Toiling with the Pāli Canon

Frederik Elwert[1], Sven Sellmer[2],
Sven Wortmann[3], Manuel Pachurka[1], Jürgen Knauth[4], David Alfter
(1) Ruhr-Universität Bochum, (2) Adam Mickiewicz University in Poznań,
(3) Universität zu Köln, (4) Georg-August-Universität Göttingen
E-mail: frederik.elwert@rub.de, sven.sellmer@amu.edu.pl,
swortmann@uni-koeln.de, manuel.pachurka@rub.de,
alfter.david@gmx.net

Abstract

The paper describes the preparation of a Buddhist corpus in the Middle Indo-Aryan language Pāli, which is available only in a flat TEI format, for content-based analysis. This task includes transforming the file into a hierarchical TEI P5 representation, followed by tokenisation (including sandhi resolution), lemmatisation, and POS tagging.

1 Introduction

The Pāli Canon is the oldest corpus of Buddhist texts and is considered authoritative by the Theravāda sect (for an overview see von Hinüber [4]). Its name is derived from the Middle Indo-Aryan language in which it was composed (Oberlies [7]). It originated as oral literature in Northern India in the 4th c. BCE, spread and grew in the course of the expansion of Buddhism and was finally committed to writing ca. in the 1st c. BCE on the island of Sri Lanka. In the subsequent period, the text still underwent some changes, but only to a much smaller degree. It consists of three text collections:

- *Vinayapiṭaka* (disciplinary rules for monks and nuns)
- *Suttapiṭaka* (religious and philosophical teachings)
- *Abhidhammapiṭaka* (detailed systematic accounts of philosophical and psychological doctrines).

The general aim of our project was conducting IT-based research on the *Suttaṭipaka* (ca. 1.75 million lexical units) from the point of view of religious studies using methods of network and topic analysis. Unfortunately, the text was not available in such a form that we could start our analyses right away; rather, we had to prepare the data first. In our paper both these (rather time-consuming) preparatory steps and some outlines of the main results will be presented. (The other two *piṭaka*s have not been dealt with in any detail by us so far, but a first assessment shows that generally the same steps as described in this paper can be applied to them, only minor modifications being necessary.)

2 Preparation

2.1 Text structure

The text of the *Suttaṭipaka* is available in digital form ([13]) in a kind of flat TEI format in the now outdated version P4. The encoding focuses mainly on the presentational aspects and was apparently designed to reflect the typographical features of the printed edition and so to allow for an adequate HTML publication.

The text is contained in 541 XML files with names bearing encoded information about their position in a four-tiered hierarchy of:

- 1st level: *piṭaka*
- 2nd level: *nikāya*
- 3rd level: *pāḷi*
- 4th level: variously named divisions, like: *vagga* ("group"), *sutta* ("teaching"), *saṃyutta* ("collection") etc.

E.g., the file name "s0202m.mul0.xml" is to be interpreted as follows:

- s *Suttapiṭaka*
- 02 2nd *nikāya*, i.e., *Majjhimanikāya*
- 02 2nd *pāḷi* in the *Majjhimanikāya*, i.e., *Majjhimapaṇṇāsapāḷi*
- m no function
- mul0 1st *vagga*,[1] i.e., *Gahapativagga* ("mul" indicating that we are dealing with the source text, not with a commentary, which is included in a separate file).

The hierarchical structure of these four levels is also represented in an XML file that uses the tree format for the WebFX framework in order to present a browsable hierarchy on the tipitaka.org website. This XML format is non-standard, but can still be used to extract the hierarchy of the individual TEI P4 files. However, inside the TEI P4 files, a flat structure is used that

[1] This index is zero-based.

gives no information about further nesting of sections in the text. The text is merely divided into a sequence of paragraphs on the same level enclosed between <p> (paragraph) tags. The attribute @rend (rendition) is used in order to record representational features of the text, but no semantic information about the role of the text parts as headings, text body, or verse is encoded, let alone text divisions. The format used thus looks like this:

```
<text>
  <body>
    <p rend="centre"> Namo tassa bhagavato arahato sammāsam-
buddhassa</p>
    <p rend="chapter">1. Gahapativaggo</p>
    <p rend="subhead">1. Kandarakasuttaṃ</p>
    <p rend="bodytext" n="1"><hi rend="paranum">1</hi><hi
rend="dot">.</hi> Evaṃ <pb ed="T" n="2.0001" /><pb ed="M"
n="2.0001" /><pb ed="P" n="1.0339" /><pb ed="V" n="2.0001" />
me sutaṃ – ekaṃ samayaṃ bhagavā campāyaṃ viharati gaggarāya
pokkharaṇiyā tīre mahatā bhikkhusaṅghena saddhiṃ. Atha kho
pesso <note>peyo (ka.)</note> ca hatthārohaputto kandarako ca
paribbājako yena bhagavā tenupasaṅkamiṃsu; […]</p>
  </body>
</text>
```

In addition to the rendition of paragraphs, some additional editorial information is given, though in a very basic form. Page breaks of different editions are indicated, and text variants in the editions are inserted as <note> elements. The paragraphs are usually numbered, though the paragraph number is redundantly encoded twice, using the @n attribute and as part of the text.

This structure was not fine-grained enough for our purposes; e.g., the *Gahapativagga* just mentioned contains accounts of ten separate conversations of different householders with the Buddha, which should be accessible individually. We therefore chose to prepare first of all a hierarchically structured TEI P5 representation with one or more additional levels (as required) by processing the whole *Suttapiṭaka* with a dedicated XSLT script. The script mainly works by recognising beginnings and endings of sections and enclosing them with <div> tags. This is possible by combining different pieces of information, partly contained in XML attributes, partly based on expert knowledge. To the first type belong attributes added to certain kinds of headings by the VRI editors (e.g., <p rend="subhead">); as an example for the second type of information formulaic expressions at the end of sections (e.g., XY-*suttaṃ niṭṭhitaṃ* “teaching XY is finished”) may be adduced. Technically this task proved to be rather tricky, mainly due to inconsistencies both in the source file and in the printed edition, so that a considerable amount of manual control, adjustments and corrections proved necessary. Eventually we managed to arrive at a file with a regular structure of up to six textual levels, the paragraph level being the lowest:

```
<text>
  <body>
    <div type="vagga">
      <head n="1">Gahapativaggo</head>
      <opener>
        <salute> Namo tassa bhagavato arahato sammāsambud-
dhassa</salute>
      </opener>
      <div type="sutta">
        <head n="1">Kandarakasuttaṃ</head>
        <p n="1"> evaṃ <pb ed="T" n="2.0001"/><pb ed="M"
n="2.0001"/><pb ed="P" n="1.0339"/><pb ed="V" n="2.0001"/> me
sutaṃ – ekaṃ samayaṃ bhagavā campāyaṃ viharati gaggarāya
pokkharaṇiyā tīre mahatā bhikkhusaṅghena saddhiṃ. atha kho
pesso <note place="app">peyo (ka.)</note> ca hatthārohaputto
kandarako ca paribbājako yena bhagavā tenupasaṅkamiṃsu; […]
       </p>
      </div>
    </div>
  </body>
</text>
```

Editorial information, like page breaks, has been carried over. Instead of the representational categories like “chapter” or “subhead,” which do not relate to the semantic structure of the corpus, object-language types like “vagga” and “sutta” have been introduced.

Additionally, poems in the text have been transformed into <lg>/<l> structures. In the original files, again also rendition information hints to the logical structure of the text:

```
<p rend="bodytext" n="30"><hi rend="paranum">30</hi><hi
rend="dot">.</hi> ‘‘Brahmunāpesā, mahānāma, sanaṅkumārena
gāthā bhāsitā –</p>
<p rend="gatha1">‘Khattiyo seṭṭho janetasmiṃ, ye
gottapaṭisārino;</p>
```

```
<p rend="gathalast">Vijjācaraṇasampanno, so seṭṭho de-
vamānuse’ti.</p>
```

The resulting files represent the structure in a more transparent way:

```
<p n="30"> “brahmunāpesā, mahānāma, sanaṅkumārena gāthā
bhāsitā –</p>
<lg type="verse">
  <l>‘khattiyo seṭṭho janetasmiṃ, ye gottapaṭisārino;</l>
  <l>vijjācaraṇasampanno, so seṭṭho devamānuse’ti.</l>
</lg>
```

2.2 Linguistic information

In addition to coming to terms with the text structure, our second goal consisted in applying lemmatisation and POS tagging because for our project it was essential to be able to access information about actors and basic concepts. Here, a considerable amount of groundwork had to be done, as practically no tools from computational linguistics were available for Pāli.

2.2.1 Tokenisation and sandhi resolution

A major problem for computational approaches to texts in Sanskrit and Middle Indo-Aryan languages is based on the fact that traditionally all kinds of euphonic sound changes, known by the term sandhi (of Sanskrit origin), are reflected in written texts, including such that result in the merging of two or more words to a single character string (e.g., *cāhaṃ* ← *ca ahaṃ* “and I”). Therefore for virtually all tasks of computational analysis the sandhi changes first have to be reversed, which mostly cannot be done in a mechanical manner because often two or more solutions are theoretically possible. Whereas for Sanskrit, sophisticated programs are available that perform quite well though not entirely without human supervision (Hellwig [2]; [3]), there are no such resources for Pāli.

The chief difference between Pāli and Classical Sanskrit consists in the fact that in Pāli, sandhi is not obligatory in all cases but mainly occurs where certain high-frequency words (especially particles like *ca* “and”, *pi* “also”, or *eva* “even”) are involved. This feature of sandhi in Pāli on the one hand excludes a truly systematic solution as it is possible and required in Sanskrit, but on the other hand enables a pragmatic approach consisting in the manual analysis of frequent “sandhi triggers”. As a result of such an analysis we formulated a set of ca. 175 rules, based on regular expressions, with the help of which the majority of sandhi changes could be undone.

The performance of this rule-based approach was evaluated by checking the number of false positives (i.e., incorrect changes) and false negatives (i.e., unreversed sandhis) in appropriate samples. In 1,555,172 input strings[2] the application of the mentioned ruled caused 93,294 changes. Of these, a random sample of 900 instances was inspected, but no errors were found, so the number of false positives seems to be very low. In order to estimate the percentage of unresolved sandhi changes we undertook a check of a random sample of 1,600 strings. In these, 13 strings featured sandhi changes, which indicates that in the whole text about 13,500 sandhi changes were left unreversed, i.e., probably ca. 87% of all changes were correctly resolved. Further optimisation is quite possible, but would be time-consuming because the most frequent types of changes have already been dealt with by the present set of rules, so every new rule will cover only a rather small number of cases.[3] As the results are good enough to serve as a basis for explorative analyses of the kind intended, we decided to leave the enhancement of the sandhi module for a later stage.

2.2.2 Lemmatisation and POS tagging

Because Pāli is a highly flective language, lemmatisation is anything but a trivial task. Our strategy was a twofold one. In a first step we prepared a reasonably comprehensive list of Pāli word forms that might occur in our text; in a second step we used these forms, together with manually prepared gold data, to train a readily available lemmatiser. For the first step, it was of great help that we could use the data of the digitised version of the *Pāli English Dictionary* (= *PED* [10]) though — because the markup of the dictionary data was very rudimentary — we had to create an improved version by extracting grammatical information (on parts of speech and other aspects) from the plain text of the dictionary articles.[4] In addition we prepared a list of proper names (which are not part of the *PED*) on the basis of the digitised version of the *Dictionary of Pali Proper Names* [7], which involved a manual selection of relevant items because this dictionary contains many items that were not only useless but actually harmful for our analyses, e.g., be-

[2] It is important to speak of “strings” here because, to refer to the example given above, *cāhaṃ* is one string representing two words: *ca* and *ahaṃ*.

[3] This claim can be made quite confidently because as a result of the lemmatisation described in the following section strings resulting from sandhi are almost always tagged as “<unknown>”, the only exceptions being sporadic cases where these strings happen to be identical with possible word forms. Analysing the set of “unknown” strings with the help of suitable regular expressions, it is therefore possible to obtain a good overview of the different types of unresolved sandhis and of their frequencies.

[4] Some aspects of this step are discussed in Knauth and Alfter [6].

cause of homonymy. Pāli also features a large number of compounds, of which we managed to identify and split about 7,500 (in terms of types). The list of word forms was built of four kinds of data:

- list of indeclinables, taken directly from the improved dictionary
- list of irregular forms of nouns, pronouns, and numerals (obtained by manual input)
- list of verb forms (by courtesy of Yukio Yamanaka, a scholar specialising in Pāli verbs); this list is still being supplemented by Dr Yamanaka and presently comprises about 1/3 of all verb forms occurring in the Pāli canon
- list of regular declension forms of nouns and adjectives; this list was prepared by a simple generator algorithm that combined noun stems with the appropriate endings (here, a certain amount of overgeneration and some false forms were inevitable, but on the whole the generator yielded quite acceptable results).

The main step of the lemmatisation and the POS tagging were both done by the well-known NLP tool "TreeTagger" (Schmid [11]; [12]); the gold data being obtained by a manual annotation of 1,090 sentences with in total 14,017 words, of which a portion of 92 sentences with 1,496 words were not used in training the tagger, but kept for evaluation purposes. As for our main task fine-grained results were not required we chose a simple tag set with the following items:

- noun
- proper noun
- verb
- particle
- punctuation
- sentence delimiter
- adverb
- adjective
- pronoun
- numeral

The performance was checked with the help of the just mentioned evaluation data set of 1,496 words. In 18 instances a sandhi combination was unresolved, so these items were not taken into account. Of the remaining tags, 121 attributions turned out to be wrong, which yields a performance rate of 91.75% correctly tagged items. Taking a closer look at the mistakes, the following types can be distinguished:

Type of mistake		Frequency
Attributed tag	Correct tag	
NOUN	ADJ	58
NOUN	VERB	37
ADJ	NOUN	15
ADJ	VERB	7
NOUN	PRON	1
PROP	ADJ	1
PROP	NOUN	1
PROP	VERB	1

From these figures it clearly emerges that two types of mistakes have by far the greatest impact:

- verbs not recognized: 45 (37.2%)
- noun for adjective or vice versa: 73 (60.3%).

Here, it can be remarked that the first type is bound to disappear almost entirely as soon as the list of verb forms will be complete, which would improve the overall performance to a range of about 95%. As far as the second error type is concerned, it will be much more difficult to minimise, because it is often objectively difficult to distinguish between adjectives and nouns in Pāli: Both categories are formally identical, nominalisation is a very common process, and the word order is quite free (though it may provide some clues), so that even a human reader regularly finds it difficult to take a decision.

The lemmatising performs at somewhat lower precision rates though a detailed assessment would require a careful reading of many passages because Pāli features a particularly high percentage of homonyms and these are treated as separate items by the TreeTagger only if they belong to a different POS class.

It is planned to make the file we prepared according to the above description available on a web repository for Old Indian texts as soon as we will have solved some minor technical issues; all tools used will be published in open source form in the near future.

2.3 Workflow

Since the TEI format is not well suited for computational linguistics tasks and analysis, the TEI representation was first transformed on the fly into the simpler weblicht TCF format (Hinrichs [5]). These data were then passed to the TreeTagger with the help of a small adapter module. Further analysis was carried out on the POS tagged and lemmatised data using the TCFnetworks package (Elwert [1]) and MALLET (McCallum [8]).

3 Corpus-based textual research

The described preparation enabled us to doing content-related research by performing topic analysis with the help of MALLET. Concretely, we selected 102 thematically more or less homogeneous text passages with a median length of 1,607 lemmata (after removal of stopwords) and generated several sets with different numbers of topics, which offer promising starting points for further, in-depth analyses by specialists. At this point, it may only be remarked that a three-topic set yielded quite an interesting thematic division, which can very roughly be described as meditation—self-cultivation—philosophy; here it is represented by the three highest-rated lemmata in each topic:

- *samādhi* ("concentrated meditation"), *kāya* ("body"), *samaṇabhrāhmaṇa* ("ascetics and brahmins")
- *vedanā* ("feeling"), *magga* ("path"), *pajānāti* ("understand")
- *dukkha* ("leading to suffering, unsatisfactory"), *citta* ("mind"), *anicca* ("impermanent")

References

[1] Elwert, Frederik (2014). TCFnetworks. URL: https://github.com/SeNeReKo/TCFnetworks.

[2] Hellwig, Oliver (2009). SanskritTagger, a stochastic lexical and POS tagger for Sanskrit. In: *Proceedings of the First International Sanskrit Computational Linguistics Symposium*, pp. 37-46.

[3] — (2010). Performance of a lexical and POS tagger for Sanskrit. In: *Proceedings of the Fourth International Sanskrit Computational Linguistics Symposium*, pp. 162-172.

[4] von Hinüber, Oskar (1996). *A handbook of Pāli literature*. Berlin: de Gruyter.

[5] Hinrichs, Erhard W., Marie Hinrichs, and Thomas Zastrow (2010). WebLicht: Web-Based LRT Services for German. In *Proceedings of the ACL 2010 System Demonstrations*, pp. 25–29. http://www.aclweb.org/anthology/P10-4005.

[6] Knauth, Jürgen and David Alfter (2014). A Dictionary Data Processing Environment and Its Application in Algorithmic Processing of Pali Dictionary Data for Future NLP Tasks. In *Proceedings of the 5th Workshop on South and Southeast Asian NLP*, pp. 65–73.

[7] Malalasekara, Gunapala P. (1937–1938). *Dictionary of Pāli proper names*. London: Murray. Digitised version URL: http://www.palikanon.com/english/pali_names/dic_idx.html

[8] McCallum, Andrew Kachites (2002). *MALLET: A Machine Learning for Language Toolkit*. URL: http://mallet.cs.umass.edu.

[9] Oberlies, Thomas (2001). *Pāli: a grammar of the language of the Theravāda Tipiṭaka*. Berlin: de Gruyter.

[10] Rhys Davids, T. W. and William Stede (1921–1925). *The Pali Text Society's Pali-English dictionary*. Digitised version URL: http://dsal.uchicago.edu/dictionaries/pali/

[11] Schmid, Helmut (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In: *Proceedings of the International Conference on New Methods in Language Processing*, 12, pp. 44–49.

[12] — (1995). Improvements in Part-of-Speech Tagging with an Application to German. In *Proceedings of the ACL SIGDAT-Workshop,* 1–9. Dublin. URL: ftp://ftp.ims.uni-stuttgart.de/pub/corpora/tree-tagger2.pdf.

[13] Vipassana Research Institute. *Chaṭṭha Saṅgāyana Tipiṭaka* version 4.0. URL: http://www.tipitaka.org.