Burmese (Myanmar) Morphological Analysis: Data and Experiments

CHENCHEN DING,
National Institute of Information and Communications Technology
HNIN THU ZAR AYE, WIN PA PA, KHIN THANDAR NWET, KHIN MAR SOE,
University of Computer Studies, Yangon
MASAO UTIYAMA, EIICHIRO SUMITA,
National Institute of Information and Communications Technology

This paper presents an comprehensive study on Burmese (Myanmar) morphological analysis, from preparation of annotated corpus to investigation based on experiments by statistical approaches. Twenty thousand Burmese sentences in news filed are annoteted with morphological information, as one component of the Asian Language Treebank Project. The annotation includes two-layer tokenization and part-of-speech (POS) annotation, to provide rich information on morphological level and on syntactic constituent level. The annotated corpus has been released under a `CC BY-NC-SA` license, which is the largest open access database of annotated Burmese when this paper was written in 2017. Detailed description of the preparation, the refinement, and the features of the annotated corpus is provided in the first half of the paper. Facilitated by the deliberately prepared corpus, experiment-based investigation on Burmese morphological analysis is presented in the second half of the paper, where standard sequence labeling approach of conditional random fields and a long short-term memory (LSTM) based recurrent neural network (RNN) are applied and discussed. We obtain several general conclusions on the Burmese morphological analysis task, covering the scheme design of output tags, the effect of joint tokinzation and POS-tagging, and the importance of ensemble to stabilize the performance of LSTM-based RNN. This study provides a solid basis for further study on Burmese processing. Burmese should no longer be referred to as a low-resourced or under-studied language, in terms of morphological analysis, attributed to this study.

CCS Concepts:•**Computing methodologies** → **Natural language processing;** *Phonology / morphology;*

Additional Key Words and Phrases: Burmese, Myanmar, annotated corpus, morphological analysis, CRFs, LSTM, RNN, ensemble

ACM Reference Format:
Chenchen Ding, Hnin Thu Zar Aye, Win Pa Pa, Khin Thandar Nwet, Khin Mar Soe, Masao Utiyama, and Eiichiro Sumita 2017. Burmese (Myanmar) Morphological Analysis: Data and Experiments. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 0, 0, Article 00 (0000), 26 pages.
DOI: 0000001.0000001

1. INTRODUCTION

In linguistics, morphology studies the formation of meaningful units and the relation among them in specific languages. As to the engineering practice of natural language processing (NLP), automatic morphological analysis can be regards as a general con-

The second author Hnin Thu Zar Aye worked as an intern student for this study at Natiosnal Institute of Information and Communications Technology, Japan, during June 2016 to May 2017.

Author's addresses: C. Ding, M. Utiyama, and E. Sumita, Advanced Translation Technology Laboratory, ASTREC, National Institute of Information and Communications Technology, 3-5 Hikaridai, Seikacho, Sorakugun, Kyoto 619-0289, Japan; email: {chenchen.ding, mutiyama, eiichiro.sumita}@nict.go.jp. Hnin Thu Zar Aye, Win Pa Pa, and Khin Thandar Nwet, Natural Language Processing Laboratory, University of Computer Studies, Yangon, No.(4) Main Road, Shwe Pyi Thar Township, Yangon, Myanmar; email: {hninthuzaraye,winpapa, khinthandarnwet}@ucsy.edu.mm.

DOI: 0000001.0000001

cept covering shallow processing related to basic meaningful units on textual data. In contrast, tasks as topic models are not belong to morphological analysis because they are not focusing on *basic meaningful units*, though the meaning of larger textual units are processed; tasks as syntactic parsing are not belong to the specific task neither, as they focus on deep, i.e., nested, structures which are not *shallow*, i.e., linear.

The specific processing within morphology analysis is diverse, depend on the different types of languages, or, in a more formal term, on linguistic typology. Consequently, most references on the morphological analysis in NLP are focusing on specific languages, as the features of languages largely affect the task in engineering. As to most inflected Indo-European languages, the process includes related tasks as stemming, lemmatization, and part-of-speech (POS) tagging of words, where a core part is around identification the stems and affixes. The same case is also for many agglutinative languages with clear *word separator* in orthography, e.g., Finnish, Turkish, and Korean [Na 2015], where the identification of various affixes turns to be a main and heavy task. For those languages without word separators in their scripts, a further word segmentation, or tokenization process is required.[1] A typical example on agglutinative languages is Japanese. According to the definition in Neubig et al. [2011]: *Japanese morphological analysis takes an unsegmented string of Japanese text as input, and outputs a string of morphemes annotated with parts of speech*. Briefly, the process is concluded as *cutting and tagging* textual strings. However, conjugated forms of numerous suffixes in Japanese should also be recovered in deeper analysis because of the agglutinativeness and the syllabic writing system [Kudo et al. 2004]. A typical example on isolation languages is Chinese, where the term of morphology analysis is less used and the *cutting* process is usually treated as a separate task called *Chinese word segmentation* [Zhao et al. 2010], because there is no further recovering process besides cutting, in tokenizing an isolate language.

This study focuses on Burmese,[2] whose features can be observed as a mixture of Chinese and Japanese. Morphologically, Burmese is highly analytic with no inflection of morphemes. Similar to Chinese, morphemes can be combined freely with no changes.[3] Syntactically, Burmese is typically head-final where the functional dependent morphemes succeeding content independent morphemes and the verb constituent working as the root of a sentence always comes at the end of a sentence. Subordinative clauses are also placed before their modifying parts and before the main clause of a sentence. All these features are identical to Japanese.[4] The Burmese word segmentation, which can be compared with the Chinese word segmentation, has been preliminarily investigated in our previous work with in-house data [Ding et al. 2016]. In this study, we conduct comprehensive study covering more details in Burmese morphological analysis on our prepared and released dataset.

From the features of Burmese, this study on Burmese morphological analysis can thus be regarded as how to *cut and tag* Burmese textual strings, without any further insertion, substitution, or deletion process in tokenization. The contribution of the study is two-folded, on both linguistics and NLP practice. Linguistically, we constructed a deliberately designed and annotated Burmese corpus, with two-layer tokenization and part-of-speech (POS) annotation, to provide rich information on mor-

[1] The terms of "word segmentation" and "tokenization" are used in an exchangeable way in this paper.

[2] The language is referred to as Burmese or Myanmar in the literature. We use Burmese in this paper because it is the name more likely to be familiar to English readers

[3] Sandhi of consonant mutation from unvoiced to voiced may happen when morphemes are combined, but the phenomenon is not reflected on writing forms. A minor exception on contracted genitive case-marker for some nouns may affect the tone and spelling, which will be mentioned in Section 3.1.3.

[4] Some modifiers of nouns can be placed after the head noun they modify, which is an exception to the head-final restriction in Burmese. This will be mentioned in Section 3.1.3.

phological level as well as on syntactic constituent level. The tokenization and POS-tagging scheme are well designed to cover various and important linguistic phenomena in Burmese. Seven rounds of cross-checking on twenty thousand Burmese sentences were conducted to achieve a consistent and precise annotation. The corpus is thus the best prepared Burmese morphologically annotated corpus in terms of quality as well as quantity when this paper is written in 2017. The corpus has been released under a CC BY-NC-SA license for research community,[5] as one component of the Asian Language Treeback (ALT) Project.[6] The guidelines of the annotation are also available for public,[7] to provide more details for users of the corpus. For NLP practice, we experiment two mainstream engineering approaches on the Burmese morphological analysis task, i.e., the classic and standard sequence labeling approach of conditional random fields (CRFs) [Lafferty et al. 2001], as well as a popular and state-of-the-art approach of long short-term memory (LSTM) based recurrent neural network (RNN) [Greff et al. 2017]. Based on the experimental results, we obtain several general conclusions on the Burmese morphological analysis task, covering the scheme design of output tags, the effect of joint tokinzation and POS-tagging, and the importance of ensemble to stabilize the performance of LSTM-based RNN. This study thus provides a solid basis for further study on Burmese processing, such as syntactic parsing and machine translation. Burmese should no longer be referred to as a low-resourced or under-studied language, in terms of morphological analysis, attributed to this study.

The remainder of the paper is organized as follows. In Section 2, we discuss related work on general approaches of word segmentation and POS-tagging, and previous work on processing Burmese. In Section 3, detailed description of the annotated corpus is provided. Since the final data and guidelines have been released, the motivation and the design of the overall annotation scheme, with several important issues in data annotation and refinement are presented in this section. Experiments of CRFs and LSTM-based RNN are presented in Sections 4 and 5, respectively, where various explorations and comparisons are organized. Section 6 contains discussion based on the experimental results. Section 7 concludes the paper and lists our future work on Burmese and other Southeast Asian languages.

2. RELATED WORK

Shallow processing tasks in NLP, such as word segmentation, POS-tagging, and chunking, can generally be modeled as a classification task to label tokens, or a structured prediction task further taking the relation between output tags into modeling, which fits the sequential feature of textual data better. A classic and standard approach for such tasks are CRFs, i.e., a structured learning method for sequential labeling. Typical works on the application of CRFs on morphological analysis are: Kudo et al. [2004] of Japanese morphology analysis, Zhao et al. [2010] of Chinese word segmentation, Na [2015] of Korean morphology analysis, and our note of Burmese word segmentation [Ding et al. 2016]. Also, non-structured approaches, such as a classifier of support vector machine (SVM), are also widely studied and applied in practice. Typical SVM-based works are: Kudo and Matsumoto [2001][8] and Neubig et al. [2011] on Japanese morphology analysis. Such approaches may apply a dynamic programming method to integrate

[5] http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/my-nova-170405.zip

[6] http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/

[7] Basic guidelines: http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/Myanmar-annotation-guideline.pdf
and supplementary instructions: http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/Myanmar-annotation-guideline-supplemantary.pdf

[8] A more detailed version in Japanese is Kudo and Matsumoto [2002].

the surrounding information on output tags to achieve a similar effect of the structured learning [Kudo and Matsumoto 2001], or simply adopt a pure point-wise way [Neubig et al. 2011] to archive fast processing by a lightweight model. However, Stratos and Collins [2015] have illustrated that well-programmed point-wise approaches can actually comparable results of standard CRFs on POS-tagging several Indo-European languages. A similar conclusion is also reached in the note of Burmese word segmentation, that the performance differs not much between CRFs and point-wise SVM. It can be considered that the capacity of a general supervised machine learning framework is adequate for the morphology analysis task in NLP, that the difference brought by structured learning is not so significant as long as a classier is well trained.

Besides classic feature-based approaches, NN-based approaches are overwhelmingly studied in NLP in recent years. An early comprehensive work in NLP is Collobert et al. [2011], where various classification and sequence labeling tasks in NLP are processed unified by NNs. Generally, an NN-based approach is a pure end-to-end processing, where the relation between input and output are modeled directly by connected nonlinear units. The crucial issues of NN-based approaches are thus in 1) the topology of the network, i.e., how to connect different basic units, and 2) the composition of basic units, i.e., how to conduct nonlinear transformation. Two typical structures of NNs are convolutional neural network (CNN) and recurrent neural network (RNN). As to NLP tasks, RNN is more popular than CNN, as it fits better the sequential features in textual data [Mikolov et al. 2010; Sutskever et al. 2014]. The nonlinear units can be a simple S-shape nonlinear function, e.g., sigmoid function or hyperbolic tangent, or a more complicated and powerful block, such as the LSTM unit, which is actually a standard component in an RNN. LSTM-based RNN thus has been a standard state-of-the-art approach in various NLP tasks. As to the related works on morphology analysis, typical studies are Chen et al. [2015] and Ma and Hovy [2016] on LSTM-based approaches for Chinese word segmentation.

On the line of research on Burmese processing, to the best of our knowledge, there are only two comprehensive works on data-driven morphological analysis: our previous note on word segmentation and Khin War War Htike et al. [2017] on POS-tagging. In our previous work, various word segmentation approaches are compared on a tokenized Burmese dataset over sixty thousand sentences. The study illustrates that data-driven supervised approaches outperform previous rule-based matching and unsupervised approaches. While the differences among different supervised approaches are not significant. The drawbacks of the study, as addressed in the note, are 1) no detailed comparison in feature engineering, 2) the annotated data have relatively inconsistence among annotators, and 3) the in-house data cannot be shared by the research community. The recent work of Khin War War Htike et al. [2017] compares various POS-tagging approaches on a POS-tagged Burmese dataset of ten thousand sentences, which the authors have released.[9] However, a serious limitation of this study is that all the experiments were conducted on golden-standard tokenization of the specific dataset. That is, various models are trained on manually tokenized data and tested on manually tokenized data as well. As the "*words*", i.e., tokens in processing, is not natural units in Burmese texts, this study is not oriented for a practical setting that all the details of tokenization are omitted by supporting there is a perfect tokenizer.

This study covers all ranges of the two previous studies. Essentially, this study is a natural extension of our previous note in terms of open access data and detailed comparison in engineering approaches. We have released well-annotated Burmese corpus around twenty thousand sentences, and experimented with representative approaches of CRFs and LSTM-based RNN with various comparisons. Experiments on the data

[9] `https://github.com/ye-kyaw-thu/myPOS`

released by Khin War War Htike et al. [2017] are also conducted and investigated in a uniformed framework. So, this study provides reliable conclusions, and a solid benchmark of the numeral results on the Burmese morphology analysis, which is reproductive and comparable for further studies.[10]

3. ANNOTATED BURMESE CORPUS

3.1. ALT Burmese Corpus

3.1.1. Overview. The overview of the the ALT project and the international collaboration can be referred to early reports of Ye Kyaw Thu et al. [2016] and Riza et al. [2016], respectively. Briefly, twenty thousand English sentences collected from *Wikinews* are manually translated into different Asian languages as the raw data. Further annotations are then conducted on each language, including tokenization (if needed), POS-tagging, phrasal structured tree-building, and word alignment with original English sentences. The Burmese language is the first Southeast Asian language we work on.

The Burmese data prepared and used in this study cover two annotation tasks: tokenization and POS-tagging. The data were preliminarily processed by different native-speaker annotators through an annotation server [Ye Kyaw Thu et al. 2016]. The preliminary annotation was largely based on the common sense of native speakers, which caused considerable inconsistence from tokenization. We thus design a unified framework for joint tokenization and POS-tagging called nova for annotating highly-analytic languages. An example of an annotated Burmese sentence is illustrated in Fig. 1. A feature of the annotation is that brackets are used to provide a two-layer annotation to adapt the ambiguities in tokenization, as well as to annotate some larger syntactic constituents which can be applied as an integrate unit for further analysis, e.g., syntactic parsing. As the guidelines for annotating Burmese data have been released, we do not mention detailed instructions here, but present an introduction on the overall process in annotation and the features of the annotated Burmese data.

3.1.2. Process of Annotation. The preliminarily annotated Burmese data were firstly mapped into the nova system, and the ambiguous and inconsistent cases in annotation were kept as much as possible by the brackets. Then a CRF-based 8-fold cross-validation on the whole dataset was applied, using tri-gram features on the syllable units [Ding et al. 2016]. We did not pursue a high precision in the cross-validation but tried to filter out inconsistent patterns in annotation. Based on the results of cross-validation, the annotation of the data was modified and improved in a systematic way. That is, we did not only focus on specific inconsistent patterns in the results, but to conclude the linguistic phenomena which were annotated inconsistently and to modify them into a consistent way. The automatic cross-validation and manual refinement were thus applied repeatedly to improve the quality of the annotation. Table I shows the statistic on data of different versions, where the original is the preliminarily annotated data and the latest 17-04-05 data are the finally released version. Figs. 2 and 3 are the visualization of the final and the second columns in Table I, respectively.

Specifically, the **#tag error** in Table I is the count of syllables with wrong tags in the cross-validation, and the **#syllable** is the total number of the syllables. The **tag error rate** (graphed in Fig. 2) is thus the quotient of the **#tag error** divided by **#syllable**.

[10] According to the description on https://github.com/ye-kyaw-thu/myPOS, the data of Khin War War Htike et al. [2017] are still "draft released". The released data are not identical to the data used in their publication and the data are further updated after we conducted the experiments reported in this paper. So it is still not a stable dataset and the numerical results on the data are thus not strictly comparable among different references up to now (2017). We used the dataset in this study in an auxiliary way, to confirm the conclusions we obtained are available on different datasets. The data used in this study can be accessed at https://github.com/chenchen-ding/mycicling

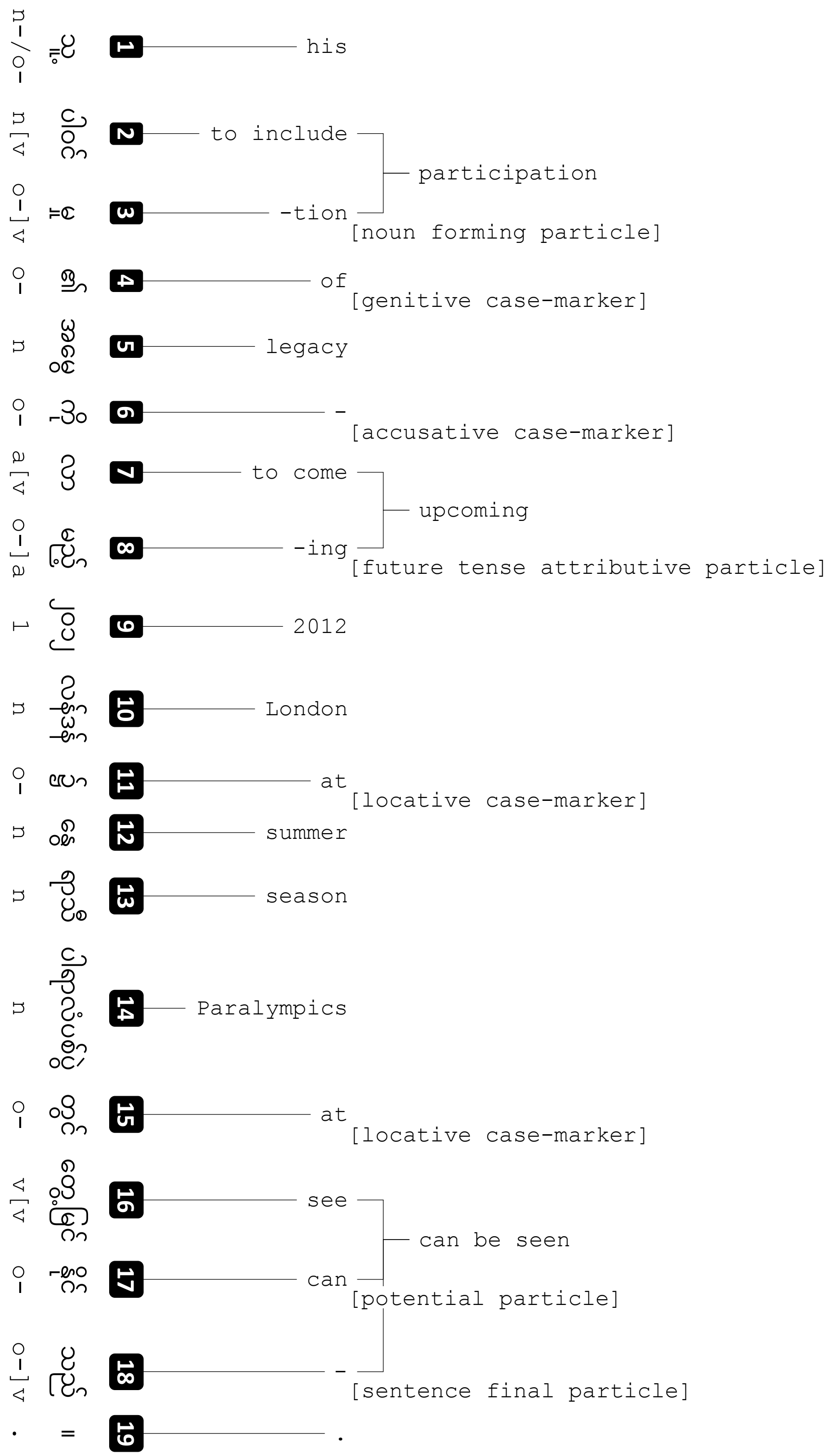

Fig. 1. A tokenized and POS-tagged Burmese sentence. English glosses for Burmese tokens and bracketed constituents are attached on upper side.

Table I. Statistics on 8-fold cross-validation of annotated Burmese data in different versions.

data version	**pattern error**		**#tag error**	**#syllable**	**tag error rate**
	type	**number**			
`original`	54,059	93,269	138,630	1,170,587	11.84%
`16-10-31`	33,048	48,672	71,939	1,171,548	6.14%
`16-11-14`	31,825	42,230	62,599	1,171,577	5.34%
`17-01-01`	28,580	37,150	52,426	1,171,107	4.48%
`17-01-15`	28,144	36,763	51,315	1,171,218	4.38%
`17-01-26`	27,598	36,177	49,980	1,170,980	4.27%
`17-03-09`	27,332	35,996	50,449	1,170,980	4.31%
`17-04-05`	26,418	35,130	48,869	1,170,922	4.17%

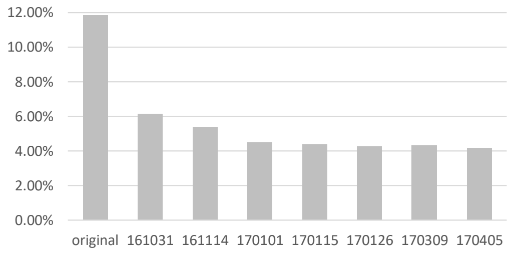

Fig. 2. Tag error rate in Table I

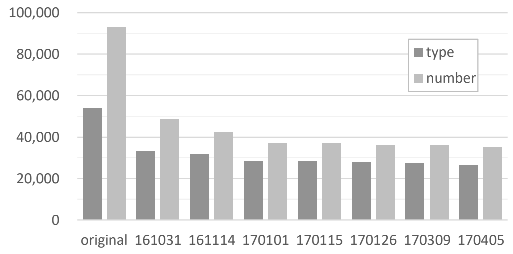

Fig. 3. Type / number of error patterns in Table I

The **type** and **number** of **pattern error** are counted by the maximum-length matching of the wrongly-tagged syllable sequences. From the version of `16-10-31` to `17-01-26` the data are annotated by four basic tags (`n`, `v`, `a`, and `o`) and brackets (`[`, `]`), which can be referred to the basic guidelines. The tags in versions of `17-03-09` and `17-04-05` are further modified by attaching functional markers (`-` and `/o-`), which can be referred to the supplementary instruction of the annotation. It can be observed that both the tag error rate and the pattern errors converged along the refinement. Notice the tag error rate increased a little from `17-01-26` to `17-03-09`, because the type of tags were enlarged by the modification, while the pattern errors decreased stably, indicating the data were always improved in terms of consistence.

In Table I, it can be observed that the number of syllables changed slightly among different versions. This is because the spellings were modified and normalized along the refinement of the annotation. There are two main normalizations as follows.

— The order of the creaky tone-marker `aukmyit` (`U+1037`) and the `virama` (`U+103A`), i.e., inherent vowel-depressor, is arranged in "`aukmyit virama`" in coding nasal-ended creaky-toned syllables, in accordance with the process in Ding et al. [2017].[11] The order of the two diacritics are used quite inconsistently in daily typing, while the order we adapt is considered as a standard order and supported by different fonts.

— The Burmese letter `wa` (`U+101D`) and the Burmese digit zero (`U+1040`) are both an o-shape character and extremely similar to each other (if not identical in some fonts).[12] The two characters are used in an exchangeable way in casual typing. We thus paid specific attention to clean up the misuse of them case by case.

[11] Notice the token 8 in Fig. 1, which is a nasal-ended creaky-toned syllable (only in spelling, and the nasal coda /ɲ/ is denasalized as /j/ in real pronunciation). The tiny circle at the lower right is `aukmyit` and the arc at the upper right is the `virama`.

[12] Notice the full circles in tokens 2 and 9 in Fig. 1. The circle in 2 is the Burmese letter `wa`, and in 9 the Burmese digit zero.

3.1.3. Features and Statistics. Based on the design and the practice of the annotation processing, the tokenized and POS-tagged Burmese data have specific features besides the quality and quantity. We conclude them as follows.

— A two-layer tokenization and POS-tagging annotated by brackets, covers a portion of ambiguities in tokenization and identifies the composition of specific constituents. Typical examples can be observed in the Fig. 1, where tokens ❷❸, tokens ❼❽, tokens ⓰⓱⓲ are composing further constituents playing certain syntactic roles.[13] The two-layer annotation addresses a large range of linguistic phenomena in Burmese, e.g, derivation, compounds, heavily-agglutinative constituents, reversed nominal-attributive constituents, and number-counter constituents.

— The tags contain analytic informations. Four basic tags are used for a sketchy annotation, and further modification of functionality (`-`) and contraction (`/o-`) are added to basis tags for more detailed information. As to the example in Fig. 1, `o-` tags are used to annotated various affixes and particles.[14] The token ❶ is a pronoun with a contracted genitive case-marker[15] where a basic tag `n` for nominal tokens is first modified by a `-` to address the functionality, and then the `/o-` is attached to annotate there is a contracted case-marker which should be tagged as `o-`.[16]

The annotated Burmese corpus thus contains abundant multi-layer information, which is feasible and flexible for various downstream NLP practical applications. The corpus also provided a refined and stable platform for academic research on investigating the techniques on Burmese morphology analysis. For this purpose, the whole corpus is split into three datasets of training, development, and test for experiments and comparison. The statistics on the Burmese corpus are listed in Table II,[17] where **small token** is the number of finally segmented tokens and the **large token** are countered by taking bracketed tokens as one token. As to the example in Fig. 1, there are 19 small tokens and 15 large tokens.[18]

Compared with the in-house data used in Ding et al. [2016], the annotated Burmese corpus in this study has less sentences, but contains more syllables and is segmented into smaller tokens. Comparing the Table II here and the Table I in Ding et al. [2016], we can find there are more than one million ($\mathcal{M}$) syllables in this ALT dataset but only less than $0.8\ \mathcal{M}$ syllables in that in-house dataset. The average sentence length is thus has a large difference that only 12.67 syllables per sentence in that in-house dataset. The difference is mainly cased by the field of the textual data in the corpora. The ALT data are composed of news articles where formal and long sentences are common, while, as stated in Ding et al. [2016], the in-house data are in a restricted filed of travel expressions, which are generally simple in syntax and vocabulary. It can also be calculated that there are 1.63 syllables per token (word) on average in the previous

[13] A verb-derived nominal expression, a verb-derived attributive expression, and a multiply-suffixed verbal expression, respectively.

[14] The basic `o` tag is used to annotate a general modifier (e.g., adverb).

[15] The creaky tone-marker `aukmyit` at the lower right.

[16] As Burmese is highly analytic, this is the only productive contraction phenomenon.

[17] The three datasets are divided by articles from the original English *Wikinews*. The lists of original URL are at `http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/index.html`. Notice the total number of sentences listed in Table II is slightly smaller than that of raw data on the linked page. This is because a little portion of wrongly translated or segmented sentences were excluded in the annotation process.

[18] There are 31 syllables in this example. Most tokens are single-syllabled; ❷, ❺, ❿, ⓭, and ⓰ are composed of two syllables; ⓮ is composed of five syllables; ❾ contains four Burmese digits which are counted as four syllables.

Table II. Statistics on the ALT Burmese corpus. (the version of 17-04-05 in Table I)

dataset	#syllable	#token		#sentence
		small	large	
training	1, 054, 829	664, 174	498, 227	17, 965
development	57, 607	36, 133	27, 081	993
test	58, 486	36, 830	27, 740	1, 007
total	1, 170, 922	737, 137	553, 048	19, 965
average syllable(s)	1	1.59	2.12	58.65

in-house data. So, the Burmese data in this study are tokenized more finely, though the topic and field are more complicated.

The data introduced in Table II will be mentioned as ALT data for short in the following Sections 4 and 5 of experiments.

3.2. CICLING Burmese Corpus

In Sections 4 and 5, we also report experimental results on the data released by Khin War War Htike et al. [2017], which will be mentioned as CICLING data. The statistics on the version we used is presented in Table III. As there is no explicit division of the dataset, we selected the test data by each eleven sentences from the whole dataset, and selected the development data by each ten sentence in a same way from the left data.

The annotation structure in this data is simpler than ALT corpus, where tokenized Burmese textual data are annotated by fifteen different tags. There was one more tag for negative particle in the original publication but then removed, as this tag was exclusively used for only one pre-positional negative particle. Even though, the temporary tag set is still not designed and applied in a refined way. For example, there are tags of abbreviation (abb), foreign words (fw), and text numbers (tn), which are not decided by syntactic roles, but by surfacing spellings. The two tags of abb and fw are mostly used for nominal tokens in the data. However, according to the examples illustrated in the instruction, abb is also used for adverbs and fw also for Arabic numbers.[19] The difference between particle (part) and post-positional marker (ppm) are also not obvious. It seems those ppm-tagged tokens are only restricted to a portion of post-positional case-markers, and other post-positioned functional tokens are all classified to be part. Generally, there is a lack of clarity and consistency on the design and the use of the tags in CICLING corpus, even not so serious a problem in practice.

Some compounds are also annotated in the corpus, by concatenating several tokens, annotated but a vertical bar. Compared to the ALT corpus, the annotation is not in a systematic and complete way, and there is no further annotation for the larger concatenated tokens.[20] In Table III, the basic and concatenated units are represented as **small** and **large token**, respectively, as a comparison of the ALT corpus. The sizes of small tokens are nearly the same on the two corpora, because the tokenization principles are not quite different from each other. On CICLING corpus, the size of large tokens does not differ much from that of small tokens, because only a few concatenation is annotated. On ALT corpus, the average length of large tokens are around 133% of small tokens, while on CICLING corpus, it is only 110%. Compared with ALT corpus, the CICLING corpus has a weak and incomplete two-layer annotation. We thus only conducted experiments over small tokens on CICLING corpus in this study, because large tokens are insignificantly different from small tokens, what is more, there is a lack of POS information on large tokens.

[19] So, a number may be annotated in three ways dependent on the script, which is unnecessarily complex.

[20] Most cases are nominal expressions from our observation.

Table III. Statistics on the CICLING Burmese corpus used in this study.

dataset	#syllable	#token		#sentence
		small	large	
training	$303,588$	$194,024$	–	$9,000$
development	$34,675$	$22,172$	–	$1,000$
test	$33,335$	$21,315$	–	$1,000$
total	$371,598$	$237,511$	$215,931$	$11,000$
average syllable(s)	1	1.56	1.72	33.78

4. BURMESE MORPHOLOGICAL ANALYSIS BY CRFS

4.1. Feature, Tag, and Tool

In this section, we presented the experiments by CRFS. Generally, the CRFs model can be formulated in a framework of a maximum-entropy principle as Eq. (1).

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda}) = \frac{\exp(\sum_j \lambda_j f_j(\boldsymbol{y},\boldsymbol{x}))}{\sum_{\boldsymbol{y}} \exp(\sum_j \lambda_j f_j(\boldsymbol{y},\boldsymbol{x}))} \tag{1}$$

Specifically to a sequential labeling task, in Eq. (1), $\boldsymbol{x}$ represents a sequence of tokens $x_0^i = x_0, x_1, \cdots, x_i$, and $\boldsymbol{y}$ represents a sequence of labels $y_0^i = y_0, y_1, \cdots, y_i$, for corresponding tokens with the same index. f_j is feature functions and λ_j is the corresponding weight for features. Thus $\boldsymbol{\lambda}$, a set of λ_j, is the parameter of the model. For a given parameter $\boldsymbol{\lambda}$ and a token sequence $\boldsymbol{x}$, the most likely labeling sequence is $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\lambda})$. In order to get a sound model parameter $\boldsymbol{\lambda}$, it should be tuned on a set of training data $\{\boldsymbol{x}^0, \boldsymbol{y}^0\}, \{\boldsymbol{x}^1, \boldsymbol{y}^1\}, \cdots, \{\boldsymbol{x}^k, \boldsymbol{y}^k\}$. Generally, the $\boldsymbol{\lambda}$ is optimized by maximizing the following log-likelihood of Eq. (1) on training instances.

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_k \big\{ (\sum_j \lambda_j f_j(\boldsymbol{y}^k, \boldsymbol{x}^k)) - \log \sum_{\boldsymbol{y}^k} \exp(\sum_j \lambda_j f_j(\boldsymbol{y}^k, \boldsymbol{x}^k)) \big\} \tag{2}$$

Here we only give an overall review of the the CRFs framework and we do not step much into the details of optimization and implementation, as there are already matured framework of algorithm and off-the-shelf tools can be referred and applied. In this paper, we focus on the feature selection and the tag set for labeling, which is the most practical issue for our morphological analysis task. The notation of $f(\boldsymbol{y},\boldsymbol{x})$ in Eq. (1) is in a quite generalized form, which can be decomposed further in specific tasks. In the interface of tokenization and POS-tagging in NLP tasks, the features used are usually binary on a sliding window for each postion of token-label pair, with local contextual information by previous and succeeding tokens. So the Eq. (1) can be transformed into Eq. (3), where x_m^n offers a context window for the label y_k at a specific position, and δ_j is the Kronecker delta, that the value of feature is 1 if and only if x_m^n and y_k match exactly, otherwise 0. Notice Eq. (3) is the probability of one training instance while δ_j is cross-instances, i.e., that the co-occurrence of specific x_m^n and y_k can appear in different instances. The weight λ_j can thus be intuitively interpreted as how "important" the co-occurrence of x_m^n and y_k should be regarded over all the given training instances. Even the features are restricted to be local windows, the normalization in Eq. (3) is still over the whole sequence of labels (i.e., the summation over y_0^i in denominator), so a $\hat{y}_0^i = \arg\max_{y_0^i} p(y_0^i|x_0^i,\boldsymbol{\lambda})$ is still searched under a global optimum.

$$p(y_0^i|x_0^i,\boldsymbol{\lambda}) = \frac{\exp(\sum_{k=0}^i \sum_j \lambda_j \delta_j(y_k, x_m^n))}{\sum_{y_0^i} \exp(\sum_{k=0}^i \sum_j \lambda_j \delta_j(y_k, x_m^n))} \quad (0 \le n \le k \le m \le i) \tag{3}$$

Consequently, there are two basic issues in CRFs that we should investigate by experiments, on y_k and x_m^n, respectively.

REL. IDX.:	-5	-4	-3	-2	-1	0	1	2	3	4	5
SYLLABLE:	ပါ	ရာ	လံ	ပစ်	ပွဲ	တွင်	တွေ့	မြင်	နိုင်	သည်	။
SMALL-TOK:	B	I	I	I	E	S	B	E	S	S	S
LARGE-TOK:	B	I	I	I	E	S	B	I	I	E	S
SMALL-POS:	B-n	I-n	I-n	I-n	E-n	S-o-	B-v	E-v	S-o-	S-o-	S-.
LARGE-POS:	B-n	I-n	I-n	I-n	E-n	S-o-	B-v	I-v	I-v	E-v	S-.

Fig. 4. IBES tagging scheme for tokens 14 15 16 17 18 19 in Fig. 1. By changing B to I and S to E, it turns an IE scheme; by changing E to I and S to B, it turns an IB scheme.

Table IV. Feature templates with different size of context window.

	uni-gram	bi-gram	tri-gram	4-gram
2	S^{0}_{0}, S^{-1}_{-1}, S^{1}_{1}	S^{0}_{-1}, S^{1}_{0}		
3	S^{0}_{0}, S^{-1}_{-1}, S^{1}_{1}, S^{-2}_{-2}, S^{2}_{2}	S^{0}_{-1}, S^{1}_{0}	S^{0}_{-2}, S^{1}_{-1}, S^{2}_{0}	
4	S^{0}_{0}, S^{-1}_{-1}, S^{1}_{1}, S^{-2}_{-2}, S^{2}_{2}, S^{-3}_{-3}, S^{3}_{3}	S^{0}_{-1}, S^{1}_{0}	S^{0}_{-2}, S^{1}_{-1}, S^{2}_{0},	S^{0}_{-3}, S^{1}_{-2}, S^{2}_{-1} S^{3}_{0}

— A feasibility tag set for y_k. The issue is trivial in a pure POS-tagging task, as in Khin War War Htike et al. [2017], that the POS tag set is used directly. In the interface of tokenization, however, the boundary of tokens should be identified and thus there are variants in notation differ from addressing the beginning or the end of a token. The size of tag set, i.e., the number of different tag types, should also be considered in practice. As more types of tags lead to a larger space of $\boldsymbol{y}$ in the normalization term of Eqs. 1 and 2, which requires more calculation in model training and decoding.

— A proper window size, i.e., the magnitude of $m - n$ in x^n_m. Generally, if we use a large window size, more contextual information can be modeled, but it will cause sparseness on features, where superfluous patterns of x^n_m and y_k, i.e., numerous δ_j, will be collected from the training instances, which makes the parameter training slow and insufficient.

We adapt the Burmese syllable as the basic unit in the morphological analysis, as in the tokenization task of Ding et al. [2016]. The syllables can be identified decisively by rules, and can be considered as stand-alone units comparable to Chinese characters.[21] Based on the syllables, we design different tagging schemes and feature templates. Fig. 4 illustrates an IBES tagging scheme for tokenization on the final part of the sentence in Fig. 1. The IBES tags can be further attached with POS-tags for joint tokenization and POS-tagging, as the two lower rows in Fig. 4. Generally, the IBES notation is the most popular scheme used in tokenization tasks, where the four tags represent beginning of a token, end of a token, inside a token,[22] and single unit as a token,[23] respectively. As simplified versions, there are IE and IB tagging schemes, where, only the end or beginning of a token is addressed. In the work of Ding et al. [2016], the IE scheme was used as a matter of fact. Basically, two different tags are enough for the tokenization task, while the performance of different tagging scheme may differ due to the nature of specific languages. There can be more complex schemes as further

[21] Or as the characters in Hangul, but more complex in the relation between composition and pronunciation.

[22] That is, the unit is neither the beginning nor the end of a token. Instead of I, M for middle is also used.

[23] That is, the unit is the beginning and the end of a token at the same time.

classification of the I tag [Zhao et al. 2010], while as stated in Kudo and Matsumoto [2002], a too complex tagging scheme may lead to more “illegal” combinations of tags,[24] which does not always help the performance but make the model and process heavier. On our specific data, the average length of token is only between one and two syllables, so we did not further classify the I tag to a more complex scheme. Only IBES, IE, and IB schemes are compared in the experiments. Table IV lists the three feature templates we used in experiments, where $\mathtt{S}^{\mathtt{n}}_{\mathtt{m}}$ stands for the syllable sequence of the relative indices within $[m, n]$. Fig. 4 illustrates the example of relative indices (upper row of REL. IDX.) when tagging the syllable with the gray background.

We used the CRF++ toolkit[25] consistently in the experiments in this section. Another popular off-the-shelf tool is the CRFsuit,[26] which is an implementation much faster than CRF++. In our preliminary experiments, we found that the CRFsuit and CRF++ have comparable performance using identical feature templates. However, CRF++ further supports bi-gram features on output tags, which can lead to slightly increased performance by further “tons of distinct features”. We find that to add non-lexicalized bi-gram features on output tags is a good trade-off in performance and the time/memory consuming in model training. So, all the experimental results reported in this section are with the features on syllables listed in Table IV and the non-lexicalized bi-gram features on output tags.[27]

4.2. Evaluation

We use the F-score (**f-s.**) consistently to evaluate and compare experimental results. Specifically, the tokens segmented out in a Burmese string are compared with the golden tokenization results. And F-score then is calculated as the harmonious average of the precision and the recall in terms of tokens. The accuracy (**acc.**) on tagging basic units, i.e., syllables, is also presented as an auxiliary measure, which is not comparable across different output tag sets. As development data are not required in training CRFs, the development data in Tables II and III are added to training data for all the experiments of CRFs. We also varied the training data size, to investigate how the quantity of training data affects the quality of performance on test data. Specifically, the training data are halved gradually until around one thousand sentences, i.e., up to one sixteenth on ALT data and one eighth on CICLING data.

Table V illustrates the tagging accuracy and F-score on tokenizing small tokens on ALT data, with different feature-tag combination and different training data size. A noticeable phenomenon is that the IE and IB scheme have much higher **acc.** than IBES scheme, while the **f-s.** is obviously lower. It is clear that the IBES scheme, though tuning the task more difficult, codes more useful information. As mentioned, some impossible tagging sequences may appear in IBES scheme. However, such inconsistency only appears rarely (once or twice) with smaller training data (on sixteenth and eighth), which is negligible. As a minor fact, IB is slightly better than IE. A related phenomenon has been mentioned in Kudo and Matsumoto [2002], that IB scheme may reserve more information in segmenting continuous chunks, while our task is actually a chunking task on Burmese syllables.[28] As to the features, bi-grams on syllables (2-) seem adequate and tri-grams (3-) bring a limited gain, but further features (4-) will decrease the per-

[24] In the IBES scheme, the sequences of IB, IS, BB, BS, EI, EE, SI, and SE are impossible. In the IE and IB schemes, all sequences are possible, except the final tag in a sequence cannot be I in IE scheme and the first tag cannot be I in IB scheme.

[25] http://taku910.github.io/crfpp/

[26] http://www.chokkan.org/software/crfsuite/

[27] The non-lexicalized bi-gram feature on output tags is annotated by a “B” according to the format of CRF++’s template. And it is a feature of $\delta(y_{k-1}, y_k)$ in the notation used in this paper.

[28] So, the IE scheme used in Ding et al. [2016] is actually the worst option.

formance, due to sparseness and over-fitting. The Table V is graphed in Fig. 6, where the performance against training data size is illustrated clearly. Generally, there is still a large space to improve the performance, when more training data provided, where the 4-gram features may boost the performance more significantly.

Table VI illustrates the tagging accuracy and F-score on joint tokenizing and POS-tagging small tokens on ALT data, where the tokenization tags are combined with POS tags to form a larger tag set, and the tokeniztion and POS-tagging are generated simultaneously. The F-score in evaluation is thus on the jointed token and POS tag. The phenomena observed in Table VI is identical to those in Table V but with lower numeral results, as the tagging task turns more difficult and evaluation turns stricter. Table VII compares the tokenization performance with and without the POS information, where $+$**pos** is the F-score on pure tokens from the results in Table VI and $-$**pos** is identical to those corresponding results in Table V. It can be observe $+$**pos** is always better than $-$**pos**. So, the POS tag information can boost the performance of tokenization, which is addressed as a future work in our previous note, and we prove it in this study. As the performance of $-$**pos** is already lower than $+$**pos**, a two-path first-tokenizing-then-POS-tagging process cannot achieve a better performance than the joint evaluation in Table VI. We can conclude that the one-path joint tokenization and POS-tagging should be a standard way in Burmese morphological analysis, as the process is simpler and the performance is better. The **f-s.** in Table VI and the F-score of $+$**pos** in Table VII are graphed in Figs. 7 and 8 respectively. As well, the performance is on test data still far from being saturated on the given training data.

Tables V, VI, and VII are a basic group of the experimental results. Tables VIII, IX, and X (graphed in Figs. 9, 10, and 11, respectively) are corresponding results on large tokens on ALT data; Tables XI, XII, and XIII (graphed in Figs. 12, 13, and 14, respectively) are corresponding results on CICLING data. The general phenomena and conclusion in the basic group can also be observed and concluded in the further two groups of experimental results. Specifically, the numerical results on large tokens in ALT data is lower than those on small tokens. Notice the average length is 1.59 on small tokens but 2.12 on large tokens. So bi-gram features are adequate to cover the range of small tokens but still insufficient for large ones. On the other hand, high order features bring the problem of sparseness, that 4-grams features do not increase the performance. Consequently, the processing on large tokens turns more difficult than that on small tokens. On the CICLING data, an obvious difference from ALT data is that the gain on tokenization precision brought by the joint tokenization and POS-tagging is not so significant (Table XIII). This may be caused by some defects in the POS system they applied, which have been discussed in Section 3.2. Consequently, information of POS tags does not provide supplementary information consistently for tokenization and thus lead to limited improvement.

There are two-layer information in the notation of ALT data, while they are processed separately in the illustrated experimental results. Tables XIV and XV (graphed in Figs. 15 and 16, respectively) provide further results in two-path processing between the two-layer annotations. In the two tables, `IBES` scheme is applied as it have been proved informative and efficient in previous results. Table XIV shows the small-to-large token processing (`S->L`), where `SYL.S->L` means the processing is still based on syllables, i.e., the syllables with labeled information of small tokens are labeled again to generate large tokens; `TOK.S->L` means the labeling is directly based on generated small tokens where the small tokens are labeled to compose large tokens. The **acc.** in tables is the labeling precision on the test data trained by the golden training data in the second path, so the numerical results is quite high. The **f-s.** in tables is the final F-score of the two-path processing, where the automatically generated results with noises are processed again by the models trained on golden training data.

The final three ranks of Table XIV is identical to the results of IBES in Table IX, provided for comparison. Although not in a large margin, the two-path processing from small tokens to large tokens is better than the one-path processing from unlabeled syllables to large tokens. And token-based two-path processing is better than syllable-based one. As to a reasonable explanation, the information of small tokens, even not accurate, can help the performance on large tokens; and small token-based features can cover a longer context, which lead to a little better results than syllable-bases features. Another benefit brought by the small-token based processing for large tokens is the boundary will be always consistent in the two-layer processing, i.e., the boundary of large tokens will always be the boundary of small tokens. The composition of Table XV of large-to-small token processing (L->S) is simpler than that of Table XIV. As the large token-based processing for small tokens is impossible, there is only syllables based processing (TOK.L->S) and the final three lines are identical to the results of IBES in Table VI, provided for comparison. The large-to-small token processing cannot provide a better performance. We hence consider, from syllables to small tokens and then from small tokens to large tokens is a natural way and thus efficient way, which build larger units by small units gradually.[29]

5. BURMESE MORPHOLOGICAL ANALYSIS BY LSTM-BASED RNN

5.1. Network Structure, Ensemble, and Implementation

In the previous section, the experimental results of CRF-based Burmese morphological processing have been illustrated and investigated. In this section, the approach is switched to LSTM-based RNN. We first describe the overall network structure and the specific approaches applied before the experiment-based evaluation. The overall network structure is illustrated in Fig. 5, where the three modules of representation, classification and structured interface is a standard configuration for structured learning in many NLP tasks [Collobert et al. 2011; Chen et al. 2015]. More modifications can be further added into the overall configuration, as using CNN in the representation module, or integrating a CRF-based module into the structured interface [Ma and Hovy 2016]. We applied the most general structure used in Fig. 5, with details based on several observations from our preliminary experiments.

— The structured interface can accelerate the converging in training, but cannot improve the performance substantially on our data. We observed that, even without the structured interface, the performance on the development data can achieve comparable results of that using the structured interface, only requiring more iterations in training. We attribute this to the ability of the representation module, that contextual information can be represented well for the classification, as a compensation for the absence of structured learning. In the structured interface, we thus only applied a standard Viterbi algorithm to model the relationship of neighboring tags, which is adequate for the task.

— The LSTM layers in representation module affects the performance obviously, that only one layer is not adequate. We used two-layer bidirectional LSTM, which is a trade-off of performance and training speed. A third layer cannot bring as much improvement as the second layer does, but increases the training time. We also tried other light-weight non-linear units as gated recurrent units (GRUs) [Cho et al. 2014] but the performance differed little. In practical, we applied a compact variant of LSTM with peepholes [Gers et al. 2002] in the representation module.

[29] There is an ultimate joint way to combine the two-layer information, including tokenization and POS tags, into single tags. However, this will lead to a huge tag set with more than 180 different tags on ALT data, which results in an unnecessarily heavy model and learning process.

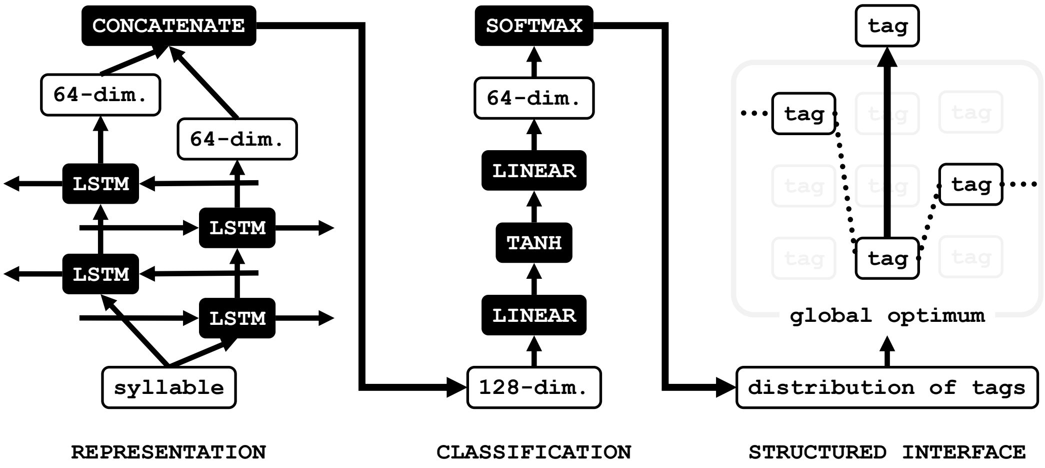

Fig. 5. Structure of the LSTM-based networks used in Burmese morphological analysis.

— The type non-linear function in classification module does not affect the performance much. So we applied the most common tanh function.
— The dimension of each layer affects the performance relatively. The dimensions illustrated in Fig. 5 are selected by the trade-off of training time and performance.
— As illustrated in Fig. 5, the embedding is based on single syllables, i.e., uni-gram of syllables. Higher orders like bi- or tri-gram of syllables turn the training heavier and slower without obvious gains in performance. We also tried pre-trained embedding, which did not bring gains in performance but could accelerate the training to converge more or less.

We have introduced the overall structure of the network we used in experiment. However, there is a serious problem in practice that the training is quite unstable. Even we used development data to control the over-fitting and by selecting models having good performance on the development data, the final performance on test data still diverged a lot with different initializations. This may be caused by the training data size, which is still not sufficient for NN-based processing.[30] We tried several parameter optimizing approaches which did not differed much in performance, and applied Adam [Kingma and Ba 2014] consistently in experiments. We tried further techniques such as dropout [Srivastava et al. 2014] in training to stabilize the performance, while the effect is not significant.

After many attempts, we discover a practical ensemble method to combine a large amount of independently fast-trained networks, to alleviate the instability in performance caused by initialization. The method is efficient and robust as well, to improve the performance by single networks. Specifically, we separately trained different networks with different initializations only with few iterations, and then combined the results by a voting ensemble. The basic idea is to release the difficulties in tuning one refined model by multiple roughly tuned models. It is also a practical solution in terms of computing resources. The fast training of multiple networks can be conducted in parallel if there are plenty computing resources or can be trained one by one successively to increase the performance of ensemble gradually under limited computing resources. The followings are the details in the ensemble.

[30] Considering the upper bound of the capacity of CRFs has not been reached yet.

— For training single networks, we applied one initial iteration without structured interface and then applied several further iterations with the Viterbi searching. Specifically, further three iterations for tokenization and further five iterations for joint tokenization and POS tagging. We selected the iteration times by observing the performance on development data, that the improvement on performance is most obvious in these rounds of iterations. As we do not pursue refined models in ensemble, this kind of fast training is adequate for ensemble.

— The ensemble is by simple voting on each output tags, to select the most common one from results of multiple models. As the ensemble is point-wisely, it may generate illegal tag sequences.[31] The problem is not serious, that under 100-model ensemble, less than 0.1% illegal sequences occurred in tokenization and less than 0.2% in joint tokenization and POS tagging. We applied a straightforward solution to collect all possible bi-grams on output tags from training data and only select the possible sequences in ensemble.[32]

— We tried some more complex ensemble schemes, such as adding weights to different models, but none achieved better results. As all the models taking part in the ensemble are "*equal*" in mechanism, we consider that to treat them in a simple and equal ways is the most nature solution.

All the above-mentioned model structures and ensemble processes were tested and decided by the performance on the development data in Tables II and III, and only the corresponding training data were used in model training for evaluation. We used the `DyNet` toolkit (version 2.0) [Neubig et al. 2017] in the implementation. Besides the above-mentioned settings, we adopted the default settings in `DyNet` for other parameters. On one GPU of Tesla K80, the training of one tokenization model took around twenty minutes and that of one joint tokenization and POS tagging model took around one hour, with the described times of iterations on the full ALT training data. We experimented up to ensemble of one hundred models. It hence would take approximately one day and a half for tokenization model training, and four days for joint tokenization and POS tagging model training, if one hundred models are trained serially.

5.2. Evaluation

The metrics used here are identical to those in the evaluation in CRFs. The training data are also halved gradually for comparison. As a small portion of development data are used in model selection rather than model training, the training data get slightly smaller than those used in CRFs' experiments. The tagging scheme of `IBES` is used consistently in all the experiments of LSTM-based RNN, as it is the most efficient scheme illustrated in previous experiments. The number of models in ensemble are also compared to investigate the effect of ensemble.

Tables XVI, XVII, and XVIII are LSTM-based RNN-version of Tables. V, VI, and VII, respectively. The results of ensemble (`ENS.`) over 5, 10, 20, 50, and 100 models are illustrated. The best and the worst single model among the total one hundred models are also listed. The results of `2-` and `3-IBES` in Tables V, VI, and VII are listed in the corresponding tables for comparison. The comparison of `ENS.-100` of LSTM-based RNN with CRFs' results in Tables XVI, XVII, and XVIII, are graphed in Figs. 17, 18,

[31] As there is the structured interface in single model training, illegal tag sequences hardly occurred except on very few training data.

[32] So, the possible sequences may change along the training data size. In tokenization, there is actually no effect as there are only four tags of `IBES`. In joint tokenization and POS tagging, smaller training data may contain less variants in combination. However, as mentioned, this is not a serious problem and effects the numerical results negligibly.

and 19, respectively.[33] The comparison of ensemble effects in Tables XVI, XVII, and XVIII, are graphed in Figs. 20, 21, and 22, respectively. It can be observed that the ensemble can bring a considerable gain in the performance, even compared with the best single model. Generally, LSTM-based RNN boosted by ensemble can achieve similar performance as that of CRFs on the full training data. In the joint tokenization and POS-tagging, the LSTM-based RNN have better performance on small training data, compared with the case of pure tokenization. It is obviously that a more informative output tag set has more significant effects, especially on small training data, as the sparseness in discrete features used in CRFs is alleviated by the embedding to a dense representation in low-dimension real space facilitated by RNN. As to the effect of ensemble, it can improve the performance even by only a few models (e.g., five), and a large amount of models bring further improvement steadily, even not significantly.

As in the evaluation of CRFs, Tables XVI, XVII, and XVIII are a basic group of the experimental results on LSTM-based RNN. Tables XIX, XX, and XXI are LSTM-based RNN-version of Tables VIII, IX, and X, which present the experimental results on large tokens in ALT data. Tables XXII, XXIII, and XXIV are LSTM-based RNN-version of Tables XI, XII, and XIII, which present the experimental results on CICLING data. All the tables are graphed in two figures: one is the comparison with the performance of CRFs (Figs. 23, 24, 25, 29, 30, and 31 for Tables XIX, XX, XXI, XXII, XXIII, and XXIV, respectively) and another shows the effect of ensemble (Figs. 26, 27, 28, 32, 33, and 34 for Tables XIX, XX, XXI, XXII, XXIII, and XXIV, respectively). From the further two groups of experiments, we can observe the same phenomena that LSTM-based RNN performs better in joint tokenization and POS tagging than in pure tokenization, even on relatively small training data. The effect of RNN is more obvious than CRFs on processing large tokens on ALT data. As discussed, low order n-gram features cannot catch enough local information while high order n-gram features cause the sparseness. This dilemma can thus be alleviated by the strength of RNN which provides more efficient feature representation. The LSTM-based RNN also outperform CRFs on joint tokenization and POS tagging CICLING data, which indicates RNN can take advantage of relatively complex and inconsistent features more efficiently than CRFs do.

The final group of experiments is the two-path process using LSTM-based RNN to generate large tokens from the results of small tokens on ALT data. Because we have found that to generate small tokens from results of large tokens is neither natural nor efficient by the experiments of CRFs, we did not experiment this way any more. The results are illustrated in Table XXV (graphed in Figs. 35 and 37) for syllable-based processing and in Table XXVI (graphed in Figs. 36 and 38) for token-based processing. In the token-based processing, we tried larger dimensions of different layers in RNN than the syllable-based processing, while no better result was reached. The results in Table XXVI is still based on the exact network structure illustrated in Fig. 5. In training the models for ensemble, one iteration without Viterbi searching plus three iterations with Viterbi searching was applied for syllable-based processing and five iterations with Viterbi searching for token-based processing. The performance is acceptable and slightly lower than that of CRFs. The ensemble effect is not obvious as well in the experiments of two-path processing. As the methodology of NN-based approach is rooted in an ultimate joint model for end-to-end processing, we consider the efficiency of RNN cannot be developed well in such a two-path processing.

[33] As the 4-IBES never reaches better results, they are omitted in tables, but only illustrated in figures.

Table V. Performance of CRFs on tokenizing small tokens on ALT data. (Fig.6)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
2-IBES	91.1%	**.901**	92.6%	**.918**	93.6%	**.928**	94.3%	.936	94.9%	.943
3-IBES	91.1%	**.901**	92.5%	.916	93.4%	.927	94.4%	**.938**	95.1%	**.944**
4-IBES	90.5%	.895	92.1%	.912	93.1%	.924	94.2%	.935	94.9%	.943
2-IE	94.1%	.880	94.9%	.894	95.6%	.907	96.0%	.916	96.3%	.922
3-IE	94.4%	.886	95.4%	.905	96.1%	.918	96.6%	.928	96.9%	.935
4-IE	94.2%	.882	95.1%	.900	95.9%	.915	96.5%	.927	96.9%	.935
2-IB	94.4%	.886	95.2%	.900	95.9%	.913	96.2%	.920	96.7%	.929
3-IB	94.6%	.890	95.5%	.907	96.2%	.921	96.7%	.931	97.0%	.937
4-IB	94.2%	.882	95.3%	.902	96.0%	.917	96.7%	.930	97.0%	.937

Table VI. Performance of CRFs on joint tokenization and POS-tagging small tokens on ALT data. (Fig.7)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
2-IBES	89.5%	**.896**	91.2%	**.912**	92.4%	**.924**	93.3%	**.934**	93.9%	**.940**
3-IBES	88.8%	.889	90.9%	.909	92.2%	.922	93.2%	.932	93.9%	**.940**
4-IBES	87.8%	.879	90.2%	.902	91.8%	.918	93.0%	.930	93.7%	.938
2-IE	91.5%	.885	92.8%	.902	93.9%	.916	94.4%	.924	94.9%	.931
3-IE	91.3%	.882	92.9%	.904	94.0%	.919	94.7%	.929	95.3%	.937
4-IE	90.5%	.871	92.3%	.896	93.6%	.914	94.6%	.927	95.2%	.936
2-IB	91.3%	.885	92.7%	.903	93.9%	.918	94.4%	.925	94.8%	.932
3-IB	91.1%	.883	92.8%	.905	93.9%	.920	94.6%	.929	95.0%	.935
4-IB	90.4%	.874	92.2%	.898	93.5%	.914	94.5%	.929	95.0%	.936

Table VII. Effects of POS tags in the tokenization performance of CRFs on small tokens on ALT data. (Fig.8)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**
2-IBES	**.908**	.901	**.922**	.918	**.932**	.928	**.942**	.936	**.947**	.943
3-IBES	.901	.901	.919	.916	.931	.927	.940	.938	.946	.944
4-IBES	.893	.895	.913	.912	.927	.924	.938	.935	.945	.943
2-IE	.896	.880	.911	.894	.924	.907	.931	.916	.938	.922
3-IE	.894	.886	.914	.905	.927	.918	.936	.928	.943	.935
4-IE	.885	.882	.907	.900	.922	.915	.934	.927	.943	.935
2-IB	.896	.886	.912	.900	.926	.913	.933	.920	.939	.929
3-IB	.896	.890	.915	.907	.928	.921	.937	.931	.942	.937
4-IB	.887	.882	.909	.902	.923	.917	.937	.930	.942	.937

Table VIII. Performance of CRFs on tokenizing large tokens on ALT data. (Fig.9)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
2-IBES	91.9%	**.882**	93.3%	.900	94.2%	.913	95.0%	.924	95.5%	.932
3-IBES	91.9%	.880	93.4%	**.901**	94.3%	**.914**	95.0%	**.925**	95.6%	**.933**
4-IBES	91.2%	.870	92.8%	.892	93.9%	.909	94.9%	.923	95.4%	.930
2-IE	94.2%	.843	95.1%	.865	95.7%	.879	96.1%	.890	96.4%	.897
3-IE	94.9%	.860	95.8%	.883	96.4%	.900	96.9%	.913	97.3%	.922
4-IE	94.6%	.853	95.6%	.879	96.3%	.897	96.9%	.912	97.2%	.921
2-IB	95.0%	.864	95.8%	.882	96.4%	.898	96.7%	.906	97.0%	.914
3-IB	95.1%	.868	96.0%	.888	96.6%	.905	97.1%	.918	97.4%	.925
4-IB	94.7%	.857	95.7%	.882	96.5%	.902	97.1%	.916	97.4%	.925

Table IX. Performance of CRFs on joint tokenization and POS-tagging large tokens on ALT data. (Fig.10)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
2-IBES	89.7%	**.872**	91.4%	**.893**	92.8%	**.911**	93.7%	**.922**	94.3%	**.929**
3-IBES	89.2%	.866	91.1%	.889	92.6%	.907	93.5%	.919	94.3%	**.929**
4-IBES	88.2%	.854	90.5%	.882	92.2%	.902	93.2%	.916	94.2%	.927
2-IE	91.6%	.852	92.8%	.874	93.8%	.891	94.7%	.906	95.1%	.913
3-IE	91.7%	.856	93.2%	.881	94.3%	.901	95.1%	.914	95.5%	.924
4-IE	91.0%	.846	92.8%	.875	94.0%	.896	94.9%	.912	95.6%	.923
2-IB	91.4%	.860	92.6%	.880	94.0%	.901	94.6%	.913	95.1%	.919
3-IB	91.2%	.859	92.8%	.884	94.0%	.903	94.7%	.915	95.3%	.923
4-IB	90.5%	.848	92.3%	.876	93.7%	.899	94.5%	.912	95.4%	.924

Table X. Effects of POS tags in the tokenization performance of CRFs on large tokens on ALT data. (Fig.11)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**
2-IBES	**.883**	.882	**.903**	.900	**.919**	.913	**.930**	.924	**.937**	.932
3-IBES	.879	.880	.900	.901	.916	.914	.928	.925	.936	.933
4-IBES	.868	.870	.893	.892	.911	.909	.924	.923	.934	.930
2-IE	.862	.843	.883	.865	.900	.879	.913	.890	.921	.897
3-IE	.869	.860	.891	.883	.910	.900	.923	.913	.931	.922
4-IE	.859	.853	.887	.879	.905	.897	.920	.912	.930	.921
2-IB	.871	.864	.890	.882	.909	.898	.921	.906	.927	.914
3-IB	.872	.868	.895	.888	.912	.905	.923	.918	.930	.925
4-IB	.863	.857	.888	.882	.908	.902	.921	.916	.931	.925

Table XI. Performance of CRFs on tokenizing CICLING data. (Fig.12)

	training data size							
	eighth		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
2-IBES	91.0%	**.902**	92.8%	**.922**	94.5%	**.939**	95.5%	.950
3-IBES	90.7%	.898	92.6%	.919	94.4%	.938	95.6%	**.951**
4-IBES	90.0%	.892	92.1%	.914	93.8%	.932	95.2%	.947
2-IE	93.9%	.878	95.4%	.905	96.1%	.920	96.8%	.933
3-IE	94.3%	.887	95.5%	.909	96.5%	.927	97.3%	.943
4-IE	93.8%	.876	95.2%	.903	96.2%	.923	97.1%	.940
2-IB	94.2%	.882	95.4%	.905	96.4%	.926	97.0%	.937
3-IB	94.1%	.882	95.5%	.908	96.6%	.928	97.3%	.943
4-IB	93.7%	.875	95.1%	.900	96.3%	.923	97.2%	.940

Table XII. Performance of CRFs on joint tokenization and POS-tagging CICLING data. (Fig.13)

	training data size							
	eighth		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
2-IBES	87.5%	**.876**	90.2%	**.902**	92.0%	**.920**	93.4%	**.934**
3-IBES	86.0%	.861	89.2%	.892	91.5%	.915	93.4%	.933
4-IBES	84.2%	.844	87.7%	.878	90.6%	.906	92.9%	.928
2-IE	89.0%	.860	91.3%	.890	92.9%	.909	94.2%	.924
3-IE	88.3%	.852	91.0%	.886	92.8%	.908	94.4%	.927
4-IE	86.9%	.837	90.0%	.874	92.1%	.901	93.8%	.921
2-IB	89.1%	.862	91.3%	.889	93.0%	.909	94.4%	.926
3-IB	88.3%	.853	91.0%	.886	92.9%	.910	94.4%	.929
4-IB	86.8%	.837	89.8%	.872	92.0%	.901	94.0%	.923

Table XIII. Effects of POS tags in the tokenization performance of CRFs on CICLING data. (Fig.14)

	training data size							
	eighth		quarter		half		all	
feat.-tag	+pos	−pos	+pos	−pos	+pos	−pos	+pos	−pos
2-IBES	**.906**	.902	**.927**	.922	**.942**	.939	.951	.950
3-IBES	.893	.898	.920	.919	.937	.938	**.952**	.951
4-IBES	.879	.892	.907	.914	.930	.932	.947	.947
2-IE	.888	.878	.913	.905	.929	.920	.942	.933
3-IE	.882	.887	.912	.909	.929	.927	.946	.943
4-IE	.870	.876	.903	.903	.925	.923	.940	.940
2-IB	.890	.882	.914	.905	.931	.926	.944	.937
3-IB	.884	.882	.912	.908	.932	.928	.947	.943
4-IB	.871	.875	.900	.900	.924	.923	.943	.940

Table XIV. Performance of CRFs on joint tokenizing and POS-tagging large tokens based on the tokenizing and POS-tagging results of small tokens on ALT data. (Fig.15)

	training data size									
	sixteenth		eighth		quarter		half		all	
feat.-tag	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.
2-SYL.S->L	98.4%	**.877**	98.6%	**.895**	98.9%	**.912**	99.0%	**.924**	99.1%	.930
3-SYL.S->L	98.2%	.867	98.5%	.892	98.8%	.909	99.0%	.920	99.1%	.930
4-SYL.S->L	98.0%	.854	98.4%	.883	98.7%	.903	98.9%	.917	99.1%	.928
2-TOK.S->L	97.6%	.874	98.2%	**.895**	98.6%	.911	98.7%	.923	98.9%	**.931**
3-TOK.S->L	97.2%	.865	97.9%	.890	98.6%	.908	98.7%	.919	98.9%	.930
4-TOK.S->L	96.7%	.850	97.5%	.879	98.2%	.901	98.6%	.916	98.8%	.927
2-SYL->L	89.7%	.872	91.4%	.893	92.8%	.911	93.7%	.922	94.3%	.929
3-SYL->L	89.2%	.866	91.1%	.889	92.6%	.907	93.5%	.919	94.3%	.929
4-SYL->L	88.2%	.854	90.5%	.882	92.2%	.902	93.2%	.916	94.2%	.927

Table XV. Performance of CRFs on joint tokenizing and POS-tagging small tokens based on the tokenizing and POS-tagging results of large tokens on ALT data. (Fig.16)

	training data size									
	sixteenth		eighth		quarter		half		all	
feat.-tag	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.
2-SYL.L->S	97.5%	.889	97.8%	.908	98.1%	.922	98.3%	.932	98.5%	.939
3-SYL.L->S	97.1%	.881	97.7%	.902	98.0%	.918	98.3%	.930	98.5%	.939
4-SYL.L->S	96.5%	.867	97.4%	.894	97.9%	.912	98.2%	.926	98.5%	.936
2-SYL->S	89.5%	**.896**	91.2%	**.912**	92.4%	**.924**	93.3%	**.934**	93.9%	**.940**
3-SYL->S	88.8%	.889	90.9%	.909	92.2%	.922	93.2%	.932	93.9%	**.940**
4-SYL->S	87.8%	.879	90.2%	.902	91.8%	.918	93.0%	.930	93.7%	.938

Table XVI. Performance of LSTM networks on tokenizing small tokens on ALT data. (Figs.17 and 20)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
ENS.-5	88.9%	.877	91.4%	.905	93.1%	.923	94.2%	.935	94.8%	.942
ENS.-10	89.0%	.879	91.4%	.905	93.2%	.924	94.3%	.936	94.9%	.943
ENS.-20	89.2%	.881	91.6%	.907	93.4%	.926	94.4%	.937	95.0%	.943
ENS.-50	89.2%	.881	91.6%	.907	93.4%	.926	94.4%	.937	95.0%	**.944**
ENS.-100	89.2%	.881	91.6%	.907	93.4%	.926	94.4%	.937	95.1%	**.944**
BEST @100	88.5%	.873	90.8%	.898	92.6%	.917	93.7%	.929	94.4%	.938
WORST@100	86.4%	.852	90.0%	.889	91.9%	.910	93.1%	.924	93.9%	.932
2-IBES	91.1%	**.901**	92.6%	**.918**	93.6%	**.928**	94.3%	.936	94.9%	.943
3-IBES	91.1%	**.901**	92.5%	.916	93.4%	.927	94.4%	**.938**	95.1%	**.944**

Table XVII. Performance of LSTM networks on joint tokenizing and POS-tagging small tokens on ALT data. (Figs.18 and 21)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
ENS.-5	88.5%	.887	90.5%	.905	92.0%	.921	93.0%	.930	93.7%	.938
ENS.-10	88.9%	.890	90.7%	.908	92.1%	.922	93.3%	.933	93.7%	.938
ENS.-20	89.2%	.893	91.0%	.910	92.3%	.924	93.3%	.934	93.8%	.939
ENS.-50	89.2%	.893	91.0%	.911	92.4%	.924	93.4%	**.935**	93.9%	.939
ENS.-100	89.2%	.894	91.0%	.910	92.4%	**.925**	93.3%	.934	93.9%	**.940**
BEST @100	87.5%	.876	89.4%	.894	91.1%	.912	92.2%	.923	93.1%	.931
WORST@100	86.4%	.867	88.6%	.887	90.3%	.904	91.7%	.917	92.6%	.928
2-IBES	89.5%	**.896**	91.2%	**.912**	92.4%	.924	93.3%	.934	93.9%	**.940**
3-IBES	88.8%	.889	90.9%	.909	92.2%	.922	93.2%	.932	93.9%	**.940**

Table XVIII. Effects of POS tags in the tokenization performance of LSTM networks on small tokens on ALT data. (Figs.19 and 22)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**
ENS.-5	.899	.877	.916	.905	.930	.923	.939	.935	.945	.942
ENS.-10	.902	.879	.918	.905	.931	.924	.941	.936	.945	.943
ENS.-20	.905	.881	.920	.907	.933	.926	**.942**	.937	.946	.943
ENS.-50	.904	.881	.920	.907	.933	.926	**.942**	.937	.946	.944
ENS.-100	.905	.881	.920	.907	**.934**	.926	**.942**	.937	**.947**	.944
BEST @100	.890	.873	.907	.898	.923	.917	.932	.929	.939	.938
WORST@100	.883	.852	.901	.889	.916	.910	.927	.924	.937	.932
2-IBES	**.908**	.901	**.922**	.918	.932	.928	**.942**	.936	**.947**	.943
3-IBES	.901	.901	.919	.916	.931	.927	.940	.938	.946	.944

Table XIX. Performance of LSTM networks on tokenizing large tokens on ALT data. (Figs.23 and 26)

	training data size									
	sixteenth		**eighth**		**quarter**		**half**		**all**	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
ENS.-5	90.6%	.862	92.2%	.884	93.7%	.905	94.9%	.924	95.6%	.933
ENS.-10	90.8%	.865	92.3%	.886	93.8%	.908	95.1%	.925	95.6%	.933
ENS.-20	90.9%	.866	92.5%	.888	93.9%	.909	95.1%	**.926**	95.7%	.934
ENS.-50	90.9%	.867	92.6%	.889	94.0%	.910	95.1%	**.926**	95.7%	**.935**
ENS.-100	90.9%	.867	92.6%	.889	94.0%	.910	95.1%	**.926**	95.7%	**.935**
BEST @100	90.0%	.854	91.8%	.879	93.2%	.899	94.4%	.915	95.1%	.926
WORST@100	88.8%	.837	90.9%	.865	92.7%	.892	93.7%	.907	94.6%	.919
2-IBES	91.9%	**.882**	93.3%	.900	94.2%	.913	95.0%	.924	95.5%	.932
3-IBES	91.9%	.880	93.4%	**.901**	94.3%	**.914**	95.0%	.925	95.6%	.933

Table XX. Performance of LSTM networks on joint tokenizing and POS-tagging large tokens on ALT data. (Figs.24 and 27)

	training data size									
	sixteenth		eighth		quarter		half		all	
feat.-tag	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.
ENS.-5	89.2%	.866	91.1%	.890	92.6%	.907	93.6%	.920	94.1%	.926
ENS.-10	89.4%	.870	91.4%	.893	92.7%	.909	93.8%	.922	94.2%	.928
ENS.-20	89.6%	.872	91.4%	.894	92.8%	.910	93.9%	.923	94.3%	.928
ENS.-50	89.8%	.874	91.6%	**.896**	92.9%	.911	93.9%	**.924**	94.4%	**.930**
ENS.-100	89.8%	**.875**	91.6%	**.896**	93.0%	**.912**	93.9%	**.924**	94.4%	**.930**
BEST @100	88.0%	.852	90.0%	.875	91.8%	.898	92.9%	.911	93.7%	.920
WORST@100	86.6%	.836	89.3%	.867	90.9%	.887	92.2%	.902	93.0%	.913
2-IBES	89.7%	.872	91.4%	.893	92.8%	.911	93.7%	.922	94.3%	.929
3-IBES	89.2%	.866	91.1%	.889	92.6%	.907	93.5%	.919	94.3%	.929

Table XXI. Effects of POS tags in the tokenization performance of LSTM networks on small tokens on ALT data. (Figs.25 and 28)

	training data size									
	sixteenth		eighth		quarter		half		all	
feat.-tag	+pos	−pos	+pos	−pos	+pos	−pos	+pos	−pos	+pos	−pos
ENS.-5	.878	.862	.900	.884	.916	.905	.929	.924	.934	.933
ENS.-10	.882	.865	.903	.886	.918	.908	.930	.925	.935	.933
ENS.-20	.883	.866	.904	.888	.920	.909	.931	.926	.935	.934
ENS.-50	.885	.867	**.905**	.889	.920	.910	**.932**	.926	**.937**	.935
ENS.-100	**.886**	.867	**.905**	.889	**.921**	.910	**.932**	.926	**.937**	.935
BEST @100	.867	.854	.887	.879	.909	.899	.921	.915	.928	.926
WORST@100	.853	.837	.879	.865	.898	.892	.912	.907	.922	.919
2-IBES	.883	.882	.903	.900	.919	.913	.930	.924	**.937**	.932
3-IBES	.879	.880	.900	.901	.916	.914	.928	.925	.936	.933

Table XXII. Performance of LSTM networks on tokenizing CICLING data. (Figs.29 and 32)

	training data size							
	eighth		quarter		half		all	
feat.-tag	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.
ENS.-5	87.6%	.866	91.1%	.902	93.5%	.928	94.8%	.942
ENS.-10	87.9%	.868	91.3%	.905	93.6%	.929	95.0%	.944
ENS.-20	88.3%	.872	91.5%	.906	93.7%	.931	95.1%	.946
ENS.-50	88.6%	.875	91.6%	.908	93.8%	.932	95.2%	.947
ENS.-100	88.6%	.875	91.6%	.908	93.8%	.932	95.2%	.947
BEST @100	87.3%	.861	90.5%	.896	92.7%	.920	94.3%	.937
WORST@100	84.9%	.838	88.7%	.876	91.8%	.911	93.4%	.928
2-IBES	91.0%	**.902**	92.8%	**.922**	94.5%	**.939**	95.5%	.950
3-IBES	90.7%	.898	92.6%	.919	94.4%	.938	95.6%	**.951**

Table XXIII. Performance of LSTM networks on joint tokenizing and POS-tagging CICLING data. (Figs.30 and 33)

	training data size							
	eighth		quarter		half		all	
feat.-tag	acc.	f-s.	acc.	f-s.	acc.	f-s.	acc.	f-s.
ENS.-5	85.7%	.859	89.3%	.895	91.3%	.912	93.1%	.930
ENS.-10	86.4%	.866	89.8%	.899	91.6%	.916	93.4%	.933
ENS.-20	86.8%	.870	90.0%	.901	92.0%	.920	93.4%	.933
ENS.-50	87.2%	.874	90.2%	**.902**	92.2%	**.922**	93.5%	.934
ENS.-100	87.1%	.873	90.1%	**.902**	92.2%	**.922**	93.6%	**.935**
BEST @100	84.5%	.848	87.8%	.879	90.2%	.902	91.9%	.918
WORST@100	83.3%	.834	86.4%	.865	89.2%	.892	91.0%	.909
2-IBES	87.5%	**.876**	90.2%	**.902**	92.0%	.920	93.4%	.934
3-IBES	86.0%	.861	89.2%	.892	91.5%	.915	93.4%	.933

Table XXIV. Effects of POS tags in the tokenization performance of LSTM networks on CICLING data. (Figs.31 and 34)

	training data size							
	eighth		quarter		half		all	
feat.-tag	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**	**+pos**	**−pos**
ENS.-5	.889	.866	.922	.902	.936	.928	.950	.942
ENS.-10	.896	.868	.925	.905	.938	.929	.951	.944
ENS.-20	.900	.872	**.927**	.906	.941	.931	.950	.946
ENS.-50	.904	.875	**.927**	.908	**.943**	.932	.952	.947
ENS.-100	.903	.875	.926	.908	**.943**	.932	**.953**	.947
BEST @100	.885	.861	.909	.896	.926	.920	.941	.937
WORST@100	.871	.838	.899	.876	.920	.911	.934	.928
2-IBES	**.906**	.902	**.927**	.922	.942	.939	.951	.950
3-IBES	.893	.898	.920	.919	.937	.938	.952	.951

Table XXV. Performance of LSTM networks on joint tokenizing and POS-tagging large tokens based on the tokenizing and POS-tagging results of small tokens on ALT data, syllable-wisely. (Figs.35 and 37)

	training data size									
	sixteenth		eighth		quarter		half		all	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
ENS.-5	96.8%	.870	97.9%	.892	98.3%	.910	98.6%	.922	99.0%	.928
ENS.-10	96.9%	.872	98.0%	.894	98.5%	.911	98.7%	.922	99.0%	.928
ENS.-20	97.1%	.873	98.0%	.894	98.6%	.911	98.7%	.922	99.1%	.928
ENS.-50	97.2%	.873	98.0%	.894	98.6%	.911	98.7%	.922	99.1%	.928
ENS.-100	97.2%	.873	98.0%	.894	98.5%	.911	98.8%	.923	99.1%	.928
BEST @100	96.3%	.868	97.4%	.890	98.1%	.908	98.6%	.921	98.9%	.927
WORST@100	94.4%	.857	96.7%	.883	97.6%	.904	98.2%	.918	98.6%	.925
2-SYL.S->L	98.4%	**.877**	98.6%	**.895**	98.9%	**.912**	99.0%	**.924**	99.1%	**.930**
3-SYL.S->L	98.2%	.867	98.5%	.892	98.8%	.909	99.0%	.920	99.1%	**.930**

Table XXVI. Performance of LSTM networks on joint tokenizing and POS-tagging large tokens based on the tokenizing and POS-tagging results of small tokens on ALT data, small token-wisely. (Figs.36 and 38)

	training data size									
	sixteenth		eighth		quarter		half		all	
feat.-tag	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**	**acc.**	**f-s.**
ENS.-5	97.7%	.873	98.2%	.893	98.5%	.910	98.7%	.922	98.9%	.929
ENS.-10	97.7%	.872	98.3%	.893	98.6%	**.911**	98.8%	.922	98.9%	.928
ENS.-20	97.8%	.873	98.2%	.893	98.6%	**.911**	98.8%	**.923**	98.9%	.928
ENS.-50	97.8%	.873	98.2%	.893	98.6%	**.911**	98.8%	**.923**	98.9%	.928
ENS.-100	97.8%	**.874**	98.2%	.893	98.6%	**.911**	98.8%	**.923**	98.9%	.928
BEST @100	97.6%	.873	98.1%	.892	98.4%	.910	98.7%	.922	98.9%	.928
WORST@100	95.3%	.858	96.6%	.882	97.8%	.905	98.3%	.919	98.6%	.926
2-TOK.S->L	97.6%	**.874**	98.2%	**.895**	98.6%	**.911**	98.7%	**.923**	98.9%	**.931**
3-TOK.S->L	97.2%	.865	97.9%	.890	98.6%	.908	98.7%	.919	98.9%	.930

6. DISCUSSION

In Sections 4 and 5, we have presented detailed experimental results to illustrate the performance and characteristics of different approaches in the task of Burmese morphological analysis, based on the data prepared in this study. Besides those differences brought by different techniques and experimental settings, the training data size is actually the most important factor affecting the performance. From an intuitive glimpse of the figures, it can be observed that the temporary annotated data around twenty thousand sentences have not yet reached the upper bound of the capacity of available machine learning approaches. The ALT data contain around $0.66\ \mathcal{M}$ small tokens and $0.50\ \mathcal{M}$ large tokens; the CICLING data have a smaller number only around $0.19\ \mathcal{M}$. Referring to several popular Chinese word segmentation datasets presented in Emerson [2005], we can find the scale on words (tokens) is from $1.10\ \mathcal{M}$ (*Peking University* *corpus*) to $5.45\,\mathcal{M}$ (*Academia Sinica corpus*), i.e., there is one order of magnitude difference in the quantity, despite the ALT Burmese data have more abundant information than only tokenization. A more comparable corpus is *The Balanced Corpus of Contemporary Written Japanese*,[34] which contains two-layer well-designed annotation [Ogura et al. 2011], has over one hundred million words. However, such a huge corpus is based on more than thirty years of NLP development on Japanese community. For the research community of Burmese processing, The ALT Burmese data prepared in this study provided a solid basis for further and development in coming years.

Based on the temporary Burmese data, we conducted experiments of CRFs and LSTM-based RNN. Although RNN achieved better performance than CRFs by a small margin in some cases, we consider the traditional CRFs are still a more proper approach than RNN in the temporary Burmese morphological analysis, considering 1) the quantity of the temporary training data, 2) the interpretability of the model, and 3) the requirement of computing resources. The empiricism rooted methodology of NN-based approaches can offer efficient end-to-end solutions, where the human designed features are largely (if not completely) substituted by huge amount of data and strong ability of computing resource. However, the processing of Burmese is still at an early stage where more experiments and investigation are required. As a matured technique, CRFs can be implemented, applied, and maintained more feasibly than NNs. CRFs can also be extend to a semi-supervised learning interface [Fujii et al. 2017] to make use of large amount unlabeled data, which accommodates the temporary case of Burmese.

In this study, the morphological analysis of Burmese is totally syllable-based. Considering the characteristics of Burmese, further sub-syllabic structures can be figured out [Ding et al. 2017], which are largely related to phonology rather than morphology. We have tried some sub-syllabic features in CRFs, and added the character embedding to NN-based approaches, but no attempt led to better performance. Specific sub-syllabic features may offer certain information of etymology, while they hardly affect the precision in automatic process.[35] As mentioned, the Burmese syllables can be compared with Chinese characters, which are relatively independent writing units, we consider the syllable should be the nature and standard unit in processing Burmese, independent from specific machine learning approaches.

7. CONCLUSION AND FUTURE WORK

This study focuses on Burmese morphological analysis, from annotated corpus preparation to experiment based investigation. Our annotated corpus of twenty thousand Burmese sentences has been released under a `CC BY-NC-SA` license for research community. We conducted experiments by standard sequence labeling approach of CRFs and a state-of-the-art LSTM-based RNN approach. Attributed to this study, Burmese should no longer be referred to as a low-resourced or under-studied language, in terms of morphological analysis.

We have two clear directions in future work. Regarding to the Burmese processing, we are preparing the final treebank of Burmese, established on the morphologically annotated sentences prepared in this study. As a typical head-final languages as Japanese, we are ready to apply parsing techniques on Japanese to Burmese. As to the scope of NLP on low-resourced languages in Asia, the next language on our schedule is Khmer. The related experiences on developing data and techniques of Burmese processing can help us annotate and process Khmer more efficiently.

[34] `http://pj.ninjal.ac.jp/corpus_center/bccwj/en/`

[35] Taking English as an example, a word beginning with *wh-* is likely to have a Germanic origin, while *rh-* can strongly indicate the word is borrowed from Greek. However, such clues do not directly contribute to identify morphological or syntactic roles of specific words.

ACKNOWLEDGMENTS

We thank many translators and annotators in the construction of Burmese ALT data. We thank Dr. Ye Kyaw Thu and Khin War War Htike for their discussion and contribution in Burmese processing.

REFERENCES

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long Short-Term Memory Neural Networks for Chinese Word Segmentation. In *Proc. of EMNLP*. 1197–1206.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. 1724–1734.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12 (2011), 2493–2537.

Chenchen Ding, Win Pa Pa, Masao Utiyama, and Eiichiro Sumita. 2017. Burmese (Myanmar) Name Romanization: A Sub-Syllabic Segmentation Scheme for Statistical Solutions. In *Proc. of PACLING*. 227–238.

Chenchen Ding, Ye Kyaw Thu, Masao Utiyama, and Eiichiro Sumita. 2016. Word Segmentation for Burmese (Myanmar). *ACM Transactions on Asian and Low-Resource Language Information Processing* 15, 4 (2016). Article No. 22.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proc. of SIGHAN workshop on Chinese language Processing*. 123–133.

Ryo Fujii, Ryo Domoto, and Daichi Mochihashi. 2017. Nonparametric Bayesian Semi-supervised Word Segmentation. *Transactions of the Association for Computational Linguistics* 5 (2017), 179–189.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research* 3, Aug (2002), 115–143.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* (2017), 1735–1780.

Khin War War Htike, Ye Kyaw Thu, Zuping Zhang, Win Pa Pa, Yoshinori Sagisaka, and Naoto Iwahashi. 2017. Comparison of Six POS Tagging Methods on 10K Sentences Myanmar Language (Burmese) POS Tagged Corpus. In *Proc. of CICLING*. id. 276.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). (In *Proc. of ICLR 2015*).

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proc. of NAACL*. 1–8.

Taku Kudo and Yuji Matsumoto. 2002. Support vector machine *wo mochiita* chunk *dōtei*. *Journal of Natural Language Processing (Shizengengoshori)* 9, 5 (2002), 3–21. in Japanese.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP*. 230–237.

John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*. 282–289.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*. 1064–1074.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Proc. of Interspeech*, Vol. 2. 1045–1048.

Seung-Hoon Na. 2015. Conditional random fields for Korean morpheme segmentation and POS tagging. *ACM Transactions on Asian and Low-Resource Language Information Processing* 14, 3 (2015), 10.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980* (2017).

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proc. of ACL-HLT*. 529–533.

Hideki Ogura, Hanae Koiso, Yumi Fujiike, Sayaka Miyauchi, Hikari Konishi, and Yutaka Hara. 2011. (2011).
http://pj.ninjal.ac.jp/corpus_center/bccwj/doc/report/JC-D-10-05-01.pdf,
http://pj.ninjal.ac.jp/corpus_center/bccwj/doc/report/JC-D-10-05-02.pdf, in Japanese.

Hammam Riza, Michael Purwoadi, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Vichet Chea, Rapid Sun, Sethserey Sam, Sopheap Seng, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. 2016. Introduction of the Asian language treebank. In *Proc. of O-COCOSDA*. 1–6.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958.

Karl Stratos and Michael Collins. 2015. Simple Semi-Supervised POS Tagging. In *Proc. of NAACL-HLT*. 79–87.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*. 3104–3112.

Ye Kyaw Thu, Win Pa Pa, Masao Utiyama, Andrew M Finch, and Eiichiro Sumita. 2016. Introducing the Asian Language Treebank (ALT).. In *Proc. of LREC*. 1574–1578.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for Chinese word segmentation. *ACM Transactions on Asian Language Information Processing* 9, 2 (2010), 5.

Received ; revised ; accepted

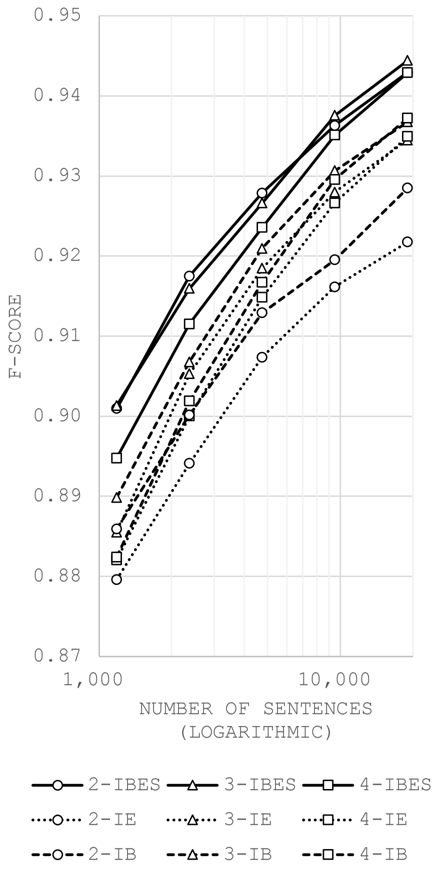

Fig. 6.   F-scores in Table V

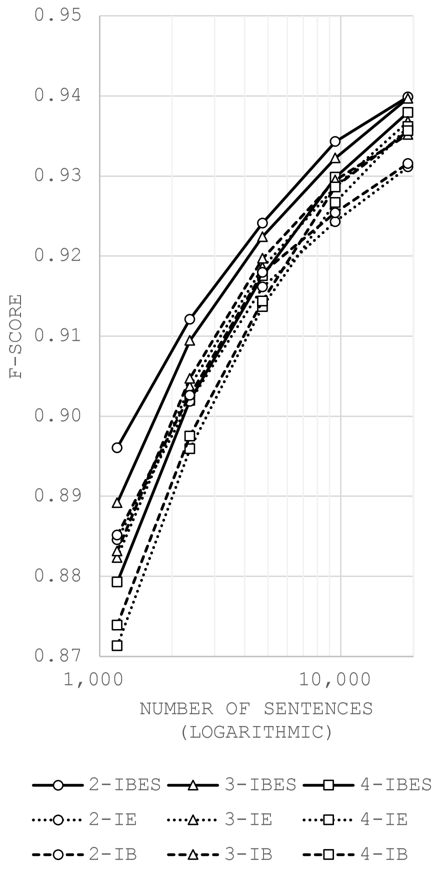

Fig. 7.   F-scores in Table VI

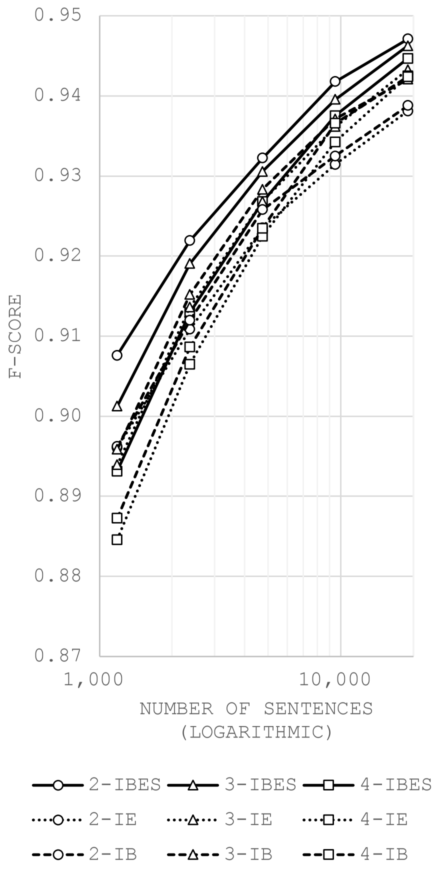

Fig. 8.   F-scores of +**pos** in Table VII

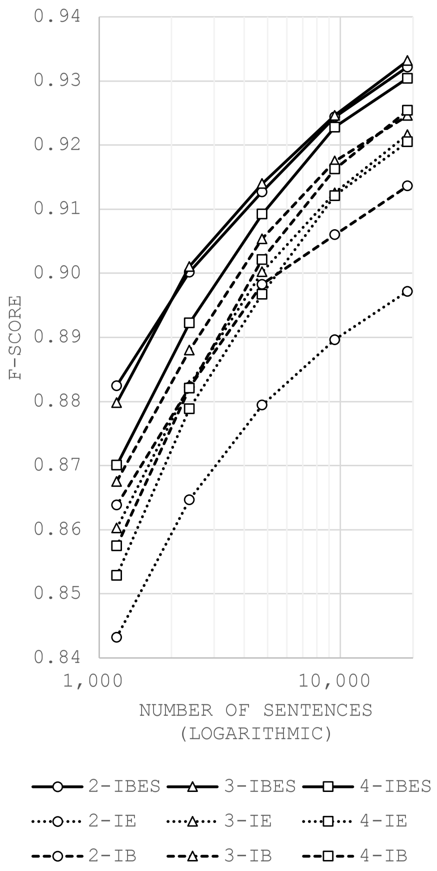

Fig. 9. F-scores in Table VIII

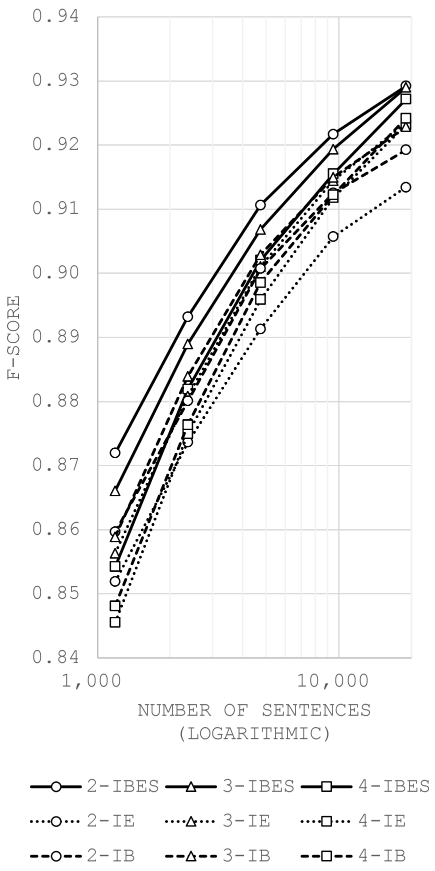

Fig. 10. F-scores in Table IX

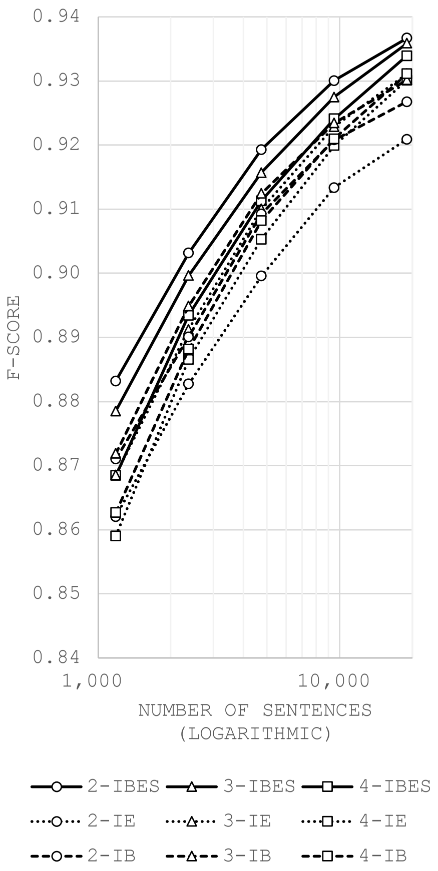

Fig. 11. F-scores of +**pos** in Table X

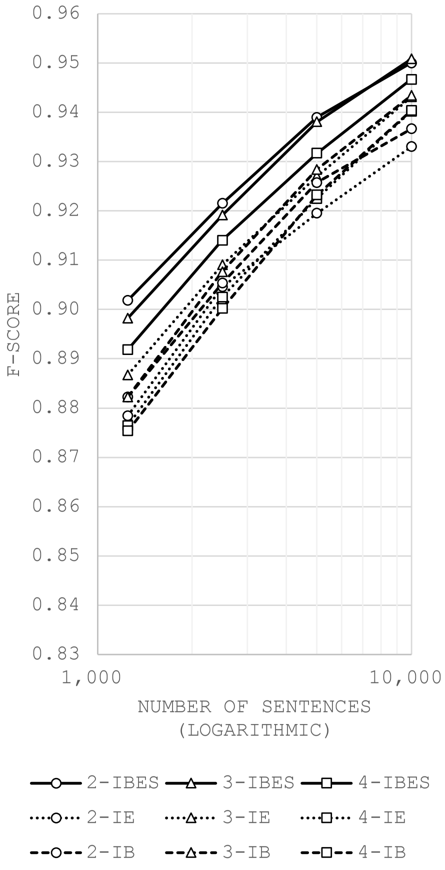

Fig. 12. F-scores in Table XI

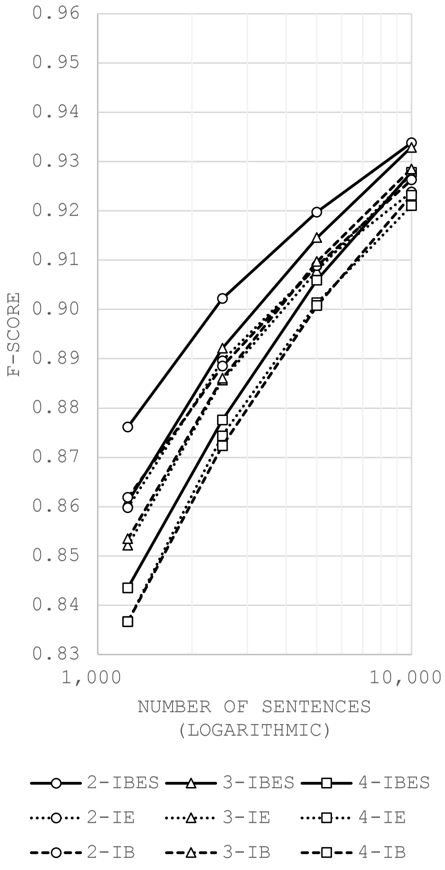

Fig. 13. F-scores in Table XII

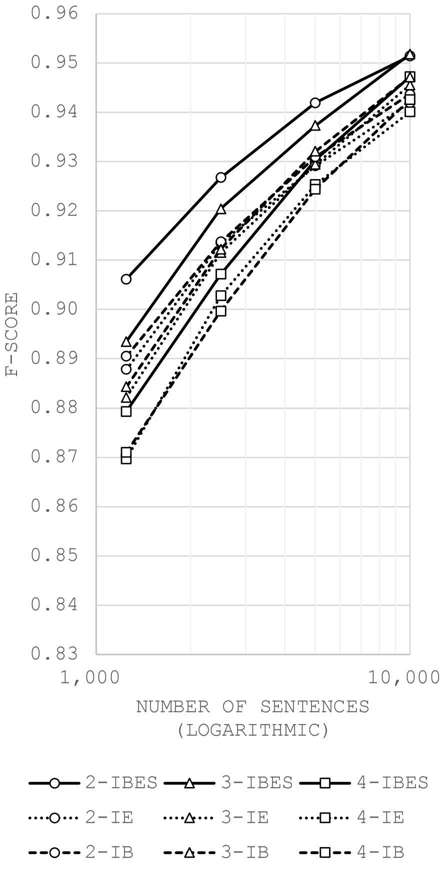

Fig. 14. F-scores of +**pos** in Table XIII

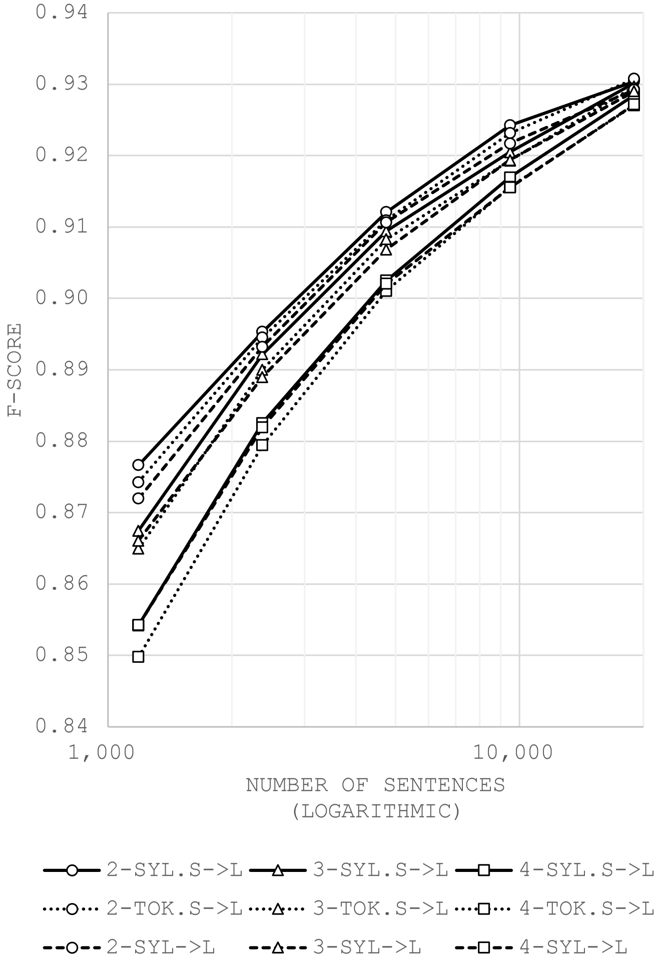

Fig. 15. F-scores in Table XIV

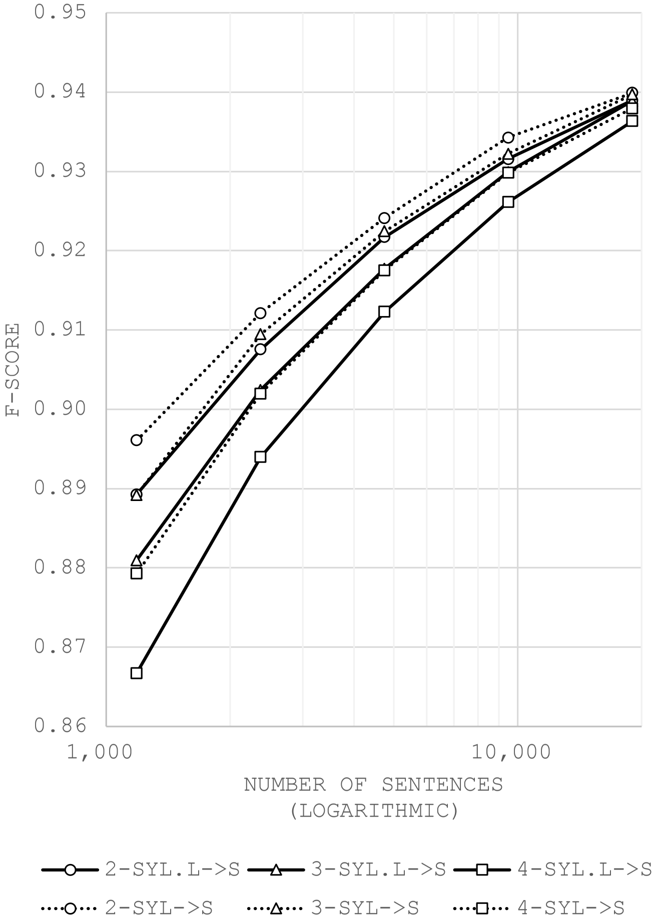

Fig. 16. F-scores in Table XV

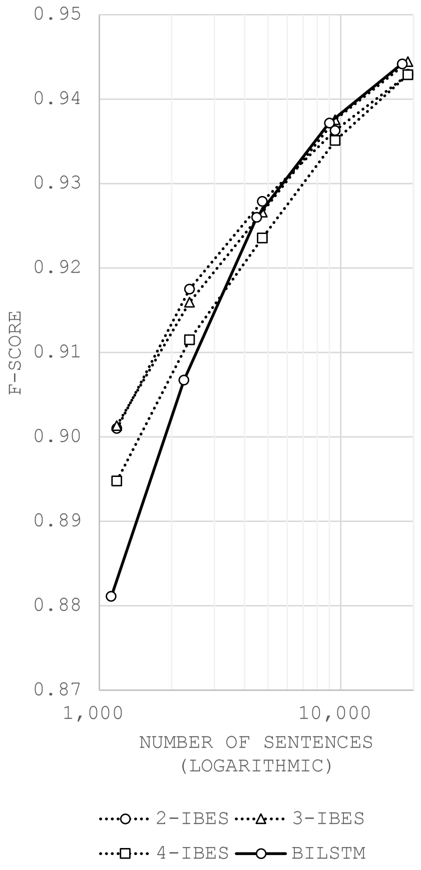

Fig. 17. F-scores in Table XVI
(comparison of LSTM (ens.-100) and CRFs)

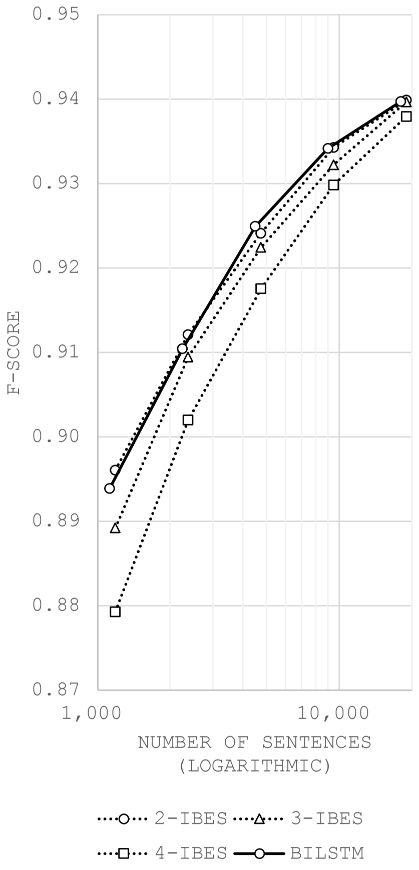

Fig. 18. F-scores in Table XVII
(comparison of LSTM (ens.-100) and CRFs)

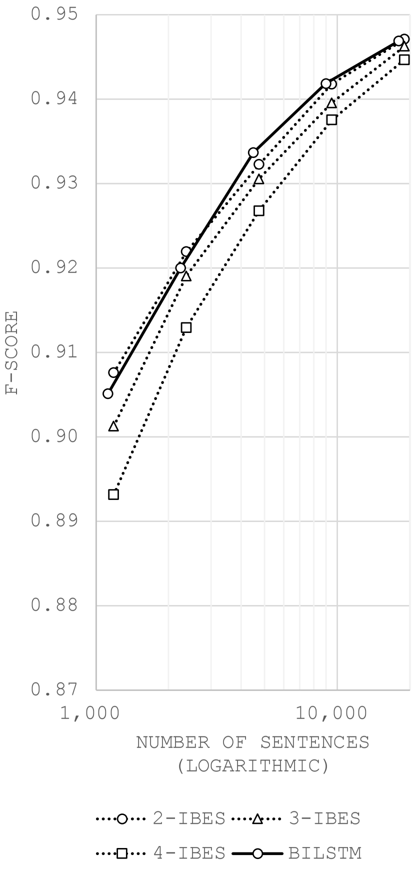

Fig. 19. F-scores of +**pos** in Table XVIII
(comparison of LSTM (ens.-100) and CRFs)

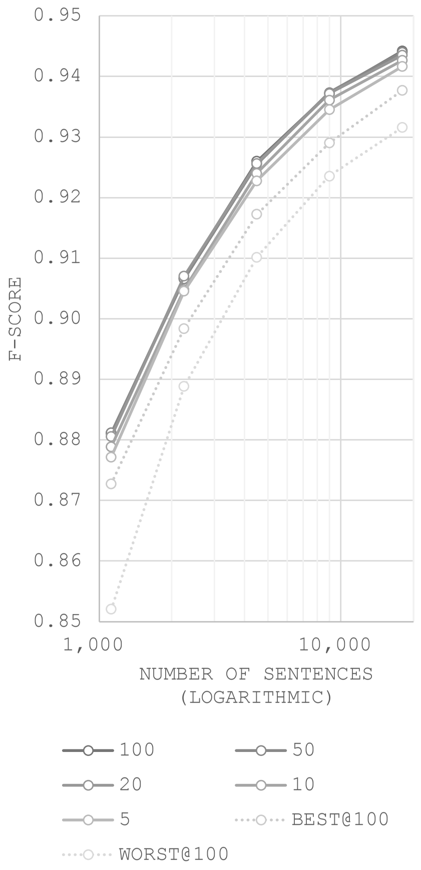

Fig. 20. F-scores in Table XVI
(comparison of the effect of ensemble in LSTM)

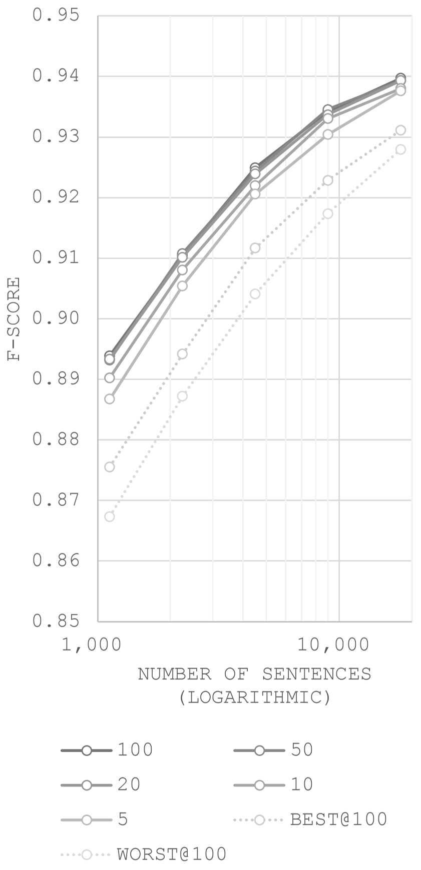

Fig. 21. F-scores in Table XVII
(comparison of the effect of ensemble in LSTM)

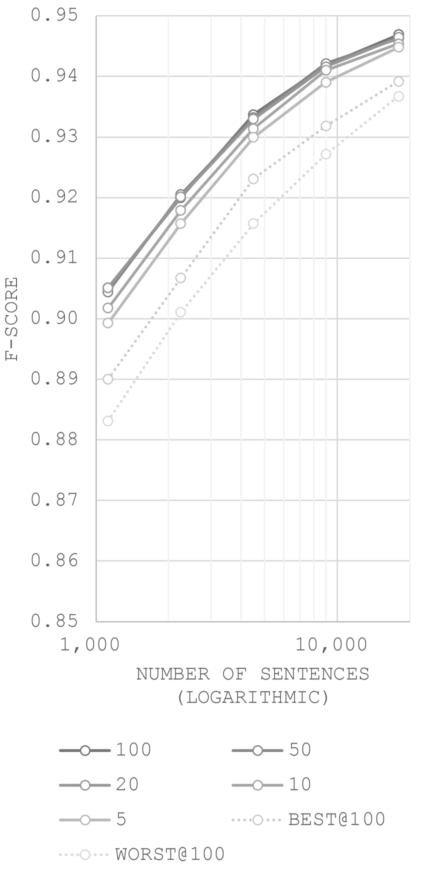

Fig. 22. F-scores of +**pos** in Table XVIII
(comparison of the effect of ensemble in LSTM)

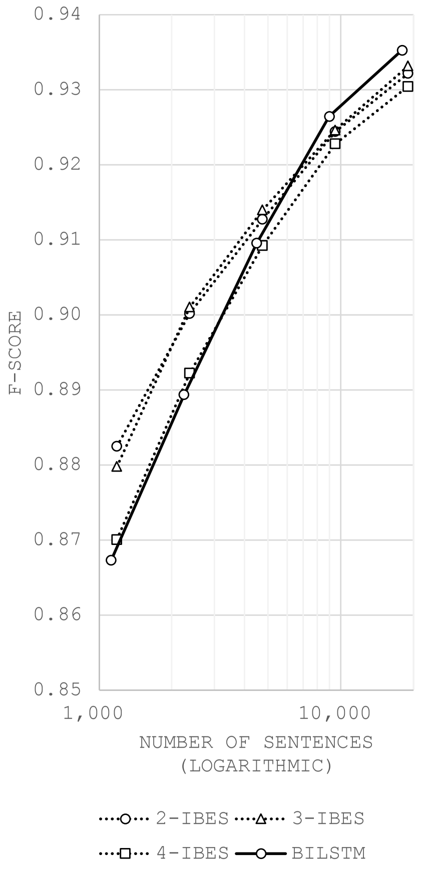

Fig. 23. F-scores in Table XIX
(comparison of LSTM (ens.-100) and CRFs)

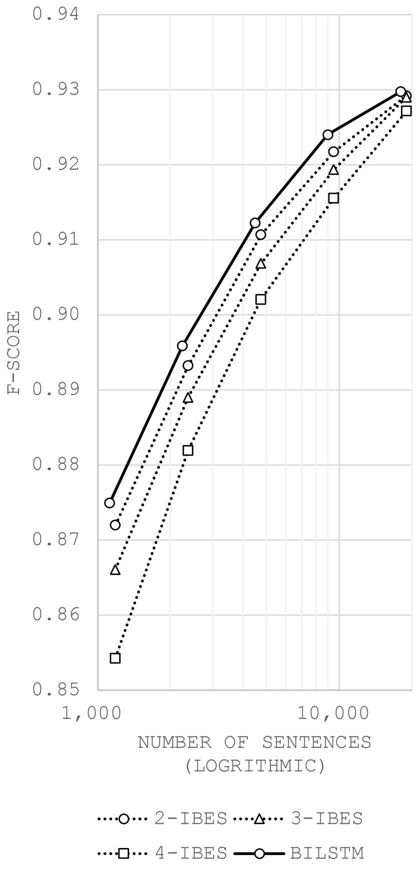

Fig. 24. F-scores in Table XX
(comparison of LSTM (ens.-100) and CRFs)

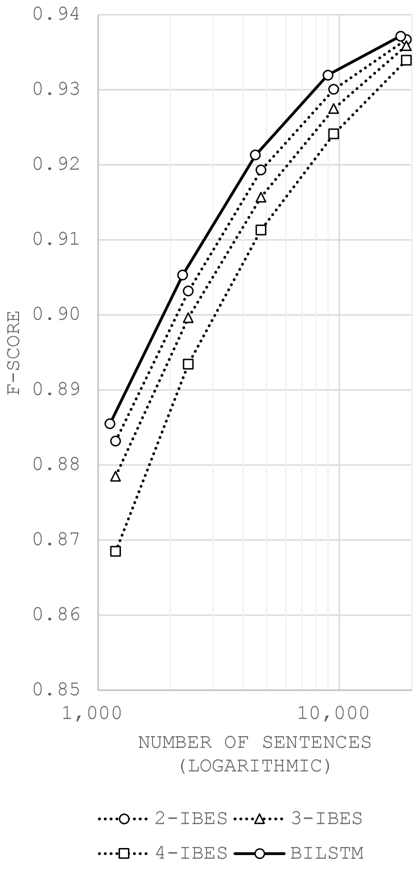

Fig. 25. F-scores of +**pos** in Table XXI
(comparison of LSTM (ens.-100) and CRFs)

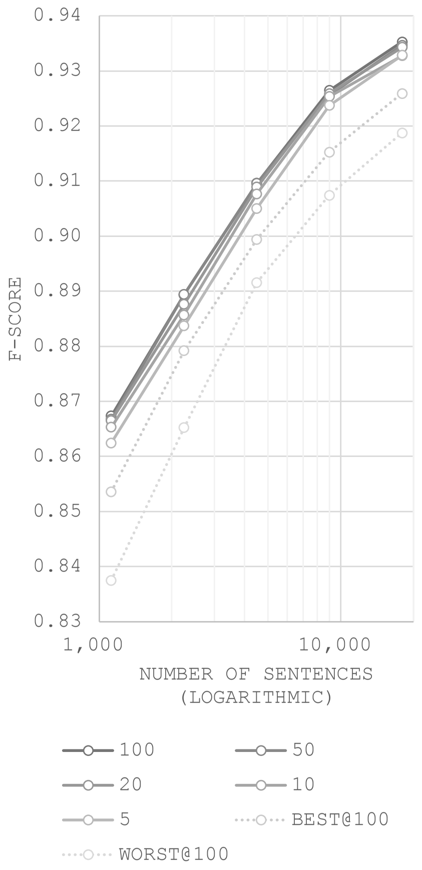

Fig. 26. F-scores in Table XIX
(comparison of the effect of ensemble in LSTM)

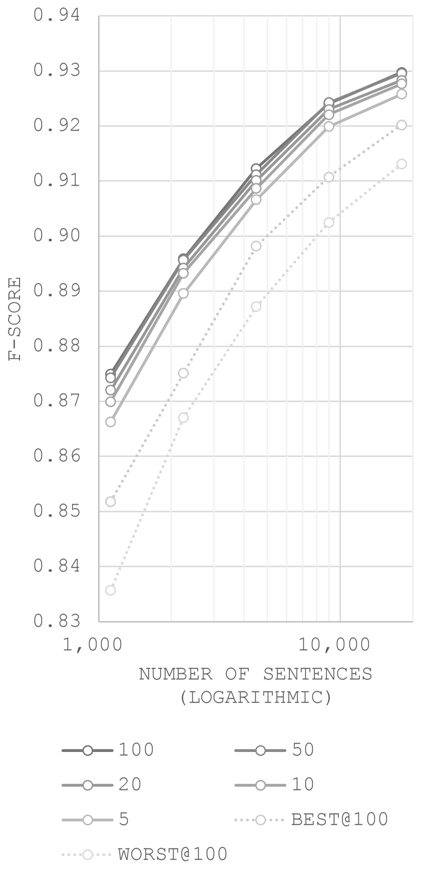

Fig. 27. F-scores in Table XX
(comparison of the effect of ensemble in LSTM)

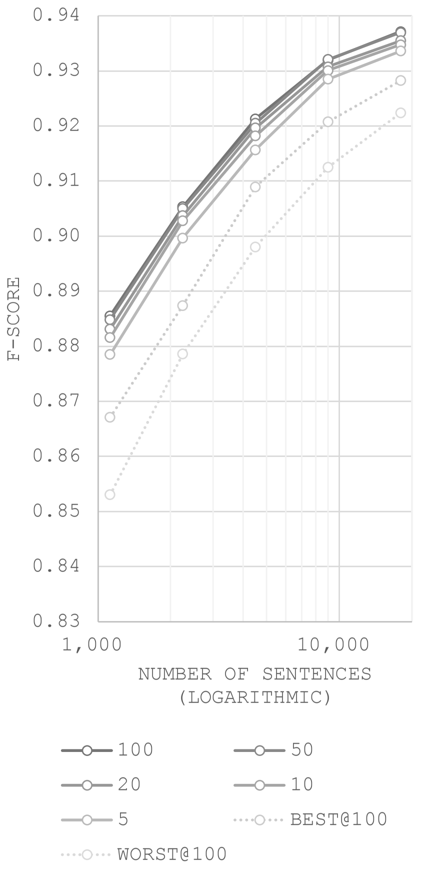

Fig. 28. F-scores of +**pos** in Table XXI
(comparison of the effect of ensemble in LSTM)

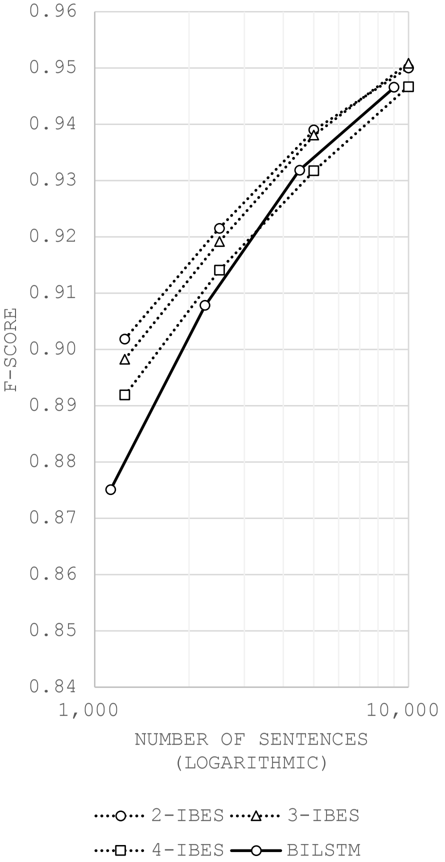

Fig. 29. F-scores in Table XXII (comparison of LSTM (ens.-100) and CRFs)

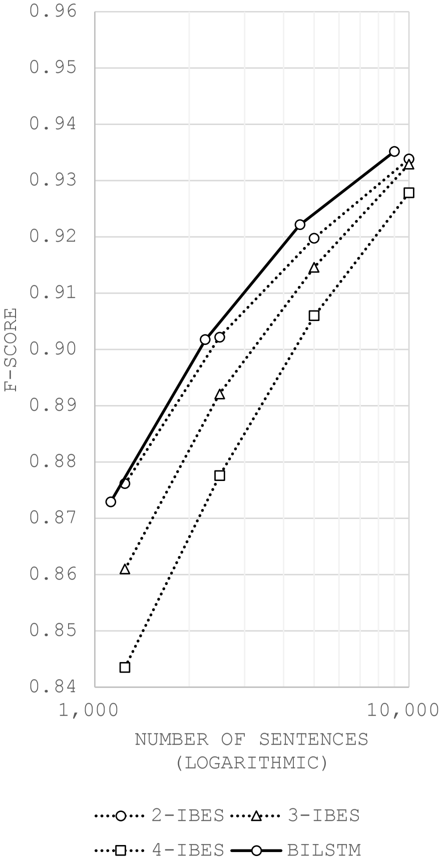

Fig. 30. F-scores in Table XXIII (comparison of LSTM (ens.-100) and CRFs)

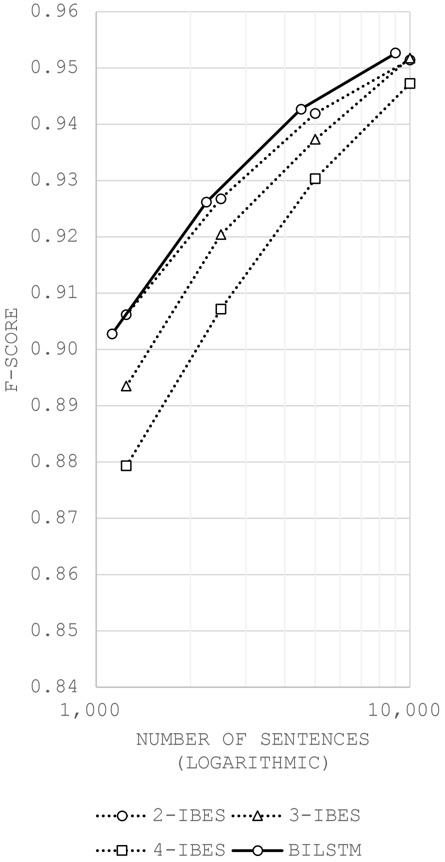

Fig. 31. F-scores of +**pos** in Table XXIV (comparison of LSTM (ens.-100) and CRFs)

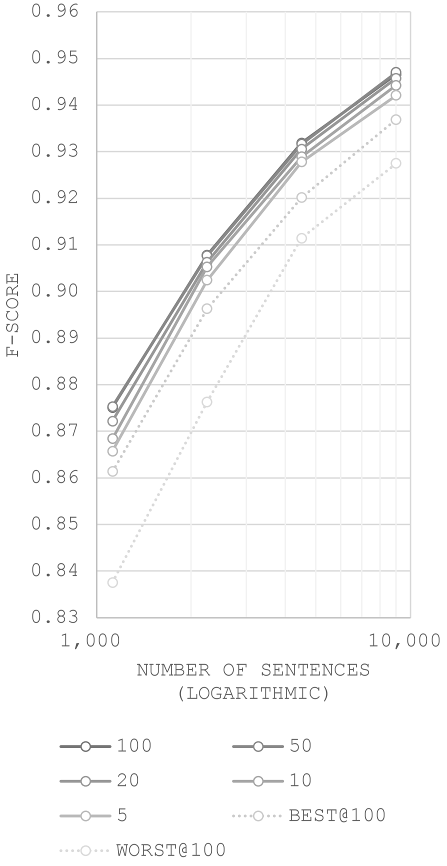

Fig. 32. F-scores in Table XXII
(comparison of the effect of ensemble in LSTM)

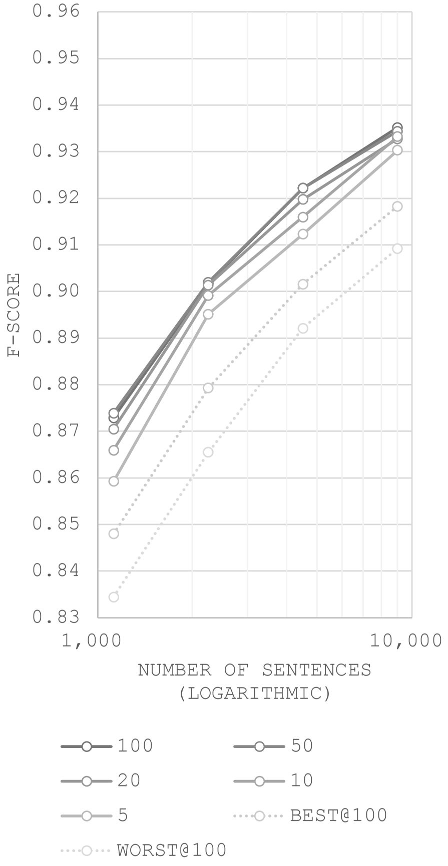

Fig. 33. F-scores in Table XXIII
(comparison of the effect of ensemble in LSTM)

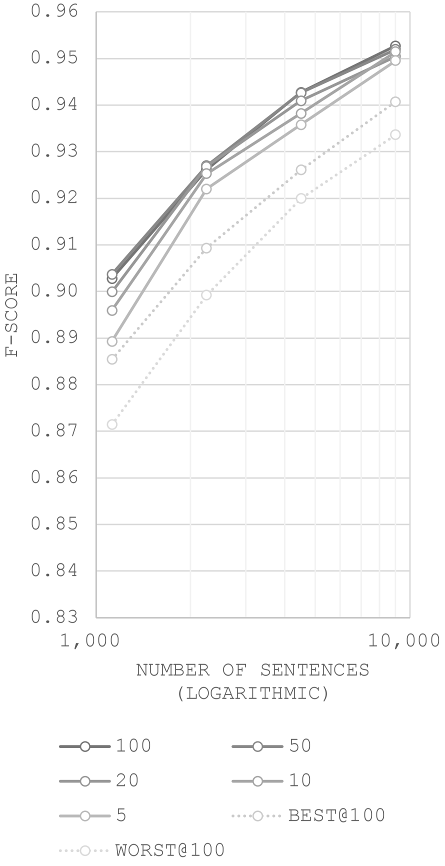

Fig. 34. F-scores of +**pos** in Table XXIV
(comparison of the effect of ensemble in LSTM)

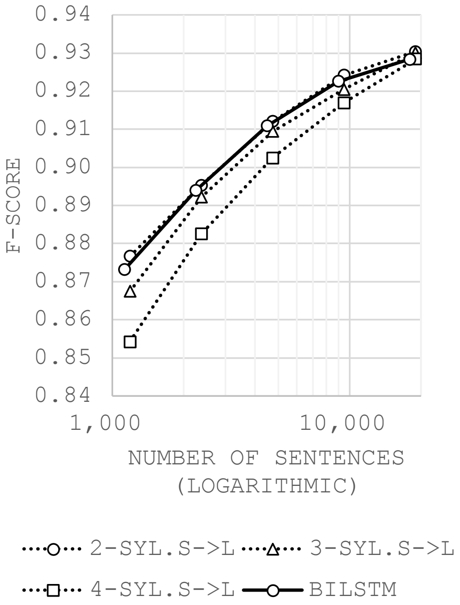

Fig. 35. F-scores in Table XXV
(comparison of LSTM (ens.-100) and CRFs)

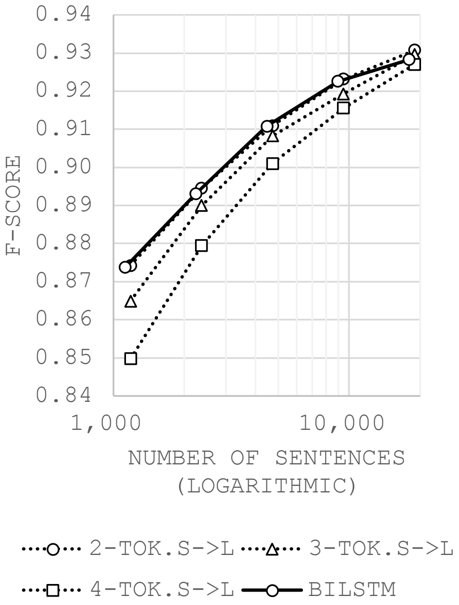

Fig. 36. F-scores in Table XXV
(comparison of the effect of ensemble in LSTM)

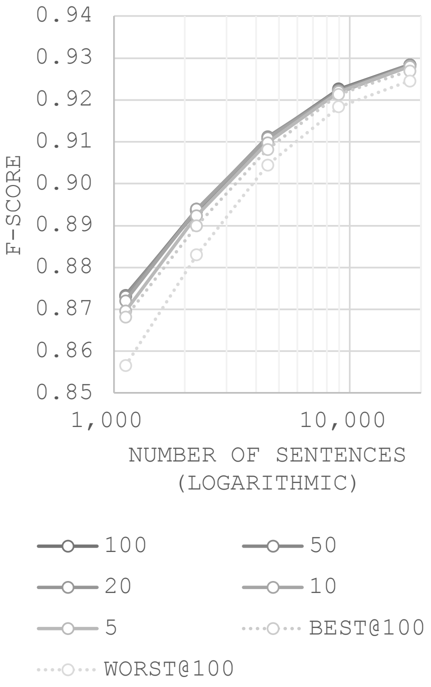

Fig. 37. F-scores in Table XXVI
(comparison of LSTM (ens.-100) and CRFs)

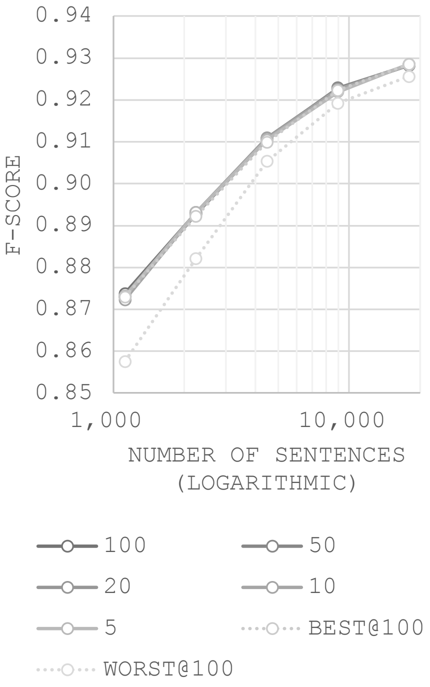

Fig. 38. F-scores in Table XXVI
(comparison of the effect of ensemble in LSTM)