Stemming and Segmentation for Classical Tibetan

Orna Almogi[1,2], Lena Dankin[3], Nachum Dershowitz[3,4(✉)], Yair Hoffman[3], Dimitri Pauls[1,2], Dorji Wangchuk[1,2], and Lior Wolf[3]

[1] Department of Indian and Tibetan Studies, Universität Hamburg, Hamburg, Germany
[2] Khyentse Center for Tibetan Buddhist Textual Scholarship, Universität Hamburg, Hamburg, Germany
[3] School of Computer Science, Tel Aviv University, Ramat Aviv, Tel Aviv, Israel
nachum@tau.ac.il
[4] Institut d'Études Avancées de Paris, Paris, France

Abstract. Tibetan is a monosyllabic language for which computerized language tools are largely lacking. We describe the development of a syllable stemmer for Tibetan. The stemmer is based on a set of rules that strive to identify the vowel, the core letter of the syllable, and then the other parts. We demonstrate the value of the stemmer with two applications: determining stem similarity of two syllables and word segmentation. Our stemmer is being made available as an open-source tool and word segmentation as a freely-available online tool.

It is worthy of remark that a tongue which in its nature was monosyllabic, when written in the characters of a polysyllabic language like the Sanskrit, had necessarily to undergo some modification.

Sarat Chandra Das, "Life of Sum-pa mkhan-po, also styled Ye-śes dpal-'byor, the author of Rehumig (Chronological Table)", *Journal of the Asiatic Society of Bengal (1889)*

1 Introduction

The Tibetan language belongs to the Tibeto-Burman branch of the Sino-Tibetan family. The language is ergative, with a plethora of (usually) monosyllabic grammatical particles, which are often omitted. Occasionally, the same syllable can be written using one of several orthographic variations, for example, *sogs* and *stsogs*. In the case of verbs, the syllable has various inflectional forms that are often homophones, a fact that can result in variants in the reading due to scribal errors. Examples of such inflectional forms are *sgrub*, *bsgrubs*, *bsgrub*, *sgrubs* (present, past, future and imperative, respectively), all of which are homophones with "stemmic identity". It should be noted that the notion of "stem" does not exist in traditional Tibetan grammar, but has been introduced here for the purpose of identifying virtually identical syllables despite varying orthographies or

A. Gelbukh (Ed.): CICLing 2016, LNCS 9623, pp. 294–306, 2018.
https://doi.org/10.1007/978-3-319-75477-2_20

inflectional forms. In some texts, orthographic abbreviations are very common: letters (consonants or vowels) are omitted within a syllable, while contracting two or more syllables into one; examples are: *bkra shis* → *bkris* and *ye shes* → *yees*. The language is also abundant in homophones without stemmic identity, which lead to scribal errors and to drift in textual content. The latter two phenomena will be disregarded here since they are irrelevant for stemming.

The Tibetan writing system is reported by tradition to have been developed circa the 7th century. It is based on the Indian Brāhmī/Gupta script with adaptations for Tibetan. Tibetologists commonly employ (as we do here) the Wylie [10] system for transliteration into Latin characters, in which no diacritics are used and thus various letters are represented by two or three consonants.

Many of the Mahāyāna Indic Buddhist texts are extant in Tibetan, and sometimes only in Tibetan. The Tibetan Buddhist canon consists of two parts: the Kangyur, which commonly comprises 108 volumes containing what is believed by tradition to be the Word of the Buddha, texts that were mostly translated directly from the Sanskrit original (with some from other languages and others indirectly via Chinese); and the Tengyur, commonly comprising about 210 volumes consisting of canonical commentaries, treatises, and various kinds of manuals that were likewise mostly translated from Sanskrit (with some from other languages and a few originally written in Tibetan).

After a brief introduction to the Tibetan syllable (Sect. 2), we describe the stemming algorithm (Sect. 3), followed by two applications: stem similarity (Sect. 4) and word segmentation (Sect. 5).

2 The Tibetan Syllable

For our purposes, we consider the Tibetan alphabet to consist of 29 consonants and 5 vowels. (Traditionally, *a* is including among a list of 30 consonants, rather than amongst the vowels.) Table 1 gives the distribution of consonants in the Tibetan Buddhist canon. (Digital data provided by Paul Hacket of Columbia University.)

The Tibetan language is monosyllabic (morphemes normally consist of one syllable): each syllable is written separately; but word boundaries – a word consists of one or more syllables – are not indicated in a text, similarly to Chinese. Table 2 presents the most frequent unigrams, bigrams, and trigrams in the corpus, along with their distributions. Notice that the Zipf's law [11], that word frequency is inversely proportional to rank in a frequency table, does not hold at the level of Tibetan syllables; see [9].

Figure 1 shows a syllable with its parts in Tibetan script (left) and a syllable appended with the grammatical particle following it (right). The script is transliterated in Wylie from left to right, with stacked letters transliterated from top to bottom.

Ligatures are standard. Three-letter ligatures are these: *rky, rgy, rmy sky, sgy, spy, sby, smy, skr, sgr, spr, sbr, smr*.

A unique feature of the language is the fact that particles can be added/omitted while the text still retains its meaning and is in fact considered

Fig. 1. left: Tibetan syllable (exemplified by *bsgrup*); right: disyllabic contraction (exemplified by *sgra'ang*)

to be the *same* text (the omission of particles, however, might result in ambiguity). Thus the presence or absence of particles can in most cases be considered inconsequential, and, therefore – at least in the context here – that component may be disregarded for most intents and purposes.

As suggested in [6], every syllable can be decomposed – in a deterministic fashion – into consonants and vowels located in specified positions. In some cases a particle is appended to the syllable preceding it, resulting in a disyllabic (or, rarely, trisyllabic) contraction. To accommodate these cases, we use an octuple (8-tuple), consisting of the following components (somewhat different from the decomposition used in [6] for standardizing the lexicographic ordering of syllables):

$$\langle \quad \text{prescript, superscript, core, subscript,} \\ \text{vowel, coda, postscript, appended particle} \rangle$$

Some of the positions may remain empty or contain a special value to indicate their absence.

In Fig. 1, stack A holds the prescript component, stack B holds the superscript, core letter, subscript, and vowel components. Stack C holds the coda (final letter), and D, the postscript. An additional position E holds the appended particle(s).

Thus, for example, the future tense *bsgrub* and the imperative *sgrubs* of the verb *sgrub* (to perform) would take the following forms:

$$bsgrub = \langle\ b,\ s,\ g,\ r,\ u,\ b,\ -,\ -\ \rangle \\ sgrubs = \langle\ -,\ s,\ g,\ r,\ u,\ b,\ s,\ -\ \rangle$$

The disyllabic contraction *sgra'ang*, to give another example, would take the form:

$$sgra'ang = \langle\ -,\ s,\ g,\ r,\ a,\ -,\ -,\ 'ang\ \rangle$$

Each location in the tuple is governed by a different set of rules. Some combinations are possible, while other combinations never occur and their appearance would suggest either a transliteration error, scribal error (or a damaged woodblock), or the presence of a non-Tibetan word, such as a Sanskrit word transliterated in Tibetan script.

Table 1. Consonant distribution within the Tibetan Buddhist canon. (The remaining consonants each appear less than 3% of the time.)

s	*d*	*g*	*b*	*r*	*n*	*y*	*p*	*m*	*ng*	*l*
10%	8%	6.5%	6.5%	6.5%	5.6%	5.5%	5.1%	4.4%	4.4%	3.4%

Table 2. Top 5 unigrams, bigrams, and trigrams in the Tibetan canon.

Unigrams		Bigrams		Trigrams	
n-gram	%	n-gram	%	n-gram	%
pa	5.6	*pa dang*	0.0071	*zhes bya ba*	0.0050
dang	2.7	*bya ba*	0.0068	*la sogs pa*	0.0028
ba	2.4	*zhes bya*	0.0067	*bya ba ni*	0.0025
par	2.3	*la sogs*	0.0045	*bcom ldan'das*	0.0020
ni	2.0	*thams cad*	0.0041	*byang chub sems*	0.0010

3 Stemming

Since syllables having the same stem may take many different forms, stemming is a crucial stage in almost every text-processing task one would like to perform in Tibetan. Usually, in Indo-European and Semitic languages, stemming is performed on the word level. However, in Tibetan, in which words are not separated by spaces or other marks, a syllable-based stemming mechanism is required even in order to segment the text into lexical items. We should point out that (heuristic) stemming does not mean the same thing as (grammatical) lemmatization, and the stemming process can result in a stem that is not a lexical entry in a dictionary. Moreover, unlike other Indo-European languages, stemming of Tibetan is mostly relevant to verbs and verbal nouns (which are common in the language). Despite being inaccurate in some cases, stemming (for Tibetan as for other languages) can improve tasks such as word segmentation and intertextual parallel detection [8]. Moreover, even for Tibetan words consisting of more than one syllable, stemming each syllable makes sense since all the inflections are embedded at the syllable level. For instance, the words *brtag dbyad* (analysis) and *brtags dpyad* (analyzed) are stemmed to *rtog dpyod* (to analyze, analysis).

The following are the main rules that govern the structure of the syllable [4].

- There are 30 possibilities for the core letter: any of the 29 consonants or the core letter *a* qua consonant.
- There are 5 vowels, one of which must be present: *a, i, u, e, o*.
- There are 3 possible superscripts: *r* (with core *k, g, ng, j, ny, t, d, n, b, m, ts, dz*); *l* (with *k, g, ng, c, j, t, d, p, b, h*); *s* (with *k, g, ng, ny, t, d, n, p, b, m, ts*).
- There are 4 subscripts: *y* (with *k, kh, g, p, ph, b, m*); *r* (with *k, kh, g, t, th, d, p, ph, b, m, sh, s, h*); *l* (with *k, g, b, z, r, s*); *w* (with *k, kh, g, c, ny, t, d,*

ts, tsh, zh, z, r, l, sh, s, h). In rare cases, the combinations *rw* and *yw* may also appear as subscripts, e.g. in the syllables *grwa* and *phywa*.
- There are 10 possible codas (final letters): *g, ng, d, n, b, m,', r, l, s*.
- There are 5 possible prescripts: the letters *g* (with *c, ny, t, d, n, zh, z, y,*[1] *sh, s, ts*), *d* (with *k, g, ng, p, b, m, ky, gy, py, by, my, kr, gr, pr, br*), *b* (with *k, g, c, t, d, zh, z, sh, s, ky, gy, kr, gr, kl, zl, rl, sl, rk, rg, rng, rj, rny, rt, rd, rn, rts, rdz, lt, sk, sg, sng, sny, st, sd, sn, sts, rky, rgy, sky, sgy, skr, sgr*), *m* (with *kh, g, ng, ch, j, ny, th, d, n, tsh, dz, ky, gy, khr, gr*), ' (with *kh, g, ch, j, th, d, ph, b, tsh, dz, khy, gy, phy, by, khr, gr, thr, dr, phr, br*).
- There are 2 possible postscripts, which come after the coda: *s*, *d* (the suffix *d* is archaic and seldom found).
- There are 6 particles that are appended at the end of syllables: *'am,'ang,'i,'is,'o,'u*. This is only possible with syllables ending with a vowel (i.e. lacking a final letter, and thus by definition also a postscript), or with the final letter *'*. The appending of the particle results in a disyllabic contraction (while the two vowels are often pronounced as diphthongs). Rarely, two particles can also be appended (e.g. *phre'u'i*). However, since for the stemming we regard the appended particle(s) as a single unit, which is not stemmed, these cases of doubled-appended syllables do not affect stemming and thus are disregarded. There are two additional possible particles that can be appended at the end of a syllable: *s* and *r*. Since both *s* and *r* are also valid codas, this may cause ambiguity. (The potential problem is partially solved for the letter *s* in the normalization stage, but a full solution is difficult to achieve.)

This gives an upper bound of $(30 \times 6 \times 4 \times 5 \times 11 =)$ 46,200 for the number of stems (ignoring the rare doubled subscripts). But because not all consonants take all subscripts and superscripts (the ample restrictions concerning the possible combinations of the prescripts remain disregarded), the actual bound is a fraction thereof: 9075 $(= 165 \times 5 \times 11)$. See Table 3.

Using the previously described tuple-representation of the Tibetan syllable, we define the stem of a syllable to be the quintuple consisting of the eight original parts minus prescript, postscript, and appended particle:

$$\langle \text{superscript}, \text{core}, \text{subscript}, \text{vowel}, \text{coda} \rangle$$

The stem can be written in the following format:

superscript
core+vowel -coda
subscript

The deleted parts do not change the basic underlying semantics of the syllable.

The stemmer works in the following manner: first, we break the syllable into a list of Tibetan letters. This stage is required because Wylie transliteration represents some Tibetan letters by more than one character (e.g. *zh, tsh*). There is, fortunately, no ambiguity in the process of letter recognition. By design,

[1] Transliterated *g.y* to differentiate from core *g* with subscript *y*.

the transliteration scheme ensures that whenever a sequence of two or three characters represents a single letter, it cannot also be interpreted in context as a sequence of distinct Tibetan letters.

Each Tibetan syllable should contain one core letter and one vowel. Other positions (subscript, etc.) are not obligatory; there should be only one letter that

Table 3. Possible superscripts and subscripts.

Core	Superscript			Subscript				Total
k	*r*	*l*	*s*	*y*	*r*	*l*	*w*	20
kh				*y*	*r*		*w*	4
g	*r*	*l*	*s*	*y*	*r*	*l*	*w*	20
ng	*r*	*l*	*s*					4
c		*l*					*w*	4
ch								*1*
j	*r*	*l*						3
ny	*r*		*s*				*w*	6
t	*r*	*l*	*s*		*r*		*w*	12
th					*r*			2
d	*r*	*l*	*s*		*r*		*w*	12
n	*r*		*s*					3
p		*l*	*s*	*y*	*r*			9
ph				*y*	*r*			3
b	*r*	*l*	*s*	*y*	*r*	*l*		16
m	*r*		*s*	*y*	*r*			9
ts	*r*		*s*				*w*	6
tsh							*w*	2
dz	*r*							2
w								1
zh							*w*	2
z						*l*	*w*	3
’								1
y								1
r						*l*	*w*	3
l							*w*	2
sh					*r*		*w*	3
s					*r*	*l*	*w*	4
h	*l*				*r*		*w*	6
a								1
								165

fits each of the seven places in the tuple, while the eighth place accommodates a syllable, one of 6 possible appended particles. We therefore start with the detection of all the vowels (by definition, each syllable contains one vowel). A contraction consisting of an appended syllable commonly contains two vowels. (As noted earlier, the rare case of a double-appended syllable has no effect on the stemming.) Syllabic contractions should contain two vowels at most.

The vowel (*a, i, u, e, o*) necessarily follows the core letter, or the subscript (*y, r, l, w*, rarely also *yw, rw*) if there is one. Examples are *bam* (*b* is the core letter); *bsgrubs* (*g* is the core letter); *'ga'* (*g* is the core letter); *zhwa* (*zh* is the core letter); *chen* (*ch* is the core letter). If the syllable begins with a vowel, the core letter in our representation is set to be *a* (meaning, we add an extra *a*), which makes *ag* a valid syllable with core letter *a*, vowel *a*, and coda letter *g*. Another example is the syllable *e* that would be represented as having core letter *a* and vowel *e*.

The stem of the syllable consists of the core letter or the stacked letter (which, in turn, consists of the core letter and a superscript, or a subscript, or both), the vowel, and the final letter (if this is found). Syllables can be considered to be stemmically identical if these are consistent, despite additions or omissions of a prescript and/or a postscript.

Under certain circumstances (commonly inflection of verbs), the core letter may be changed. However, the change is not arbitrary, and usually occurs among phonetically "related" letters, such as *k/kh/g; c/ch/j; t/th/d; p/ph/b.*

Possible changes can be found in the vowel, while still retaining the same basic meaning (and the same stem). Most commonly the vowel *o* in verbs changes to *a* and vice-versa, reflecting a change in tense. Since other vowel changes are unfortunately also possible, it seems impossible to identify a pattern. The only viable solution would be to work with a list of verbs and their inflections, or alternatively, to consider a vowel change as substantial (thus failing to recognize the stemmic identity).

1. In the case of an appended syllable, the first vowel is considered to be the main one in the syllable and is placed in the vowel location of the tuple.
2. The second vowel can only be part of one of the 6 possible appended particles. We place the entire particle in the eighth location of the tuple, without breaking it into its component letters, as we are only interested in the complete particle. Any other case is considered a non-Tibetan syllable that is transliterated in Tibetan script, and consequently the syllable is not stemmed.
3. Following the detection of the vowel of the syllable to be stemmed, we place the letters before and after this vowel in the appropriate places in the tuple, according to the constraints for each position. If no legal decomposition is found, the syllable is not stemmed.

The final stage is normalization. As it turns out, there are groups of Tibetan letters that can be replaced one with the other without changing the basic meaning of the syllable. Since we are interested in grouping all syllables that are ultimately stemmically identical into one and the same stem, we normalized all tuples according to the following rules:

A. *a, o* are the same when in the vowel position.
B. *c, ch, j, zh, sh* are the same in the core letter position.
C. *ts, tsh, dz, z* are the same in the core position.
D. *s* in the coda position may be simply omitted.[2]

(We have glossed over a few additional special cases and peculiarities that are dealt with in the stemmer code.)

Once we have the tuple corresponding to the syllable, we extract the components ⟨superscript, ..., final letter⟩ to obtain a quintuple that represents the syllable's stem. For *bsgrub* and *sgrubs*, future tense and imperative of the verb *sgrub*, the stemming process will generate the same stem: *sgrub*.

The stemming tool is available at http://www.cs.tau.ac.il/~nachumd/Tools.html; the code is at https://github.com/lenadank/Tibetan_Stemmer.git.

For the similarity measures and word-segmentation tasks, described in the following sections, each letter is encoded by a number. The particles, as previously mentioned, are encoded as themselves, so overall we have a total of 41 possible values for the various locations in the tuple (29 consonants [excluding *a*], 5 vowels, 6 particles, blank).

4 Learning Similarity

The stemmer, as described above, extracts the information encoded in each Wylie transliterated syllable and makes it explicit. An important task, given two syllables, is to evaluate their stemmic similarity. One can, for example, examine the tuple representation of the syllables and count the number of places in which they differ. While this is a reasonable baseline, it does not take into account the relative importance of each component. This approach also misses the importance of each substitution made. Some substitutions can be considered silent or synonymous; others change the meaning completely; and there is a continuous spectrum in between.

Assessing the relative importance of each substitution by experts is infeasible. We, therefore, use metric learning algorithms for this task. Specifically, we employ Support Vector Machines (SVMs) as described below. The dataset we used for learning close stemmic similarities contains a list of syllables divided into 1521 sets. It was extracted from a list of verb inflections provided by Paul Hacket.

Since this inflection dataset is limited in extent, we model each substitution as two independent changes, and the importance of each substitution, per location in the tuple, is computed as the sum of two learned weights. One weight is associated with the letter in one syllable and the other associated with the parallel letter in the second. This way, instead of a quadratic number of parameters, we have only a linear number.

[2] The reason for omitting the coda *s* for the sake of normalization is that in cases where it is added to form the past tense, which results in a syllable that appears to have a stem with coda *s*, we treat this *s* as equivalent to the postscript *s* often added to form the past tense.

4.1 The Learned Model

Given the stemmer's output for two Wylie encoded syllables $x_i, y_i \in \mathbb{R}^5$, we first re-encode it as a more explicit tuple by using an encoding function. Three types of such functions are considered.

In the first type, the encoding is simply the identity function. This is a naïve approach in which the rather arbitrary alphabetic distance affects the computed metric.

The second type encodes each possible letter in each of the five locations (superscript, ..., final letter) as one binary bit. The bit is 1 if the associated location in the stemmed representation has the value of the letter associated with the bit, and 0 otherwise. This representation learns one weight for each letter at each location. If the two syllables x_i and y_i differ in three locations out of the five, the learned model would sum up six weights: each of the three locations would add one weight for each of the two letters involved in the substitution.

The third type of encoding is based on information regarding equivalence groups of letters. In other words, substitutions within each group are considered synonymous. There are five groups with more than one letter:

1. *g, k, kh*
2. *c, ch, j, zh, sh*
3. *d, t, th* close
4. *b, ph, p*
5. *z, dz, tsh, ts*

The rest of the letters form singleton groups. The total number of groups is 21.

Let f be the encoding function. The learned model has a tuple of parameters w, which has the same dimension as $f(x)$, and a bias parameter b. It has the form: $w^\top |f(x_i) - f(y_i)| + b$, that is, a weighted sum of the absolute differences between the encoding functions of the two stemmed syllables.

During training, synonymous and non-synonymous pairs of syllables are provided to the SVM algorithm [3]. Each pair is encoded as a single tuple $|f(x_i) - f(y_i)|$, and an SVM with a linear kernel is used to learn the parameters w and b.

4.2 Evaluation

The dataset contains 1521 sets of verbs and their inflectional forms. The sets are divided into three fixed groups in order to perform a cross validation accuracy estimation. In each cross validation round, two splits are used for training and one for testing. Within each group, all pairs of syllables from within the same set (inflections of the same verb) are used as positive samples. There are 110–140 such pairs in each of the splits. Ten times as many negative samples are sampled.

Table 4 presents the results of the experiments. The area under the ROC curve (AUC) is used to measure classification success. We compare two methods: one does not employ learning and simply observes the Euclidean distance

Table 4. Comparison of the three encoding functions used for metric learning. Results for both the Euclidean (L2) distance and SVM-based metric learning are shown. The reported numbers are mean AUC ± SD over three cross-validation splits.

Method	Naïve	Binary	Equivalence groups
L2 distance	0.7990 ± 0.0149	0.9282 ± 0.0148	0.9534 ± 0.0203
SVM metric learning	0.9129 ± 0.0211	0.9723 ± 0.0212	**0.9808 ± 0.0154**

$\|f(x_i) - f(y_i)\|$; the other is based on learning the weights w via SVM. We compare the three functions f described above: (i) Naïve, (ii) Binary, and (iii) Equivalence groups.

As can be seen, the Equivalence group function significantly outperforms the other functions. It is also evident that learning the weights with SVM is preferable to employing a constant weight matrix (which results in a simple Euclidean distance).

5 Word Segmentation

The problem of word segmentation, viz. grouping the syllables into words, is of major importance. Since no spaces or special characters are used to mark word boundaries, the reader has to rely on language models so as to detect the word boundaries.

5.1 Design

The approach we take is based on a flavor of recurrent neural networks (RNNs) called "long short-term memory" (LSTM) [5]. LSTMs have been used in the past for word segmentation of Chinese text [2]. Our work differs from previous work in that we rely on the tuple representation of the syllable, while previous works represent each syllable out of a sizable group of syllables as an atomic unit. Our input tuple is therefore much more compact.

The word-segmentation pipeline consists of the following steps.

First, we represent each syllable using an encoding function that is similar to the Binary function of Sect. 4: each possible letter in each location is assigned a single bit to indicate its existence at this location.

Then, for each syllable, the surrounding syllables are collected to form a context-window of size 5. At the text boundaries, we pad with 0's. This context is represented as a single tuple that results from concatenating five tuples: up to two syllables before, and two syllables after, the current syllable.

Since we consider 5 syllables per time frame, 8 parts, and up to 41 symbols per part, the size of our representation is 1640. As mentioned, when the context-window extends beyond the beginning or end of a sentence, tuples of 0's are used for padding.

The neural network is presented in Fig. 2. It consists of a single LSTM layer with 100 hidden units, followed by a softmax layer. Following the convention of

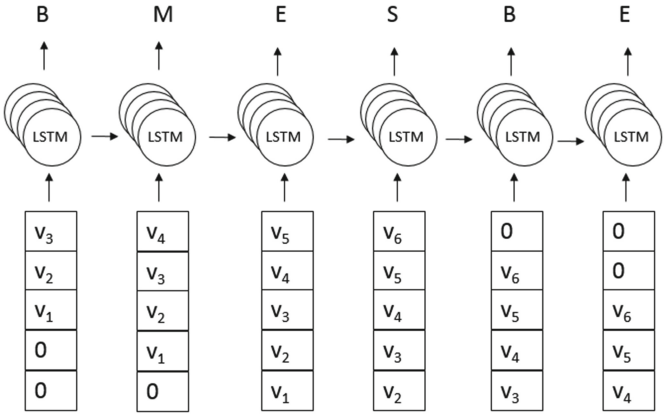

Fig. 2. The LSTM network used for word segmentation.

Table 5. The performance of the word segmentation neural network.

Context size	Classification type	Precision	Recall	F1 score
3 syllables	{B, M, E, S}	0.89	0.88	0.88
	Binary	0.93	0.93	0.93
5 syllables	{B, M, E, S}	0.90	0.90	0.90
	Binary	**0.95**	**0.95**	**0.95**
7 syllables	{B, M, E, S}	0.90	0.90	0.90
	Binary	0.94	0.94	0.94

previous work [2], for each syllable there are 4 possible target labels, indicating whether the syllable is the beginning of a word (B), middle of the word (M), end of the word (E), or constitutes a single-syllable word (S).

An alternative network was also trained, which has only two target labels, 1 when the current syllable is the end of a word and 0 when it's not. This is sufficient for segmenting words – we stop each word after a label of 1 and immediately begin another. Note that in the 4-label network, there is no hard constraint that precludes, for example, a word-middle (M) to appear right after the end of a word (E) or a single syllable word (S). In the binary alternative we propose, consistency is ensured by construction.

5.2 Evaluation

Training data was downloaded from the Tibetan in Digital Communication project (http://larkpie.net/tibetancorpus). The training set consisted of 36,958 sentences and the test set of 9239 sentences. Overall, there are 349,530 words consisting of an average of 1.46 syllables per word.

Training of the network was accomplished using cross-entropy loss and with the Adam learning rate scheduling algorithm [7]. A dropout layer of 50% is added before the LSTM layer.

Table 5 summarizes the performance of the neural network when trained with context-windows sizes of 5 and 7. As can be seen, a context size of 5 syllables works somewhat better, and the proposed binary network outperforms the multilabel network.

The online segmentation tool is available at http://www.cs.tau.ac.il/~nachumd/Tools.html. A screenshot for a sample text is shown in Fig. 3.

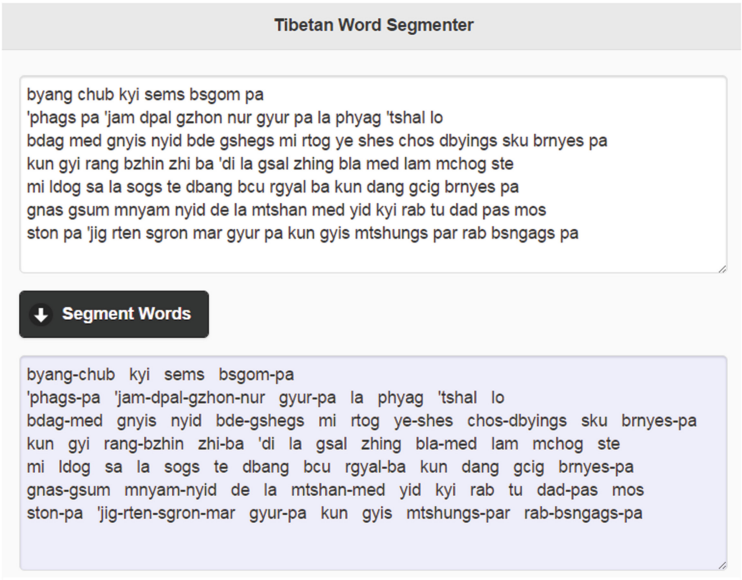

Fig. 3. A screenshot of the online word segmentation application.

6 Conclusion

We have seen the practicality of designing a rule-based stemmer for the syllables of a monosyllabic language like Tibetan. This contributes to an analysis of the morphology of the Tibetan syllable and provides a basis for the development of additional linguistic tools.

We plan on experimenting with the possibility of semi-supervised learning of such a stemmer, and comparing results with this rule-based approach.

The creation of a practical stemming tool for Tibetan made it possible for us to build a reasonable word-segmentation algorithm. We also plan to use it for the development of intelligent search and matching tools for classical Tibetan.

Acknowledgements. We would like to express our deep gratitude to the other participants in the "Hackathon in the Arava" event (held in Kibbutz Lotan, Israel, February 2016; see [1]), who all contributed to the development of new digital tools for analyzing Tibetan texts: Kfir Bar, Marco Büchler, Daniel Hershcovich, Marc W. Küter, Daniel Labenski, Peter Naftaliev, Elad Shaked, Nadav Steiner, Lior Uzan, and Eric Werner. We thank Paul Hacket for crucially providing the necessary data.

This research was supported in part by a Grant (#I-145-101.3-2013) from the GIF, the German-Israeli Foundation for Scientific Research and Development, and by the Khyentse Center for Tibetan Buddhist Textual Scholarship, Universität Hamburg, thanks to a grant by the Khyentse Foundation. N.D.'s and L.W.'s research was supported in part by the Israeli Ministry of Science, Technology and Space (Israel-Taiwan grant #3-10341). N.D.'s research benefitted from a fellowship at the Paris Institute for Advanced Studies (France), with the financial support of the French state, managed by the French National Research Agency's "Investissements d'avenir" program (ANR-11-LABX-0027-01 Labex RFIEA+).

References

1. Almogi, O., Dankin, L., Dershowitz, N., Wolf, L.: A hackathon for classical Tibetan J. Data Min. Digital Humanit. (to appear)
2. Chen, X., Qiu, X., Zhu, C., Liu, P., Huang, X.: Long short-term memory neural networks for Chinese word segmentation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, pp. 1197–1206. Association for Computational Linguistics, September 2015. http://aclweb.org/anthology/D15-1141
3. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018
4. Hahn, M.: Lehrbuch der klassischen tibetischen Schriftsprache, Indica et Tibetica, 7th edn., vol. 10. Indica et Tibetica Verlag, Marburg (1996)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
6. Huang, H., Da, F.: General structure based collation of Tibetan syllables. J. Comput. Inf. Syst. **6**(5), 1693–1703 (2010)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, May 2015. http://arxiv.org/pdf/1412.6980v8.pdf
8. Klein, B., Dershowitz, N., Wolf, L., Almogi, O., Wangchuk, D.: Finding inexact quotations within a Tibetan Buddhist corpus. In: Digital Humanities (DH 2014), Lausanne, Switzerland, pp. 486–488, July 2014. http://nachum.org/papers/textalignment.pdf
9. Liu, H., Nuo, M., Wu, J.: Zipf's law and statistical data on modern Tibetan. In: COLING (2014)
10. Wylie, T.V.: A standard system of Tibetan transcription. Harvard J. Asiatic Stud. **22**, 261–267 (1959)
11. Zipf, G.K.: Human Behaviour and the Principle of Least Effort. Hafner Pub. Co., New York (1949)