Detecting Sentence Boundaries in Sanskrit Texts

Oliver Hellwig
Düsseldorf University, SFB 991
ohellwig@phil-fak.uni-duesseldorf.de

Abstract

The paper applies a deep recurrent neural network to the task of sentence boundary detection in Sanskrit, an important, yet underresourced ancient Indian language. The deep learning approach improves the F scores set by a metrical baseline and by a Conditional Random Field classifier by more than 10%.

1 Introduction

Most NLP tasks that deal with written texts take it for granted that sentences are separated reliably by punctuation marks, although punctuation has been added quite late to many writing systems. The large corpora in Old- and Middle-Indian languages, which belong to the central sources for understanding the history of South Asia, generally lack dedicated punctuation marks. This paper applies deep recurrent neural networks (RNN) to a combination of morphological and lexical features for detecting sentence boundaries (SB) in Sanskrit, the oldest and most important of these Indian languages.

Traditional editions of Sanskrit texts use a *scriptio continua*, which lacks several orthographic elements that structure texts in modern Western languages. Single words are frequently not separated by blank spaces due to missing orthographic regulation, or because the words are merged through the euphonic rules called *sandhi* ("connection").[1] Moreover, Sanskrit texts don't have a consistent and unambiguous system for marking SBs. Editors and scribes insert so called (double) *daṇḍa*s ("sticks", indicated by | and || in this paper) to mark the end of metrical structures. The position of these *daṇḍa*s can be derived directly from the prosodic structure of a text, and *daṇḍa*s always occur at the end of text lines, which coincide with half-verses in most printed editions. While single *daṇḍa*s mark the end of a half-verse, double *daṇḍa*s should, at least theoretically, indicate, where a stanza in the given metre is completed. Double *daṇḍa*s typically occur after every second line or half-verse of a metrical text, because the stanzas are finished at these points. In this function, they are meant to improve the readability of a text. As many sentences terminate at the end of a half-verse or of a stanza, *daṇḍa*s provide a good baseline for punctuation prediction (refer to Table 3). Many editors, however, also insert double *daṇḍa*s after a single or after three metrical lines, when they feel that a sentence is completed at these positions.[2] In this way, the purely metrical motivation of double *daṇḍa*s is mingled with the new function of a punctuation mark – leaving aside the fact that the philologically interesting inner-line SBs cannot be marked in the *daṇḍa* system.

Linguistic peculiarities of Sanskrit complicate the task. English, for example, encodes a large amount of its syntax through a strongly regulated word order, and structures its sentences by subordinating conjuctions. While these data provide a lot of the information necessary for restoring punctuation, Sanskrit has a rather loose word order with a tendency to subject-object-verb constructions, it uses conjunctions quite sparingly, and their position provides only weak indications for the presence of SBs. As the Indian

[1]The two words *parvatasya agre*, for example, are merged into one string *parvatasyāgre* by the rule *a*+*a*=*ā*; refer to Kielhorn (1888, 6ff.) for an overview. *Sandhi* is one of the main problems for Sanskrit NLP.

[2]Refer to Hopkins (1901, 194): "The number of verses in a (...) stanza may be decreased or increased by one or two (...). Sometimes, however, where one or three hemistichs make a stanza, it is merely a matter of editing."

grammatical tradition has emphasized (Section 2), determining the boundaries of a sentence is equivalent to grasping its full semantic meaning.

The need for reliable punctuation on sentence level is beyond question. Access to full sentences is central for NLP tasks such as dependency parsing or role labeling. In addition, detecting SBs is also important from a philological perspective. The metrical texts considered in this paper belong to a tradition of (pseudo-)oral poetry that still survives in parts of India (Smith, 1987). The constituent structure of sentences (e.g., extensive right branchings) or the presence of enjambements, which are easily detected when SBs are known, provide important evidence for understanding the transition of these epics from an oral to a written state (Sellmer, 2015; Parry, 1930). More generally, Sanskrit provides a challenging application scenario for NLP due to the richness of its phonetics (*sandhi*), morphology, lexicon, and semantics. In spite of its historical importance, it is heavily underressourced from the perspective of NLP, and the size of its corpus prevents a purely manual annotation of linguistic phenomena.[3]

The remainder of the paper is structured as follows. Section 2 sketches how a sentence was defined in the tradition of classical Indian grammar, and summarizes related research from NLP and automatic speech recognition. Section 3 reports results of a test annotation, details the annotation guideline, and describes the data prepared for this study. Section 4 introduces the features and the deep learning model. Section 5 describes the evaluation baselines given by prosodical markers and a CRF model, discusses the performance of the model, and identifies critical areas. Section 6 summarizes the paper.

2 Related Work

Classical Sanskrit was systematically de- and prescribed in the famous grammar Aṣṭādhyāyī of Pāṇini (around 350 BCE, Scharfe (1977)), who used Sanskrit as a metalanguage, and applied methods such as rewrite rules and rule inheritance for minimizing the text length (Kiparsky, 2009). While the Aṣṭādhyāyī deals exhaustively with phonetics and morphology, syntax only plays a subordinate role. Its main syntactic contribution is the *kāraka* theory, which describes the interaction between nominal case suffixes and verbs (Cardona, 1976, 215ff.). The grammatical tradition following Pāṇini provided empirical, verb-centered definitions of sentences (Matilal, 1966, 377ff.). Because many Sanskrit sentences don't overtly express the copula "to be", these definitions gave rise to extended discussions about the underlying grammatical and cognitive structures of sentences such as *puṣpaṃ raktam* (flower:NSG red:NSG), which may mean "a red flower" or the complete sentence "the flower is red" (Deshpande, 1991). Missing copulae introduce a high degree of ambiguity in the SB detection task, as will be seen in Section 5.3. The later philosophical school of Nyāya concentrated on the conditions that make a sentence meaningful and complete for a competent speaker of Sanskrit (Matilal, 1966, 385ff.), and that include the semantic compatibility of the words (*yogyatā*) and their correct grouping (*saṃnidhi*; see Kulkarni et al. (2015)). If an utterance fulfills these conditions, it creates the intended cognition (*śābdabodha*) in the listener. So, the Indian tradition claims that only a competent speaker can determine the boundary of a sentence, but does not provide formal criteria for deciding if a sentence is complete or not.

Related research in NLP mainly deals with punctuation restoration in speech transcripts, and in languages such as Chinese that traditionally don't use punctuation marks for structuring syntactic sequences. Liu et al. (2005) contrast Hidden Markov Models (HMM), Maximum Entropy classifiers and Conditional Random Fields (CRF). They obtain a significant decrease of the SB detection error when processing lexical and automatically induced POS features using a CRF. Baldwin and Joseph (2009) perform simultaneous case and punctuation restoration in English texts. They process automatically annotated lexical, POS, and chunk features with a linear kernel Support Vector Machine. The authors report the highest F score for punctuation restoration, when they iteratively label the training and test sets with the output of the classifier, and retrain with the augmented feature space ("iterative retagging"). Zhao et al. (2012) train CRFs on the task of inserting punctuation in Chinese text, using features from different annotation levels of a Chinese treebank, and observe an increase in the F score, when higher level features such as

[3]There exist no reliable estimations of the real size of Sanskrit literature. The GRETIL website (`http://gretil.sub.uni-goettingen.de/`), which provides digital transcripts of a few percent of all printed Sanskrit texts, may contain around 15 million lexical tokens (estimation of the author; numbers may be significantly higher due to Sandhi). Large parts of the Sanskrit literature are still only transmitted as manuscripts.

POS tags or chunks are combined with lexical information. Tilk and Alumäe (2015) model the restoration of commas and periods in Estonian speech transcripts with a two-stage Long Short-term Memory (LSTM) approach. The first LSTM is trained on a large written corpus with lexical information in 1-hot-encoding as predictors and the associated punctuation as predicted classes. Following Seide et al. (2011, 26), the authors combine the output of the last hidden layer of this text LSTM with duration features from a smaller corpus of punctuated speech transcripts. This combined feature set is fed into a second LSTM that performs the final classification.

Algorithms based on short range models (n-grams, HMMs) or those requiring strict positional information may not be applicable to Sanskrit for several reasons. Sanskrit has a relatively free word order (Gillon and Shaer, 2005; Hock, 2013), and encodes many syntactical relations through its morphology, so that the positional information inherent in an n-gram model may not contribute as strongly as in English or Chinese. In addition, Sanskrit NLP suffers from data sparsity in the lexical domain. The corpus on which the models are trained contains 3,950,000 disambiguated lexical tokens. New data for pretraining a lexical model cannot be generated on the fly, because the phonetic phenomenon of Sandhi introduces a high degree of ambiguity (Hellwig, 2015b), and sufficiently large digitized Sanskrit corpora are missing.

CRFs as used by Liu et al. (2005) and Zhao et al. (2012) are more flexible than HMMs in modeling the feature space involved in SB detection, because their input features can, in principle, come from arbitrarily long ranges around a focus word, and because they are trained to maximize the classification accuracy. RNNs as used by Tilk and Alumäe (2015) are equally able to capture the long-range interactions between morphology, lexicon, and output symbols that can be hypothesized to play an important role in SB detection. Section 5 will compare their efficiency in the present task. The problems of exploding and vanishing gradients (Pascanu et al., 2013) can be handled with Long Short-Term Memory units (LSTM, for the vanishing ones (Hochreiter and Schmidhuber, 1997), combined with a gradient cutoff) or with Hessian free training of the network (Martens and Sutskever, 2011). Stacked LSTMs as used by Sutskever et al. (2014) with bidirectional units (Schuster and Paliwal, 1997) seem to provide a promising approach for labeling SBs in Sanskrit.

3 Data

3.1 Test annotation

The discussion in Section 2 has shown, that the Sanskrit grammatical tradition does not provide a solid basis for developing a practical annotation guideline for SBs. As a consequence, ten sequences of at least two metrical lines that contain complex syntactic phenomena were annotated independently by three external annotators and the author of the paper. Given the small size of the data set, this annotation was not primarily meant to determine the true inter-annotator agreement (IAA), but rather to obtain quantitative support for ambiguous cases in the annotation guideline. The lines were tokenized according to Western editorial standards without resolving Sandhis and compounds.

Assuming that a period can be inserted after each of the 360 tokens, the annotation yielded an IAA of 0.805, using Fleiss' κ (Fleiss, 1971). When only those tokens are considered after which at least one annotator inserted a period, the IAA drops to $\kappa = 0.312$. A detailed analysis shows that almost all unanimous annotations concern periods that coincide with (double) *daṇḍa*s, while there is substantial disagreement about inner-line periods.

3.2 Guideline

Drawing from the results of the initial annotation and from ideas proposed in Matilal (1966), this paper defines a Sanskrit sentence as a sequence of words that contains at least an overtly expressed finite verb (type s_1; minimal sentence length: one word[4]), or two non-verbal elements with an unexpressed copula denoting equivalence or existence[5] (type s_2). s_1 and s_2 can be expanded by (recursive and/or compound) subordinate clauses and matrix sentences. As a direct consequence, sentences on the s_1 or s_2 levels that

[4]Sentences such as *gacchāni* 'I shall go' don't need to overtly express the personal pronoun *aham*.

[5]Existence: *hastināpure vaṇik* "[there is/was a] merchant in [the city of] Hastināpura"; equivalence: *puṣpaṃ raktam*, see page 2.

are connected by a (coordinating) conjunction such as *ca* ‘and’ are interpreted as separate sentences in this paper. The following three cases need special consideration:

Overtly Expressed Subjects No period is inserted between main clauses separated by a coordinating conjunction such as *ca* ‘and’, if the first sentence overtly expresses the subject, and the following sentences use the same subject without overtly expressing it.

The particle *iti* The particle *iti* ‘thus’ marks the end of a direct speech, or of a personal opinion presented as a direct speech. The direct speech terminated by *iti* is interpreted as a matrix sentence and, therefore, not separated by an SB.

Formulae and interjections Interjections and formulaic phrases are marked as separate clauses, if they are not embedded as matrix sentences.[6]

3.3 Data

One annotator used the guideline (Section 3.2) to mark sentence ends in 226 chapters with 96,292 lexical tokens, which were drawn from the metrical texts in the Digital Corpus of Sanskrit (DCS, Hellwig (2015a)[7]). Although each chapter constitutes a single long sequence with unknown punctuation, most metrical texts simulate an oral presentation by inserting the stock line “[some person] said[8]” between closed narrative blocks. Therefore, the chapters have been split up into a total of 609 of such blocks (“sequences”), which represent the individual statements of the persons participating in a conversation. The epic Mahābhārata (MBH) contributes most of the data. Because the text has probably grown over centuries and incorporated diverse written and oral sources (Brockington, 1998), the predominance of the MBH does not bias the data unduly towards the style of one author.

A total of 9,562 SBs has been annotated. 85.6% of the SBs coincide with (double) *daṇḍa*s, which provide a strong baseline for SB detection (Table 3). The annotated chapters contain a total of 9,027 word types, 3,838 of which are hapax legomena. The sentences have a mean length of 10 tokens (median: 8), and 90% of all sentence lengths are found in the interval [3,20].

4 Experimental Settings

4.1 Features

This section describes which features were considered for SB detection, and motivates their use. Their influence on the prediction accuracy is reported in Section 5.1.

***Daṇḍa* information** Because *daṇḍa*s provide a strong baseline for SB detection (see Table 3), and omitting them drastically reduces the F score in all configurations, they are used as features in all settings. (Double) *daṇḍa*s are encoded as dummy variables.

Morphological Information Sanskrit has a rich, though partly ambiguous Indo-Aryan morphology. Nouns, adjectives, pronouns, and declinable verbal participles are inflected in eight cases, three numbers (including dual), and three genders, while finite verbal forms occur in three numbers, three persons, and over tenses and modes (aspects). Although Sanskrit also uses conjunctions to join subordinate and main clauses, verbal subordination is typically expressed by the indeclinable absolutive (gerund; *tvānta* and *lyabanta*).[9]

As morphology provides strong indications for the inner structure of a sentence, it is included in the feature set either in 1-hot- (**1h**, each observed combination of morphological subfeatures is mapped to

[6]Refer to Wackernagel (1978 (reprint from 1896, II, 5) on short sentences with a particle-like function.

[7]This corpus collects 279 texts from different domains with 3,950,000 tokens with gold-annotations on the morphological, lexical, and word semantic level.

[8]*vyāsa uvāca* “[The sage] Vyāsa said” is a typical example of these stock lines, which are always terminated by a single *daṇḍa*, and not written in śloka metre.

[9]A typical toy example for this construction runs like: *rāmo* (‘Rāma’ NSG) *vanaṃ* (‘forest’ ASG) *gatvā* (‘go’ ABS) *sītāṃ* (‘Sītā’ ASG) *paśyati* (‘see’ PR3.SG), “Rāma, having gone to the forest, sees Sītā.” = “After Rāma has gone to the forest, he sees Sītā.”

a distinct position in a 1-hot vector v_M) or in a decomposed encoding, in which each position of v_M encodes the presense or absence of a subfeature such as ‘nominative’ or ‘perfect tense’ (**dec**, refer to the featurization in Cotterell and Schütze (2015, 1289)). Both encoding modes (**1h**, **dec**) don’t distinguish between different morphological derivations of tenses and modes.

Because the DCS does not contain syntactic annotations, morphological information is also used to generate possible syntactic links. Given a context size of $s = 5$, a word w_p at position p in a sequence, and the set of words $W_q = \{w_q| \, |q - p| \leq s, q \neq p\}$, a link between w_p and w_q is generated, if (1) $p < q$ and w_p belongs to the same compound (*samāsa*) as w_q, (2) w_p and w_q are nominal forms with the same case, number, and gender, (3) if one of w_p and w_q is a verb and the other one a congruent nominative, (4) if one of w_p and w_q is an absolutive and the other one a finite verb, or (5) if w_p and w_q belong to a set of correlative conjunctions and pronouns such as *yadā* ‘when’-*tadā* ‘then’ or *yad* ‘which’-*tad* ‘that’. These links are encoded as two sums weighted with $\frac{1}{|p-q|}$ for the left and right contexts. Although the existence of a link does not guarantee that w_p and w_q belong to the same sentence, t-tests of the weighted values with the SB labels as binary factor yield highly significant test statistics of $t = -31.99$ (left) and $t = 45.70$ (right), such that testing the predictive power of these features appears justified.

Lexical Information Sanskrit has a rich vocabulary, and Sanskrit authors put importance on the use of synonyms. An unsophisticated text such as the MBH, for example, uses 35 synonyms to denote the warrior Arjuna, or 14 for the concept “mountain”. As a consequence, one faces considerable data sparsity, when lexical information is used in 1-hot encoding. Low dimensional embeddings built from reduced vector space models (VSM, Turney and Pantel (2010)) or from neural networks (Bengio et al., 2003; Mikolov et al., 2011) have been shown to offer a workaround for this problem. Therefore, word embeddings generated with the `word2vec` tool (Mikolov et al., 2011)[10] are used as lexical features in all configurations marked with **w2v**.

As an alternative to a fully lexicalized model, the setting **indecl** uses the set of the most frequent 100 conjunctions and indeclinables in 1-hot encoding as the sole lexical information. This setting is motivated by the idea that these words indicate the basic structure of a sentence.

Yuret (1998) has shown that pointwise mutual information (PMI) between words can be used for building dependency structures. Therefore, normalized PMI for a window of size $s = 5$ around each w_p is added to the feature space in analogy to the syntactic links described above. PMI is either calculated from lexical information (**lpmi**), or from a mixture of lexical and word semantic data (**lspmi**).

4.2 Network Architecture and Settings

The RNN consists of a linear input layer with a dropout rate of 0.1 (Hinton et al., 2012), one or more bidirectional LSTM layers without peephole units, and a softmax output layer. All network weights are randomly initialized with a uniform distribution in $[-0.01, +0.01]$. The initial learning rate is set to 0.0008, and linearly decreased to the value of 0.0001. Training is performed with stochastic gradient descent, gradient clipping at the LSTM units, and a constant momentum of 0.95. Because the output of the network is a single binary variable, it is decoded using a threshold of 0.5. The model is implemented in C++.

The experiments reported in Section 5 are performed with a ten-fold cross-validation (CV). In order to make the results comparable to each other, the same random split of the data was used in all experiments.

5 Evaluation

5.1 Feature Selection

In order to assess how the features (Section 4.1) influence the classification results, flat networks with dropout and one bidirectional LSTM are trained on subsets of morphological and lexical features. Table 1 shows that the decomposed morphological encoding creates better results than the 1-hot encoding. It may be conjectured that the decomposed version can estimate the relevance of rare morphological features (e.g., genitive dual) from their more frequent subfeatures (e.g., genitive in all numbers). The

[10]Settings: Trained with full chapters, i.e. *daṇḍa*s were ignored; embedding size: 200, bow, window size: 10, 5 iterations.

Enc.	Links?	P	R	F
dec	no	85.08	**79.41**	**82.15**
dec	yes	**85.58**	77.92	81.57
1h	no	84.83	78.42	81.50
1h	yes	84.62	78.02	81.19

Table 1: Influence of morphological features on precision and recall of LSTMs. Enc.: encoding type; links: hardcoded morphological links used? LSTM architecture: dropout → bidirectional LSTM → softmax, 100 hidden units, 10 CVs, 50 iterations; lexical features: **freqindecl**, lexical links: **lpmi**

Lexicon	Links	P	R	F
none	**none**	83.50	78.59	80.97
freqindecl	**lpmi**	84.44	78.68	81.46
freqindecl	**none**	84.86	77.65	81.10
freqindecl	**lspmi**	85.08	79.24	82.05
w2v	**lpmi**	= Table 1, row 1		
w2v	**none**	**85.78**	79.01	82.26
w2v	**lspmi**	85.50	**79.31**	**82.28**

Table 2: Influence of lexical features; settings as in Table 1; decomposed morphological features (**dec**), no morphological links

hard-coded morphological links don't improve the performance, and are therefore discarded from the feature set.

Table 2 shows that lexical features have a noticeable, though not too large effect on the classification results. While the fully unlexicalized setting produces the worst results, the combination of word embeddings (**w2v**) with semantically enriched lexical links (**lspmi**) produces the best F scores (Table 2). This result demonstrates that neural language models create meaningful embeddings even from small training corpora, although the full lexical disambiguation of the training data is certainly helpful in learning proper representations.

The LSTM models for the final evaluation are trained on decomposed morphological features, neural embeddings of size 200, and semantically enriched lexical links. The basic architecture follows the description in Section 4.2, and the number of inner bidirectional LSTM layers is set to 2 or 3.

5.2 Comparing baseline models and LSTMs

Table 3 presents the baselines and the results for different LSTM architectures. The prosodical baselines are calculated by inserting an SB either at each *daṇḍa* or double *daṇḍa* ("baseline *daṇḍa*"), or at each double *daṇḍa* only ("baseline double *daṇḍa*"). As remarked in Section 1, editors tend to move double *daṇḍa*s by one line, if they are able to indicate an SB in this way. Therefore, double *daṇḍa*s present a rather precise baseline for SB detection in metrical Sanskrit texts, although their recall is low.

As many previous papers apply CRFs to SB detection (Section 2), CRFs trained with morphological features and different levels of lexical features are used as a second set of baselines. The central rows in Table 3 show that the F scores of CRFs are only slightly higher than those of the prosodical baselines. Error analysis reveals that CRFs base their predictions mainly on *daṇḍa* information, which explains the comparatively small differences between the F scores of the two baselines (75.55 vs. 73.80).

Bidirectional LSTMs significantly outperform both baselines. Comparing Tables 1, 2 and 3 shows that their F scores increase with their depth, i.e. the number of stacked LSTM layers. When using a deeper architecture, the strongest improvements are observed in the model recall. The best single model in Table 3 almost reaches precision and recall of the two metrical baselines, and improves the F score of the double *daṇḍa* baseline by almost 13%.

Classifier	Architecture	P	R	F
Prosodical baseline	*daṇḍa*	59.34	**85.54**	70.07
	double *daṇḍa*	**89.34**	62.86	73.80
CRF	no lex.	83.85	68.75	75.55
	freqindecl	83.82	68.74	75.54
	w2v	85.73	67.24	75.37
LSTM	2, dropout	86.92	80.70	83.70
	3	87.07	75.62	80.94
	3, dropout	88.53	85.13	**86.79**

Table 3: Comparison of baselines and LSTM architectures; settings for CRF: **lpmi**, **dec**, features extracted from a window of size ± 6 around each word, 10 CVs; settings for LSTM: **w2v**, embedding size: 200, **lspmi**, no morph. links; 100 hidden units in each bidirectional LSTM layer, 10 CVs, 50 epochs

		Length class				
b_i-e_i-i_i	Proportion	1 ($\leq$ 4 words)	2 (5-9 w.)	3 (10-14 w.)	4 (15-29 w.)	5 ($\geq$ 30 w.)
1-1-1	65.08	733 (41.91)	2234 (64.94)	2348 (80.25)	865 (66.03)	43 (31.39)
0-1-1	12.01	516 (29.50)	507 (14.74)	82 (2.80)	36 (2.75)	7 (5.11)
1-0-1	11.47	369 (21.10)	499 (14.51)	171 (5.84)	50 (3.82)	8 (5.84)
1-1-0	6.98	8 (0.46)	90 (2.62)	238 (8.13)	272 (20.76)	59 (43.07)
0-0-0	0.38	5 (0.29)	9 (0.26)	10 (0.34)	9 (0.69)	3 (2.19)
0-0-1	1.41	89 (5.09)	32 (0.93)	6 (0.21)	6 (0.46)	2 (1.46)
1-0-0	1.61	15 (0.86)	33 (0.96)	49 (1.67)	46 (3.51)	11 (8.03)
0-1-0	1.07	14 (0.80)	36 (1.05)	22 (0.75)	26 (1.98)	4 (2.92)

Table 4: Sentence based evaluation of the output of the best RNN from Table 3, stratified by sentence length classes (columns 3ff.). Column 1: $b_i = 1$: b(eginning) of sentence s_i is detected; $e_i = 1$: e(nd) detected; $i_i = 1$: the i(nner) part of s_i does not contain superfluous SBs.

5.3 Discussion

Table 3 demonstrates that RNNs clearly outperform both baselines, and that stacking the bidirectional LSTM layers further improves the performance. However, the results don't tell much about the actual usability of the SB labeler, especially about how many full sentences were labeled correctly, and which sentence structures or types are prone for errors. To assess these questions, a metric similar to the "strict" evaluation in Liu and Shriberg (2007) is used. Instead of measuring the annotation precision for single instances of SBs, this metric considers if full sentences have been annotated correctly, and where errors occur in their annotation. For every sentence s_i in the gold annotation it is tested, if the RNN has marked the beginning b_i of s_i (= the end of s_{i-1}) and the end e_i of s_i correctly, and if it has inserted additional SBs in between b_i and e_i (variable i_i). A sentence is accepted as correct in this evaluation, if b_i and e_i are correct ($b_i = 1$, $e_i = 1$), and if there exist no superfluous SBs between them ($i_i = 1$).

Table 4 shows the proportions of the $2^3 = 8$ combinations of these three binary labels for the output of the best RNN from Table 3 (3 hidden layers, dropout). The model achieves an overall "strict" accuracy of 65.08% (configuration 1-1-1, column 2). If the configurations in which only one SB has been missed (0-1-1, 1-0-1) or in which wrong SBs have been inserted between two correctly labeled SBs (1-1-0) are accepted as partial matches, this "lenient" accuracy goes up to 95.54%.

It has been noted in Section 1 that sentences starting or ending in the middle of a text line convey important text historical information. In addition, the CRF failed almost completely to detect these more unusual SBs. In order to examine how these cases are handled by the LSTM, the output of the best LSTM from Table 3 has been stratified according to the start and end positions of the sentences. It may be expected that sentences starting at the beginning of a text line (b) and ending at a *daṇḍa* (d) or double *daṇḍa* (d2) may have lower error rates than those starting and/or ending in the middle of a line (m). The results displayed in Table 5 support this hypothesis. The highest accuracy rates are observed for the

Start-end	Corr.	1 err.	> 1 err.	Acc.
b-m	463	673	149	36.03
b-d	1185	495	72	67.64
b-d2	4034	1044	65	78.44
m-m	27	37	34	27.55
m-d	146	202	68	35.10
m-d2	368	461	39	42.40

Table 5: Sentence based evaluation of the output of the best RNN from Table 3, stratified by start and end positions of sentences. b: Sentence starts at the b(eginning) of a text line, m: in the m(iddle); d/d2: sentence ends at a *daṇḍa*/double *daṇḍa*

configurations b-d and b-d2, while accuracy rates for *-m-* configurations are clearly lower. Although these cases constitute only a minority of the training data, and their accuracy rates may rise when more labeled data are available, the presence of the (double) *daṇḍa* feature (page 4) is certainly most relevant for the observed differences in the accuracy levels.

Columns 3ff. of Table 4 provides another view of the same data, which have been stratified with regard to length classes of sentences. As could be hypothesized from Table 5, the highest accuracy is observed for length class 3, which contains, among others, all sentences that extend over two lines between two double *daṇḍa*s (subset of configuration b-d2). A closer inspection of class 5 (sentences containing at least 30 words) shows, that most of the correct instances of this class have between 30 and 50 words, although the model also marks two very long sentences correctly. MBH 1.19.3-15, a description of the ocean, is a right-branching construction typical for poetic style (*kāvya*). The initial phrase *dadṛśāte tadā tatra samudram* ("Then, both of them saw the ocean there") is expanded by several lines of accusative constructions that depend on the head word *samudram*. Apart from congruent adjectives and appositions, the expansions also contain subordinate participle clauses. This means that the whole sentence can not be reduced to an easily memorizable pattern in the form verb-adverb*-acc*, and demonstrates that stacked LSTM units are in principle able to capture such long-range syntactic dependencies.

A considerable number of errors is produced by short sentences that start or end in the middle of a text line (*-m-* configurations), and for which only one boundary is detected correctly (configurations 0-1-1 and 1-0-1 with length class 1 in Table 4). One of the syntactical patterns that produce most of the errors in this class consists of sequences of words in nom. sg. lacking a copula as observed in MBH 1.147.11:

ātmā	*putraḥ*●	*sakhā*	*bhāryā*●
self:NSGM	son:NSGM	friend:NSGM	wife:NSGF

"[The] son [is the] self. [The] wife [is a] friend."

Another problematic pattern is formed by sequences of the form verb-sg acc-sg* (MBH 6.41.64):

anumānaye	*tvāṃ*●	*yotsyāmi …*
ask:PR1.SG	you:ASG	fight:FUT1.SG

"I ask you [for permission]. I will fight …"

As soon as more training data are available, a Viterbi search over decoded sentence patterns or an additional CRF layer (Huang et al., 2015) may help to reduce the number of such errors.

6 Conclusion

Although the proposed deep bidirectional LSTM model clearly outperforms the metrical and CRF baselines, its accuracy is currently not high enough for performing a reliable unsupervised annotation of SBs. As the evaluation of short sentences has shown, many of the problematic cases cannot be solved on the morpho-syntactic level, but require comprehensive lexical and word semantic information. This finding suggests that a larger amount of training data, including more tokenized texts and more annotated SBs, may improve the performance. In this context, the LSTM model will be used for pre-annotating SBs. Another line of future research will concentrate on the representation of input features. Recent studies such as Labeau et al. (2015), but also Hellwig (2015b) for Sanskrit have demonstrated that the processing of morphologically rich languages may benefit from using sub-word units, skipping lexicalization altogether, or integrating it into a "deeper level" of the network architecture. Given the complexity of Sanskrit phonetics and the richness of its vocabulary, such an approach may prove useful, as soon as more SB annotations are available.

References

Timothy Baldwin and Manuel Paul Anil Kumar Joseph. 2009. Restoring punctuation and casing in English text. In *Advances in Artificial Intelligence*, Springer, pages 547–556.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

John Brockington. 1998. *The Sanskrit Epics*. Brill, Leiden.

George Cardona. 1976. *Pāṇini. A Survey of Research*. Mouton, The Hague - Paris.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the 2015 Annual Conference of the NACL*, ACL, pages 1287–1292.

Madhav M. Deshpande. 1991. Pāṇinian syntax and the changing notion of sentence. In Hans Heinrich Hock, editor, *Studies in Sanskrit Syntax*, Motilal Banarsidass Publishers, Delhi, pages 31–43.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76(5):378–382.

Brendan Gillon and Benjamin Shaer. 2005. Classical Sanskrit, "wild trees", and the properties of free word order languages. In Katalin É. Kiss, editor, *Universal Grammar in the Reconstruction of Ancient Languages*, De Gruyter, Berlin, Boston, pages 457–494.

Oliver Hellwig. 2015a. Morphological disambiguation of Classical Sanskrit. In Cerstin Mahlow and Michael Piotrowski, editors, *Systems and Frameworks for Computational Morphology*. Springer, Cham, pages 41–59.

Oliver Hellwig. 2015b. Using Recurrent Neural Networks for joint compound splitting and Sandhi resolution in Sanskrit. In Zygmunt Vetulani and Joseph Mariani, editors, *Proceedings of the 7th LTC*. pages 289–293.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Hans Henrich Hock. 2013. Some issues in Sanskrit syntax. In Peter M. Scharf and Gérard Huet, editors, *Proceedings of the Seminar on Sanskrit syntax and discourse structures*. Paris.

E. Washburn Hopkins. 1901. *The Great Epic of India. Its Character and Origin*. Charles Scribner's Sons, New York.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Franz Kielhorn. 1888. *Grammatik der Sanskrit-Sprache*. Dümmler Verlag, Berlin.

Paul Kiparsky. 2009. On the architecture of Pāṇini's grammar. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, Springer, Berlin, Heidelberg, pages 33–94.

Amba Kulkarni, Preeti Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How free is 'free' word order in Sanskrit? In Peter M. Scharf, editor, *Sanskrit syntax*. pages 269–304.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained POS tagging. In *Proceedings of the 2015 Conference on EMNLP*. pages 232–237.

Yang Liu and Elizabeth Shriberg. 2007. Comparing evaluation metrics for sentence boundary detection. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*. volume 4, pages 182–185.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using Conditional Random Fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 451–458.

James Martens and Ilya Sutskever. 2011. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1033–1040.

Bimal Krishna Matilal. 1966. Indian theorists on the nature of the sentence (vākya). *Foundations of Language* 2:377–393.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. pages 196–201.

Milman Parry. 1930. Studies in the epic technique of oral verse-making i: Homer and Homeric style. *Harvard Studies in Classical Philology* 41:73–147.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*.

Hartmut Scharfe. 1977. *Grammatical Literature*. A History of Indian Literature, Volume 5, Fasc. 2. Otto Harrassowitz, Wiesbaden.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Frank Seide, Gang Li, Xie Chen, and Dong Yu. 2011. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pages 24–29.

Sven Sellmer. 2015. *Formulaic Diction and Versification in the Mahābhārata*. Adam Mickiewicz University Press, Poznań.

John D. Smith. 1987. Formulaic language in the epics of India. In B. Almqvist, S.Ó. Catháin, and P.Ó. Héalaí, editors, *The heroic process: Form, Function, and Fantasy in Folk Epic*. Glendale Press, pages 591–611.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Ottokar Tilk and Tanel Alumäe. 2015. LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*. Dresden, Germany.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37(1):141–188.

Jakob Wackernagel. 1978 (reprint from 1896). *Altindische Grammatik*. Vandenhoek und Ruprecht, Göttingen.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

Yanqing Zhao, Chaoyue Wang, and Guohong Fu. 2012. A CRF sequence labeling approach to Chinese punctuation prediction. In *26th Pacific Asia Conference on Language, Information and Computation (PACLIC 26)*. pages 508–514.