*Int. J. Advance Soft Compu. Appl, Vol. 18, No. 1, March 2026*
*Print ISSN: 2710-1274, Online ISSN: 2074-8523*
*Copyright © Al-Zaytoonah University of Jordan (ZUJ)*

# A Hybrid Deep Neural Network and Morphological Knowledge to Enhance Arabic Lemmatization

**Samir Belayachi [1], Azzeddine Mazroui[2]**

[1]Department of Computer Science, Faculty of Sciences, Mohammed First University
e-mail: samirbelayachi@gmail.com
[2]Department of Computer Science, Faculty of Sciences, Mohammed First University
e-mail: azze.mazroui@gmail.com

**Abstract**

***Lemmatization is performed in many natural language processing applications during the preprocessing stage, enabling more efficient text analysis and the extraction of relevant information. The Arabic language is associated with the following challenges of lemmatization: the morphological richness of the language, the high usage of concatenations and the omission of the diacritical marks. To overcome those issues, we will introduce a new lemmatizer to improve the functionality of a deep learning framework. The latter will be a BLSTM network, with an additional filtering layer based on morphological features. To alleviate the effect of out-of-vocabulary words, a statistical layer built on Hidden Markov Models was introduced. Tests done on a reference corpus showed that accuracy was enhanced by a margin of over nine percentage points on the use of the two layers (filtering and statistical). In addition, the results were compared with state-of-the-art lemmatizers on an independent corpus which proved that the proposed approach was superior.***

**Keywords**: *NLP, Arabic Language, Morphological Analysis, Lemmatization, Bidirectional Long Short-Term Memory network.*

## 1 Introduction

The massive and ever-growing creation of digital materials in the form of text, voice and images has been achieved due to the extensive use of the internet and social media. It is, however, a challenge to extract useful information out of this data using automated analysis. As a result, the generation of sophisticated natural language processing (NLP) programs has remained an agenda of the information technology practitioners.
A basic step in NLP is morphological analysis, which is the study of the inner structure of words [1]. It attempts to break down a word into morphemes (the smallest units of meaning of a word) and identify their grammatical functionality. This analysis can be either contextual considering surrounding words, or context-free where words are considered independently. The contextual morphological analysis uses the surrounding words to provide an accurate segmentation of each word under analysis. On the other hand, the

***Received 12 August 2025; Accepted 9 February 2026***

context-free method gives the various possible ways in which a word can be divided into morphemes.

Lemmatization, which involves reducing inflected words to their base form [2], is a morphological analysis technique widely applied in NLP during the preprocessing phase. For nouns, the lemma corresponds to their masculine singular form (when applicable) without clitics. For instance, the respective lemmas of the words ولمدارسهم /wlmdArshm/1 (and for their schools) and كالمعلمات /kAlmElmAt/ (like the teachers) are مدرسة /mdrsp/ (school) and معلم /mElm/ (teacher). For verbs, the lemma refers to the verb form without clitics, conjugated in the past tense, in the third person singular masculine. For example, the lemma of سيكتبونه /syktbwkh/ (they will write it) is كتب /ktb/ (he wrote). For particles, the lemma is simply the particle itself, devoid of clitics. Therefore, the lemma of بالذي /bAlDy/ (with that) is الذي /AlDy/ (the one).

Lemmatization is essential for various NLP tasks, including indexing web pages [3], information retrieval [4], text generation [5], text readability assessment [6], machine translation [7], and sentiment analysis [8].

There are three main main approaches for developing lemmatizers. First, rule-based and lexicon-based approaches, which use explicit linguistic knowledge encoded in rules and dictionaries to provide good accuracy in regular cases but may be less effective when dealing with the complexity of Arabic morphology. Second, machine learning-based approaches, which rely on statistical models and require large annotated corpora to learn and predict word lemmas [9]. Third, hybrid approaches which combine linguistic rules, lexicons and machine learning techniques to leverage the advantages of each method ([10], [11], [12]). Even though hybrid methodologies are often linked to the improvement of performance outcomes, their implementation may be technically challenging. Furthermore, the error rates in empirical assessment on a representative and independent corpus, when a variety of state-of-the-art lemmatization algorithms are employed has shown to be relatively high. This highlights the necessity of coming up with an even more effective lemmatization solution.

In this paper, we present a new lemmatizer that was built in a hybrid manner. First of all, the morphological data provided by the Alkhalil Morpho Sys analyzer [13] was incorporated with a Bidirectional Long Short-Term Memory (BiLSTM) network and the performance was high relative to the state-of-the-art models. Nonetheless, a review of the errors showed that not all errors were caused by out-of-vocabulary words (OOV). To overcome this problem, a statistical layer with Hidden Markov Models (HMMs) was added, which also enhanced performance in the course of the evaluations.

Other parts of this paper are structured in the following way: Section 2 contains the extended review of the literature about the existing Arabic lemmatizers. Section 3 describes the hybrid approach used to develop the proposed lemmatizer. It reviews the definition of HMM models, the steps of the Viterbi algorithm, and the smoothing method chosen to mitigate the effect of OOV words. Section 4 compares experimental findings, comparing our lemmatizer to state-of-the-art systems. Finally, Section 5 sums up this manuscript by discussing essential findings and recommendations regarding further research activity.

## 2 Related Work

The study of how to extract lemmas in MSA texts has been well researched and hybrid methods have been found most effective. We give here a historical survey of some of the most popular lemmatizers used in literature. Madamira [14] is an open-source library of

1 Buckwalter transliteration http://www .qamus.org/transliteration.htm

morphological analysis of Arabic language, containing a lemmatizer. Madamira has two stages of lemmatization. The SAMA parser [15] does a context free morphological analysis first. That is to say, it produces several possible lemmas of each word it analyzes. The system then uses an SVM classifier and language models which take into account word contexts, to pick the best lemma in relation to each word out of the generated alternatives. The authors say that this tool has an approximate accuracy of about 96.6%. Farasa, too, is a set of open-source software that is specialized in Arabic morphosyntactic disambiguation and was developed by the Qatar Computing Research Institute [10]. Along with a lemmatizer, Farasa also has a segmentation module, a diacritization system and a morphosyntactic tagger (POS). These tools rely on an SVM model with greedy algorithm [16]. Results of lemmatizer are not diacritized and the accuracy of the lemmatizer is about 97%. The authors suggested a primary approach which is based on machine learning and a secondary dictionary-based approach for designing a lemmatizer in [17]. They created a three-million entry exhaustive dictionary and annotated a two-million tokens corpus. The two lemmatizers resulting from such work may work separately or in unison to produce better results. The clitics and the base forms were integrated with morphosyntactic tagging and word segmentation that allowed the system to reach over 95% accuracy. Alkhalil Lemmatizer [11] is an open-source Arabic lemmatizer, which uses a two- step hybrid model. It creates a list of possible lemmas of every word by the means of the AlKhalil Morpho Sys parser in a context-free environment first. Then, a disambiguation step is involved that uses the Hidden Markov Models and the Viterbi algorithm. This step is a context-based selection of the most pertinent lemma of the list generated. Accuracy of this hybrid approach according to the authors is greater than 94%. The Arabic lemmatizer is also a part of the suite of Arabic language processing parsers and capabilities offered by the CAMeL Tools [12]. They are available either through command-line interfaces or Python APIs. The disambiguation methods integrate deep learning with conventional morphological processing. The researchers indicate that the accuracy of experiments which are done in Modern Standard Arabic texts was about 95%. To enhance the coverage as well as the speed of the Arabic morphological analysis, Ibn- Ginni [18] merges the strengths of two of the popular open-source morphological analyzers: BAMA [19], which is known to be the fastest, and Alkhalil Morpho Sys, which is known to be the most comprehensive. Ibn-Ginni expands coverage of BAMA with a huge database of three million unique words, which allows analysis of 0.6 million other words in comparison to BAMA alone. In addition, it has a high analysis rate of an average of 0.3 milliseconds per word. The morphological database and the tool are open-source and can be found at GitHub. SinaTools [9] is an open-source (Python) package that provides an Arabic natural language processing (NLP) and understanding. It contains a single bundle of a few NLP applications, such as Named Entity Recognition (NER) in both flat and nested format, Word Sense Disambiguation (WSD), Semantic Relatedness, Synonymy Extraction and Evaluation, Lemmatization, Part-of-speech Tagging, Root Tagging, and others. It further provides corpus processing and text stripping utilities and diacritic- sensitive word matching. SinaTools will make it easier to integrate Arabic NLP solutions with systems and workflows. The authors claim that lemmatizer scores approximately 91% as F1-score. The lemmatizers being analyzed perform very well on the evaluation dataset they were evaluated on. To check the consistency of these findings in other corpora, we performed an evaluation using the independent and free Nemlar corpus [20], tagged with lemma labels and containing 237,337 words. We did not include in our comparative analysis the Arabic lemmatizers, Farasa and Alkhalil. Farasa was not included as the output in the program does not have diacritics limiting the ability to completely disambiguate it. Alkhalil was omitted since it was trained on Nemlar corpus, the same sample that we evaluated, and this may have bias in it and resulted in the unfairness of the comparison. The outcome as in Table 1 shows that there is a decline in the accuracy as compared to the findings that had initially been reported by the original authors. Hence, we realized that lemmatizers are very critical in NLP application. Therefore, we aimed to advance to come up with a new lemmatizer that would have a better performance.

Table 1: Accuracy of various lemmatizers on the Nemlar test corpus

| Lemmatizer | Accuracy |
|---|---|
| Madamira | 85.80% |
| CaMelTools | 85.20% |
| SinaTools | 76% |

## 3 Methodology

This section gives a detailed summary of development of the lemmatizer and highlights the successive improvements using which performance improved and linguistic issues were tackled. The methodology is having three major iterations of the lemmatizer. Each one of them makes specific contributions to accuracy and to the treatment of out-of-vocabulary words.

### 3.1 Linguistic resources

The development of our lemmatizer is based on four essential resources:

- A corpus C, labeled at the lemma level for the training and testing phases of the lemmatizer.
- A word vocabulary Vw, containing all words that BiLSTM model can process as input.
- A morphological lexicon Lmor, linking words in Vw to their morphological information.
- A lemma vocabulary Vl, listing all potential outputs of the BiLSTM model.

#### 3.1.1 Labeled Corpus *C* at the lemma level

To train our lemmatizer, we built a rich and diverse annotated corpus extracted from the corpus presented in [21]. This corpus contained approximately 100 million words and was compiled from several online news sites as well as e-books. It reflects the linguistic diversity and regional variations of Arabic, as it originates from several Arab countries and covers a wide range of domains.

From this corpus, we derived a subcorpus *C* of about 1 million words, which was semi-automatically annotated with lemma tags [22]. In the first stage, the Alkhalil lemmatizer was applied to assign a lemma tag to each word in subcorpus *C*, after which manual verification was carried out by two linguistic annotators. This annotated subcorpus serves as the foundation for the training and evaluation phases of our lemmatizer.

#### 3.1.2 Word vocabulary $V_w$

As part of our model's development, a key step involved constructing a vocabulary list that captures the richness of the Arabic language. This ensures the BiLSTM model could learn the structures and nuances present in the initial corpus. We extracted the 100,000 most frequent words from the corpus developed in [21], including particles and stop words, to serve as the basis for the embedding phase, during which words were transformed into feature vectors. The numerical representation enables artificial intelligence algorithms to process the words, allowing the model to learn semantic relationships and enhancing its performance in tasks such as homonyms (words that share the same spelling but have different meanings) disambiguation and contextual understanding.

#### 3.1.3 Morphological lexicon $L_{mor}$

In the subsequent steps, we need to use the different potential lemmas of each word in the vocabulary $V_w$. To identify these lemmas, we use the morphological analyzer AlKhalil Morpho Sys [13], which analyzes words in isolation and provides all possible segmentations into proclitics, stems and enclitics for each analyzed word. The analyzer also supplies the lemma, root, pattern, POS tag and syntactic state for each segmentation. Using this information, we constructed the lexicon $L_{mor}$, which links each word in $V_w$ to all its potential lemmas provided by the Alkhalil Morpho Sys. Table 2 shows a sample of this lexicon.

Table 2: A sample of the morphological lexicon $L_{mor}$.

| Lemmas | Vocabularies |
|---|---|
| عِلْمِيَّة - عَلَمِيَّة - عَلَمِيّ – عِلْمِيّ | العلمية |
| مَرَّة – مِرَّة – مُرَّة | للمرة |
| تُرْبِيّ – تُرْبِيَّة – تَرْبِيَة – تَرَبِّيَة – تَرْبِيَّة | التربية |

#### 3.1.4 Lemma vocabulary $V_l$

From the morphological lexicon of lemmas $L_{mor}$ associated with vocabulary words $V_w$, we generated a lemma vocabulary $V_l$ containing 41,864 lemmas.
We present in table 3 the statistics related to all these linguistic resources.

Table 3: Linguistic resource statistics.

| Resource | Size |
|---|---|
| Word vocabulary $V_w$ | 100,000 |
| Lemma vocabulary $V_l$ | 41,864 |
| Labeled Corpus $C$ | 1,000,000 |

### 3.2 Lemmatizer architecture

#### 3.2.1 Baseline Lemmatizer

An initial implementation of the lemmatizer was described as a Bidirectional Long Short-Term Memory (BiLSTM) model which is called the baseline lemmatizer. The reason why this BiLSTM was chosen is because it is capable of capturing long-range dependencies and contextual relationships in text series. It uses two cells of LSTM that process the input sentence in both ways forward and backward, which resembles the process of reading a sentence twice. This provides the model with a better insight into the context in which a word is used and results in more accurate lemmas predictions, by taking advantage of the two-way aspect. Outputs of the neural network are lemmas which are the result of each word fed into the neural network. Therefore, the vocabulary $V_w$ is the input of network and the vocabulary $V_l$ is the output.
We trained and tested this model on the labeled corpus $C$ divided into 80% training and 20% testing in a random manner. We are using our own BiLSTM lemmatizer that can take sequence of length up to 57 tokens, each token is in the form of 128-dimensional embedding vectors all of which characterizes the semantic property of the token in the language. The LSTM cells of the hidden layer will have 128 units each. For training

purposes, a batch size of 32 was used and 10 epochs. The results of the accuracy and error rate on the test set are summarized in Table 4.

Table 4: Accuracy and error rate of the baseline model in the test set.

| Model | Accuracy | Error rates |
|---|---|---|
| Baseline BiLSTM Lemmatizer | 97.825% | 2.175% |

The baseline model demonstrated strong performance, achieving an error rate of approximately 2.175%. This high accuracy can be partly explained by the fact that both the training and test sets were derived from the same corpus *C*. To evaluate the model's generalization ability on an independent test set, we conducted additional testing using the Nemlar corpus [20]. For this evaluation, the baseline model was retrained on the entire *C* corpus before being tested on the Nemlar dataset. The performance results from this assessment are presented in Table 5.

Table 5: Accuracy and error rate of the baseline model on the Nemlar corpus.

| Model | Accuracy | Error rates |
|---|---|---|
| Baseline BiLSTM Lemmatizer | 80.76% | 19.24% |

We observe a significant decline in the performance of the baseline model when tested on an independent corpus. Moreover, comparing these results with those of state-of-the-art lemmatizers presented in Table 1, we note that the baseline model performs worse than the Madamira and CaMelTools lemmatizers.
The limited performance of the baseline model can be attributed to its operational mechanism, which requires selecting a lemma from the exhaustive output lemma dictionary for each input word, instead of restricting the selection to morphologically compatible lemmas. To enhance the accuracy of the baseline BiLSTM lemmatizer, we introduced a second version of the lemmatizer (the filtered lemmatizer), which incorporates advanced morphological filtering.

### 3.2.2 Filtered BiLSTM Lemmatizer

To enhance the accuracy of the baseline model, we developed a second version of the lemmatizer that incorporates advanced morphological filtering. This approach utilizes the morphological information of the analyzed word to refine the selection of potential lemmas. Instead of searching for the lemma across the entire dictionary, the model now considers only a reduced set of candidates that are morphologically compatible with the given word. By narrowing the search space, this method significantly reduces ambiguity and improves prediction accuracy. The integration of morphological analysis with the BiLSTM neural network enables this enhanced version of the lemmatizer to achieve superior performance by focusing the model's attention toward the most relevant options.
Specifically, we introduced a morphological filter between the BiLSTM's hidden layer output and the final softmax activation function σ (see Figure 1). This filter operates by multiplying the vector $H_t$, corresponding to the word $w_t$ and computed by the hidden layer, with a binary vector Ft. The non-zero elements of Ft are set to 1 at positions corresponding to the potential lemmas of $w_t$ in the lemma vocabulary $V_l$. Each word in the word vocabulary $V_w$ is stored alongside its potential lemmas in the lexicon $L_{mor}$. This filtering mechanism ensures that the lemmatizer selects a lemma only from the morphologically compatible candidates provided by the AlKhalil Morpho Sys analyzer. By leveraging morphological information more effectively, this approach significantly improved the lemmatizer's accuracy (see Table 6).

The evaluation of this model was conducted under the same conditions established for the baseline model. Specifically, we utilized the C corpus for training and the Nemlar corpus for testing. Table 6 presents the results of this evaluation, alongside with the results detailed in Table 1 and Table 5, for comparative analysis.

Fig. 1: Architecture of the filtered BiLSTM lemmatizer.

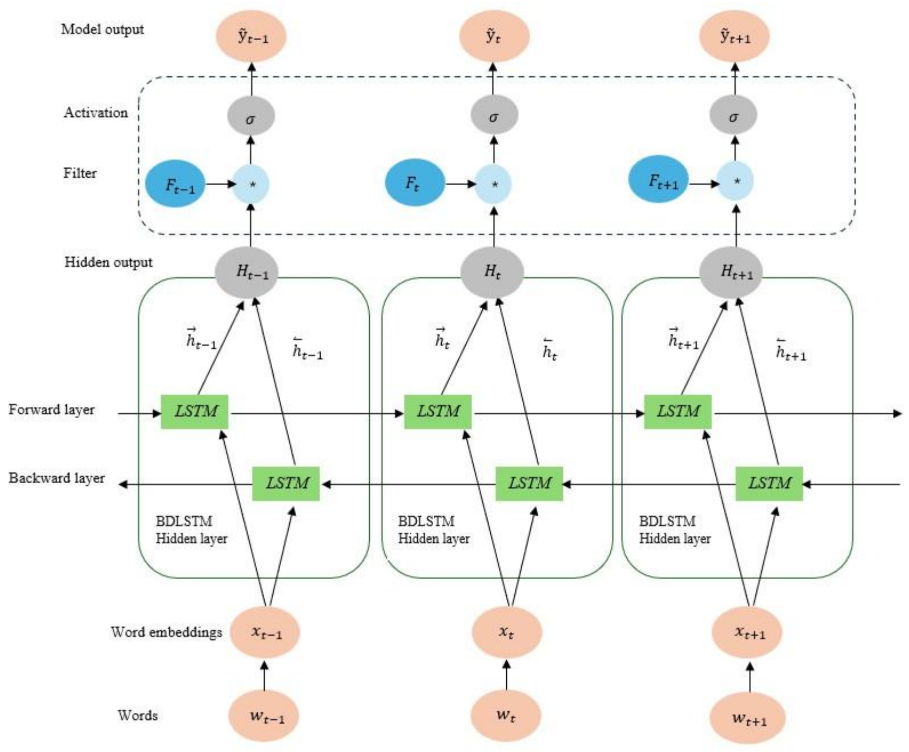


Table 6: Accuracy and error rate of the baseline model on the Nemlar corpus.

| Model | Accuracy | Error rates |
|---|---|---|
| Baseline BiLSTM Lemmatizer | 80.76% | 19.24% |
| Filtered BiLSTM lemmatizer | 89.06% | 10.94% |
| Madamira | 85.80% | 14.20% |
| CaMelTools | 85.20% | 14.80% |
| SinaTools | 76% | 24 % |
| Baseline BiLSTM Lemmatizer | 80.76% | 19.24% |
| Filtered BiLSTM lemmatizer | 89.06% | 10.94% |

The effectiveness of morphological filter is evident in these results. The inclusion of morphological information resulted in a significant error rate drop and the error rate improved by more than 8 percentage points. The baseline BiLSTM lemmatizer had an error rate of 19.24% and the filtered lemmatizer reduced this down to a mere 10.94%. In addition, we compare our results with the best lemmatizers in the world, which highlights the strength of our methodology. Our filtered model displays a significant decrease in the error rates and it scores more than 4 points higher than Madamira (10.94% vs. 14.80%) and almost 4 points higher than CaMelTools (10.94% vs. 14.80%). These findings relate to the high level of accuracy attained through incorporation of morphological knowledge.

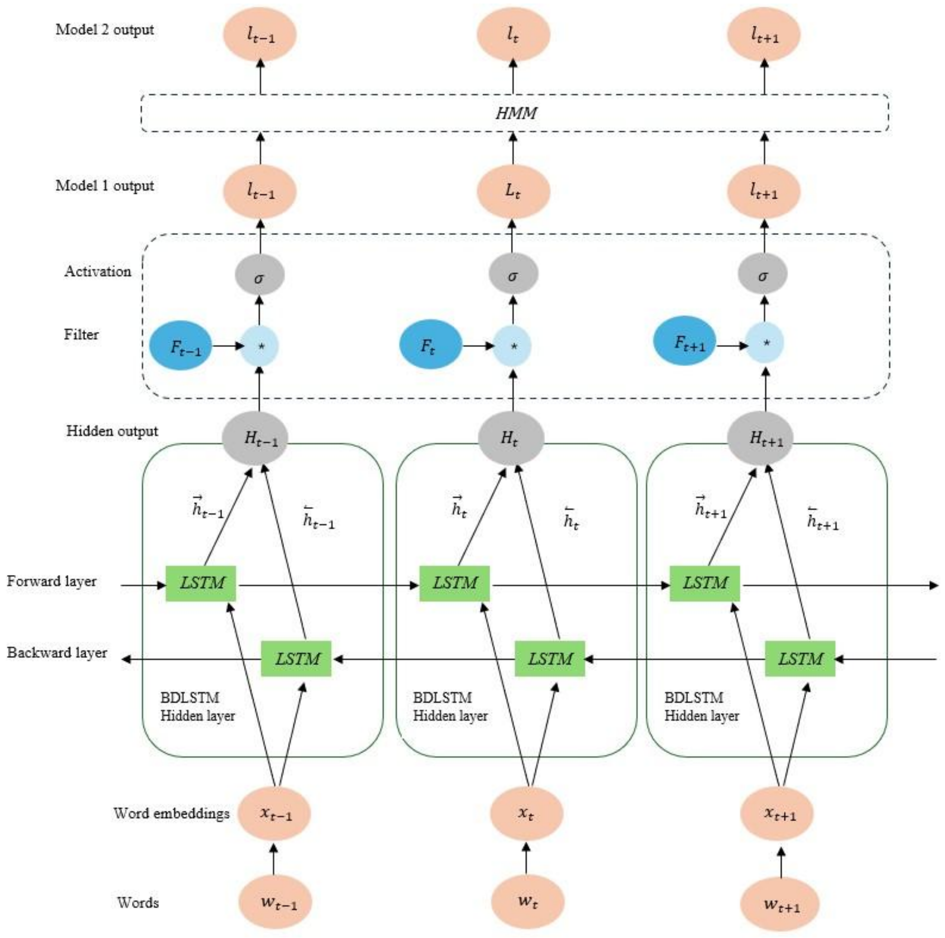
The study of the lemmatization errors of the filtered model revealed that a large part of the errors may be attributed to the effect of out-of-vocabulary words (the words in the test set

that are not represented in the $V_w$ vocabulary of the model). In the next section, We resolved this issue by training our filtered lemmatizer in a statistical layer with Hidden Markov Models.
In order to alleviate the adverse effects of the out-of-vocabulary words on the lemmatizer, we added a statistical layer to the filtered BiLSTM model. We start by explaining the operating principle of the Alkhalil lemmatizer that was designed in [11] and was the inspiration of the addition of this statistical layer to our model. Alkhalil lemmatizer is a system of morphological disambiguation of the Arabic language. It takes a milder approach that incorporates two modules to determine the right lemmas to be used on words in a sentence. The former module does a context-free analysis of each word in the Alkhalil Morpho Sys analyzer and thus produces a list of possible lemmas. The second module takes advantage of the context of the sentence: a statistical disambiguation module, which is a statistical component based on HMM models. This component picks the most likely lemma out of the initial candidates offered by the morphological analysis.

### 3.2.3 HMM BiLSTM Filtered Lemmatizer

Our HMM-filtered BiLSTM lemmatizer is an extension of the filtered BiLSTM version. For words belonging to the vocabulary $V_w$, the list of candidate lemmas is restricted to the single lemma generated by the filtered BiLSTM lemmatizer. On the other hand, for OOV words (i.e. absent from $V_w$), the list of candidate lemmas consists of the lemmas generated by the Alkhalil Morpho Sys analyzer. This configuration implies that only OOV words present an ambiguity in terms of their lemmatization. The additional layer of the HMM-filtered BiLSTM lemmatizer uses an HMM model to identify, from the list of lemmas associated with each OOV word, the most likely lemma based on its context.
The structure of the HMM-filtered BiLSTM lemmatizer is shown in figure 2. We indicate the special lemma assigned to a word $w_t$ in the vocabulary $V_w$ by the Filtered BiLSTM Lemmatizer model as $l_t$. In the case of wt not in the Vw vocabulary, Lt is the full list of lemmas that were produced by the Alkhalil Morpho Sys analyzer.

Fig. 2: Architecture of the HMM BiLSTM filtered lemmatizer.



Recall that HMMs are a class of probabilistic graphical models used to predict a sequence of unknown variables $(X_t)_t$, taking values in the set of hidden states (the lemmas), from a sequence of observed variables $(Y_t)_t$, taking values in the set of observed states (the words). Our goal is to find the most likely sequence of lemmas $(l_1^*, l_2^*, \ldots, l_k^*)$ for the words in the sentence $Ph = \{w_1, w_2, \ldots, w_k\}$, satisfying the following relation:

$$(l_1^*, \ldots, l_k^*) = \underset{\substack{l_i^{j_i} \in L_i \\ 1 \le i \le k}}{argmax}\, Pr\left(l_1^{j_1}, \ldots, l_k^{j_k} | w_1, \ldots, w_k\right)$$

where $L_i = \{l_i^1, l_i^2, \ldots, l_i^{n_i}\}$ denotes the set of candidate lemmas associated with the word $w_i$. Recall that $n_i = 1$ if $w_i \in V_W$. To determine the most probable sequence of lemmas $(l_1^*, l_2^*, \ldots, l_k^*)$, we suppose that the HMM is of order 1, i.e., the processes $(X_t)_t$ and $(Y_t)_t$ satisfy the following two Markov conditions:

- $Pr\left(X_{t+1} = l_j / X_t = l_i, \ldots, X_1 = l_1\right) = Pr\left(X_{t+1} = l_j / X_t = l_i\right) = a_{ij}$

  where $a_{ij}$ is the transition probability from hidden lemma $l_i$ to hidden lemma $l_j$

- $Pr\left(Y_t = w_r / X_t = l_i, Y_{t-1} = w_{r_{t-1}}, X_{t-1} = l_{i_{t-1}} \ldots, Y_1 = w_{r_1}, X_1 = l_{i_1}\right)$

  $= Pr(Y_t = w_r / X_t = l_i) = b_i(w_r).$

  where $b_i(w_r)$ is the probability of observing word $w_r$ given the hidden lemma $l_i$, and we use the algorithm of Viterbi [24], which is particularly suitable for the search for the optimal path. It is based on two functions:

- $\varphi(t, l_t^r)$ which represents the maximum probability among all paths of length $t-1$ leading to the lemma $l_t^r$, i.e.:
$$\varphi(t,\, l_t^r) \;=\; \max_{\substack{l_i^{r_i} \in L_i \\ 1 \le i \le t-1}} \; Pr\left(l_1^{r_1}, \cdots, l_{t-1}^{r_{t-1}}, l_t^r / w_1, \cdots, w_{t-1}, w_t\right)$$
- $\psi(t, l_j^t)$ which memorizes the previous lemma that maximized the function $\varphi$:
$$\psi(t, l_t^r) = arg \max_{l_{t-1}^s \in L_{t-1}} \varphi(t-1, l_{t-1}^r)\, Pr(\, l_t^r / l_{t-1}^s)$$

The two Markov hypotheses allow the function $\varphi$ to be computed recursively as follows:
$$\varphi(t, l_t^r) = \left(\max_{l_{t-1}^s \in L_{t-1}} \varphi(t-1, l_{t-1}^s) \times \Pr(\, l_t^r / l_{t-1}^s)\right) \Pr(\, w_t / l_t^r).$$
To extract the optimal path, we apply the Viterbi algorithm according to the following four steps:

**First step (initialization):** For *1 ≤ k ≤ n₁*, calculate $\varphi\left(1, l_1^k\right)$ the probability that the sentence begins with $w_1$ accompanied by the lemma $l_1^k$.

**Second step (recursive calculation)**: For *2 ≤ t ≤ n* and *1 ≤ r ≤ nₜ*,
$$\varphi(t, l_t^r) = \left(\max_{l_{t-1}^s \in L_{t-1}} \varphi(t-1, l_{t-1}^s) \times \Pr(\, l_t^r / l_{t-1}^s)\right) \Pr(\, w_t / l_t^r)$$
$$\psi(t, l_t^r) = \arg \max_{l_{t-1}^s \in L_{t-1}} \varphi(t-1, l_{t-1}^s) \Pr(\, l_t^r / l_{t-1}^s)$$

**Third step (final state):**
$$\psi(n+1) = \arg \max_{l_n^s \in L_n} \varphi(n, l_n^s)$$

**Fourth step (deduction of the optimal path):**
$$l_n^* = \psi(n+1) \quad \text{and for } t = n\text{-}1 : 1 \quad l_t^* = \psi(t, l_{t+1}^*)$$

To run the Viterbi algorithm, we first need to estimate the initial probability $\pi = (\pi_i)$, where $\pi_i = P(X_1 = l_i)$, the transition matrix $A = (a_{ij})$ and the emission matrix $B = (b_i(t))$. These parameters are estimated by applying to a labelled training corpus $C$ using a maximum likelihood method [Manning & Schütze, 1999], supplemented by the Absolute Discounting smoothing technique [Ney & Essen, 1991]. This smoothing reduces the impact of OOV words. The parameters are therefore estimated as follows:

$$\pi_i = \frac{\max\,(m_i - D, 0)}{M} + \frac{D}{M} P_{abs}(l_i) N_{1+}(I\, \bullet)\;;\; a_{ij} = \frac{\max\,(n_{ij} - D, 0)}{n_i} + \frac{D}{n_i} P_{abs}(l_j) N_{1+}(l_i\, \bullet)\;;$$
$$b_i(t) = \begin{cases} \frac{m_{it} - D}{n_i}, & if\ m_{it} \neq 0 \\ \frac{N_i \times D}{n_i \times Z_i}, & elsewhere \end{cases}$$

with the constant $D = 0.5$ and $P_{abs}\left(l_j\right) = \frac{n_j}{N}$,

- $m_i$: number of sentences of the training corpus $C$ of which $l_i$ is the lemma of the first word.
- M: number of sentences in the training corpus *C*.
- $N_{1+}(I\, \bullet)$: number of lemmas corresponding to the words that appear at the beginning of sentences in corpus $C$.
- $n_{ij}$: number of occurrences of the transition from the lemma $l_i$ to the lemma $l_j$ in the corpus *C*.

- $n_i$: number of words in the corpus $C$ annotated with the lemma $l_i$.
- $N_{1+}(l_i \bullet)$: number of words whose lemmas appear at least once after the lemma $l_i$ in the corpus $C$.
- $m_{it}$: number of times that the word $w_t$ appear with the lemma $l_i$ in the corpus $C$.
- $N_i$: number of words annotated in the corpus $C$ with the lemma $l_i$.
- $Z_i$: number of words not annotated in the corpus with the lemma $l_i$, for which the Alkhalil analyser still generates this lemma.

The proposed solution combines the advantages of filtered BiLSTM and HMM models, and subsequently offers a more accurate lemmatization system even in the presence of out-of-vocabulary words.

To evaluate the HMM-filtered BiLSTM lemmatizer, we compared its performance with that of six previous models. For this purpose, we trained the HMM-filtered BiLSTM lemmatizer on corpus $C$, the same dataset used to train both the baseline and filtered models. The open-source Alkhalil lemmatizer was also retrained on corpus $C$ to ensure a fair comparison. Finally, all six models were evaluated on the independent Nemlar corpus. Table 7 presents the results, including both accuracy and processing speed (measured in words analyzed per second).

Table 7: Accuracy and error rate of the baseline model on the Nemlar corpus.

| Model | Accuracy | Speed |
|---|---|---|
| Baseline BiLSTM Lemmatizer | 80.76% | 776 |
| Filtered BiLSTM lemmatizer | 89.06% | 374 |
| HMM-filtered BiLSTM Lemmatizer | 90.10% | 297 |
| Alkhalil lemmatizer | 85.51% | 2,637 |
| Madamira | 85.80% | 786 |
| CaMelTools | 85.20% | 235 |
| SinaTools | 76% | 38,973 |

Experimental results indicate that integrating an HMM-based statistical layer into the filtered BiLSTM lemmatizer leads to a notable improvement in accuracy. This additional layer effectively addresses the issue of out-of-vocabulary words, resulting in an accuracy increase of over one percentage point (90.10% for the HMM-filtered BiLSTM lemmatizer vs. 89.06% for the filtered BiLSTM lemmatizer). Among the evaluated tools, SinaTools exhibits the lowest accuracy, while Madamira, CaMelTools and the Alkhalil Lemmatizer achieve comparable performance. However, the two latest versions of our lemmatizer (HMM-filtered BiLSTM and Filtered BiLSTM) significantly outperform the other lemmatizers. In terms of processing speed, SinaTools is the fastest, followed by the Alkhalil Lemmatizer and then Madamira.

Table 8 presents examples of sentences containing OOV words that are not analyzed by the filtered BiLSTM but are correctly handled by the HMM-filtered BiLSTM.

Table 8: Some examples from the output of the HMM-filtered BiLSTM.

| Output HMM-filtered BiLSTM | Output filtered BiLSTM | Sentence |
|---|---|---|
| تَمَّ - إِثْبَات - سِنّ - طَلَب - دَفْتَر- قَيْد - سِجِلّ | تَمَّ -إِثْبَات - سِنّ - طَلَب - OOV - قَيْد - سِجِلّ | يتم إثبات السن بالطلب وبدفتر قيد السجل |
| احْتَفَلَ - مِصْرِيّ - ثَالِث - عِشْرِين- مِنْ - يُولْيُو - كُلّ - عَام | احْتَفَلَ - مِصْرِيّ - OOV - عِشْرِين - مِنْ - يُولْيُو - كُلّ -عَام | يحتفل المصريون بالثالث والعشرين من يوليو كل عام |
| قَدَّرَ - قِيمَة - وَقْت - شَكْل - كَبِير - أَحْسَنَ - تَنْظِيم - اسْتِغْلَال | قَدَّرَ - قِيمَة - وَقْت - شَكْل - كَبِير -OOV - تَنْظِيم -OOV | يقدرون قيمة الوقت بشكل كبير ويحسنون تنظيمه واستغلاله\| |
| حَدَّدَ - قَانُون - حُكْم - مُوَازَنَة - مُؤَسَّسَة - هَيْئَة - عَامّ - حِسَاب | حَدَّدَ - قَانُون - حُكْم - مُوَازَنَة -مُؤَسَّسَة - هَيْئَة - عَامّ - OOV | يحدد القانون أحكام موازنات المؤسسات والهيئات العامة وحساباتها |

Table 9 presents examples of sentences where the system fails to correctly lemmatize all the words in the sentence. In cases where the lemmatizer does not return the correct lemma, the correct form is provided in parentheses.

Table 9: Some examples of errors produced by the HMM-filtered BiLSTM.

| Output HMM-filtered BiLSTM | Sentence |
|---|---|
| سُرْعَة - تَوْلِيد - كَلَام – مُكَافَأَة (مُكَافِئ) - سُرْعَة - نُطْق - بَشَرِيّ - طَبِيعِيّ | سرعة توليد للكلام مكافئة لسرعة النطق البشري الطبيعية |
| سِيَاسَة - تَوْظِيف - هَامَ - مُرْتَكِز (مُرْتَكِ) | سياسة التوظيف وأهم مرتكزاتها |
| زُهَيْر (زَهِير) - كَاظِم - عُبُود | زهير كاظم عبود |
| وَضَعَ (وَضْع) - أَسَاس - شَأْن - إِدَارِيّ - شَرِكَة - تَنْظِيم - شَأْن - عَامِل | وضع أسس الشؤون الإدارية بالشركة وتنظيم شؤون العاملين |
| عَمِلَ - عَلَى - أَفَادَ (تَفَاد) - ذَا - وَسِيلَة - تَالِي | ونعمل على تفادي ذلك بالوسائل التالية |

## 6 Conclusion and Future Work

Lemmatization is an important part of preprocessing many NLP applications, especially where the language being studied is morphologically rich, as is the case with Arabic. In this work, we presented two variations of a hybrid lemmatizer that is particularly Arabic-oriented lemmatizer. The initial one is a bidirectional recurrent neural network with a morphological filtering mechanism that uses morphological information during the Alkhalil Morpho Sys analyzer. This method improves the selection of lemmas to a great extent since the output space is confined to morphologically valid lemmas. Nevertheless, neural models can be vulnerable to cases of out-of-vocabulary words and hence cannot generalize well. In order to overcome this shortcoming, we constructed a second one with a statistical layer using Hidden Markov Models. This improvement allows the model to address unseen words more appropriately with the help of probabilistic transitions between lemmas.

Experimental tests prove the fact that both hybrid versions obtain significant enhancements over the current Arabic lemmatizers, being more accurate and stable. These results can point to the efficiency of transitional solutions to the language complexity of Arabic, specifically, the integration of deep learning with rule-based and statistical algorithms. The

solutions to our research are opening the way to more precise and flexible lemmatization solutions, which will be able to serve a broad spectrum of NLP tasks, such as information retrieval, text summarization and machine translation.
The future studies include enhancement of the proposed lemmatizers by looking into alternative neural network models and the application of reinforcement learning algorithms, especially through the application of large language models. We also intend to generalize these hybrid solutions to other morphological analysis solutions such as stemmers, POS taggers and diacritization systems.

## References

[1] Habash, N.Y.: Introduction to Arabic Natural Language Processing. Morgan & Claypool Publishers, (2010)

[2] Boudchiche, M., Mazroui, A. (2020): Spline functions for Arabic morphological disambiguation. Applied Computing and Informatics

[3] Dilekh, T. (2011): Implémentation d'un outil d'indexation et de recherche des textes en arabe. PhD thesis, Université de Batna 2

[4] Hull, D.A. (1996): Stemming algorithms: A case study for detailed evaluation. Journal of the American Society for Information Science 47(1), 70–84

[5] Gunther, T., Furrer, L. (2013): Lemmatisation and pos-tagging for historical texts: a comparison of six taggers and two lemmatization methods. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 27–35

[6] Nassiri, N., Lakhouaja, A., Cavalli-Sforza, V. (2018): Modern standard Arabic readability prediction. In: Lachkar, A., Bouzoubaa, K., Mazroui, A., Hamdani, A., Lekhouaja, A. (eds.) Arabic Language Processing: From Theory To Practice. *Communications in Computer and Information Science*, vol. 782, pp. 120–133. https://doi.org/10.1007/978-3-319-73500-9 9 .

[7] Berrichi, S., Mazroui, A. OCT 21-27, 2018: Benefits of morphosyntactic features on english-arabic statistical machine translation. In: ElMohajir, M., AlAchhab, M., ElMohajir, B., Jellouli, I. (eds.) 2018 IEEE *5th international congress on information science and technology* (IEEE CIST'18), pp. 244–248 (2018). IEEE; IEEE Morocco Sect; Al Akhawayn Univ; Royaume Maroc, Ministere Habous Affaires Islamiques; IEEE Morocco Comp & Commun Joint Chapter. IEEE *5th International Congress on Information Science and Technology* (IEEE CiSt), Marrakech, Morocco.

[8] Touahri, I., Mazroui, A.: Studying the effect of characteristic vector alteration on arabic sentiment classification. Journal Of King Saud Universitycomputer And Information Sciences 33(7), 890–898 (2021) https://doi.org/10.1016/j.jksuci.2019.04.011

*[9]* Hammouda, T., Jarrar, M., Khalilia, M.: Sinatools: Open source toolkit for arabic natural language processing. Procedia Computer Science 244, 388–396 (2024) https://doi.org/10.1016/j.procs.2024.10.213 . *6th International Conference on AI in Computational Linguistics*

[10] Abdelali, A., Darwish, K., Durrani, N., Mubarak, H. (2019): Farasa: A fast and furious segmenter for arabic. (2016). https://doi.org/10.18653/v1/N16-3003

[11] Boudchiche, M., Mazroui, A.: A hybrid approach for Arabic lemmatization. International Journal of Speech Technology 22(3), 563–573

[12] Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., Inoue, G., Eryani, F., Erdmann, A., Habash, N. (2020): Camel tools: An open-source python toolkit for Arabic natural language processing. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 7022–7032

[13] Boudchiche, M., Mazroui, A., Bebah, M.O.A.O., Lakhouaja, A., Boudlal, A. (2017): Alkhalil morpho sys 2: A robust arabic morpho-syntactic analyzer. Journal of King Saud University-Computer and Information Sciences 29(2), 141–146
[14] Pasha, A., Al-Badrashiny, M., Diab, M., 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. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation* (LREC'14), pp. 1094–1101. European Language Resources Association (ELRA), Reykjavik, Iceland. http://www.lrec-conf.org/proceedings/lrec2014/pdf/593Paper.pdf
[15] Maamouri, M., Graff, D., Bouziri, B., Krouna, S., Bies, A., Kulick, S. (2010): Ldc standard arabic morphological analyzer (Sama) version 3.1 ldc2010l01. Web Download. Philadelphia: Linguistic Data Consortium
[16] Y. Zhang, R.B.K.D. C. Li (2015): Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In: Proc. 2015 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol., Association for Computational Linguistics, pp. 42–52 https://doi.org/10.3115/v1/N15-1005
[17] Freihat, A.A., Abbas, M., Bella, G., Giunchiglia, F. (2018): Towards an optimal solution to lemmatization in Arabic. Procedia Computer Science 142, 132–140 https://doi.org/10.1016/j.procs.2018.10.468 . Arabic Computational Linguistics
[18] Nazih, W., Fashwan, A., El-Gendy, A., Hifny, Y.: Ibn-ginni (2023): An improved morphological analyzer for arabic. ACM Transactions on Asian and Low-Resource Language Information Processing
[19] Buckwalter, T. (2004): Buckwalter Arabic morphological analyzer (Bama) version 2.0. linguistic data consortium (ldc) catalogue number ldc2004l02. Technical report, ISBN1-58563-324-0
[20] Attiya, M., Yaseen, M., Choukri, K. (200: Specifications of the Arabic written corpus produced within the Nemlar project. URL http://www. nemlar. org.(Cit´e en pages 29, 39, 60 et 67)
[21] Belayachi, S., Mazroui, A. (2021): Comparison of the lexicons of contemporary and the classical arabic language. *4ème édition des Journées Doctorales de l'Ingénierie de la Langue Arabe*, Mohammed V University Rabat
[22] Belayachi, S., Mazroui, A. (2022): Semi-automatic labeling of an MSA corpus. *Conférence internationale Nouvelles Tendances en Informatique (NTI2022)*, Mohamed First University Oujda
[23] Ney, H., & Essen, U. (1991, April). On smoothing techniques for bigram-based natural language modelling. In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing* (pp. 825-828). IEEE.
[24] Neuhoff, D. (1975). The Viterbi algorithm as an aid in text recognition (Corresp.). *IEEE Transactions on Information Theory*, *21*(2), 222-226.

## Notes on contributors

***Samir Belayachi*** is currently pursuing a PhD in Natural Language Processing. His research interests include Arabic language processing, particularly Arabic morphology, syntactic analysis and morphology-aware deep learning models. His work aims to bridge linguistic knowledge and neural methods to enhance the accuracy and robustness of Arabic NLP applications.

***Azzeddine Mazro*ui** is a Full Professor of Computer Science and Director of the NLP team at the Research in Computer Science Laboratory, Mohammed First University. His research interests encompass Arabic morphological and syntactic analysis, machine translation, sentiment analysis, and corpus development within the fields of computational linguistics and information science.