Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 142 (2018) 132–140

Procedia Computer Science

www.elsevier.com/locate/procedia

The 4th International Conference on Arabic Computational Linguistics (ACLing 2018), November 17-19 2018, Dubai, United Arab Emirates

Towards an Optimal Solution to Lemmatization in Arabic

Abed Alhakim Freihat[a,*], Mourad Abbas[b], Gábor Bella[a], Fausto Giunchiglia[a]

[a]*Department of Information Engineering and Computer Science, University of Trento, via Sommarive, 5, 38123 Trento, Italy*
[b]*Computational Linguistics Department, CRSTDLA, Algeria*

Abstract

Lemmatization—computing the canonical forms of words in running text—is an important component in any NLP system and a key preprocessing step for most applications that rely on natural language understanding. In the case of Arabic, lemmatization is a complex task because of the rich morphology, agglutinative aspects, and lexical ambiguity due to the absence of short vowels in writing. In this paper, we introduce a new lemmatizer tool that combines a machine-learning-based approach with a lemmatization dictionary, the latter providing increased accuracy, robustness, and flexibility to the former. Our evaluations yield a performance of over 98% for the entire lemmatization pipeline. The lemmatizer tools are freely downloadable for private and research purposes.

Keywords: lemmatization; Arabic; natural language processing; machine-learning-based lemmatization; dictionary-based lemmatization

1. Introduction

Lemmatization consists of assigning to the surface form of each word in a text its corresponding *lemma*, that is, its canonical form as the word is commonly found in a dictionary. As such, lemmatization decreases morphological variations in text, in turn facilitating operations such as semantic analysis [1], information retrieval [2], question answering [3], or search [4]. For this reason, lemmatization is a crucial preprocessing operation in a wide range of applications that involve dealing with natural language.

The difficulty of the lemmatization task greatly depends on the nature of the language. In morphologically poor languages such as English, lemmatization can be considered an easy task already solved by simple normalization rules and a list of exceptional cases. For morphologically rich—highly inflecting or agglutinative—languages such as Arabic, on the other hand, it remains difficult and requires diverse, more complex approaches that are often specific to the language [5].

The major challenges specific to Arabic lemmatization, and NLP in general, are the rich morphology [6], which includes agglutinative properties [7], and the optional—and mostly omitted—marking of short vowels in writing [8]. This last property results in pervasive lexical ambiguity even considering the corresponding part of speech: for example, the past tense verb صرت could be vocalized as صِرْتُ with the corresponding lemma صَارَ/*become* or as صَرَّ with

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: abdel.fraihat@gmail.com

1877-0509
10.1016/j.procs.2018.10.468

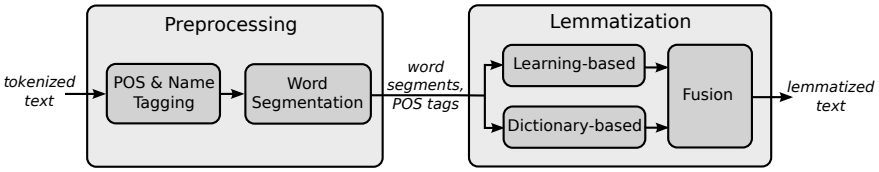


Fig. 1. The high-level NLP pipeline architecture for lemmatization

the corresponding lemma ضَرَّ/*grit*. As deciding for the correct lemma is ultimately a word sense disambiguation problem, such cases put considerable stress on the quality of lemmatization. Tools that are capable of outputting multiple solutions in an order of preference are in this sense more robust as they potentially allow the disambiguation problem to be delayed to later, syntactic or semantic processing steps.

There has been extensive research so far on solving the lemmatization problem for Arabic. While several approaches were proposed, there are no more than a handful of actual tools available. Existing tools typically combine multiple techniques to achieve efficient lemmatization. The *Alkhalil* lemmatizer [9] first applies morpho-syntactic analysis to the input sentence to generate all potential word surface forms. Then, among these only one form is selected per word using a technique based on hidden Markov models. The accuracy of the tool is reported to be about 94%. Another lemmatizer is *Madamira* [10] which relies on preliminary morphological analysis on the input word that outputs a list of possible analyses. As a second step, it predicts the correct lemma using language models. The accuracy of the tool is 96.6%. The *Farasa* lemmatizer [11] uses a dictionary of words and their diacritizations ordered according to their number of occurrences. The accuracy reported for Farasa is 97.32%.

Beside these tools, there are other proposed approaches: for example, [12] propose a pattern-based approach while [13] and [14] present rule-based solutions.

In this paper, we present a new, freely available lemmatization tool that is composed of the fusion of a machine-learning-based classifier as a main lemmatizer and of an auxiliary dictionary-based lemmatizer. The underlying idea is that the classifier, that relies on context, is well-suited to solving cases of lexical ambiguity while the dictionary-based extension provides an extra performance boost, easy extensibility by new lemmas (e.g., neologisms), as well as the possibility to retrieve multiple possible lemmas per word form for subsequent analysis. The two lemmatizers are, however, implemented as separate tools and can also be used independently, each one yielding results beyond 95% of accuracy. Their high performance is partly explained by the preceding, fast and lightweight steps of POS tagging, morphological analysis, and word segmentation—earlier contributions of the authors reused in this work—that provide rich morphological information to the lemmatizers.

The lemmatizer was implemented as a component in a new, free, comprehensive pipeline for Arabic NLP [15] and is freely available for private or research purposes.[1] Both the 3-million-entry dictionary and the 2-million-token annotated corpus used to train the classifier were entirely generated by the authors and are contributions of this paper.

2. The Lemmatization Pipeline

As is the case of most approaches, our lemmatizer operates over an input pre-annotated by previous preprocessing steps. The pipeline specific to our method is shown in figure 1 and is composed of the following main steps:

1. *Preprocessing:* taking whitespace-tokenized Arabic text in input, we pre-annotate the text through the following operations:

   (a) *POS and name tagging:* tokens are annotated by a machine-learning-based sequence labeler that outputs both POS and named entity tags, later used by the lemmatizer;
   (b) *word segmentation:* using the POS output, cliticized words are segmented into a proclitic, a base word, and an enclitic, making the subsequent lemmatization step simpler.

[1] http://www.arabicnlp.pro/alp/

2. *Lemmatization:* the segmented and pre-annotated text is fed into the following lemmatizer components:

   (a) *dictionary-based lemmatizer:* words are lemmatized through dictionary lookup;
   (b) *machine-learning-based lemmatizer:* words are lemmatized by a trained machine learning lemmatizer;
   (c) *fusion:* the outputs of the two lemmatizers are combined into a single output.

In the following sections we present each component in detail, with a focus on lemmatization components as the pre-processors have already been discussed in our earlier work [15].

3. Preprocessing

The role of preprocessing is to enrich the input of the lemmatizer (and other subsequent components) by morphological and other contextual information, based on which the lemmatization task is simplified. With respect to state-of-the-art tools [9, 10], the preprocessing required by our lemmatization approach is lightweight and fast. The tokenized input text is first enriched by part-of-speech tagging, implemented as a fast machine-learning-based sequence labeler. Then it is further segmented by a very simple word segmenter component that further reduces the complexity and ambiguity of words. The simplicity of the process, presented in detail in [15], is explained by the rich morphological information output by the POS tagger, based on which word segmentation becomes a nearly trivial task. For example, the word وباستخدام (*and by using*) is first identified by the POS tagger to contain two proclitics and an inflected base word. On the basis of this result the segmenter outputs the word segment sequence <و, ب, استخدام> (<*and*, *by*, *using*>) and the corresponding POS tags <C, P, SMN> (<conjunction, preposition, singular masculine noun>).

In this section we provide a brief overview of the preprocessing from the point of view of the subsequent lemmatization task.

3.1. POS and Name Tagging

In the lemmatization pipeline, the main goal of the POS and name tagger is to reduce the ambiguity of words by extracting information from their morphology and context:

- whether the word is part of a name or not;
- the corresponding part of speech;
- whether the word contains proclitics or enclitics (prefixes or suffixes).

The tagger is implemented as a single machine learning component, described in our earlier work [15] and freely available online.[2] In the following we provide only a brief presentation of it, in order to demonstrate the level of detail it provides to the subsequent lemmatizer.

```
<TAG>     ::= <PREFIX> <BASETAG> <POSTFIX>
<BASETAG> ::= <POSTAG> | <NERTAG>
<PREFIX>  ::= <PREFIX> | <PROCLITIC> "+" | ""
<POSTFIX> ::= <POSTFIX> | "+" <ENCLITIC> | ""
```

A tag is thus composed of a mandatory base tag and of zero or more proclitics and enclitics concatenated with the "+" sign indicating word segments. A base tag, in turn, is either a POS tag or a named entity (NER) tag. We do not consider name tags in the rest of the paper as they are irrelevant for lemmatization beyond the fact that names are skipped by the lemmatizer.

On a coarse-grained level, POS tags are divided into the following categories:

[2] http://www.arabicnlp.pro/alp/

```
و_C+PSTV أضاف أن_ACC الأبحاث_D+PIN والتحريات_C+D+PFN التي_REL
باشرتها_PSTV+PRO مصالح_PIN الأمن_D+SMN مكنت_PSTV من_P تحديد_SMN هوية_SFN
انئنين_NM من_P المشتبه_D+SMAJ فيهم_P+PRO ،_PX المتورطين_D+PMN بشكل_P+SMN
مباشر_SMAJ في_P عمليات_PFN القرصنة_D+SFN المعلوماتية_D+SFAJ ،_PX
فيما_P+REL ينحصر_PRSV دور_SMN باقي_NQ الموقوفين_D+PMN في_P ربط_SMN
```

Fig. 2. An example piece of text annotated with parts of speech and morphology

```
و_C أضاف_PSTV أن_ACC ال_D أبحاث_PIN و_C ال_D تحريات_PFN التي_REL
باشرت_PSTV ها_PRO مصالح_PIN ال_D أمن_SMN مكنت_PSTV من_P تحديد_SMN هوية_SFN
انئنين_NM من_P ال_D مشتبه_SMAJ في_P هم_PRO ،_PX ال_D متورطين_PMN ب_P
مباشر_SMAJ في_P عمليات_PFN ال_D قرصنة_SFN ال_D معلوماتية_SFAJ ،_PX
فيما_P+REL ينحصر_PRSV دور_SMN باقي_NQ ال_D موقوفين_PMN في_P ربط_SMN ال_D
```

Fig. 3. An example output of the word segmenter

```
<POSTAG> ::= <NOUN> | <ADJECTIVE> | <VERB> | <ADVERB> | <PREPOSITION> | <PARTICLE>
```

Coarse-grained parts of speech [16] are enriched with verb tenses and morphological features, the goal of which is to solve a large part of lexical ambiguity problems already on the level of POS tagging:[3]

```
<NOUN>      ::= ( <NUMBER> <GENDER> "N" ) | "PIN"   /* PIN: broken plural noun */
<ADJECTIVE> ::= ( <NUMBER> <GENDER> "N" ) | "PIAJ"  /* PIAJ: broken plural adjective */
<NUMBER>    ::= "S" | "D" | "P"                     /* Singular, Dual, or Plural */
<GENDER>    ::= "M" | "F"                           /* Masculine or Feminine */
<VERB>      ::= ( <PASSIVE> <TENSE> "V" ) | "IMPV"  /* IMPV: imperative verbs */
<PASSIVE>   ::= "P" | ""                            /* empty for active verbs */
<TENSE>     ::= "PST" | "PRS"                       /* Past, Present */
```

Examples of noun tags are SMN and SFN meaning *singular masculine* and *singular feminine*, respectively. They enable us to differentiate between, for example, the words رجل_SMN/*man* and رجل_SFN/*leg*.

Examples of verb tags are PSTV and PRSV (past and present tense) that enable us, for example, to differentiate between تحمل_PSTV/*(he) sustained* and تحمل_PRSV/*(she) carries*. In figure 2, we provide an example of an annotated text.

3.2. Word Segmentation

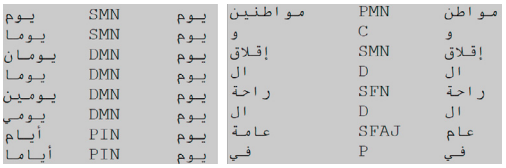
Word segmentation is executed based on the segmentation information embedded within POS tags. Its serves a double goal: to reduce the amount of distinct word forms, resulting in smaller and more robust lemmatizers, as well as to reduce lexical ambiguity due to multiple possible interpretations. For example, word segmentation of reduces the number of possible word forms of the lemma قلم from several hundreds of clitisized nouns { قلم, قلمي, بقلمي, وبقلمي, القلم,...} to six forms { قلم, قلمان, قلما, قلمين, قلمي, أقلام} only. On the other hand, word segmentation reduces the lexical ambiguity in cases such as لسعة which may be single word (*sting*) or a clitisized word (*for capacity*).

The input of the segmentation component is a word and its corresponding POS tag. The output is a list of tokens that correspond to the proclitic, base, and enclitic components of the POS tag. Given that the presence of clitics is identified upstream, segmentation becomes a simple rule-based string splitting task, as described in detail in [15]. An example output of the segmentation tool is shown in figure 3.

4. Lemmatization

The principal component of our lemmatization approach is a machine-learning-based classifier. It takes as input word segments and their corresponding POS tag, also taking context (words and tags) into account. The learning-based

[3] We omitted some tags, such as the named entity tags, which are irrelevant for lemmatization.



يوم	SMN	يوم
يوما	SMN	يوم
يومان	DMN	يوم
يوما	DMN	يوم
يومين	DMN	يوم
يومي	DMN	يوم
أيام	PIN	يوم
أياما	PIN	يوم

(a)

مواطنين	PMN	مواطن
و	C	و
إقلاق	SMN	إقلاق
ال	D	ال
راحة	SFN	راحة
ال	D	ال
عامة	SFAJ	عام
في	P	في

(b)

Fig. 4. Examples of the contents of (a) the dictionary and (b) of the corpus used to train the classifier.

approach is justified by the inherent ambiguity of diacritic-free Arabic words whose meanings are typically deduced, by humans and machines alike, from context. While the preliminary POS tagging resolves a great deal of ambiguity, some cases still remain such as the verb form يكون which may be the verb form of the verb كان, or the verb كون.

A downside of learning-based lemmatization is that more rare and exceptional cases, such as أسنة (*spears*), may not be covered by its training corpus, which leads to lemmatization mistakes. The addition of new cases requires the re-training of the classifier. Another inconvenience is that classifiers—such as OpenNLP that we used—typically commit on a single output result, which may or may not be correct. In case of such ambiguity, from the full set of possible lemmas further NLP processing steps may be able to provide a correct results based on, e.g., syntactic or semantic analysis. In order to support these cases, we complement the learning-based lemmatizer by a dictionary-based one. The dictionary lemmatizer can be run independently, but we also provide a simple fusion method that combines the results of the two lemmatizers as described below.

Both lemmatizer components were implemented using the Apache machine learning based toolkit OpenNLP,[4] using the Maximum Entropy classifier.

4.1. Dictionary Lemmatization

The dictionary of a db lemmatizer consists of a text file containing, for each row, a word, its POS tag and the corresponding lemma, each column separated by a tab character. An example of the of the dictionary is shown in column (A) of figure 4.

In case of ambiguous word forms (i.e., a word form POS-tag pair that has several lemmas), the corresponding lemmas are separated by ”#” character. For example lemmas of the word form تزور are زور#زار

In the following we describe the method we used to build the dictionary. The used corpus is the same corpus we used for segmenting, POS-tagging, and named entity recognition as described in our previous work [15].

1. **Segmentation:** The corpus was segmented as explained in the previous section. The result of this step was generating a segmented corpus that contains more than 3.1 million segmented tokens.
2. **POS-tag based classification:** In this step, we classified the word forms according to their POS-tag.
3. **Inherent feminine and adjectival feminine classification:** In this step, we classified the feminine nouns into inherent feminine and adjectival feminine nouns. For example, the noun أسرة_SFN/*family* is inherent feminine while the noun أسيرة_SFN/*prisoner* is adjectival. This differentiation is important because the lemma of adjectival nouns is the masculine singular form of the noun أسير while there is no masculine singular lemma for أسيرة.
4. **Plural type classification:** In this step, we classified the singular, and dual nouns (after extracting their singular forms) according to their plural type into six classes as shown in Table 1. This classification enables us to build the possible word number_gender forms of a given lemma automatically. For example, the class SMN_PMN has six different possible number_gender forms. On the other hand, using the feminine classification lists in previous step, enabled us to differentiate between the SMN_PFN and SFN_PFN. Tn the class SMN_PFN, the lemma of a singular feminine noun (SFN) is the singular masculine noun (SMN). In the class SFN_PFN on the other hand,

[4] http://opennlp.apache.org/docs/1.9.0/manual/opennlp.html#tools.cli.lemmatizer

the lemma is the singular feminine noun itself. The adjectives were classified into three classes. The first class is similar to the class SMN_PMN which allows six different word forms. The second class contains a seventh possible form which is the broken plural adjective form. The third class contains PIAJ as a single possible plural form. For example, ماهر belongs to this class since it has two possible plural forms ماهرون and مهرة.

5. **Lemmas extraction:** This step is semi-automatic as follows:

   - **Manual:** Assigning the lemmas to broken plural nouns and adjectives was performed manually.
   - **Automatic:** Based on the morphological features in the tags, it was possible to extract lemmas for singular, dual, masculine plurals, feminine plurals adjectives and nouns. We also used rules to extract the verb lemmas such as removing the affixes وا.

6. **lemmas Enrichment:** Using the lemmas from the previous step, we have enriched the corpus with new verbs, adjectives, and nouns. For example, if the lemma of a plural noun or adjective was missing, we added it to the noun and adjective lemmas lists.
7. **Dictionary generation:** The files produced so far are as follows.

   (a) **Noun files:** Three files for masculine, feminine, and foreign nouns. The lemmas in these files were classified according to Table 1. There is a fourth file that contains quantifiers, pronouns, adverbs, ...
   (b) **Adjectives:** Three files for adjectives, comparatives, and ordinal adjectives. The lemmas in the adjective file are classified according to Table 1.
   (c) **Verbs:** One file that contains all extracted verb lemmas.

   Using these files, the dictionary was generated as follows:

   - **Nouns and adjectives generation:** According to the plural class, the noun and adjective forms were generated. The ا case ending, or changing ة to ت were also considered in this step.
   - **Verbs generation:** For each verb in the verb lemmas list, we automatically generated the verb conjugations in present, past, imperative cases. We considered also accusative (فعل منصوب) and asserted verbs (فعل مجزوم).
   - **Dictionary building:** Using the results from previous step, we built the dictionary as described in Figure 4 (B), where the lemmas of ambiguous surface forms were joined into a single string using the # operator.

Table 1. Plural classes

Class	Possible Word Forms	Example
SMN_PMN	SMN,SFN,DMN, DFN, PFN, PMN	مؤمنون,مؤمنات,مؤمنتان,مؤمنان,مؤمنة,مؤمن:مؤمن
SMN_PFN	SMN,DMN,PFN	إجراءات,إجراءان,إجراء,:إجراء
SFN_PFN	SFN,DFN,PFN	كتابات,كتابتين,كتابة,:كتابة
SMN_PIN	SMN,DMN,PIN	أعلام,علمان,علم,:علم
SFN_PIN	SFN,DFN,PIN	أسر,أسرتان,أسرة,:أسرة
FWN_PFN	FWN,DMN,PFN	تلفزيونات,تلفزونان,تلفزيون,:تلفزيون

4.2. Machine Learning Lemmatization

The format of the corpus here is similar to the dictionary described in previous section. The only difference is that we order the entries according to their original position in the sentence in the segmented corpus. An empty line

indicates the end of a sentence. An example of the of the training corpus is shown in column (B) of figure 4. We used the segmented corpus from previous section to build the lemmatization corpus was performed in two steps:

1. **Lemmas assignation:** In this step, we used a dictionary lemmatizer to assign the word forms to their corresponding lemmas. In case of preposition, particles, and numbers, the lemma of the word form was a normalized form of the word form itself.The lemmas of named entities were also the named entities them selves. In this step, if a word form was ambiguous, all its possible lemmas were assigned.
2. **Validation:** In this step, we disambiguated the lemmas of the ambiguous word forms Manually.

The size of the generated corpus is 3,229,403 lines. The unique word forms after discarding the digits is 59,049 as specified in Table 2.

Table 2. Lemmas and unique word forms distribution in the corpus of the mlb lemmatizer

POS	Number of lemmas	Number of word forms
Noun	18,165	26,337
Adjective	6,369	13,703
Verb	4,258	19,009
Named entity	20,407	20,407
Particle	605	649

In a final step, we added all generated word forms and their corresponding lemmas from the dictionary described in previous section to the corpus. This increased the size of the corpus to 3,890,737 lines.

4.3. Fusion

While the learning-based lemmatizer outputs for each word a single candidate lemma, from the dictionary multiple solutions could be retrieved even for a single part of speech (for example, the verb form يسلم can be a verb form of the verb سلم *gave up*, or أسلم *converted to Islam*). The goal of the simple fusion component is to produce a final result from these solutions. The final output is a list of one or more lemmas in a decreasing order of confidence.

The idea underlying the fusion method is that we *usually* trust the dictionary to be capable of providing a correct solution space (a small set of possible lemmas), while we *usually* trust the classifier to return the most likely lemma from the previous set. However, in the case of out-of-corpus words the classifier may return incorrect results extrapolated from similar examples, such as returning the lemma تهمن for the word form يتهمن. Thus, whenever a lemma is returned by the classifier that is not included in the dictionary, it will still be included as a solution but with a lower confidence.

Accordingly, our simple fusion method is as follows. We take as input the results output by the two lemmatizers, namely $L_{\mathrm{DIC}} = \{l_1, ..., l_n\}$ for the dictionary-based one and $L_{\mathrm{CL}} = \{l\}$ for the classifier-based one, and output L_{F}, the fusion result. We start by comparing the results of the two lemmatizers:

- if $|L_{\mathrm{DIC}}| = 1$ and $l_1 = l$, i.e., the outputs are identical, then the solution is trivial, we return either output and we are done: $L_{\mathrm{F}} = \{l\}$;
- otherwise, two further cases are distinguished:
  - if $l \in L_{\mathrm{DIC}}$, that is, the dictionary contains the classification output, then we prioritize the result of the classifier by making it first (i.e., the preferred lemma): $L_{\mathrm{F}} = \{l, l_1, ..., l_n\}$;
  - otherwise, we add the classifier result as the last element: $L_{\mathrm{F}} = \{l_1, ..., l_n, l\}$.

5. Evaluations

For evaluation we used a corpus of a 46,018-token text, retrieved and assembled from several news portals (such as Aljazeera news portal[5] and Al-quds Al-Arabi news paper[6]). We excluded from the evaluation the categories of tokens that cannot be lemmatized: 5,853 punctuation tokens, 3,829 tokens tagged as named entities, 482 digit tokens, and 10 malformed tokens (i.e., containing typos, such as أوروبيةبالموافق instead of أوروبية بالموافق). Thus the number of tokens considered was 35,844.

In order to have a clear idea of the efficiency of the lemmatization pipeline, we evaluated it in a fine-grained manner, manually classifying the mistakes according to the component involved. This allows us to compute a comprehensive accuracy for the entire pipeline as well as evaluate individual components: the POS tagger, the segmenter, each lemmatizer, as well as the fusion lemmatization. The evaluation data files are available online.[7]

Table 3. Types of mistakes committed by the learning-based lemmatizer, and their proportions

Type of mistake	Occurrences	Example
POS tag (coarse-grained) mistakes	199	تبريج_SMN instead of تبريج_PRSV
Morphological tag (fine-grained) mistakes	201	كروم_SMN instead of كروم_PIN
Segmentation tag mistakes	103	أخذتنا_PSTV instead of أخذت_PSTV and نا_PRO
Classifier mistakes: nonexistent lemma	158	جدا instead of وجد for يجدوا_PRSV
Classifier mistakes: wrong disambiguation	12	كان instead of كون for يكونوا_PRSV
Dictionary mistakes: missing word form	1,207	كاشطة,دواعم,قرفصاء
Fusion mistakes	50	نلن ,عوالم ,دواعم

The fine-grained evaluation is summed up in table 3.[8] *Nonexistent lemma* stands for cases where the POS tag and the segmentation were correct, yet the classifier gave a wrong, non-linguistic result. *Wrong disambiguation* means that the lemmatizer chose an existing but incorrect lemma for an ambiguous word form.

Table 4. Accuracy values computed for various components of the lemmatization pipeline

Component	Evaluation method	Accuracy
preprocessing	all mistakes (POS, morphological, segmentation)	98.6%
classifier-based lemmatizer	in isolation	99.5%
classifier-based lemmatizer	in isolation, built-in OpenNLP cross-validation	99.7%
classifier-based lemmatizer	entire pipeline	98.1%
dictionary-based lemmatizer	in isolation	96.6%
dictionary-based lemmatizer	entire pipeline	95.2%
fusion lemmatizer	entire pipeline	98.4%

The accuracy measures reported in table 4 were computed based on the results in table 3. On these we make the following observations. The performance of preprocessing (98.6%) represents an upper bound for the entire lemmatization pipeline. In this perspective, the near-perfect results of the classifier (99.5% when evaluated in isolation, 98.1% on the entire pipeline) are remarkable. We cross-checked these results using the built-in cross-validation feature of

[5] http://www.aljazeera.net/

[6] http://www.alquds.co.uk/

[7] http://www.arabicnlp.pro/alp/lemmatizationEval.zip

[8] While after tagging and segmentation the number of (segmented) tokens rose to 62,694, we computed our evaluation results based on the number of unsegmented tokens.

OpenNLP and obtained similar results (99.7%). The dictionary-based lemmatizer reached a somewhat lower yet still very decent result (96.6% in isolation, 95.2% on the entire pipeline), due to the 1207 OOV word forms. The fusion of the two lemmatizers, finally, improved slightly on the classifier: of the 170 mistakes made by the classifier, 120 could be correctly lemmatized using the dictionary. Thus the fusion method reached a full-pipeline result of 98.4%, only a tiny bit worse than the performance of preprocessing itself.

6. Conclusion and Future Work

We presented an optimization approach for Arabic lemmatization, based on the combination of machine learning and a lemmatization dictionary, that provides excellent accuracy. Beside the result itself, the addition of a lemmatization dictionary provides additional robustness to the underlying NLP pipeline. Firstly, it makes the lemmatizer easy to extend by new lemmas that could potentially be mislabeled by the classifier. Secondly, it allows the lemmatizer to return not only one result but an order list of candidate lemmas, allowing the decision to be delayed to subsequent NLP components.

Both the machine learning model and the dictionary were built using a corpus of 2.2 million tokens annotated and manually validated by the authors. The dictionary, the trained model, and corresponding tools are all free for research purposes upon request.

The presented tool was implemented as a component of the *ALP* comprehensive NLP pipeline. We plan to extend the current pipeline with new components such as a vocalizer, a phrase chunker, a dependency parser, or a multiword expression detector.

References

[1] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surv.*, vol. 41, pp. 10:1–10:69, Feb. 2009.
[2] V. Balakrishnan and L.-Y. Ethel, "Stemming and lemmatization: A comparison of retrieval performances," vol. 2, pp. 262–267, 01 2014.
[3] A. A. Freihat, M. R. H. Qwaider, and F. Giunchiglia, "Using grice maxims in ranking community question answers," in *Proceedings of the Tenth International Conference on Information, Process, and Knowledge Management, eKNOW 2018, Rome, Italy, March 25-29, 2018*, pp. 38–43, 2018.
[4] F. Giunchiglia, U. Kharkevich, and I. Zaihrayeu, "Concept search," in *The Semantic Web: Research and Applications* (L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl, eds.), (Berlin, Heidelberg), pp. 429–444, Springer Berlin Heidelberg, 2009.
[5] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," vol. 8, pp. 14:1–14:22, Dec. 2009.
[6] M. Gridach and N. Chenfour, "Developing a new approach for arabic morphological analysis and generation," *CoRR*, vol. abs/1101.5494, 2011.
[7] K. Shaalan, "A survey of arabic named entity recognition and classification," *Comput. Linguist.*, vol. 40, pp. 469–510, June 2014.
[8] O. Hamed and T. Zesch, "A Survey and Comparative Study of Arabic Diacritization Tools," *JLCL: Special Issue - NLP for Perso-Arabic Alphabets.*, vol. 32, no. 1, pp. 27–47, 2017.
[9] M. Boudchiche, A. Mazroui, M. Ould Abdallahi Ould Bebah, A. Lakhouaja, and A. Boudlal, "Alkhalil morpho sys 2: A robust arabic morpho-syntactic analyzer," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 29, pp. 141–146, Apr. 2017.
[10] A. Pasha, M. Al-Badrashiny, M. T. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, "Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic.," in *LREC*, vol. 14, pp. 1094–1101, 2014.
[11] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, "Farasa: A fast and furious segmenter for arabic," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pp. 11–16, Association for Computational Linguistics, San Diego, California, 2016.
[12] M. Attia, A. Zirikly, and M. T. Diab, "The power of language music: Arabic lemmatization through patterns," in *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon, CogALex@COLING 2016, Osaka, Japan, December 12, 2016*, pp. 40–50, 2016.
[13] E. Al-Shammari and J. Lin, "A novel arabic lemmatization algorithm," in *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, AND '08, (New York, NY, USA), pp. 113–118, ACM, 2008.
[14] T. El-Shishtawy and F. El-Ghannam, "An accurate arabic root-based lemmatizer for information retrieval purposes," *CoRR*, vol. abs/1203.3584, 2012.
[15] A. A. Freihat, G. Bella, H. Mubarak, and F. Giunchiglia, "A single-model approach for arabic segmentation, pos tagging, and named entity recognition," in *2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*, pp. 1–8, April 2018.
[16] I. Zeroual, A. Lakhouaja, and R. Belahbib, "Towards a standard part of speech tagset for the arabic language," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 2, pp. 171 – 178, 2017. Arabic Natural Language Processing: Models, Systems and Applications.