African Journal of Science, Technology and Social Sciences. 2(2) 2022, 77-84

OPEN ACCESS

AFRICAN JOURNAL OF SCIENCE, TECHNOLOGY AND SOCIAL SCIENCES

Journal website: https://journals.must.ac.ke

MERU UNIVERSITY OF SCIENCE AND TECHNOLOGY

A Publication of Meru University of Science and Technology

SwaRegex: a lexical transducer for the morphological segmentation of Swahili verbs

George Mutwiri,*[1] Mutua Makau,[1] Amos Omamo[1]

[a] *School of Computing and Informatics. Meru University of Science and Technology.*

ARTICLE INFO

KEY WORDS

Natural language processing

Morphological segmentation

Low-resourced language

Lemmatization

Part-of-speech tagging

Word segmentation

Segmentation

ABSTRACT

The morphological syntax of the Swahili verb comprises 10 slots. In this work, we present SwaRegex, a novel rule-based model for the morphological segmentation of Swahili verbs. This model is designed as a lexical transducer, which accepts a verb as an input string and outputs the morphological slot occupied by the morphemes in the input string. SwaRegex is based on regular expressions developed using the C# programming language. To test the model, we designed a web scraper that obtained verbs from an online Swahili dictionary. The scrapper separated the corpus into two datasets: dataset A, comprising 163 verbs Bantu origin; and dataset B, containing the entire set of 715 non-Arabic verb entries obtained by the web scrapper. The performance of the model was tested against a similar model designed using the Xerox Finite State Tools (XFST). The regular expressions used in both models were the same. SwaRegex outperformed the XFST model on both datasets, achieving a 98.77% accuracy on dataset A, better than the XFST model by 41.1%, and a 68.67% accuracy on dataset B, better than the XFST model by 38.46%. This work is beneficial to prospective learners of Swahili, by helping them understand the syntax of Swahili verbs, and is an integral teaching aid for Swahili. Search engines will benefit from the lexical transducer by leveraging its finite state network when lemmatizing search terms. This work will also create more opportunities for more research to be done on Swahili.

* Corresponding author: George Mutwiri. Email: mutwirigeorge3@gmail.com

https://doi.org/10.58506/ajstss.v1i2.119

AFRICAN JOURNAL OF SCIENCE, TECHNOLOGY AND SOCIAL SCIENCES ISSN :2958-0560

Introduction

Morphological segmentation is a subfield of morphological analysis that involves identifying morpheme boundaries within text (Liu et al., 2021). Words in a language comprise a set of morphemes arranged so as to conform to the syntax rules of the language. Natural language processing (NLP) acts as a bridge between computers and natural human language. For this to be possible, it is important that computers be able to identify the morpheme constructs that make up words in the language in conjunction with the morphological rules that apply to the language. This in turn makes it possible for the computer to analyze words based on their constituent morphemes and also to generate words based on morphological rules.

While researchers have recently shown more interest in poorly resourced languages, there remains scant work on certain aspects of these languages. In this paper, we present SwaRegex, a lexical transducer for Swahili verbs. The paper is structured as follows. We first review related literature. Following that, finite transducers, the morphology of the Swahili language and the Swahili verb, the design of SwaRegex and its XFST counterpart, and experimental results are discussed.

Related Works

There are two methods by which morphological segmentation can be performed: rule-based and machine learning methods. Machine learning methods improve their performance automatically by relying on experience and datasets (Mahesh, 2018). These methods can be supervised, semi-supervised, or unsupervised. Supervised methods rely on labeled data to learn, while unsupervised methods rely on unlabeled data. Semi-supervised methods, however, rely on both labeled and unlabeled data (Li et al., 2021). Machine learning takes time in order for training to occur. It is also incapable of distinguishing between derivational and inflectional morphology in natural language processing (Mott et al., 2020).

Rule-based methods rely on a set of rules that define how the correct input is mapped to the relevant output. Rule-based methods are suitable when computational language resources are unavailable or inadequate (Ammari & Zenkouar, 2021a). These rules are usually in the form of regular expressions, which are translated into finite state networks. This means that rule-based methods achieve more accurate results since they perform only according to the available rules. However, these methods post poor results on input whose rules have not been defined.

In rule-based methods, regular expressions can be compiled into lexical transducers. A lexical transducer is a finite state transducer that accepts a surface form of a word as input and produces its lexical form, and vice versa. The lexical form of a word is made up of its lemma and tags that depict its morphological roles. The lemma of a word is its dictionary representation (Möhrs & Cajo, 2020). Creating rules for lexical transducers necessitates an understanding of finite state automata as well as the morphological rules and syntax of the target language.

Morphological analysis focused on morphological alternations in Kinyarwanda was presented by (Muhirwe, 2007). The analyzer was rule-based, employing finite state machines. The model was implemented in Twolc (Two-level compiler) and based on the Xerox finite state tool. The author did not, however, publish the results of their model.

(Katushemererwe & Issue, 2010) presented a finite state method for the automatic analysis of Runyakitara nouns. All noun lexemes in the language were built into fsm2. Nouns were extracted from a Runyakitara dictionary and manually coded into noun subclasses. The system was evaluated using a dataset extracted from a weekly newspaper and an orthography reference book, both in a different language. Their model recorded a precision of 80% on 4472 words and a recall of 80% on the 5599-word corpus.

Runyagram, a formal system for the morphological segmentation of Runyakitara verbs based on the fsm2 interpreter, was presented by (Fridah & Thomas, 2010). Just like their similar model for nouns (Katushemererwe & Issue, 2010), Runyagram finite state transducer was comprised of a special symbol file, a grammar file, and a replacement rule file containing about 34 rules. The grammar contained about 330 rules and was converted into a finite-state acceptor containing about 1200 transitions and about 800 states. The system was tested against 3971 verbs from an orthography reference book and a dictionary of another Bantu language. The system scored a recall of 86% and a precision of 82%.

While the above research focuses on a variety of poorly resourced languages, our model is keen on the morphology of Swahili verbs. No work has been presented before on the morphological segmentation of Swahili verbs. Our model implements regular expressions, a rule-based technique for morphological segmentation. We chose rule-based over machine learning-based techniques because rule-based techniques are best suited for languages with limited resources (Ammari & Zenkouar, 2021a).

Methods and Materials

Method

Finite State Transducers

A finite state transducer is a finite state automaton that allows not only analysis of input against lexical forms, but also generation of the base form of a word given its lexical form. A finite state automaton is made up of five elements Q, ∑, q_0, F, and T. These elements have the following roles:

Q – a finite set of states

∑ - the alphabet (finite set of input symbols)

q_0 - a state in Q that is the start state of the automaton

F – a finite set of final states. F is a subset of Q

T – a transition function that determines when the automaton moves from one state to another

A finite state transducer has an additional element to its finite state automaton: the output function, which is responsible for yielding output from the automaton. (Abdulla et al., 2019; Gerdjikov, 2018) describe a finite state transducer as a finite automaton tool defined by:

Q a finite set of states;

Σ input alphabet;

Γ the output alphabet;

I ⊆ Q the set of initial states

F ⊆ Q the set of final states

δ ⊆ Q × (Σ ∪ {λ}) × (Γ ∪ {λ}) × Q, the transition function

Given a finite state transducer y:x, y denotes the lexical string while x, the surface string. The upper-side symbol is y while x is the lower-side symbol. When the input string is x, the transducer traverses its finite state automaton, searching for an arc labeled x leaving the start state. If such an arc exists, the network returns the lexical symbol, which in this case is y. This operation is called analysis/lookup, and it returns the lexical string for a given surface input. By traversing the finite state network backwards, this transducer can also be used to determine whether y is a valid string of the language. This process is called generation/ look down, and it takes as input the lexical strings and returns the corresponding surface strings (Karttunen, 2001).

Materials

This section is organized into two sub-sections. The first explores the Swahili language and the morphological structure of its verbs. The second section discusses the structure of SwaRegex.

Swahili

Swahili is an agglutinative (Shikali et al., 2019) East African language with over 80 million speakers. The language is written in the Roman alphabet and has only 5 vowels (a, e, i, o, u), 23 consonants (b, c, d, f, g, h, i, j, k, l, m, n, p, r, s, t, v, w, y, th, dh, gh, sh) and 6 nasal consonants (ny, ng', ng, nd, mb, mw). Every Swahili word concludes with a vowel (Ngugi et al., 2010). (Schönhof & Wilkans, 2012) suggest that Swahili morphology involves agglutination, the noun class system, concordance, conjugation (or inflection), and use of locatives.

The basic word order in Swahili is SVO (subject verb object), with the subject and object optional, for example, *Kijana anacheza mpira* (The young boy/girl is playing the ball). '*Kijana*' is the subject and '*mpira*' the object. Removing the subject and object in this example leaves only the verb "*anacheza*", which is a morphologically correct simple sentence (Kituku, 2019).

Morphological Syntax of the Swahili Verb

The following is the syntactical format for the Swahili verb (Deo, 2016):

Pre-Initial (*optional*): Negative marker *ha-* (NEG1); mutually exclusive with NEG2. When the Formative 1 slot is marked by the present continuous marker *-na-* and the Pre-Initial slot marked by *-ha-*, the final vowel should be marked by the negative present indicative *-i*.

Initial (*optional*): Subject Concord (SC); *Pronouns*

Post-Initial (*optional*): Negative marker *si-* (NEG2); mutually exclusive with NEG1; *conjunctive/ subjunctive*. When used, the final vowel/mood is marked by the subjunctive marker *-e*.

Formative 1 (*optional*): TAM; *-na-*, *-li-*, *-ta-*, *-ka-*, *-me-*, *-sha-*, -k*i-*, *-nge-*, *-ngali-*, *-ngeli-*

Formative 2 (*optional*): Subordinator (SUBO); *object/ relative pronoun; -ye-*. This slot also takes the same morphemes as the Post-Final.

Pre-radical (*optional*): Object concord (OC) *-ni-* or reflexive marker *-ji-*.

Verbal base: verb stem lexical base or root

Pre-Final (*optional*): verbal derivational suffixes (SUFF)

Final: TAM; *-e* (subjunctive); *-a*(indicative); *-i* (neg. pres. Indicative)

Post-Final (*optional*): Plural addressee marker (PLA)/ *plural imperative: -ni*; reference marker (REF)/ relative pronoun – particle *-o*. If this slot is occupied, the tense slot (Formative 1) is not marked at all.

Swahili verbal inflections

The verb is made up of a root, that remains unchanged no matter the inflection used (Mdee, 2016). Swahili verbal roots take on affixes when inflected.

Inflection on verbs is applied after the verbal root. An exception to inflection in Swahili is proper nouns: they are never inflected (Steinberger et al., 2011). Once inflected, the meaning of the verb changes, which also requires changes in the subject and/or object concordance. (Deo, 2016) suggests the following verbal inflections and the possible suffixes they take:

Active suffix (**ACT**): *-a, -i, -u, -e*

Applicative/ applied/ dative/ prepositional (**APPL**): *ia, -ea, -ilia, -elea*

Associative/ reciprocal suffix (**REC**): *-an*

Causative (**CAUS**): *-isha, -esha, -iza, -eza, -sha, -za, -ya, -sa*

Contactive suffix:(**CONT**) *-et, -at*

Conversive/ reversive / separative suffix: (**CONV**) – *-u-*, *--ul-*, *-ol-*, *-o-*

Durative suffix (**DUR**): *-a*

Intensive suffix (**INTE**): *-ili--*, *-ele-*

Passive (**PASS**): *-wa, -iwa, -ewa, -liwa, -lewa*

Potential suffix (**POTE**): *-ka*

Static suffix – (**STC**): *-ma*

Stative/ neuter suffix (**STV**): *-ka, -ika, -leka, -ika, -eka, -uka*

Reciprocal Stative suffix (**RECSTV**): *-na*

Reflexive suffix (**REFL**): *-ji-*

SwaRegex

The structure of SwaRegex was divided into two sections: the web scraper and the lexical transducer.

The Web Scraper

A web scrapper was designed to automatically scrape off verbs from the "Translation Dictionary" website (https://www.translationdirectory.com/dictionaries/dictionary031.htm). Since the website lists a dictionary of Swahili words, verbs are not placed in any particular order. However, verbs are denoted by a succeeding "*v.*" tag on entries in the dictionary. Verbs also appear in their inflected forms. The first character of each verb is a hyphen. In order to obtain only the roots of these verbs, the scrapper was tuned to remove the hyphen, the Pre-Final, and the Final morphemes. The scrapper was also tuned to ignore entries containing spaces and to separate verbs of Arabic origin from verbs of Bantu origin. Arabic-derived verbs are those that end in either "*e*", "i", and "*u*". This separation is because verbs of Arabic origin are inflected differently from those of Bantu orignin. Malformed verb entries were also excluded from the scraped data.

The scrapper saved two separate files: dataset A, which contained163 verb phrases that did not have any inflectional morphemes, and dataset B, which contained all the non-Arabic verb entries obtained from the website. In total, dataset B contained 715 entries.

The Lexical Transducer

SwaRegex was developed in Visual Studio using C# version 8.0. The model contained two sets of regular expressions. The first set defined morphemes for each morphological slot, while the second set described possible combinations of morphemes from the first set of expressions that form a morphologically correct verb.

To define the morphemes allowed in each morphological slot, ten variables were defined, one for each slot. Each variable was a regular expression specifying all the sets of morphemes allowed to occupy the morphological slot. These regular expressions were enclosed in regex groups so the transducer could identify the particular morphological slot when a match was successfully made for an input verb. However, some morpheme combinations in various slots were defined as separate variables due to the following reasons:

The morphemes occur in extremely separate contexts. For instance, the morpheme "*ji*" (reflexive), which only succeeds "*si*" (Post-Final).

The occurrence of a morpheme in more than one slot distorts the output of the transducer. Take for instance '*a*' appears at both the Initial and the Final. If the morpheme appears in any one slot within the input verb, the transducer marks both slots as occupied.

The regular expressions representing the morphological syntax of the Swahili verb were also constructed in the same way. These comprised the second set of expressions. String interpolation was used to dereference variables representing the morphological slots when constructing regular expressions. The matching function for the regular expression was set to ignore case during the matching operation.

The same set of regular expressions were copied onto the XFST tool. This tool has been used before in developing and testing finite-state transducers for various languages (Sahala et al., 2020). Next, we describe the implementation of the model in both SwaRegex and its XFST counterpart.

Implementation of SwaRegex

We implemented SwaRegex in Visual Studio as a console project in C# 8.0. The variables containing the first set of regular expressions that represented morphemes allowed in each morphological slot were defined as follows:

```
string P1 = "(?<P1>ha)";

string P2 = "(?<P2>wa|u|i|ya|vi|zi|u|pa|ku|m|tu|li|ki)";

string P3 = "(?<P3>si)";

string P4 = "(?<P4>na|ka|me|sha|taka|li)";

string P5 = "(?<P5>ye|yo|lo|cho|vyo|zo|po|ko|mo|ye)";

string P6 = "(?<P6>mw)";
string P8 = "(?<P8>an|az|i|ian|ik|iki|ikiw|ikiz|ikw|ili|iliw|ish|ishiw|iz|e|ek|ele|ew|esh|et|ez|k|lek|lew|li|liw|m|n|ol|sh|shan|s|u|uk|ul|uli|uliw|ush|w|y|z|zik|zw)";

string P9 = "(?<P9>i)";

string P10 = "(?<P10>ni)";
```

P7, the verb roots variable, was defined in the same way.

8 more slots were defined to avoid duplicates in the above morphological slots. This was to improve the performance of the models, where duplicates resulted in more than one output from the transducers. These slots are defined below:

```
string PA = "(?<PA>ja)";

string PB = "(?<PB>ji)";

string PD = "(?<PD>e)";

string PE = "(?<PE>a)";

string PF = "(?<PF>nge|ngeli|ngali)";

string PG = "(?<PG>ta)";

string PH = "(?<PH>o)";

string PJ = "(?<PJ>hu)";
```

Next, we show the second set of variables, which defines possible morpheme combinations that form a morphologically correct verb. These rules are defined as follows:

```
string Rule1 => $"^({P1})({P2})?({PA}|{PF}|{PG})?({P6}|{P10}|{PB}|{P2})?({P7})({P8}|{PE}|{PH})?({P9}|{PE})({P10})?$";

string Rule2 => $"^({P3})({PA}|{PF}|{PG})?({P2}|{P6}|{PB})?({P7})({P8}|{PE}|{PH})?({P9}|{PD}|{PE})({P10})?$";

string Rule3 => $"^({P2}|{PE})({P3}|{P4}|{PF})?({P2}|{P6}|{PB})?({P7})({P8}|{PE}|{PH})?({P9}|{PD}|{PE})({P10})?$";

string Rule4 => $"^({P2}|{PE})({P4}|{PF})({P5}|{PH})?({P2}|{P6}|{P10}|{PB})?({P7})({P8}|{PE}|{PH})?({P9}|{PD}|{PE})({P10})?$";

string Rule5 => $"^({P2}|{PE})({P4}|{PF}|{PG})?({P2}|{P6}|{P10}|{PB})?({P7})({P8}|{PE}|{PH})?({PE})({P5}|{PH})?({P10})?$";

string Rule6 => $"^({P2}|{PE})({P4}|{PF})?({P2}|{P6}|{PB})?({P7})({P8})?({PE})({P5}|{PH})?({P10})?$";

string Rule7 => $"^({P7})({P8})?({PE})({P10})?$";

string Rule8 => $"^({PJ})({P7})({P8})?({P9}|{PE})$";
```

The above ruleset was combined into one general regex, against which input was matched as follows:

```
var match = Regex.Match(verb, $"{Rule1}|{Rule2}|{Rule3}|{Rule4}|{Rule5}|{Rule6}|{Rule7}|{Rule8}", RegexOptions.IgnoreCase);
```

Implementation of the XFST model

The first set of regular expressions representing possible combinations of morphemes for each slot are as follows on the XFST terminal:

```
define P1 ha;
define P2 wa|u|i|ya|vi|zi|u|pa|ku|m|tu|li|ki;
define P3 si;
define P4 na|ka|me|sha|taka|li;
define P5 ye|yo|lo|cho|vyo|zo|po|ko|mo|ye;
define P6 mw;
define P8 an|az|e|ek|ele|ew|esh|et|ez|i|ian|ik|iki|ikiw|ikiz|ikw|ili|iliw|ish|ishiw|iz|k|lek|lew|li|liw|m|n|ol|s|sh|shan|u|uk|ul|uli|uliw|ush|w|y|z|zik|zw;
define P9 i;
define P10 ni;
```

P7, the verb roots, were also defined in the same way.

Next, the following command was passed to XFST to develop a finite state network against the above morphological slots:

```
read regex
[[P1] (P2) (PA | PF | PG) (P6 | P10 | PB | P2) [P7] (P8 | PH | PE) [P9 | PE] (P10)] |
[[P3] (PA | PF | PG) (PB | P2 | P6) [P7] (P8 | PH | PE) [P9 | PE | PD] (P10)] |
[[P2 | PE] (P3 | PF | P4 | P3  PF) (PB | P2 | P6) [P7] (P8 | PH | PE) [PD | PE | P9] (P10)] |
[[P2 | PE] [P4 | PF] (P5 | PH) (PB | P10 | P2 | P6) [P7] (P8 | PH | PE) [PD | PE | P9] (P10)] |
[[P2 | PE] (P4 | PF| PG) (PB | P10 | P2 | P6) [P7] (P8 | PH | PE) [PE] (P5 | PH)
```

```
(P10)] |
     [[P2 | PE] (P4 | PF) (PB | P2 | P6) [P7]
(P8) [PE] (P5 | PH) (P10)] |
     [[P7] (P8) [PE] (P10)]|
     [[PJ] [P7] (P8) [P9 | PH]]
     .o. [PA -> A, PB -> B, PD -> D, PE -> E,
PF -> F, PG -> G, PH -> H, PJ -> J, P1 -> 1,
P2 -> 2, P3 -> 3, P4 -> 4, P5 -> 5, P6 -> 6,
P7 -> 7, P8 -> 8, P9 -> 9, P10 -> 10];
```

The notation ".o." in XFST denotes composition. In this case, it implies that morphemes in an input verb that correspond to a slot, say PA, will cause the output of the transducer to be A. We used this notation to convert the finite state network into a transducer.

Analysis and Discussion

A total of eight morphological rules for the Swahili verb were defined. The web scraper obtained a total of 776 distinct Swahili verbs from the online dictionary. From these, 507 were roots, 163 were verbs of Bantu origin, and 61 were verbs of Arabic origin.

Both models were tested against the two datasets. The same set of rules was defined for each model. Each model was pointed to the same dataset. The models had no problems reading the file. However, for the XFST model, the encoding of the file had to be changed from UTF8 to ANSI. The performance of SwaRegex on both datasets is shown in Table 1, while the performance of the XFST model is shown in Table 2.

Dataset	Total verbs	Correctly segmented	Accuracy
A	163	161	98.77%
B	715	491	68.67%

Table 1: *Performance of SwaRegex on both datasets*

Dataset	Total verbs	Correctly segmented	Accuracy
A	163	95	57.67%
B	715	216	30.21%

Table 2: *Performance of the XFST model on both datasets*

Dataset A Test Results

SwaRegex correctly segmented 161 out of the 163 verbs in dataset A. This translated to 98.77% accuracy. This performance was possible because dataset A was devoid of noise. The XFST model correctly segmented 95 verbs. This translated to 57.67% accuracy. Since XFST does not provide a way of evaluating how it parses strings, it was not possible to evaluate its performance on the datasets. However, printing the finite state network at the top of its stack revealed that some paths were malformed. This was, therefore, attributed to the design limitations of XFST.

Dataset B Test Results

SwaRegex correctly segmented 491 entries from this dataset. This translated to an accuracy of 68.67%. The XFST model correctly segmented 216 entries, giving it 30.21% accuracy. This drop in performance was a result of noise in the data. To investigate this, the roots obtained by the web scraper were closely examined. The examination revealed that some root entries in this dataset were erroneous. These errors occurred while the scrapper stripped the inflectional morphemes from the entries in the online dictionary.

Comparison with other models

SwaRegex achieves better results than those proposed by (Rahi et al., 2020). Their model averages an accuracy of 95.2% on 2,500 Maithili verbs in XFST. Their model involves manually sorting words into inflection types and classes.

SwaRegex also outperforms (Ammari & Zenkouar, 2021b), whose model presented an accuracy of 87% when performing morphological analysis on 28,000 Amazigh words in XFST.

A similar experiment by (Chahuneau et al., 2013) achieved an accuracy of 78.2% on predicting Swahili words, which is still outperformed by our model.

Runyagram (Katushemererwe & Issue, 2010) was a morphological segmentation experiment on Runyakitara, one of the Bantu languages in Uganda. Their experiments were conducted on fsm2, a similar finite-state transducer environment to xfst. Still, SwaRegex achieved better results than Runyagram, which scored an accuracy of 82%.

Discussion

Regular expressions remain a robust means of matching input text against a set of rules. They are a widely used technique in programming today. They are important when evaluating whether user input conforms to a variety of rules. One such application is user input validation in web forms, where regular expressions are used to ensure that email addresses and phone numbers are in the correct format (Davis et al., 2018).

When applying morphological analysis to languages with limited resources, regular expressions

are essential, as demonstrated by SwaRegex, a lexical transducer for Swahili verbs. SwaRegex has demonstrated remarkable results when determining whether verbs are formatted correctly. This function is helpful in many contexts where Swahili is spoken.

First, Swahili-speaking computer users will benefit from SwaRegex, where search keywords can be analyzed morphologically to improve their search experience. One such area where SwaRegex is applicable is in search engines, where the search keywords need to be broken down into roots in order for the engine to present the most relevant results to the user. Used this way, SwaRegex plays the role of a lemmatizer, where Swahili verbs are broken down into their lemma or roots.

SwaRegex can also be useful to new Swahili learners. SwaRegex provides learners with a platform that helps them learn the morphological syntax of the Swahili verb. Learners will learn how to position morphemes to occupy the morphological slots of the verb in order to create morphologically correct verbs.

Conclusion and Future work

SwaRegex, a lexical transducer for the morphological segmentation of Swahili verbs, is presented in this study. The model is rule-based, and the morphological syntax of the verb is represented by regular expressions. A web scraper was used to populate the dataset with verbs from an online dictionary dynamically. The same regular expressions were combined into a lexical transducer in the XFST terminal to evaluate SwaRegex's performance. SwaRegex outperformed its XFST cousin in terms of performance. We intend to develop the morphological rule set in our model in the future to accommodate different inflections.

References

Abdulla, P. A., Faouzi Atig, M., Chen, Y. F., Diep, B. P., Holik, L., Rezine, A., & Rummer, P. (2019). Trau: SMT solver for string constraints. *Proceedings of the 18th Conference on Formal Methods in Computer -Aided Design, FMCAD 2018*. https://doi.org/10.23919/FMCAD.2018.8602997

Ammari, R., & Zenkouar, L. (2021a). Amazigh-sys: Intelligent system for recognition of amazigh words. *IAES International Journal of Artificial Intelligence*. https://doi.org/10.11591/IJAI.V10.I2.PP482-489

Ammari, R., & Zenkouar, L. (2021b). APMorph: Finite-state transducer for Amazigh pronominal morphology. *International Journal of Electrical and Computer Engineering*. https://doi.org/54.559[3] 5/ijece.v11i1.pp699-706

Chahuneau, V., Schlinger, E., Smith, N. A., & Dyer, C. (2013). Translating into morphologically rich languages with synthetic phrases. *EMNLP 8679 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*.

Davis, J. C., Coghlan, C. A., Servant, F., & Lee, D. (2018). The impact of Regular Expression Denial of Service (ReDoS) in practice: An empirical study at the ecosystem scale. *ESEC/FSE 8674 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. https://doi.org/10.1145/3236024.3236027

Deo, S. N. (2016). Pairwise Combinations of Swahil Applicative with Verb Extensions. *Nordic Journal of African Studies*, *25*(1), 52–71.

Gerdjikov, S. (2018). Note on the Lower Bounds of Bimachines. In *arXiv*.

Karttunen, L. (2001). Applications of finite-state transducers in natural language processing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/54.544[1] /7-540-44674-5_2

Katushemererwe, F., & Issue, S. (2010). *Fsm8 and the Morphological Analysis of Bantu Nouns – First Experiences from Runyakitara*. *4*(1), 58–69.

Kituku, B. (2019). *Grammar engineering for Swahili*. *08*(06). http://repository.dkut.ac.ke:8080/xmlui/handle/123456789/1280

Li, B., Cheng, F., Zhang, X., Cui, C., & Cai, W. (2021). A novel semi-supervised data-driven method for chiller fault diagnosis with unlabeled data. *Applied Energy*. https://doi.org/54.545[0] /j.apenergy.2021.116459

Liu, Z., Jimerson, R., & Prud'hommeaux, E. (2021). *Morphological Segmentation for Seneca*. https://doi.org/10.18653/v1/2021.americasnlp-1.10

Mahesh, B. (2018). Machine Learning Algorithms-A Review. *International Journal of Science and Research*.

Mdee, J. (2016). Patterns of Swahili Verbal Derivatives: An Analysis of their Formation. *Huria: Journal of the Open University of Tanzania*, *21*(1), 43–51.

Möhrs, C., & Cajo, S. T. (2020). The microstructure of a lexicographical resource of spoken German: Meanings and functions of the Lemma Eben. *Rasprave Instituta Za Hrvatski Jezik i Jezikoslovlje*. https://doi.org/10.31724/RIHJJ.46.2.25

Mott, J., Bies, A., Strassel, S., Kodner, J., Richter, C., Xu, H., & Marcus, M. (2020). *Morphological Segmentation for Low Resource Languages*. *May*, 3996–4002.

Muhirwe, J. (1983). *Computational Analysis of Kinyarwanda Morphology : The Morphological*. *1*(1), 78–87.

Ngugi, K., Okelo-Odongo, W., & Wagacha, P. . (2010). Swahili text-to-speech system. *African Journal of*

Science and Technology. https://doi.org/54.8758/ajst.v6i1.55170

Rahi, R., Pushp, S., Khan, A., & Sinha, S. K. (2020). A Finite State Transducer Based Morphological Analyzer of Maithili Language. *ArXiv Preprint ArXiv:2003.00234*.

Sahala, A., Silfverberg, M., Arppe, A., & Lindén, K. (2020). BabyFST - Towards a finite-state based computational model of ancient babylonian. *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings*.

Schönhof, & Wilkans, A. (2012). On the question of transitive and intransitive verbs in Swahili. *Lingua Posnaniensis*. https://doi.org/54.68$^{1\,2}$/v54566-012-0008-y

Shikali, C. S., Sijie, Z., Qihe, L., & Mokhosi, R. (2019). Better word representation vectors using syllabic alphabet: A case study of Swahili. *Applied Sciences (Switzerland)*. https://doi.org/54.77^{3}4/app^{3}5^{2}7^{0}8^{2}

Steinberger, R., Ombuya, S., Kabadjov, M., Pouliquen, B., Della Rocca, L., Belyaeva, J., de Paola, M., Ignat, C., & van der Goot, E. (2011). Expanding a multilingual media monitoring and information extraction tool to a new language: Swahili. *Language Resources and Evaluation*. https://doi.org/54.544^{1}/s10579-011-9155-y