Morphological disambiguation from stemming data

Antoine Nzeyimana*
University of Massachusetts, Amherst, MA 01003, USA
anthonzeyi@gmail.com

Abstract

Morphological analysis and disambiguation is an important task and a crucial preprocessing step in natural language processing of morphologically rich languages. Kinyarwanda, a morphologically rich language, currently lacks tools for automated morphological analysis. While linguistically curated finite state tools can be easily developed for morphological analysis, the morphological richness of the language allows many ambiguous analyses to be produced, requiring effective disambiguation. In this paper, we propose learning to morphologically disambiguate Kinyarwanda verbal forms from a new stemming dataset collected through crowd-sourcing. Using feature engineering and a feed-forward neural network based classifier, we achieve about 89% non-contextualized disambiguation accuracy. Our experiments reveal that inflectional properties of stems and morpheme association rules are the most discriminative features for disambiguation.

1 Introduction

For morphologically rich languages, morphological analysis and disambiguation plays a critical role in most natural language processing (NLP) tasks. When inflections are generated by piecing together multiple morphemes, a large and sparse vocabulary is produced, requiring tools to unpack the individual morphemes for downstream NLP tasks such information extraction and machine translation. A key characteristic of these languages is that morphemes often have specific meanings (often relating to properties of the words they form or referring to contextual entities) and their combination into words is mostly regular. Figure 1 shows typical morphological units contained in the word 'ntuzamwibeshyeho' (*Never underestimate him/her*).

While several morphologically rich languages such as Turkish, Arabic and Modern Hebrew already have mature tools for morphological segmentation (Cöltekin, 2010) (Çöltekin, 2014) (Itai and Segal, 2003) (Habash and Rambow, 2006), Kinyarwanda still lacks appropriate tools for the task. A key limitation in the effort is the need to have high quality datasets manually annotated by language experts. With limited funding opportunities, research on NLP for low resource languages lags behind recent advancements made for NLP on high resource languages. In this work, we leverage an easy to collect stemming dataset and transform it into a resource for morphological disambiguation. While the focus here is on Kinyarwanda verbal forms, the method can be applied to other morphologically rich languages. Collecting stemming data is much faster and less prone to errors than full morphological segmentations which require subtle linguistic knowledge.

Through a maximum entropy model, we are able to combine morphological properties of stems with inflectional similarity information from word embeddings to accurately disambiguate candidate segmentations from a rule-based morphological analyzer. Our work here pertains to non-contextual verb-phrase disambiguation but is a key step towards contextual disambiguation. We believe that this work will allow rich morphology to be incorporated in new models for Kinyarwanda and improve downstream NLP tasks on the language.

* Part of this work was done while the author was a graduate student at the University of Oregon, Eugene, OR 97403, USA

Proceedings of the 28th International Conference on Computational Linguistics, pages 4649–4660
Barcelona, Spain (Online), December 8-13, 2020

ntuzamwibeshyeho							
nti-	**-u-**	**-za-**	**-mu-**	**-ii-**	**-beshy-**	**-e-**	**-ho**
Negation	**Subject** (2nd pers/sing.)	**Tense** (future)	**Direct Object** (1st class/human/sing.)	**Reflexive** (wrt. subj)	**Stem**	**Aspect** (imperative)	**Prep.**
not	**you**	**will**	**him/her**	**self**	**lie**	**(imperative)**	**about**

Figure 1: Morphological segmentation of the word 'ntuzamwibeshyeho'. Each morpheme unit has a specific meaning or function in order to get the meaning of the whole word. The word is an inflection of the lemma 'kwibeshya' (*to err*) which a derivation from 'kubeshya' (*to lie*) by adding a reflexive *-ii-* . Therefore, a literal morpheme-by-morpheme translation of the word would be '*Never lie to yourself about him/her*' while the real meaning is '*Never underestimate him/her*'.

Morphological segmentation	**Probability**
- nti u/1 - za - - - mu - **ibeshy** - - - - - - - e ho	0.184
- nti u/7 - za - - - mu - **ibeshy** - - - - - - - e ho	0.152
- nti u/5 - za - - - mu - **ibeshy** - - - - - - - e ho	0.137
- nti u/7 - za - - - mu ii **beshy** - - - - - y - e ho	0.074
- nti u/1 - za - - - mu ii **beshy** - - - - - y - e ho	0.074
- nti u/5 - za - - - mu ii **beshy** - - - - - y - e ho	0.074
- nti u/5 - za - - - mu ii **beshy** - - - - - - - e ho	0.062
- nti u/1 - za - - - mu ii **beshy** - - - - - - - e ho	0.061
- nti u/7 - za - - - mu ii **beshy** - - - - - - - e ho	0.058
- nti u/1 - za - - - mu ii **besh** - - - - - y - e ho	0.039
- nti u/5 - za - - - mu ii **besh** - - - - - y - e ho	0.037
- nti u/7 - za - - - mu ii **besh** - - - - - y - e ho	0.034
- nti u/1 - za - - - mu - **ibeshy** - - - - - y - e ho	0.009
- nti u/5 - za - - - mu - **ibeshy** - - - - - y - e ho	0.004
- nti u/7 - za - - - mu - **ibeshy** - - - - - y - e ho	0.001

Table 1: Potential segmentations for the inflected verb 'ntuzamwibeshyeho' as produced by a finite state based morphological analyzer and ranked by our disambiguation method. Hyphens(-) represent potential morpheme slots that are not filled for this instance. The stem or root is shown in bold. The top three options only differ in the entity class for the subject – **u/5** is for humans, 3rd person singular – **u/1** also for humans but 2nd person singular – while **u/7** is for inanimate object. Notice that the disambiguation tool predicted that the subject mostly likely is a 2nd person. The tool has also chosen to base on the derived stem '*-ibeshy-*' (*to err*) rather than the original root '*-beshy-*' (*to lie*), thus reflecting more focus on semantics.

2 Related work

Computational morphology has been studied for decades, but most implementations and evaluations have been conducted on languages that are not related to Kinyarwanda. Finite state methods for morphological analysis have been proposed by Beesley and Karttunen (Karttunen, 2000) and have been popular for morphological analysis. Our morphological analyzer is based on the underlying principle of two level morphology (Koskenniemi, 1983), but our custom implementation does not follow the exact formalism of finite state transducers. We rather focus on refining rules that are specific to Kinyarwanda through extensive empirical examination. In (Muhirwe, 2007), a morphological alternation model for Kinyarwanda was presented using Xerox tools, but no empirical evaluation was conducted. In (Garrette et al., 2013), an experiment was conducted on learning POS taggers for Kinyarwanda and Malgasay using a small dataset of frequent words annotated by linguists. While POS tagging is an important task, morphological

analysis is even more important for morphologically rich languages because it reveals more information than what can be covered by a finite set of tags. Other work on Kinyarwanda has been more linguistic in nature (Kimenyi, 1980); (Jerro, 2016), especially due to that Kinyarwanda is considered as a more generic prototype of the larger group of Bantu languages, owing to its rich morphology and tonal system.

The problem of morphological disambiguation has been researched on for other morphologically rich languages such as Turkish, Arabic and Modern Hebrew. Proposed methods range from statistical ones (Hakkani-Tür et al., 2002) (Cohen and Smith, 2007), to rule based approaches (Yuret and Türe, 2006), and more recently using recurrent neural networks (Zalmout and Habash, 2017). Most of these methods are usually trained and evaluated on richly annotated tree-banks which are not available for low resource languages like Kinyarwanda. The only labeled data required by our method is a list of inflection/stem pairs which can be conveniently collected by untrained native speakers. Another major difference is also that our disambiguation focuses on uncontextualized morphology of Kinyarwanda verbal forms. We reserve contextual disambiguation and part of speech tagging for a future study.

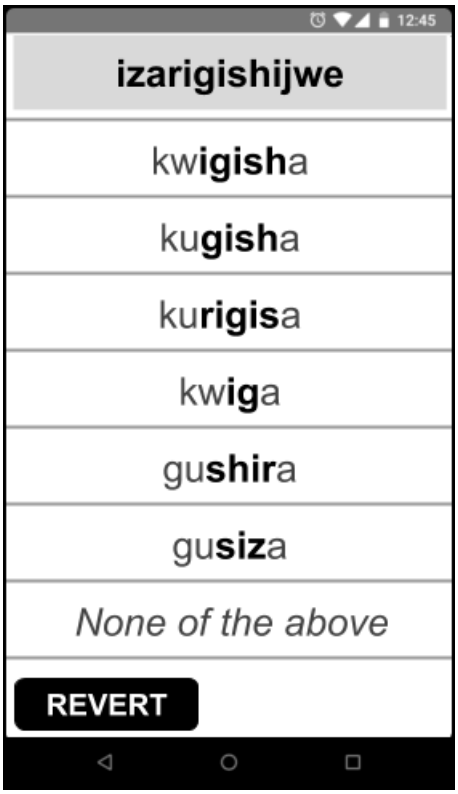

Figure 2: A smart-phone interface is used by volunteers to stem Kinyarwanda verb forms. For instance, the analysis of the inflected form 'izarigishijwe'(...*which have been made disappear*) can result in ambiguous stems – 'igish'(kwigisha: *to teach*), 'gish'(kugisha: *to ask [for advice]*), 'rigis'(kurigisa: *to make disappear*), 'ig'(kwiga: *to study*), 'shir'(gushira: *to end*) and 'siz'(gusiza: *to prepare a place for construction*)

3 Methods

3.1 Dataset development

The dataset for this project comes from a crowd-sourcing effort where users labeled inflected forms of Kinyarwanda verbs with corresponding lemma. From a web-crawled corpus, our toolkit detects potential verb inflections, auto-segments them and asks volunteers to choose the right lemma from a proposed list of candidates. For convenience of use, volunteers are asked to lemmatize the inflected verbs using a simple mobile application (see Figure 2) and the labeled data are sent to a back-end server. The raw labeled dataset was filtered for potential random user inputs by using a baseline classifier and removing data for users who performed poorly on the otherwise least ambiguous instances. While more than 200

volunteers used the stemming application, only data from 37 annotators was found to be consistent and then used in this study.

3.2 Morphological analysis

Our morphological analysis is based on finite state methods (Karttunen, 2000). Table 2 shows a repertoire of Kinyarwanda verb morphemes and examples of when they are used. The morphotactics, which dictate the ordering of morphemes, are modeled with a hierarchical graph shown in Figure 3. It is this graph of morpheme slots that make Kinyarwanda verbal system very productive. In theory, thousands of different morpheme slot sequences can be produced by this graph, but in reality, there are more semantic and syntactic restrictions.

In addition to the basic morphotactics graph model, morpho-graphemic rules and other morpheme association rules are added to the analyzer using small constraint-enforcement language. The language is expressive enough to allow a researcher to incorporate complex grammatical regularities. For example, a rule such as '{V;NEG;ta} ⇒ {!V;PRE_IN;nti}' prevents having two negation markers in the same verb inflection. Also, '{V;PRE_IN;si} ⇒ {V;SUBJ;n}' enforces the negative pre-prefix 'si' to be used only with first person singular. The rule '{V;STEM;#1} ⇒ {V;STEM;/^[hzcvrz]$/}' limits the number of single character stems, while '{V;STEM;/^gamij$/} ⇒ {V;ASP;e}' allows the irregular verb '-gamij' (*to aim*) to only take aspect marker suffix '-e'.

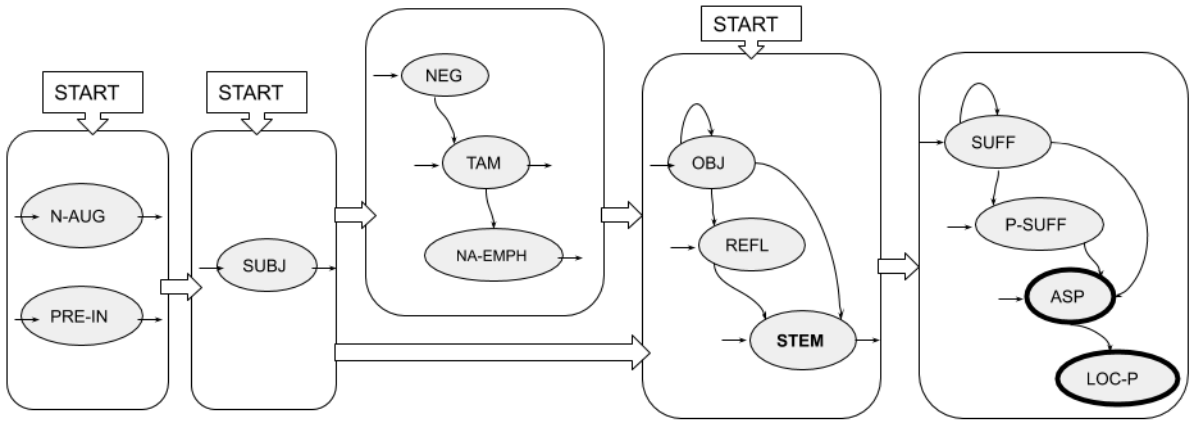

Figure 3: A hierarchical graph model of Kinyarwanda verbal morphotactics.

3.3 Classification

We handle morphological disambiguation as a classification problem with a variable number of classes (candidate stems) for each instance. We compute two types of features from each morphological segmentation and feed those features to a feed-forward neural network and finally produce probabilities with a softmax function. We train the network to minimize a cross-entropy loss function.

Note that, since we have a variable number of instance-specific class labels (candidate stems), the classifier is not really discriminating between a fixed set of classes but rather ranking segmentations based on the features they present. Having our labels being only the stem part of segmentation, we also need to account for the fact that there are multiple possible segmentations with the same valid stem. We account for this in our cross-entropy loss function given in Equation 1.

$$\mathcal{L}_{CE} = -\sum_{j}^{M} p_j log(\hat{p}_j) \tag{1}$$

where:

* M is the number of candidate segmentations produced by the rule-based analyzer

* $\hat{p}_j$ is the hypothesis probability assigned to candidate segmentation j from the soft-max layer

Morpheme slot	**Morphemes**	**Example**
1. Nominal augment (N-AUG)	**u**, a, i	**u**-a-som-ye: uwasomye *'the one who read'*
2. Pre-prefix (PRE-IN)	si, **nti** (*negative*), **ni** (*imperative*), **ni** (*conditional*)	**nti**-mu-a-som-ye: ntimwasomye '*you didn't read*' **ni**-mu-som-e: nimusome '*read*' **ni**-mu-som-a: nimusoma '*if you read*'
3. Subject class marker (SUBJ)	**n**, u, tu, mu, a, u, **ba**, u, i, ri, a, **ki**, bi, i, zi, ru, ka, tu, bu, ku, ha, **ku**, **mi**	**n**-a-som-ye: nasomye '*I read (past)*' **ba**-za-som-a: bazasoma '*they will read*' **ki**-a-som-w-ye: cyasomwe '*which was read*' **ku**-som-a: gusoma '*to read*' i-**mi**-som-ir-e: imisomere '*way of reading*'
4. Negation (NEG)	**ta**	tu-**ta**-som-a: tudasoma *'we don't read'*
5. Tense–Apect–Modality (TAM)	i, a, ra, **ara**, za, aza, ka, raka, **kaza**, ki, **racya**, oka, aka	ba-**ara**-som-ye: barasomye *'they read (past)*' ba-**kaza**-som-a: bakazasoma *'and they will read'* ba-**racya**-som-a: baracyasoma *'they still read*'
6. Emphasizing prefix (NA-EMPH)	**na**	u-a-**na**-som-a: wanasoma *'you can **even** read*'
7. Third object class (OBJ 3) marker (tri-transitivity)	mu, ba, wu, yi, ri, ya, **ki**, bi, yi, zi, ru, ka, tu, bu, ku, ha	n-ara-**ki**-ha-mu-som-ir-ye: narakihamusomeye *'I read **it** for her while there'*
8. Second object class (OBJ 2) marker (di-transitivity)	ku, tu, ba, mu, ba, wu, yi, ri, ya, **ki**, bi, yi, zi, ru, ka, tu, bu, ku, ha	n-ara-**ki**-mu-som-ir-ye: narakimusomeye *'I read **it** for her'*
9. First object class (OBJ 1) marker (transitive verbs)	**n**, ku, tu, ba, mu, ba, wu, yi, ri, ya, ki, bi, yi, zi, ru, ka, tu, bu, ku, ha	a-ara-ki-**n**-som-ir-ye: yarakinsomeye *'he read it **for me**'*
10. Reflexive (REFL)	ii	a-**ii**-som-ir-aga: yisomeraga *'he was reading for himself'*
11. STEM	-	ku-**som**-a: gusoma *'to read'*
12-13. Suffix (SUFF) (repeatable)	ir, ish, **ik**, iz, y ur, uk, an	ki-ra-som-**ik**-a: kirasomeka *'it is readable*'
14. Passive suffix (P-SUFF)	w	ki-a-som-**w**-ye: cyasomwe *'it was read by'*
15. Aspect marker (ASP)	a, e, **aga**, ye, i, aho	a-a-som-**aga**: yasomaga *'he was reading*'
16. Locational post-suffix (LOC-P)	yo, ho, **mo**	a-a-som-ye-**mo**: yasomyemo *'he read inside*'

Table 2: A repertoire of Kinyarwanda verbal morphemes.

* p_j is the reference probability which we set to:

$$p_j = \begin{cases} \frac{1}{n}, & \text{if segmentation } j \text{ has the right stem,} \\ & (n \textit{ being the number of such segmentations}). \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

3.4 Feature extraction

Similarity features

The first type of features estimates how similar a given segmentation is to other inflections of the same stem, effectively handling the stem disambiguation part of the problem. Formally, given a candidate segmentation x with a stem s, first we produce a set of N common inflections of the same stem $\{y_i\}_i^N \in Infl(s) \cap \mathcal{V}$ by associating the stem with common standard affixes, generating the surface forms and making sure that these surface forms are part of the word embedding vocabulary $\mathcal{V}$. We choose K nearest inflections to x among $\{y_i\}_i^N$ (by cosine similarity) and estimate the final similarity scores as:

$$f_m(\{d_e(x, y_i)\}_i^K), \quad (3)$$

where:

* $\{\cdot\}_i^N$ notation means a set of N elements indexed by i
* $f_m(\cdot)$ is a mean function; we use both arithmetic, geometric and harmonic means as features.
* $d_e(x, y_i)$ is the normalized angular similarity between the word embedding vectors for x and y_i, i.e.:

$$d_e(x, y_i) = \sigma(1 - \frac{1}{\pi} \arccos\left(\frac{\boldsymbol{e}(x)^T \boldsymbol{e}(y_i)}{\left\|\boldsymbol{e}(x)\right\| \left\|\boldsymbol{e}(y_i)\right\|}\right)), \quad (4)$$

  with $\boldsymbol{e}(\cdot)$ being the word embedding lookup function.

* $\sigma(\cdot)$ is a normalizing sigmoid function of the form:

$$\sigma(z) = [1 + exp(-8\frac{z - min_f}{Max_f - min_f})]^{-8}, \quad (5)$$

  with min_f and Max_f being tunable hyper-parameters for each type of feature f demarcating the active range of the feature.

Additionally, we use 2-dimensional euclidean distances of the token- and document- corpus frequencies between x and y_i to estimate how popular a given segmentation is in the corpus in relation to the popularity of the inflection set $\{y_i\}_i^K$:

$$d_t(x, \{y_i\}_i^K) = \sigma(\frac{1}{K} \sum_{i=1}^K \sqrt{(t_c(x) - t_c(y_i))^2 + (t_d(x) - t_d(y_i))^2}), \quad (6)$$

where:

* $t_c(z) = \sigma(token\_count(z))$, i.e. the sigmoid-normalized number of times z appear in the corpus
* $t_d(z) = \sigma(document\_count(z))$, i.e. the sigmoid-normalized number of corpus documents containing z

Finally,

$$f_m(\{\frac{t_c(y_i) + t_d(y_i)}{2}\}_i^K) \quad (7)$$

and

$$\frac{t_c(x) + t_d(x)}{2} \quad (8)$$

are included as separate features.

Morphological indicator features

The second type of features evaluates the appropriateness of "morphological features" present in a given segmentation versus typical features associated with its stem in the training dataset. These indicator features include the use of special morphemes, morpheme associations and special morpho-graphemic rules inherent in the segmentation. For example, passivization (transformation from active to passive form) is expressed by a special suffix but not all verbs can be used in passive form. The same applies to transitivity (the number and type of object pronouns a verb can take), the use of special suffixes, personal pronouns, locatives, and so on. Essentially, the M linguistically-motivated indicator features $f_i(x)$ are compared to their selection ratio scores in the training dataset. By selection ratio scores, we mean:

$$f_m(\{\sigma(\frac{chosen[f_i,s]}{proposed[f_i,s]})\}_i^M) \tag{9}$$

and separately

$$\sigma(\frac{chosen[f_i,s]}{proposed[f_i,s]}), \tag{10}$$

where:

* $chosen[f_i, s]$ is the number of times stem s has been chosen as the valid stem for any morphological segmentation having morphological feature f_i.

* $proposed[f_i, s]$ is the number of times stem s has been proposed (either chosen or rejected) among candidate lemmas for any segmentation having morphological feature f_i.

The list of indicator features f_i used in our experiments are given in Appendix A.

Feature extraction example

Here we present a working example of how features are extracted. Given the input the input word '**gatwikirwa**' to be lemmatized by the annotator, we follow the following steps to extract the training data.

Step 1. Morphological analysis: The morphological analyzer first produces candidate segmentations as provided in Table 2. The morphological indicator features (from Appendix A) of each candidate analysis are shown in the same table.

No	Morphological analysis (morpheme sequence)	Indicator features from Appendix A
1	- - ka - - - - - tu - ik - - ir - - - w a -	1, 5, 17, 24, 26
2	- - ka - - - - - tu ii kir - - - - - - w a -	1, 5, 8, 24
3	- - ka - - - - - - - twik - - ir - - - w a -	1, 17, 24, 26
4	- - ka - - - - - - - twikirw - - - - - y - a -	1, 23
5	- - ka - - - - - - - twikirw - - - - - - - a -	1
6	- - ka - - - - - - - twikir - - - - - - w a -	1, 24

Table 3: Example of morphological analysis for input word '**gatwikirwa**' (*1. covered or 2. burned*)

Step 2. Annotator input: The annotator is presented with a list of candidate lemma to choose from:

gatwikirwa
1. kw**ik**a (*to sink, fall, drop)*
2. gu**kir**a (*1. to be healed, releaved or 2. to be rich)*
3. gu**twik**a (*to burn)*
4. gu**twikirw**a (*to be covered)*
5. gu**twikir**a (*to cover)*

Assuming the annotator chooses 'gu**twikir**a' as the correct lemma, then the annotated raw data consists of the 5 labeled pairs: gatwikirwa/**ik**:0/1, gatwikirwa/**kir**:0/1, gatwikirwa/**twik**:0/1, gatwikirwa/**twikirw**:0/2 and gatwikirwa/**twikir**:1/1.

Step 3. Generate inflection sets $S_j = \{y_i\}_i^N$ for each candidate stem j: The morphological analyser formulates a set of common inflections for each candidate stem:

1. **ik**: $S_1 = \{$kwika, turitse, turika, twitse, twikaga, yaritse, arika, ...$\}$
2. **kir**: $S_2 = \{$gukira, arakize, turakira, dukize, bakize, tuzakire, ...$\}$
3. **twik**: $S_3 = \{$gutwika, uzatwika, uzatwike, hatwitsemo, atwikamo, ...$\}$
4. **twikirw**: $S_4 = \{$gutwikirwa, tugatwikirwa, yaratwikiwe, agatwikirwa, ...$\}$
5. **twikir**: $S_5 = \{$gutwikira, yatwikiriza, yatwikirije, akitwikira, ...$\}$

Step 4. Compute similarity features: The input word $x =$'gatwikirwa' is then paired with each of the nearest K entries y_i in the inflection sets S_j to compute similarity scores $d_e(x, y_i)$, $t_c(.)$, $t_d(.)$ and $d_t(x, S_j)$ according to equations 4-8. Example values are provided below.

d_e('gatwikirwa', 'kwika') $= 0.075$	d_t('gatwikirwa', 'kwika') $= 0.05$
d_e('gatwikirwa', 'gukira') $= 0.003$	d_t('gatwikirwa', 'gukira') $= 0.01$
d_e('gatwikirwa', 'gutwika') $= 0.822$	d_t('gatwikirwa', 'gutwika') $= 0.61$
d_e('gatwikirwa', 'gutwikirwa') $= 0.992$	d_t('gatwikirwa', 'gutwikirwa') $= 0.81$
d_e('gatwikirwa', 'gutwikira') $= 0.986$	d_t('gatwikirwa', 'gutwikira') $= 0.72$

Step 5. Compute morphological indicator features: The morphological indicator features shown in Table 3 will be used to populate the 'proposed' and 'chosen' tables in equations 9-10. For instance, given the morphological analyzer output in Table 3 and annotator choice as **gatwikirwa/twikir**, the tables are going to be populated as in the following example:

proposed[24, 'twikir']	incremented by 1	
chosen[24, 'twikir']	incremented by 1	i.e. '*cover*' is more likely to be passivized
proposed[8, 'kir']	incremented by 1	
chosen[8, 'kir']	decremented by 1	i.e. '*be rich*' is less likely to be subject reflexive
proposed[1, 'twikirw']	incremented by 2	
chosen[1, 'twikirw']	decremented by 2	i.e. the passivized form is just demoted

Step 6. Neural network input/output for training: For every candidate segmentation, the neural network receives an input of 64 real values composed of 3 from equation 3, 1 from equation 6, 3 from equation 7, 1 from equation 8, 3 means (geometric, arithmetic and harmonic) of all similarity features, 3 values from equation 9, 47 values from equation 10 (1 for each indicator feature in Appendix A) and 2 binary (0 or 1) features indicating the popularity of the lemma in two lexical resources (small/restricted and large). An extra sigmoid-normalized weighted average similar to equation 6 was added to account for the popularity of the type of inflection used to generate each element in the inflection set. In this example, the target probability $p_j = 1$ for the pair gatwikirwa/**twikir** and $p_j = 0$ for the other 4 pairs.

4 Experimental setup

The first step in our experiments was to generate stem morphological indicator features from user annotations as explained in section 3.4. The second step involves preparing the dataset for training and evaluation. After features are extracted, we split the data between in training and validation set and then train a baseline classifier using only the data from user annotations. We up-sample by 4 factors the annotations from the best trained annotator who is equipped with subtle linguistic understanding of

Kinyarwanda verb morphology. We use the baseline classifier to then predict the stem for the entire unlabeled vocabulary of Kinyarwanda verbal forms. We rank the predicted stems by prediction uncertainty (entropy). The most uncertain instances are sent back to annotators for labeling in a batched active learning fashion (Settles, 2009). For active learning, we send batches of the top 10000 uncertain samples for which the entropy $H > 1$. We also enrich our labeled set with the most confident predictions in a semi-supervised manner. For semi-supervised learning, we only take examples for which the baseline model has labeled with at least 0.95 top probability (P_1), having at least 3 competing stems, $(P_1 - P_2) > 0.95$ and entropy $H < 0.1$.

After expanding our labeled data through active learning and semi-supervised learning, we repeat the first step of feature extraction to form our final training and evaluation dataset, which contains about 170,000 examples. We split our dataset into training, development and test set in the ratio of 70%+15%+15% respectively. All our models are trained with gradient descent using ADAM update rule (Kingma and Ba, 2014) using 0.01 learning rate in 256-sized mini-batches and for 50 epochs. For our best fine-tuned model, we re-train the model with large batches of 4000 examples each using LAMB method (You et al., 2019) for 100 more epochs. Since all features are precomputed, training each feed-forward neural network takes less than 5 minutes on an quad-core machine. We use POSIX C and C++11 for this project with the only external dependency being Eigen matrix algebra library [1].

5 Experimental results and discussion

Model size – We evaluated the model robustness by varying the number of hidden units in the feed-forward neural nets (Table 4). Surprisingly, the size of the of the model doesn't affect the performance. Even a small network of two hidden layers of 6 and 3 units respectively achieves almost the same accuracy as the network of 3 layers of 32,16,8 hidden units. We also observed very little over-fitting, having the same level of accuracy on both training, development and test set. We believe that this persistent performance is probably due to the semi-supervised method we used and possibly that the summarizing features (i.e. $f_m(\cdot)$) precomputed explain most of the label variations.

Size of feedforward layers	Morphological Embedding	FastText Embedding
64-32-16-8	89.32	89.30
64-32-8	89.39	89.30
64-6-3	89.28	89.19

Table 4: Test set accuracy for various model sizes using all 64 input features

Feature subsets – We then evaluated different feature subsets to assess which ones had greater impact on the final performance. All the results presented in Table 5 used a the small model of two layer, 64-6-3 feed-forward units. The three statistics (arithmetic, geometric and harmonic means) of morphological features account for most of the accuracy while using the individual morphological features under-performs. This is probably due to that the small nature of neural network used doesn't allow it to effectively learn these statistics. The difference in the performance of inflectional similarity features may be attributed to the differences in the two pre-trained word embeddings used. The vocabulary of our "Morpho" embeddings is almost as twice as big than the fastText (Bojanowski et al., 2017) one, even though they are trained on the same corpus and both are based on the Skip-Gram model (Mikolov et al., 2013). So, comparing them requires carefully setting proper hyper-parameters min_f and Max_f for the normalizing function $\sigma(\cdot)$ in equation 4.

Annotator performance – Our final evaluation looked at how our fine-tune model rated different individual annotator labels depending on their linguistic training level and the mode of active learning used (Table 6). The 3 reported annotators in the table were identified contacts of the author and together contributed more than 30% of the labeled data. Our interpretation of the results is that the model might

[1] http://eigen.tuxfamily.org/index.php?title=Main_Page

Feature subset	Morpho. Embedding	FastText Embedding
Summary statistics	87.92	88.29
Inflectional similarity with word embeddings	66.47	68.74
Inflectional similarity with token frequency	51.94	44.24
Morphological features summary	88.16	88.36
Morphological feature details	85.52	85.30
All features combined and training longer	**89.51**	**89.55**

Table 5: Test set accuracy using various feature subsets using 6-3 hidden units. The fine-tuned model was trained longer using 32-8 MLP setup

Annotator training level	Uniformly random samples	Most uncertainty samples
Graduate student	84.9	61.9
College graduate	80.7	-
High school graduate	77.9	55.2

Table 6: Relative performance (training set accuracy) of different annotators with varying levels of linguistic training.

be relying too much on easy examples pulled in through semi-supervised learning and noise introduced by individual annotators. The level of annotator training also has a clear impact on the performance.

Sources of ambiguity – There are inherently multiple sources of ambiguity when one encounters a Kinyarwanda verbal expression. Achieving full disambiguation requires having access to complete contextual information. This information may even be encoded only in the tonal system (Kimenyi, 2002) and thus unavailable in written form. In fact, reading written Kinyarwanda requires careful real-time disambiguation by the reader because tones are not marked in text. Contextual information is also needed for semantic disambiguation. For example, the verb **'yarigishije'** can mean both 'a-ara-igish-iz-ye' (*he taught*) or 'a-a-rigis-iz-ye' (*he made disappear*). Without the semantic context, both segmentations are possible. Sentence level disambiguation may also benefit from contextual agreements through the Bantu noun class system. Our annotation process is also affected by lemmatization ambiguity and the blurred boundary between inflection and derivation. For example it is subjective whether the verbs **kwivuga** 'ku-ii-vug-a' (*to talk about self*), **kuvuza** 'ku-vug-y-a' (*'to make sound with (some object)'*) and **kuvugisha** 'ku-vug-ish-a' (*to talk to (someone)*) are themselves lemma forms or just inflections of **kuvuga** 'ku-vug-a' (*to talk*).

6 Conclusion

This work focused on morphological disambiguation of Kinyarwanda verb forms using maximum entropy model on new crowd-sourced stemming dataset. High disambiguation accuracy was achieved through careful feature engineering. Intuitively curated inflectional features emerged as important parsimonious predictors. Future work should look at how to directly use morpheme embedding methods as a way to more generically represent both semantics and morphology in a unified form. Achieving total disambiguation ultimately requires complete contextual information which may not be available in written form.

Acknowledgements

This research was partly made possible by access to the S3C Laboratory facility in the Department of Geography at the University of Oregon. We thank the volunteers who contributed to the stemming dataset through Kinyarwanda Stemmer mobile application. We also thank the anonymous reviewers for their insightful feedback.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Shay B Cohen and Noah A Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 208–217.

Cagri Cöltekin. 2010. A freely available morphological analyzer for turkish. In *LREC*, volume 2, pages 19–28.

Çagri Çöltekin. 2014. A set of open source tools for turkish natural language processing. In *LREC*, pages 1079–1086.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 583–592.

Nizar Habash and Owen Rambow. 2006. Magead: a morphological analyzer and generator for the arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688.

Dilek Z Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410.

Alon Itai and Erel Segal. 2003. A corpus based morphological analyzer for unvocalized modern hebrew. In *Proceedings of Machine Translation for Semitic Languages: Issues and Approaches, Workshop at MT Summit IX (MT-SUMMIT-IX)*.

Kyle Jerro. 2016. The locative applicative and the semantics of verb class in kinyarwanda. *Diversity in African languages*, page 289.

Lauri Karttunen. 2000. Applications of finite-state transducers in natural language processing. In *International Conference on Implementation and Application of Automata*, pages 34–46. Springer.

Alexandre Kimenyi. 1980. *A relational grammar of Kinyarwanda*, volume 91. Univ of California Press.

Alexandre Kimenyi. 2002. *A tonal grammar of Kinyarwanda: an autosegmental and metrical analysis*, volume 9. Edwin Mellen Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*, volume 11. University of Helsinki, Department of General Linguistics Helsinki, Finland.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jackson Muhirwe. 2007. Computational analysis of kinyarwanda morphology: The morphological alternations. *International Journal of computing and ICT Research*, 1(1):85–92.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 1(5).

Deniz Yuret and Ferhan Türe. 2006. Learning morphological disambiguation rules for turkish. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 328–334.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713.

Appendix A. Morphological indicator features used by the classifier

No	Feature key	Feature explanation
1	f.with_subjects	Has a subject marker
2	f.missing_subj	Doesn't have a subject marker
3	f.with_human_subjects	Has human subject marker
4	f.with_location_subjects	Has locative subject marker
5	f.with_objects	Has object marker
6	f.with_human_objects	Has human object marker
7	f.with_location_objects	Has locative object marker
8	f.trans_any_refl	Has reflexive marker
9	f.trans_any_obj3	Has any object marker (transitive)
10	f.trans_any_obj23	Has two object markers (ditransivive)
11	f.trans_any_obj123	Has three object markers (tritransitive)
12	f.obj2_obj3_suff	Ditransitive with any suffix
13	f.obj3_suff	Transitive with any suffix
14	f.obj2_obj3_refl_suff	Ditransitive with reflexive marker and any suffix
15	f.obj3_refl_suff	Transitive with reflexive marker and any suffix
16	f.suff_ish	Has suffix **-ish**
17	f.suff_ir	Has suffix **-ir**
18	f.suff_iz	Has suffix **-iz**
19	f.suff_an	Has suffix **-an**
20	f.suff_ik	Has suffix **-ik**
21	f.suff_uk	Has suffix **-uk**
22	f.suff_ur	Has suffix **-ur**
23	f.suff_y	Has suffix **-y**
24	f.suff_w	Has passive suffix **-w**
25	f.suff_ir_y	Has suffixes **-ir** followed by **-y**
26	f.suff_ir_w	Has suffixes **-ir** followed by **-w**
27	f.suff_an_y	Has suffixes **-an** followed by **-y**
28	f.suff_iz_y	Has suffixes **-iz** followed by **-y**
29	f.suff_y_w	Has suffixes **-y** followed by **-w**
30	f.post_suff	Has locative post-suffix
31	f.mg_rule_r_y_none	Uses suffix rule **r + y** $\rightarrow$ **y** e.g. biragoye: bi-ra-gor-ye (*it is difficult*)
32	f.mg_rule_r_y_z	Uses suffix rule **r + y** $\rightarrow$ **z** e.g. yarabuze: a-ara-bur-ye (*went missing*)
33	f.obj3_ka_ku	Has object markers **-ka** or **-ku**
34	f.suff_ish_ish	Has suffixes **-ish** followed by **-ish**
35	f.suff_ir_ir	Has suffixes **-ir** followed by **-ir**
36	f.comb_obj3_suff_w	Has an object marker, a suffix and a passive suffix **-w**
37	f.with_nloc_obj_no_suf	Has non-locative object marker but without a suffix
38	f.suff1_suff2	Has two suffixes
39	f.suff1_suff2_suff3	Has three suffixes
40	f.ni_imperative	Uses the imperative prefix **ni-**
41	f.ni_conditional	Uses the conditional prefix **ni-**
42	f.mg_rule_t_y_s	Uses suffixation rule **t + y** $\rightarrow$ **s** e.g. narose: n-a-rot-ye (*I dreamed*)
43	f.mg_rule_t_y_sh	Uses suffixation rule **t + y** $\rightarrow$ **sh** e.g. nafashe: n-a-fat-ye (*I took*)
44	f.suff_an_ir	Has suffixes **-an** followed by **-ir**
45	f.suff_ur_y	Has suffixes **-ur** followed by **-y**
46	f.suff_ur_w	Has suffixes **-ur** followed by **-w**
47	f.suff_uk_y	Has suffixes **-uk** followed by **-y**