provide a deeper understanding of language for performing more sophisticated systems. For example, graphs can provide a structured way to represent knowledge in Question Answering (QA) systems. If a user asks, "Where was William Shakespeare born?" and the knowledge graph of QA systems contains information "William Shakespeare—born in–¿ England," where "William Shakespeare" and "England" are entities and "born in" is a relationship between them.

The answer can be obtained through many steps by navigating the graph. First, the QA system analyzes the question and identifies the entities and their relationships. In this case, the entity is "William Shakespeare" and the

Table 2. Some affixes that can be attached to LD words

POS	Segmentation	English meaning	LD word	No
interrogative word +verb+ preposition +pronoun	شن+صَار+في+هم	what did happen to them	شنصَارفيهم	1
interjection + interrogative word +tens marker+ gender and number marker +verb+	تي+من+ب+ي+قول+هَا+ل+ك	psst, who will say it to you?	تيمنبيقولهَالك	2
conjunction + negator prefix + verb+ pronoun+ pronoun+ negator suffix	مَا+شبح+ت+هَا+ش	I did not see it	ومَاشبحتهَاش	3
negating word+ noun+ pronoun	مش+شور+ك	it is not your business	مشّورك	4
negator prefix + verb+ negator suffix	م+هذرز+ش	he does not talk	مهذرزش	5
tens mark+ gender and number marker +verb+ gender and number marker + preposition+ pronoun	ب+ن+قعمز+و+ل+هم	we will stay with them	بنقعمزولهم	6

Table 3. Lists of the prefixes and suffixes that can be attached to LD words

Prefixes	و ب ه ف ت ن ي عَ ح م ل ال لل لي ان كل عَند عَلَى تي شن من مَا يَا مش معَا في
Suffixes	ه ت ن و ي ل ك ة ش هم نَا كم هَا ات ني تي وَا ين هن هو هي همَا لَا

relationship is ”born in”. In the next step, the system navigates the graph, starting from the ”William Shakespeare” node, to find an edge or relationship matching ”born in”. The node ”England” which is directly connected to the ”born in” edge, represents the answer. Obviously, the use of graph representations enables QA systems to efficiently answer questions. As graphs offer significant advantages in NLP, this paper aims to use a graph as an effective computational model to tackle the agglutination problem in agglutinative and low-resource languages, which is one of the NLP challenges.

6 Word Segmentation Algorithm

Affixes can be a chain or a sequence of morphemes. A sequence means a set of morphemes next to each other in a set order. Thus, the ordering of morphemes is considered a key idea for solving word segmentation. In this case, a special data structure for representing the ordering of affixes and their relationships is needed. To this end, WDG can be used as a linear-chain model to solve word segmentation problem. Graphs are a type of data structure that can be used to model a wide range of scientific challenges, including NLP problems [22]. WDG can be defined as a triple G = (V, E, w), where V is the set of nodes, E is the set of edges, and w: E $\rightarrow$ N is a function that assigns a weight w(e) to each edge $e \in E$. Typically, w(e) $\geq$ 0 [5].

The algorithm used WDG to model a word segmentation problem for representing the affixes and their ordering relationships. Where WDG nodes are used to represent affixes, each node represents one affix. While directed edges or arrows from one node to another are utilized to represent ordering relationships, weighted edges are also used to represent three relationships: prefix-to-prefix, suffix-to-suffix, and prefix-to-suffix relationships.

The prefix-to-prefix relationship is used to represent the relationship between two prefixes with an edge weight of 1, while an edge weight value of 2 is used to indicate the suffix-to-suffix relationship, which represents the relationship between two suffixes. The prefix-to-suffix relationship, which can be represented by a weight value of 3, is used to represent a relationship between the last prefix and the last suffix in a word. Figure 1 shows three relationships for the LD word ”منبيقولهَالك”.

In order to separate affixes from words, it is necessary to correctly identify the location of affixes in a given word. The starting point of prefixes can be easily identified, as it is always at the beginning of the words. In contrast, the starting point of suffixes is always unknown and difficult to identify, but the end of suffixes can be easily identified because it is always at the end of the word. For this reason, the direction of dividing suffixes in this algorithm is to start from the end to the beginning of the word, while the direction from

the beginning to the end is for splitting prefixes. Obviously, both directions are opposite. Figure 1 shows the starting points of prefixes and suffixes for segmenting the word "منبيقولهَالك".

A given word in the algorithm can be easily segmented by matching its affixes to the graph nodes. The first step in the segmentation process is to match the first affix to the starting nodes in the graph. Therefore, the starting nodes in the graph are distinguished from the other nodes. The segmentation process starts if there is a match between the first affix and any starting nodes. Then it keeps matching between word affixes and graph nodes until it either reaches the end node or no matching takes place. The path between the starting node and the end node represents word affixes. However, a given word only includes prefixes if the segmentation path only has edge weights with values of 1. While the given word would only contain suffixes if all edge weights in a segmentation path had a value of 2. In contrast, if the edge weights in the path contained values of 1 followed by 3, and zero or more values of 2, then the given word would have prefixes and suffixes. Figure 1 illustrates three ordering relationships with their edge weights for the word "منبيقولهَالك".

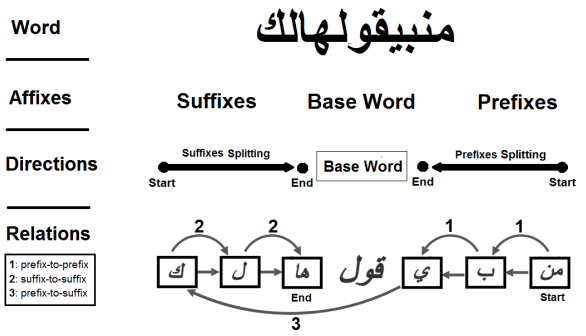

Fig. 1. Directions and relations for segmenting the word "منبيقولهَالك"

To illustrate how the algorithm works, Figure 2 shows a graph that can be used as an example for segmenting different word forms. The node numbers with a red color were added to the graph to clarify how the algorithm works.

Word segmentation in the algorithm can be done by following a certain path in a graph. For example,

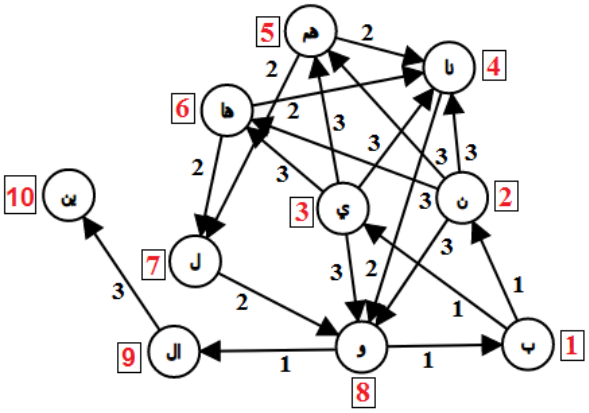

Fig. 2. WDG example for splitting some word affixes

Table 4 includes different word forms that can be segmented by using the WDG in Figure 2. The word "بنقعَمز" in the first row, which means "I want to sit" in English, can be segmented by matching the first letter "ب" to node number one, which is the starting node in the graph. After that, the matching between node two and the second letter "ن" will take place. Obviously, at node number two, the segmentation process terminates upon no further matches. At this point, the segmentation path would then have two nodes, numbers one and two. These nodes represent the word affixes. As a result of segmentation, the base word "قعَمز" as well as the two prefixes "ن" and "ب" are obtained.

The last row in Table 4 shows a special case when both the first letter "و" and the last two letters "هم" of the word "وبنشبحولهم", which means in English "and we will see them", would match node numbers eight and five, respectively. The question here is: what is the starting node? If there were two starting nodes, one for a prefix and the other for a suffix, the algorithm would begin from the node that represents a prefix.

In this case, the starting node would be node number eight. Consequently, there will be matching between node eight and the first prefix "و". Next, node number one will match the next prefix "ب". Then it keeps matching until it reaches node number two. In this situation, when the

last suffix "هم" matches the node number five, the relationship of prefix-to-prefix will be immediately changed into a prefix-to-suffix relationship with an edge weight value of 3.

Then it moves from node two to node five. Likewise, when the next suffix "ل" matches the node number seven, it moves from node five to node seven, and the prefix-to-suffix relationship will be changed into a suffix-to-suffix relationship with an edge weight value of 2. It would then keep following until it reached node number eight, which is the end node. The starting and finishing nodes are obviously the same node, which is node number eight.

The segmentation path would finally include six nodes, numbered 8, 1, 2, 5, 7, and 8, which represent the word affixes: "و+ب+ن+و+ل+هم" with edge weights of 1, 1, 3, 2, and 2, respectively. The types of affixes can be identified by their edge weight values. The segmentation result would eventually be the basic word "شبح", the prefixes "و + ب+ن" and the suffixes "و+ل+هم".

The algorithm can also deal with negation. The graph in Figure 3 shows an example of how negation circumfixes can be segmented. The nodes "و" and "م" are distinguished as starting nodes in the graph. The segmentation of the word "ومقَالولَاش", which means "they did not say it to him" in English, would begin by matching the first prefix "و" to the graph node "و", which is the starting node. The next step would be to match the next node "م" to the next prefix "م". After that, match the next node "ش" to the suffix "ش". Then it would move on to the next node "لَا", and it would keep matching until it reached node "و", which is the end node. The final result of segmenting would be "م" and "و" are prefixes, "قَال" is the base word, and "ش", "لَا", and "و" are suffixes. In Figure 3, the nodes of the segmentation path are highlighted in red.

To enhance the performance of the algorithm, a language resource called an exception list is used, which is a list of words that have no affixes but whose original letters appear as affixes. It is used to match a given word to the words in a list. If there is a match, no segmentation is done; otherwise, the affixes would be split. The list was collected from two different language resources: the LD Twitter corpus [3] and the LD-MSA bilingual dictionary [2].

The augmentation of the list size results in an enhancement of the algorithm's efficiency. Finally, the affixes graph was transformed into an adjacency matrix, and the PHP programming language was used to implement the algorithm. The algorithm steps are shown in Algorithm 1.

In this algorithm, many symbols are used as variables. Where the "GW" symbol represents a given word, "AF" denotes an affix that can be segmented from a given word. Furthermore, the "CN" symbol indicates the active or current node in the graph. The number of direct successor nodes of the "CN" node is represented by the "m" symbol, while "n" stands for the number of starting nodes. The "DN(i)" represents direct successor nodes of the current node "CN". The "Prefixes_list" and "Suffixes_list" symbols are used to save prefixes and suffixes that can be split from a given word, respectively. Finally, "SN(i)" represents the starting nodes in the graph. The demo version of the algorithm is available online[2]. .

7 Scalability and Complexity

The algorithm utilizes WDG to represent affixes. The WDG can be represented by an adjacent matrix, which is a square 2D array. The size of the adjacent matrix is NxN, where N is the number of nodes or affixes. The time complexity of this algorithm was measured using the big O notation metric. Time complexity is used to estimate the time required to execute the return output. The algorithm has a time complexity of O(nˆ2). The time complexity of the algorithm can be changed based on the data structure that is used to represent a WDG. Moreover, the algorithm has a simple and flexible structure that separates code from data (graph data), making the code more scalable.

This separation enables the algorithm to effectively handle different scales of data, from small to large graphs, without diminishing its

[2] http://dra.net.ly/segment/segmenting.php

Algorithm 1 Pseudo-code of the Word Segmentation Algorithm

Input: a word
Output: affixes and a remaining word
Read adjacency matrix of affixes graph
Read exception words list
Let Prefixes_list, Suffixes_list, and AF ← Null
$GW \leftarrow$ the given word
if $GW \notin$ exception words list **and** length(GW) > 2 **then**
    Get a list of all starting nodes $SN(i = 1, 2, \ldots, n)$ in the graph
    for $i \leftarrow 1$ to n **do**
        if beginning of GW matches $SN(i)$ **then**
            $AF \leftarrow SN(i)$
        end if
        if ending of GW matches $SN(i)$ **then**
            $AF \leftarrow SN(i)$
        end if
    end for
    if $AF \neq$ Null **then**
        if AF is a prefix **then**
            $Prefixes\_list \leftarrow AF$
        end if
        if AF is a suffix **then**
            $Suffixes\_list \leftarrow AF$
        end if
        Remove AF from GW
        Let CN be the starting node
        Move to CN in the graph
        while true **do**
            Get list of all direct successors $DN(i = 1, 2, \ldots, m)$ of node CN
            for $i \leftarrow 1$ to m **do**
                $W \leftarrow$ weight between CN and $DN(i)$
                if $W = 1$ **and** beginning of GW matches $DN(i)$ **then**
                    $Prefixes\_list \leftarrow Prefixes\_list + DN(i)$
                else if $W \neq 1$ **and** ending of GW matches $DN(i)$ **then**
                    $Suffixes\_list \leftarrow Suffixes\_list + DN(i)$
                end if
                if there is a match **then**
                    Remove $DN(i)$ from GW
                    $CN \leftarrow DN(i)$
                end if
            end for
            if no match or successor list is empty **then**
                exit while
            end if
        end while
    end if
end if
Print: Prefixes_list, GW, Suffixes_list

Table 4. Examples of segmenting different word forms

Path weights	Suffixes	Prefixes	Base word	Segmentation path	English meaning	LD word	No
1	Null	ب+ن	قعَمز	1,2	I want to sit	بنقعَمز	1
1,1,3	و	ب+ي	هدرز	1,3,8	and they will talk	يهدرزو	2
2	نَا+هم	Null	شلبخ	5,4	we beat them	شلبخنَاهن	3
1,3	ين	و+ال	فَالح	8,9,10	and the smart ones	وَالفَالحين	4
1,1,3,2,2	و+ل+هم	و+ب+ن	شبح	8,1,2,5,7,8	and we will see them	وبنشبحولهم	5

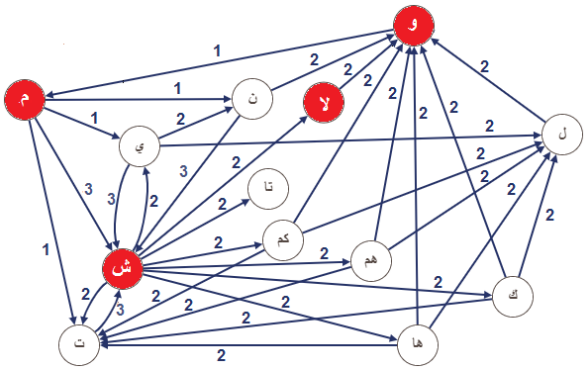

Fig. 3. WDG example for segmenting negation affixes

performance or efficiency. WDG is well-suited for word segmentation because it is a relatively simple task that can be represented by a small number of nodes. In this work, only 43 nodes were used to represent affixes, which is a relatively small number.

8 Evaluation and Discussion

Languages with limited resources, such as LD, face challenges in obtaining standard evaluation datasets. Consequently, a new test dataset of 1,200 words has been mainly created from the LD Twitter corpus [24] and other resources. This dataset includes a diverse collection of words that have been formed with different affixes. Furthermore, in the absence of any existing LD work that can be used for comparative evaluation of the algorithm, the authors decided to conduct a manual evaluation of their algorithm.

First, the test dataset was manually segmented by linguists, and then the segmented data was compared to the results of the algorithm to assess its accuracy.

The test dataset was divided into three portions. 50% of the test dataset contains words with affixes, while words without affixes make up 25%. The last 25% includes words that have no affixes, but some of the original letters appear as affixes. The test dataset finally included words with affixes and clitics, as well as some compound words.

However, the performance of the algorithm was measured using precision, recall, and F1 score metrics. The algorithm achieved a perfect F1 score of 0.886 on the test dataset. Table 5 shows examples of the algorithm's results.

8.1 Error Analysis

For more details, see Table 6, which includes some examples that have been selected to demonstrate various types of the algorithm failures. The first row shows the wrong segmentation of the letter "ل". Although it is an original letter of the base

word "قول", this letter was segmented because the matching took place between the letter "ل" and a node in the segmentation path.

The second row shows a special case where one node contains the value of the combination of two other nodes. The mistake occurred when the node "من" was segmented instead of segmenting the node "م" followed by the node "ن".

The third row shows the wrong segmenting case, which occurs when there is more than one relationship between two nodes. The nodes "ي" and "ش" actually have two relationships: the prefix-to-suffix and the suffix-to-suffix relationships. Only one relationship between any two nodes can be defined in this algorithm. In this case, the prefix-to-suffix relationship was predefined. As a result, the algorithm generated this mistake because it was unable to handle the suffix-to-suffix relationship.

The last row of the table contains the word "بيَادجو", which means "trip" in English. The word was imported from Italian. Although it had no affixes, it was segmented because its original letters appeared as affixes.

The overall accuracy of the algorithm could be improved by adding words whose original letters appear as affixes to the exception list. Moreover, the failures in the second and third rows in Table 6 can be addressed by dividing the entire graph into many smaller graphs. Additionally, the failures in the third row can also be solved by introducing a new graph weight that encompasses both the prefix-to-suffix and the suffix-to-suffix relationships. Furthermore, despite the success of segmentation, Out-Of-Vocabulary (OOV) or unknown words were obtained in some cases. For example, the segmentation result of the word "حوَازتي", which means "my farm" in English, is "حوَاز", which is an OOV word.

9 Pros and Cons of the Algorithm

They conducted a theoretical comparison of the algorithm to other approaches for word segmentation in order to demonstrate its strengths and weaknesses. The algorithm has several advantages over a rule-based approach. It is more scalable because it separates code from data. It can also be visually displayed in a variety of ways, such as graphs and plots. A rule-based approach requires more manual work to write and generate rules, and it can be time-consuming. Furthermore, the algorithm shares some drawbacks with a rule-based approach, such as the inability to handle unknown cases. If the graph is incomplete or wrong, the algorithm may not provide accurate or complete results. In addition, a machine learning approach can learn from new data, making it more scalable and adaptable than the algorithm. On the other hand, the algorithm does not rely on training data for word segmentation, thereby providing a reliable solution for low-resource languages. Moreover, it is simple and can be readily implemented using any programming language that supports a 2D array to represent a graph, and no additional code libraries are needed.

10 Applicability of the Algorithm

The proposal algorithm was designed to use a graph for addressing agglutinative problems, particularly in low-resource languages. The algorithm was implemented and evaluated on LD, a member of the Arabic dialect family. The algorithm showed effective performance in segmenting LD words. It also theoretically exhibits more applicability due to its capacity for easy adaptation and maintaining consistent performance across different languages, where their affixes can be represented in WDG format. For example, in addition to LD, the proposed algorithm can be readily applied and adapted to efficiently segment words from other Arabic dialects, as their affixes resemble those of LD and can be represented as WDG.

11 Conclusion and Future Work

This paper introduces an algorithm for segmenting LD words by modeling the word segmentation problem as a WDG. A Twitter corpus of 5,000 tweets was used to conduct a deep linguistic study

Table 5. Correct word segmentation examples in the algorithm

Correct segmentation	The algorithm	English meaning	LD word
و+م+عَند+ك+ش	و+م+عَند+ك+ش	and you do not have	ومعَندكش
مَا+هذرز+نَا+ش	مَا+هذرز+نَا+ش	we did not talk	مَاهذرزنَاش
ب+جو+ك	ب+جو+ك	in your mood	بجوك
وَاجد	وَاجد	much	وَاجد
قطوس+كم	قطوس+كم	your cat	قطوسكم

Table 6. Examples of words that were incorrectly segmented by the algorithm

Correct segmentation	The algorithm	English meaning	LD word
م+ت+قول+ي+ش	م+ت+قو+ل+ي+ش	you do not say to me	متقوليش
م+ن+حب+ش	من+حب+ش	I do not like	منحبش
م+في+ش+ي	م+فيش+ي	there is not	مفيشي
بيَادجو	ب+ي+ادج+و	trip	بيَادجو

to identify and collect a set of affixes that can be segmented by the algorithm. A new test dataset of 1,200 unique words was created to evaluate the performance of their algorithm.

The algorithm was manually examined to determine its accuracy on the test dataset. The results showed that the algorithm achieved an overall accuracy of 88.6%, which is satisfactory. The algorithm achieved promising results without the need for external data resources, making it a viable solution for not only Arabic dialects but also agglutinative languages, such as MSA, Turkish and Finnish.

As for future work, they intend to provide a training dataset for LD word segmentation for machine learning applications. They also plan to develop several crucial NLP tools for the LD in the near future, such as the Named Entity Recognition (NER) and Part of Speech (POS) tagger.

References

1. **Abdelali, A. (2004).** Localization in modern standard arabic. Journal of the American Society for Information Science and technology, Vol. 55, No. 1, pp. 23–28.

2. **Alhammi, H., Haddar, K. (2022).** Toward building a bilingual dictionary for libyan dialect-modern standard arabic machine translation. KDIR, pp. 75–81.

3. **Alhammi, H. A., Alfard, R. A., Ramadan, A. (2018).** Building a twitter social media network corpus for libyan dialect. International Journal of Computer Electrical Engineering, Vol. 10, No. 1.

4. **Almuhareb, A., Alsanie, W., Al-Thubaity, A. (2019).** Arabic word segmentation with long short-term memory neural networks and word embedding. IEEE Access, Vol. 7, pp. 12879–12887.

5. **Bapat, R., Kalita, D., Pati, S. (2012).** On weighted directed graphs. Linear Algebra and its Applications, Vol. 436, No. 1, pp. 99–111.

6. **Berrimi, M., Moussaoui, A., Oussalah, M., Saidi, M. (2020).** Arabic dialects identification: North african dialects case study. 3rd Conference on Informatics and Applied Mathematics, IAM 2020, RWTH Aachen University, pp. .

7. **Carstairs-McCarthy, A. (2017).** Introduction to English Morphology: words and their structure. Edinburgh university press.

8. **Darwish, K., Abdelali, A., Mubarak, H. (2014).** Using stem-templates to improve arabic pos and gender/number tagging. LREC, pp. 2926–2931.
9. **Darwish, K., Mubarak, H. (2016).** Farasa: A new fast and accurate arabic word segmenter. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pp. 1070–1074.
10. **Diab, M., Hacioglu, K., Jurafsky, D. (2007).** Arabic computational morphology: Knowledgebased and empirical methods, chapter automated methods for processing arabic text: From tokenization to base phrase chunking.
11. **Eldesouki, M., Samih, Y., Abdelali, A., Attia, M., Mubarak, H., Darwish, K., Laura, K. (2017).** Arabic multi-dialect segmentation: bi-lstm-crf vs. svm. arXiv preprint arXiv:1708.05891.
12. **González Martínez, A., López Hervás, S., Samy, D., Arques, C. G., Moreno Sandoval, A. (2013).** Jabalín: A comprehensive computational model of modern standard arabic verbal morphology based on traditional arabic prosody. International Workshop on Systems and Frameworks for Computational Morphology, Springer, pp. 35–52.
13. **Habash, N., Roth, R., Rambow, O. (2009).** Morphological analysis and disambiguation for modern standard arabic. Language Resources and Evaluation, Vol. 43, No. 4, pp. 381–405. DOI: 10.1007/s10579-009-9083-0.
14. **Habash, N., Roth, R., Rambow, O., Eskander, R., Tomeh, N. (2013).** Morphological analysis and disambiguation for dialectal arabic. Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 426–432.
15. **Hamilton, W. L. (2020).** Graph representation learning. Morgan & Claypool Publishers.
16. **Inoue, G., Khalifa, S., Habash, N. (2021).** Morphosyntactic tagging with pre-trained language models for arabic and its dialects. arXiv preprint arXiv:2110.06852.
17. **Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., et al. (2018).** Marian: Fast neural machine translation in c++. arXiv preprint arXiv:1804.00344.
18. **Khalifa, S., Zalmout, N., Habash, N. (2016).** Yamama: Yet another multi-dialect arabic morphological analyzer. Proceedings of COLING 2016, the 26th international conference on computational linguistics: system demonstrations, pp. 223–227.
19. **Kudo, T. (2018).** Subword regularization: Improving neural network translation models with multiple subword candidates. arXiv preprint arXiv:1804.10959.
20. **Mohamed, E., Mohit, B., Oflazer, K. (2012).** Annotating and learning morphological segmentation of egyptian colloquial arabic. LREC, pp. 873–877.
21. **Monroe, W., Green, S., Manning, C. D. (2014).** Word segmentation of informal arabic with domain adaptation. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 206–211.
22. **Nastase, V., Mihalcea, R., Radev, D. R. (2015).** A survey of graphs in natural language processing. Natural Language Engineering, Vol. 21, No. 5, pp. 665–698.
23. **Obeid, O., Inoue, G., Habash, N. (2022).** Camelira: An arabic multi-dialect morphological disambiguator. arXiv preprint arXiv:2211.16807.
24. **Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R. (2014).** Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. Lrec, Vol. 14, pp. 1094–1101.
25. **Ruokolainen, T., Kohonen, O., Sirts, K., Grönroos, S.-A., Kurimo, M., Virpioja, S. (2016).** A comparative study of minimally supervised morphological segmentation.

Computational Linguistics, Vol. 42, No. 1, pp. 91–120.

26. **Salameh, M., Bouamor, H., Habash, N. (2018).** Fine-grained arabic dialect identification. Proceedings of the 27th international conference on computational linguistics, pp. 1332–1344.
27. **Samih, Y., Eldesouki, M., Attia, M., Darwish, K., Abdelali, A., Mubarak, H., Kallmeyer, L. (2017).** Learning from relatives: Unified dialectal arabic segmentation. Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pp. 432–441.
28. **Sennrich, R., Haddow, B., Birch, A. (2015).** Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909.
29. **Tawfik, A., Emam, M., Essam, K., Nabil, R., Awadalla, H. H. (2019).** Morphology-aware word-segmentation in dialectal arabic adaptation of neural machine translation. Proceedings of the Fourth Arabic Natural Language Processing Workshop, pp. 11–17.

Article received on 30/01/2025; accepted on 24/07/2025.
**Corresponding author is Husien Alhammi.*