SELECTING PROPER TOOLS FOR THE PROCESSING OF KOREAN, ARABIC AND INDONESIAN CORPUS

Prihantoro

Universitas Diponegoro

prihantoro2001@yahoo.com, prihantoro@undip.ac.id

Abstract

One of the fundamental issues in corpus processing begins with how the corpus processing tools can recognize the characters in the documents. This issue is considered trivial for romanized languages, which adopt the same writing system as English (such as Indonesian). While most corpus processing tools are designed to process English, or the romanized version of the documents, tools to handle languages with their own writing systems needs to be built. Consider Korean, a language with syllable block writing system, or Arabic, a language with consonant skeleton writing system. The organization of the characters in these languages is not completely concatenative like English. The description of the writing system in these languages must underlie the design of the tools. Character recognition issue can be solved and much more complex processing tasks can be performed. This paper will demonstrate AntConc, Geuljabi and Unitex and show how these tools recognize characters in these languages.

Keywords: Corpus Processing Tools, writing system, character recognition, automatic retrieval

1. Writing System

How many languages in this world are still in use? The exact number might be difficult to determine. However, *Ethnologue,* (Lewis, Simons, & Fennig, 2013) provided a brief description of 6700 languages. Although the number of languages of the world reaches thousands, some of these languages are just spoken, without written record. Some more languages are spoken and written. A question that I addressed for the latest group is how many of these languages have their own writing system. There are many languages in current use, that employ romanized characters for their writing system. Indonesian, for example, is a language spoken by almost 300 million people. However, this language does not have any writing system and prefers to employ romanized characters[[1]](#footnote-2).

Corpus processing tools that are equipped for the processing of English (and other romanized languages) are many. Can it be used to process corpora in which the data is written in non-romanized alphabet like Arabic or Korean? If not, what processing tools are present for languages with their own writing system like Arabic or Korean? This paper is an effort to answer these two questions. I, in the fourth section, will demonstrate the use of some corpus processing tools to process romanized and non-romanized languages. I believe that selecting the proper tool is a preliminary step before you go to more complex corpus processing tasks (Mc Enery & Hardie, 2012) like statistical functions, automatic retrieval, data annotation, output generation etc.

1. Corpus Processing Tasks

Many corpus-processing tools, these days, are present on internet and can be downloaded for free. They can be used offline with the data that you customize yourself. Some corpora are made online, and also equipped with facilities to perform processing tasks. But for these online corpora, we usually cannot use our own texts or corpus. The corpus is kept on the server and cannot be modified.

The simplest task that corpus-processing tools can do is word count. It measures how many words are running in the text corpus. Another task is word rank. It may rank the word from the most to least frequent. These tasks can be done with raw corpus, as they are merely statistical. Some of these tasks can also be performed, even by non-corpus tools. Consider ‘word count’ and ‘find’ features in MS Word, a word processor program. This can be done as the tasks are purely character based. For more sophisticated task, the corpus needs to be annotated.

When the corpus is annotated, much more complex tasks beyond character match can be performed. For instance, you can search the corpus on the basis of part of speech (POS), semantic, or word relation, depending on the type of annotation present in the corpus.

Many researches focus on developing English corpus processing tools. This is because English is widely used around the world. The tool can also be used to process the corpus in different language, in condition that it employs romanized characters in the text. However, the function will be limited to only character-match based task (frequency count, word and character count).

1. Challenges For Non Romanized Languages

Most corpus processing tools are designed for corpus where the texts are written by using roman characters. However, some tasks that are considered simple like word count or word frequency might be quite challenging when it comes to the processing of languages with non-romanized characters, as usually the default language of computers is English. We can overcome this by selecting the proper encoding. But the problem is not all corpus processing tools have the same encoding system.

Character recognition for corpus with romanized characters like English and Indonesian is a simple processing task. However, it is a quite challenging issue for non-romanized language like Arabic or Korean. When you process corpus Korean or Arabic corpus with English/romanized characters based corpus-processing tool, then you might see strange symbols as in Figure 1:

Figure 1. The failure of English Module to Recognize Korean Script



The problem is because the two languages differ in terms of not only the characters but also the writing systems. English, which happen to be written in roman characters, is concatenative: each character is arranged horizontally by attaching one to another from left to right. Korean, on the other hand is written by using syllable block system. Each block is a syllable, where it is composed of at least 2 to maximum 4 characters. The characters are not arranged concatenatively.

**Figure 2. Script Syllabification of <친구> [chin.gu] in Korean VS The Concatenation of**

**Friend in English and *teman* in Indonesian**

|  |  |
| --- | --- |
| 친구 **[chin.gu]** |  |
| **[chin]** | [**gu]** |
|  |  |
| **ㅊ** [ch] | **ㅣ**[i] |  | **ㄱ** [g] |
| **ㄴ**[n] | **ㅜ** [u] |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t | e | m | a | n |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| f | r | i  | e | n | d |
|  |

Each syllable in Korean script is composed uniquely. The token <친구> [chin.gu] is composed of two syllable**.** By referring to figure two, you are also aware that Korean script is composed both horizontally and vertically (the first syllable reads from left-right-up-down, while the second syllable reads up-down). Consider the difference with Indonesian and English. The Indonesian and English writing system is the same, using roman alphabets and concatenated from left to right.

What has been said about Arabic writing system? It reads from right to left rather than left to right. It also employs diacritics that are useful as vowel marker. In arabic orthography, words are composed of consonants skeleton, where the vowels are represented by diacritics. Even though diacritic is not always necessary as most arabic speakers are quite aware of the meaning just by the visual of the roots (the consonant skeleton). Consider figure three that shows the description of a consonantal skeleton [bis.mi]:

**Figure 3. The Consonantal Skeleton in Arabic for [bismi]**

[m] [s] [b]



[i] [i]

Speakers of arabic scripts (or at least people who have competence to read the script) are aware that the main character is actually the consonant and the vowels are reflected by diacritics. The consonant is obligatory while the vowels (marked by the diacritic) is optional. Often the diacritic (shown by broken line) is removed but speakers of arabic can still understand the meaning. In figure three, the consonant skeleton is [bsm] and inflected by two diacritics for vowel [i] attached on [b] and [m].

In this paper, I compare the processing of corpora. The can be categorized based on the the writing system (romanized VS non-romanized), and the language (Indonesian, Arabic, Korean). I will process these corpora by two kinds of tool. The first one is one-language platform (designed only for one language) and the second one is multi-languages tool (designed for more than one language).

1. Selecting The Proper Tool

*One-Language Platform*

Antconc (Anthony, 2006) is a tool written by Laurence Anthony for the purpose of English corpus processing. It is a very light, but very powerful in processing raw corpus. For new corpus users, it is recommended to work with this tool before proceeding further to more advanced tool. As a light tool, Antconc has many functionalities. Retrieval in Antconc can be performed by target word or regular expression[[2]](#footnote-3). It may retrieve single or multiword expression. The way Antconc presents the finding in the corpus is also interesting. It can display the result in standard concordance lines. Another way to display the result is by creating dispersion plot so that the users will understand the density of the target words with reference to the corpus texts. Antconc is also equipped with basic statistic collocation count.

Figure 4. Concordance Line with Target Word: Money



However, this tool can also be used to process languages with romanized characters, like Indonesian. The presence of this tool is actually useful for languages that do not have specific corpus processing tool yet. Corpus research for these languages can begin without having to wait for the creation of specific corpus processing tools.

Figure 5. Concordance Line with Target Word in Indonesian: *uang* ‘money’



This corpus-processing tool, however, cannot process corpus written in non-romanized languages like Arabic or Korean. Error is most likely to happen when you push this kind of tool to process documents with the characters that it does not recognize, it might result on strange symbols as shown in figure 6, where the original text can be seen in figure 7:

Figure 6. Using English Module to Recognize Arabic Text



Figure 7. Original Arabic Text



 The same thing will happen if this tool is forced to process Korean texts, as they are written in non-romanized characters. What users can do in this point is to seek a specific corpus processing tool for the language. Now, consider Geuljabi[[3]](#footnote-4), a corpus processing tool dedicated for the processing of *Hangeul* (Korean characters).

Figure 8. Geuljabi Display



 Geuljabi can process several texts at once. It also has basic functionalities as Antconc: basic concordance, text retrieval, statistical tasks and etc. The advantage of Geuljabi, besides its module can recognize Korean characters, is its ability to tag or annotate the text. The annotation is very useful as users may also search on the basis on linguistic information (not only character-based match):

Figure 9. Search 명 as a Dependent Noun



The format of the search is as follow: target token\_<word class>. If your search is just character based, this space can be left empty. Figure 10, shows the retrieval of target word명as a dependent noun. The specification of word class is important as one token might be the representation of several word class in isolation, but in sentence context they fall only to one class. See the result in figure 10:

Figure 10. Concordance for the search of 명 as a Dependent Noun



 The result of the retrieval shows that the target word is almost always accompanied by a number (digital numeric) in every line. The target word명in Korean is a numeral classifier. As it name suggests it has to co-occur with numeral, in either digital numeric or Arabic numeral. Although retrieval in this annotated corpus consume some more memory, but users have the benefit of performing linguistic information based retrieval. Can *Geuljabi* be used to process Indonesian corpus? The answer is yes, to some extent. However, it cannot perform annotation for Indonesian texts as the lexical resource is not designed for Indonesian lexicon. Basic corpus processing tasks, the character based match, however, can still be performed.

*Multi-Languages Platform*

 Tools with multi languages platform can process corpora in different language. Besides some basic processing tasks, it also has the ability to do advance processing tasks with annotated corpus. One of the examples is Unitex[[4]](#footnote-5) (Paumier, 2008). Unitex is a multi-language platform corpus-processing tool. Before you use Unitex on your corpus, it requires you to select the language of your corpus. See figure 11:

Figure 11. Language Selection in Unitex



 When you choose one language, Unitex will refer to the language resources (lexical resource, language characters, sentence separator, tokenizer and etc). The resources for each language are unique. For instance, the character recognition for Arabic is different from Korean. Using Korean module to recognize Arabic, or the other way around, will cause errors to happen. Now consider Arabic inflection module in figure 12. The mode is called Semitic mode:

Figure 12. A Toy Semitic Inflection Grammar



 Lemma in Arabic is organized by consonantal skeleton, where the vowels are used to fill this skeleton (Paumier, 2008). The skeleton for inflection grammar in figure 12 is represented by number <123>. The lexical resource is as follow:

yakotobu,ktb.V:aP3ms

In this entry lines, the grammar code is <V:aP3ms>, where the root is <ktb> and the inflected form is <yakotobu>. The root might take different inflection that results on different word forms such as <kitab>, <kutiba> and etc. One research of Arabic by using Unitex was conducted by Traboulsi (2009). He managed to extract named entities (proper noun) in Arabic by using local grammar graphs (LGGs)[[5]](#footnote-6) in Unitex. Figure xxs shows the extraction of organization name in Arabic:

Figure 13. Extracting Arabic Organization Name



Unlike Arabic, Korean writing system obeys different logic. Korean is agglutinative language, where the characters are composed in syllabic block style. Unitex complies with this logic. Instead of recognizing characters concatenatively, it arranges the characters in syllable block style (see the upper part of figure 14). Beyond simple corpus processing task, Unitex can also perform morphological analysis on Korean text as shown by figure 15:

Figure 14. Syllable Block Writing as Opposed to Concatenative Writing



Figure 15. Sentence Automaton: Morphological Analysis of Korean



Figure 15 shows a sentence automation where the verb is morphologically analyzed (Paumier, 2008). The verb in green graph is the surface form, while the parallel three parts (see the dotted lines) is the basic forms by morphological analysis. The first part <V+i> is the main verb, where 오다 is the main verb. The second part 어 <pass+Morph> is the past tense morpheme, while the 습니다 <st+dec+hso> is the politeness marker.

One of the advantages of Unitex is that users can customize their own resources. One research by using self customized resource on Unitex was about the automatic conversion of archaic English to modern English (Prihantoro, 2011). See figure 16:

Figure 16. Input (archaic english) – Output (modern English) Concordance



 Up to now, Unitex does not have official Indonesian language module. However, (Prihantoro, 2011) has designed a small size resource as a prototype to show that Unitex can be used to process Indonesian corpus. He designed a grammar to recognized morphophonemic variants of the derived forms. By using a grammar code in Unitex, the derivation of a lemma can be obtained. See the retrieval of derivations of lemma *bor* ‘to drill’:

Figure 17. Derivation of Lemma *bor* ‘to drill’



The grammar has managed to retrieve the derivational verbs *dibor* (passive), *mengebor* (active) and noun *bor*. This research has shown that the prototype must be developed, and content of the Indonesian language resources (especially the lexical resource) in Unitex must be enrich to perform standard Indonesian corpus processing.

1. Conclusion

How to select the proper tool to process your corpus? First is to understand a tool that has the ability to process your corpus (depending on the language). Second is to find tools that suit your corpus processing purpose. If your processing tasks are considered basic like frequency count, character based retrieval and simple concordance. However, if you want some more advanced tasks that require the annotation of the corpus, then you will need to find tools with resources to perform annotation function. The third is to find tools that support your computing skills. Some tools do not require computing skills at all, but some tools do.

 One challenge in corpus linguistics is how to create tools that are user-friendly, even for users with zero computational skills. I have to admit that there are several corpus-processing tools dedicated for the processing of Indonesian created by computer scientists, such as Morphind (Larasati, 2012) and (Wicaksono & Purwarianti, 2010). However, people with no prior computing skills might have difficulties in operating these tools. On my personal evaluation, the three software that I have briefly presented, AntConc, Geuljabi, and Unitex, are ‘quite’ friendly to the users. The ‘click’, ‘drag’, and ‘drop’ methods employed by the three tools are considered less complex than the syntax of coding or scripting, which just fits for linguists with no computational background.

REFERENCES

Anthony, L. (2006). Concordancing with AntConc: An introduction to tools and techniques in corpus linguistics. JACET Newsletter , 155-185.

Gross, M. (1997). The Construction of Local Grammars. In E. Roche, & Y. Schabes, Finite State Language Processing (pp. 329-354). Massachusetts: MIT Press.

Larasati, S.-D. (2012). IDENTIC CORPUS: Morphologically Enriched Indonesian-English Parallel Corpus. LREC , 902-906.

Lewis, M.-P., Simons, G.-F., & Fennig, C.-D. (2013). Ethnologue: Languages of the World, Seventeenth edition. Dallas, Texas: SIL International.

Mc Enery, T., & Hardie, A. (2012). Corpus Linguistics: Method, Theory and Practice. Cambridge: Cambridge University Press.

Paumier, S. (2008). Unitex Manual. Paris: Universite Paris Est Marne La Valee & LADL.

Prihantoro. (2011b). Local Grammar Based Auto Prefixing Model for Automatic Extraction in Indonesian Corpus (Focus on Prefix MeN-). Proceedings of International Congress of Indonesian Linguists Society (KIMLI) (pp. 32-38). Bandung: Universitas Pendidikan Indonesia Press.

Prihantoro. (2011). Transducer for Auto-Convert of Archaic to Present Day English: A Support for Computer Assisted Language Learning. Proceeding for The 16th English in Southeast Asia Conference. Yogyakarta: Sanata Dharma University Press.

Traboulsi, H. (2009). Arabic Named Entity Extraction: A Local Grammar Based Approach. Proceedings of the International Multiconference Science and Information Technology (pp. 139-143). Mragowo, Poland: Polish Information Processing Society.

Wicaksono, A.-F., & Purwarianti, A. (2010). HMM Based Part-of-Speech Tagger for Bahasa Indonesia. Proceeding of the Fourth International MALINDO Workshop (MALINDO2010). Jakarta: Malindo Press.

1. It is interesting as there are some vernacular languages in Indonesia, like Lampungnese, Javanese, Balinese have their own writing system. Unfortunately, they are not widely used in daily communication. There are some columns in the local newspaper specially dedicated to the preservation of the language. However, not many use the vernacular writing system. Instead, romanized characters are used. On one hand, the use of romanized characters these days may reach wider coverage. But on the other hand, the use of romanized characters have several shortcomings. Writing system often reflects the phonetics and the phonology of the language, and roman characters do not always conform to the phonetics and the phonology of the language. [↑](#footnote-ref-2)
2. Machine readable notation to perform automatic retrieval in computer text [↑](#footnote-ref-3)
3. http://builder.hufs.ac.kr/user/boardList.action?command=view&siteId=dicora&boardId=4938951&page=1&boardSeq=7915847&search=&column=&categoryId=&categoryDepth=&parent= [↑](#footnote-ref-4)
4. http://www-igm.univ-mlv.fr/~unitex/ [↑](#footnote-ref-5)
5. Local Grammar is an abstract machine to perform natural language processing task [↑](#footnote-ref-6)