

# Open Source Corpus Analysis Tools for Malay

Timothy Baldwin\* and Su'ad Awab†

\* Dept. of Computer Science and Software Eng.  
University of Melbourne  
Victoria 3010 Australia

tim@csse.unimelb.edu.au

† English Language Department  
Faculty of Linguistics & Languages  
University of Malaya  
50603 Kuala Lumpur, Malaysia

suad@um.edu.my

## Abstract

Tokenisers, lemmatisers and POS taggers are vital to the linguistic and digital furtherment of any language. In this paper, we present an open source toolkit for Malay incorporating a word and sentence tokeniser, a lemmatiser and a partial POS tagger, based on heavy reuse of pre-existing language resources. We outline the software architecture of each component, and present an evaluation of each over a 26K word sample of Malay text.

## 1. Introduction

Despite rapid growth in language resource (LR) development and advances in the semi-automation of the LR development process, the sheer number of languages throughout the world means that the bulk of languages lack LRs vital to language technology research. In particular, while static LRs such as monolingual and bilingual dictionaries exist in some form for a significant proportion of the world's languages, text processing tools such as lemmatisers and part-of-speech (POS) taggers tend to exist for only a select few “resource-heavy” languages such as English, German and Japanese. There are significant opportunities, however, for leveraging preexisting LRs and text processing tools, to reduce the overhead in developing new text processing capability. It is this process of resource reuse that forms the basis of this research.

The focus of this paper is on the open source development of text processing tools for Malay, a language with a population of >200m speakers which is severely underrepresented in text processing terms. In particular, we attempt to provide a snapshot of resources available for Malay text processing, and describe the development of: (1) a Malay tokeniser based on an English tokeniser, and (2) a Malay lemmatiser based on a monolingual Malay dictionary and descriptive grammar of Malay morphology. In both cases, we employ a “recycle and reuse” strategy, in redeploying pre-existing tools and LRs for our purposes.

The original motivation for this research stems from an interest in Malay multiword expressions (MWEs: Sag et al. (2002)), focusing on their morpho-syntactic variability. In order to (1) identify MWEs in Malay corpus data, and (2) analyse the flexibility of individual MWEs, it was essential that we had access to a lemmatiser, due to the rich morphology of the Malay language (Section 2.). In addition, in order to limit such an analysis to intrasentential contexts, we needed to be able to identify sentential boundaries through sentence tokenisation. Finally, so as to be able to condition our analysis on the POS of word tokens in different contexts, we ideally required access to a POS tagger. Given the lack of pre-existing tools for Malay that provided these capabilities (Section 3.), we embarked upon a course of build-

ing a tokeniser (Section 4.) and lemmatiser (Section 5.), as described below.

## 2. Malay Morphology

Malay is an agglutinative language with rich morphology (Abdulla Hassan, 1974; Asmah Hj Omar, 1988). The three basic morphological operations are:

1. affixation
2. reduplication
3. compounding

Below, we outline the nature of each of these processes.

### 2.1. Affixation

Affixation is the most commonly used morphological process. There are three types of affixes: prefixes, suffixes and infixes.

Common prefixes include *me-*, *pe-*, *be-*, *ter-*, *se-*, *ke-* and *di-*, while common suffixes include *-i*, *-kan*, *-nya*, *-lah*, *-kah*, *-mu* and *-ku*. There are three infixes in Malay: *-el-*, *-em-* and *-er-*. Examples of infixation are *geletar* “shiver/tremble”, *gemilang* “bright” and *gerigis* “serrated”. However, infixation is not productive, and instances of infixation are generally treated as distinct word forms by lexicographers.

Affixation in Malay is highly productive and often accompanied by phonological variation in the stem. For example, the combination of the verbal prefixed *me-* with the stem *paksa* “force” produces *memaksa* “to force”.

Multiple affixation can take place in a single word, with up to four affixes as in *diperbanyakannya* “made plenty” which consists of *di-* + *per-* + *banyak* + *-kan* + *-nya* (lemma = *banyak* “a lot”).

In preparing the lemmatizer, a CD version of Kamus Dewan Edisi Ketiga (Taharin, 1996), a standard Malay dictionary published by DBP, was used.

### 2.2. Reduplication

There are three basic types of reduplication: full duplication, partial duplication, and rhyming and chiming.

Full duplication occurs in *kuda-kuda* “trestle” (from *kuda* “horse”) whilst *kekura* “tortoise” is an example of partial re-duplication (from the stem *kura* “tortoise”). Phonological changes are involved in rhyming and chiming: *lauk* “dish” becomes *lauk-pauk* “all sorts of dishes” and *kayu* “wood” combines with *kayan* “wood” to form *kayu-kayan* “different sorts of wood”.

### 2.3. Compounding

As in English, compounding fuses simplex words (either lemmas or derived forms) together into single-word compounds, possibly marked with a hyphen and possibly without any indication of the original word boundary (cf. *trade-off* vs. *tradeoff* in English). A Malay example of this is *adat-istiadat* “customs and traditions”, which constitutes a single word token but is made up of the component words *adat* “custom” and *istiadat* “custom/tradition”.

Compounds provide a valuable component of our analysis of MWEs, as, similarly to English, compounding tends to indicate a high level of lexicalisation and low level of compositionality.

## 3. Language Resources and Requirements of Malay

In this section, we review the LRs necessary for carrying out morpho-syntactic corpus analysis of Malay, including such tasks as corpus-driven MWE extraction/identification. In doing so, we both review relevant LRs currently available for Malay, and outline the peculiarities of Malay as relevant to the process of LR development in the case that no appropriate LR exists.

First, and most obviously, we require a **corpus**. Pre-existing Malay corpora of note include the Dewan Bahasa & Pustaka (DBP) Corpus consisting of 114m word tokens of taken from sources ranging from modern literature to textbooks, and the Malay Concordance Project, a corpus of 3m words of classical Malay text.<sup>1</sup> Unfortunately, the former is not publicly available while the latter has obvious limitations in studying modern Malay. Given this, we took it upon ourselves to compile a corpus of modern Malay text from web data. To date, we have collected a little over 1m words of newspaper text published over the period 2000–2005, ranging in genre from legal to technical to informal content.

Second, we require a **concordancer** to compare and contrast the token occurrences of a given word/MWE. As Malay largely mirrors English with regard to character encoding and punctuation conventions, it is possible to use preexisting concordance tools developed primarily for English.

Third, we require a **sentence tokeniser** in order to constrain MWE extraction/identification and limit our concordance windows to sentence units. As noted above, there are relatively few divergences between Malay and English punctuation conventions, with the obvious caveat that the set of commonly-used abbreviations which are marked by word-final full stops differs greatly between the two languages.

Fourth, we require a **lemmatiser** to strip off the affixes from the headword, and deal with the rich morphology of Malay (incorporating such phenomena as affixation, reduplication and compounding – see Section 2.). To illustrate, we would hope to lemmatise each of *berkena*, *berkeanaan*, *mengena*, *mengenai*, *kena-mengena*, *mengenakan*, *terkena*, *perkenakan* and *pengenaan* into the lemma *kena* “hit/be on target”. While there has been limited work on building a Malay lemmatiser (e.g. Beesley and Kartunnen (2003)), it has tended to focus on capturing particular morphological quirks of Malay rather than broad coverage. We thus had little option but to develop our own lemmatiser.

Finally, we would ideally like to have access to a **POS tagger**. Here, there are no tagged corpora of Malay and no publicly accessible POS taggers, once again forcing us to develop our own (limited) tagger.

Importantly, we are committed to sharing these tools with the linguistic and computational linguistic communities, to which end all tools have been published open source, under the terms of the GNU Lesser General Public License (LGPL). The software bundle can be downloaded from:

<http://lt.csse.unimelb.edu.au/resources/malay-toklem/>

## 4. Tokeniser

The first module in our corpus analysis toolkit is a tokeniser, which takes raw text input and: (1) segments it into word tokens (i.e. performs *word tokenisation*), and (2) identifies the sentence boundaries (i.e. performs *sentence tokenisation*). For example, given the following input:

```
Tetapi agak mengecewakan apabila
Astro menyediakan empat saluran yang
dimuatkan dengan produk beridentiti
Cina; TVBS Asia, AEC, Phoenix dan
Wah Lai Toi. Di TVBS Asia terdapat
slot berita Mid Day Headlines, EAC
dengan In E-News. Lain-lain slot
berita semasa ialah Chat Room dan
Super Sunday
```

our tokeniser would (ideally) generate the following output:

```
^ Tetapi agak mengecewakan apabila
Astro menyediakan empat saluran yang
dimuatkan dengan produk beridentiti
Cina ; TVBS Asia , AEC , Phoenix
dan Wah Lai Toi . ^ Di TVBS Asia
terdapat slot berita Mid Day
Headlines , EAC dengan In E-News
. ^ Lain-lain slot berita semasa
ialah Chat Room dan Super Sunday .
```

Word tokenisation consists primarily of inserting white space between word tokens and punctuation marks, and normalising white space. This is largely trivial, except for full stops which, in Malay as in English, are ambiguous between sentence boundaries and abbreviation markers. Sentence tokenisation provides the means to disambiguate full stops, and tag full stops which act as sentence boundaries via the insertion of a caret (^).

<sup>1</sup><http://www.anu.edu.au/asianstudies/proudfoot/MCP/Q/mcp.html>

## 4.1. Methodology

Past research on sentence tokenisation (especially for “resource-heavy” languages such as English) can be categorised as: (1) being rule-based, based primarily on stop word lists of common abbreviations (e.g. Briscoe and Carroll (2002)), (2) employing a supervised classifier (e.g. Reynar and Ratnaparkhi (1997)), or (3) employing an unsupervised classifier (e.g. Schmid (2000)). For our purposes, we opted for the rule-based approach, due to (1) the relative successes of rule-based methods for English, (2) the lack of annotated data to build a supervised classifier with, and (3) the desire to come up with a quick-turnaround solution.

The simplest and most effective means of developing a rule-based sentence tokeniser, given the high level of similarity in punctuation and case conventions between English and Malay, was to adapt a pre-existing rule-based English sentence tokeniser to the Malay language. The particular sentence tokeniser we chose for this purpose was that used within the RASP system (Briscoe and Carroll, 2002). In order to adapt this to Malay, the main change required was the replaced of the abbreviation “stop word” list for English (used, e.g., to predict that a full stop proceeding *Dr* is most probably part of the abbreviation rather than a sentence boundary) with one for Malay. Our stop word list for Malay is made up of the following common abbreviations:

Abd., Ab., Mohd., Md., Muhd., Bhd., Drs., Dt.,  
Inc., Sdn., St., Jln., Kapt., kg., kump, LL.B.,  
LL.M, Lt., per., Pn., Pt., Rp., Tmn., Tn., Tkt.,  
Tj., Y.bhg

## 4.2. Implementation

Similarly to the RASP tokeniser, we implemented our rule-based tokeniser as a command-line utility in the flex language. While the tokeniser was developed in a Linux environment, it can equally be compiled and used within Mac OS X, or alternatively the CygWin environment for Windows<sup>TM</sup>.

## 4.3. Evaluation

We evaluated the sentence tokenisation performance of our tokeniser by manually disambiguating sentence boundaries in a ~26K word subset of our corpus data. The F-score of the sentence tokeniser was found to be 99.4%. The time taken to tokenise the full 1M word corpus was around 29 seconds on a 1.7GHz Pentium M processor.

## 5. Lemmatiser

The next step in the text processing pipeline is a lemmatiser. Here, we built a system from scratch, based in part on the KAMI Malay-English lexicon (Quah et al., 2001). KAMI was originally constructed for the purposes of machine translation, and consists of approximately 68K word-forms, each of which is optionally listed with a basic part of speech, English and Chinese translation, and lemma. Due to the morphological richness of Malay, the raw lexical coverage of KAMI over our Malay corpus was 83.7% at the token level and 31.4% at the type level, underlying the importance of building our own lemmatiser.

From KAMI, we extracted: (1) all simplex words which are listed with a lemma and POS, and (2) all simplex words which are listed only with a POS. We then made various corrections and additions to these word lists based on a development dataset taken from our corpus. The total number of words with lemma and POS information is ~14K, and the total number of words with only POS information is ~31K.

## 5.1. Methodology

The core of the lemmatiser is a set of around 40 overlapping regular expressions, each of which is attuned to a specific affix type and strips off that affix and phonologically normalises the remaining string. It is important to realise that the deaffixation rules can overlap, e.g. for the input *menarik*, one possible rule would strip off the *me-* prefix and normalise the stem to *tarik* whereas another would strip off *me-* and return *narik* as the word stem.

Each deaffixation rule potentially has POS-based constraints associated with the input and output, e.g. in the case of both rules above, the input and output must both be a verb. We check that such constraints are satisfied on every rule application, and delete any paths which lead to a conflict in POS constraints. Through the use of an agenda which stores the outputs from the various rules along with the POS constraints imposed by each rule, we are able to generate a lattice tracing all possible deaffixation paths and the POS constraints associated with each.

The final step of lemmatisation is then to select the best path through the lattice. This is done by first classifying each path according to whether the final lemma hypothesis occurs in the KAMI lexicon (with or without the predicted POS), what the final POS is predicted to be, what the length (in characters) of the final hypothesised lemma is, and whether the original wordform was contained within KAMI and listed with a matching lemma. We combine these together heuristically to select the most plausible path through the lattice, and output the lemma and (optionally) POS predicted by the lemmatisation rules in question. The following is our list of heuristics, in descending order of preferability:

1. the word is listed in KAMI with a POS tag and lemma
2. the word is listed in KAMI as a functional word<sup>2</sup>
3. the word is reduplicated, and the lemma is contained in KAMI
4. after deaffixation, the lemma is predicted to be a verb
5. the word is listed in KAMI as a content word
6. after deaffixation, the lemma is predicted to be a noun or adjective

For those heuristics which draw on the KAMI lexicon, the lemma and POS tag are taken straight from the lexicon (or in the case that no lemma is listed, the word is considered

<sup>2</sup>In practical terms, functional words are non-content words, i.e. all words other than nouns, verbs and adjectives.

to be the lemma); for the deaffixation-based heuristics, the lemma and POS are the product of the final rule application.

In the instance of multiple paths being selected, we choose the path associated with the smallest lemma output (i.e. the lemma with the smallest character count), and use a pseudo-random tie-breaking mechanism if this fails to resolve the ambiguity. Note that this is often insufficient to resolve POS tag ambiguity, as it can happen that the same word/lemma is associated with multiple POS tags. Here, in the current version of the lemmatiser, we make no attempt to disambiguate the POS tag.

## 5.2. Implementation

The lemmatiser is implemented in Perl, and makes use of: (1) a word list containing (word, POS, lemma) tuples, and (2) a word list containing (word, POS) tuples (where no lemma was listed in KAMI). As with the sentence tokeniser, the lemmatiser is run from the command line.

## 5.3. Evaluation

The performance of the lemmatiser was evaluated over the same ~26K word mixed-domain sample as above, relative to hand-annotated gold-standard lemma data. The overall lemmatisation accuracy was a creditable 94.5% at the word token level, and 85.0% at the word type level. We have yet to evaluate the accuracy of the POS tag outputs, due to a lack of gold-standard data and also the fact that our POS tags are often ambiguous between multiple tags. The time taken to lemmatise and tag the full 1M word corpus was around 32 seconds on a 1.7GHz Pentium M processor.

## 6. Conclusion

We have developed an open source tokeniser, lemmatiser and (partial) POS tagger for Malay, and demonstrated the accuracy of each module over mixed-domain corpus data. In making these tools available for public use, we have provided valuable impetus to the fields of corpus and computational linguistics for Malay.

## Acknowledgements

The research in this paper was supported in part by the research collaboration between the University of Melbourne and NTT Communication Science Research Laboratories, Nippon Telegraph and Telephone Co., Ltd, and also the Australian Research Council through Special Initiative (E-Research) grant number SR0567353, “An Intelligent Search Infrastructure for Language Resources on the Web”. We also thank John Carroll for providing access to RASP, which provided a valuable reference point in writing our Malay tokeniser.

## 7. References

- Abdulla Hassan. 1974. *The Morphology of Malay*. Dewan Bahasa dan Pustaka, Kuala Lumpur, Malaysia.
- Asmah Hj Omar. 1988. *Susur Galur Bahasa Malayu*. Dewan Bahasa dan Pustaka, Kuala Lumpur, Malaysia.
- Kenneth R. Beesley and Laurie Karttunen. 2003. *Finite State Morphology*. CSLI Publications, Stanford, USA.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1499–1504, Las Palmas, Canary Islands.

Chiew Kin Quah, Francis Bond, and Takefumi Yamazaki. 2001. Design and construction of a machine-tractable Malay-English lexicon. In *Asialex 2001 Proceedings*, pages 200–205, Seoul, Korea.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proc. of the 5th Conference on Applied Natural Language Processing (ANLP)*, pages 16–9, Washington D.C., USA.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In Alexander Gelbuk, editor, *Computational Linguistics and Intelligent Text Processing: Third International Conference: CICLing-2002*, pages 1–15. Springer-Verlag, Hiedelberg/Berlin, Germany.

Helmut Schmid. 2000. Unsupervised learning of period disambiguation for tokenisation. Technical report, IMS, University of Stuttgart.

Mashitah Taharin, editor. 1996. *Kamus Dewan Edisi Ketiga*. Dewan Bahasa dan Pustaka, Kuala Lumpur, Malaysia.