Demonstration of ChaKi.NET – beyond the corpus search system

Masayuki ASAHARA♢ Yuji MATSUMOTO♣ Toshio MORITA ♠
♢ National Institute for Japanese Language and Linguistics,
National Institutes for the Humanities, Japan
♣ Nara Institute of Science and Technology, Japan.
♠ Sowa Research Co., Ltd.

Abstract

ChaKi.NET is a corpus management system for dependency structure annotated corpora. After more than 10 years of continuous development, the system is now usable not only for corpus search, but also for visualization, annotation, labelling, and formatting for statistical analysis. This paper describes the various functions included in the current ChaKi.NET system.

1 Introduction

The corpus management tool ChaKi[1] (Matsumoto et al., 2005) was originally released in 2004. In version 3.0, the user interface was rewritten using the .NET framework, and was renamed ChaKi.NET.

The system was originally created as a corpus search system for dependency-analysed Japanese corpora. The String Search, Tag Search, and Dependency Search functions can be used to search dependency-parsed corpora at the string, POS-tag and dependency structure levels. A dependency-parsed corpus is converted into an SQLite DB file or stored on a MySQL server. In the case of SQLite DB files, corpus database files are shared by simply copying them to a new system. The system has been enhanced continuously and used for other purposes such as corpus visualization, annotation, labelling, and formatting for statistical analysis. In this paper, we present these functions of ChaKi.NET.

2 Visualization

2.1 Visualization of Dependency Tree

ChaKi.NET was originally developed as the viewer for the output of a dependency analyser named CaboCha [2]. Figures 1 and 2 show the diagonal and horizontal visualization modes, respectively. The extended CaboCha format and CoNLL-X format[3] can be imported into ChaKi.NET. The Japanese examples are from the BCCWJ-DepPara syntactic dependency and coordinate structure annotation of the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa et al., 2014). Because Japanese is a strictly head final language, the diagonal mode is often used for annotation. The lower panel of Figure 2 shows a Universal Dependency tree (German). In the ACL community, the direction of dependency relation arrows is from head to dependent. However, in the Japanese NLP community, we prefer the one from dependent to head, regarding the dependency relation as the modification relation. In ChaKi.NET, the direction of arrows can be specified by the user.

2.2 Visualization of SEGMENT, LINK, and GROUP

We believe that most annotations on text corpora can be abstracted into the following three types: SEGMENT, LINK, and GROUP. SEGMENTs are regions in a sentence such as phrases and named entities. LINKs are directed relations between two SEGMENTs; these can indicate syntactic dependency, semantic dependency (predicate argument relation), and temporal relationships between two events. GROUPs are equivalence classes determined by an equivalent relation between SEGMENTs; these include coordinate structures and coreferences.

[1] `https://en.osdn.jp/projects/chaki/releases/`
[2] `https://taku910.github.io/cabocha/`
[3] `http://ilk.uvt.nl/conll/#dataformat`

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 49–53, Osaka, Japan, December 11-17 2016.

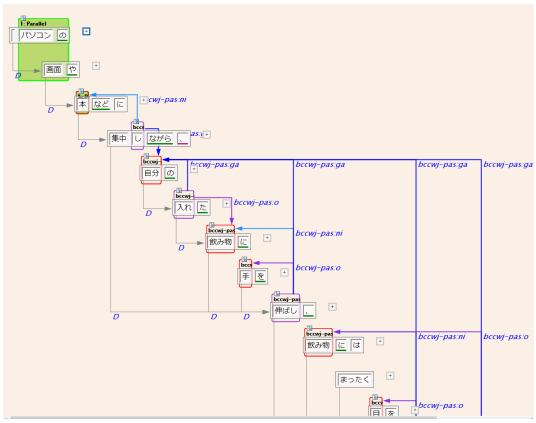
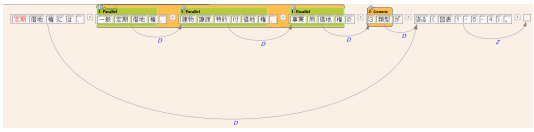
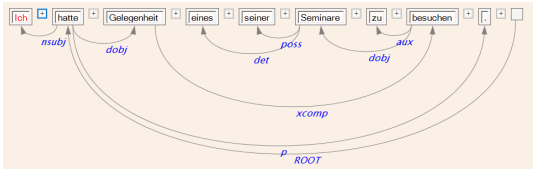


Figure 1: Diagonal mode for dependency tree visualization



Japanese Example: arrow direction is from dependent to head



German Example: arrow direction is from head to dependent

Figure 2: Horizontal mode for dependency tree visualization

Figure 3 shows a visualization example from BCCWJ-DepParaPAS, i.e., the dependency structure with predicate-argument relations and coreference relations for BCCWJ. A thick blue arrow denotes a 'ga' relation, which is a subject-predicate relation. A thick purple arrow denotes an 'o' relation, which is an object-predicate relation.

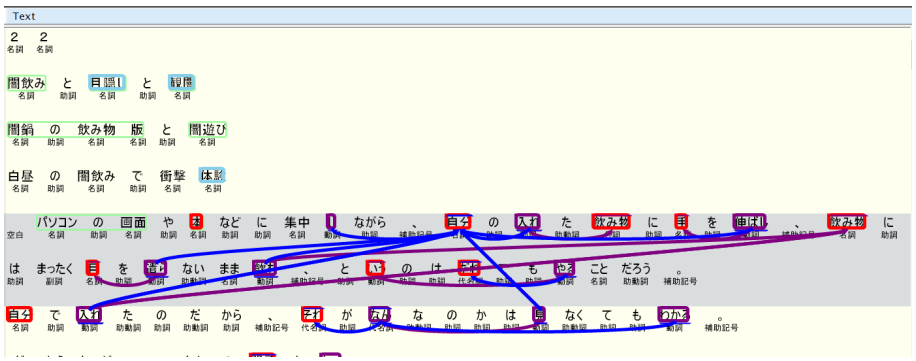


Figure 3: Text mode for visualization of predicate argument structure

2.3 Visualization of Projection

ChaKi.NET can import more than one corpus with alignment information into one database. We refer to this as 'projection' from one image to another in relational algebra. Parallel corpora can be visualized using the projection function. The aligned words are highlighted by colours.

The projection functions can be used for various types of parallel analysis:

- Word segmentation variation (BCCWJ): `https://youtu.be/L-Arl9oDUm8`
  BCCWJ includes two units – 'Short Unit Word' and 'Long Unit Word' (Figure 4).
- The Japanese-English parallel corpus (BCCWJ-Trans): `https://youtu.be/SZL8P5_Z-Xg`
  The corpus is word aligned (Figure 5).
- Dialect (Kitakyushu, Fukuoka, Japan) and standard Japanese: `https://youtu.be/_b1zLHMK_i8`
  The dialect corpus is Bunsetsu-segmented with Katanaka transcription. The dialect is translated into standard Japanese, which is POS tagged and dependency parsed (Figure 6).

We also plan to use this function with a historical Japanese corpus containing translations into contemporary Japanese.

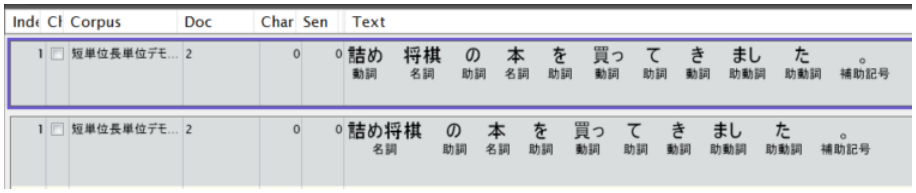


Figure 4: Visualization of two word segmentation standards

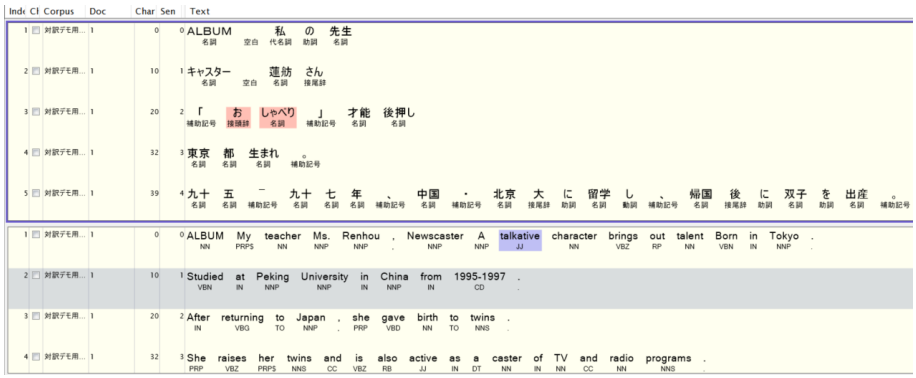


Figure 5: Visualization of Japanese-English parallel corpus

2.4 Visualization of Time

ChaKi.NET can store the start time, end time, and duration of words or morphemes for speech transcription corpora. The demo for 'Corpus of Spontaneous Japanese' (CSJ) (Maekawa et al., 2000) can be accessed at `https://youtu.be/Qod6J14X9mU`.

2.5 Combination of Projection and Time

The BCCWJ EyeTracking Corpus (Asahara et al., 2016) contains the reading time data of 24 experiment subjects, obtained from BCCWJ samples. We can define two word orders – the reading order of the subject and the word order in the original text. For the former order, we can define the start time, end time, and duration. For the latter order, reading time is aggregated into the following three duration types: first pass duration, regression path duration, and total duration. First pass duration is the time spent in a word region before moving on or looking back. Regression path duration is the time from

Figure 6: Visualization of dialect and standard Japanese parallel corpus

the time that the eye first enters a word region until the time it moves beyond that region, and includes regression time. Total duration is the sum of all fixations in a word region. Figure 7 shows a visualization of the BCCWJ EyeTracking Corpus. The demo for the BCCWJ EyeTracking Corpus can be viewed at `https://youtu.be/H2ySz09n_sA`.

Figure 7: Visualization of BCCWJ EyeTracking Corpus

3 Annotation and Labelling

3.1 Annotation

ChaKi.NET can call a morphological analyser (MeCab) [4] and a Japanese dependency analyser (CaboCha); this functionality is invoked when a user drags and drops a text file onto ChaKi.NET's menu bar. The word segmentation and POS tags of the analyser output can be corrected by a morpheme panel.

Using a mouse operation, the dependency structure can be modified via the dependency tree panels shown in Figures 1 and 2. SEGMENT, LINK, and GROUP are also modified using the panels.

3.2 Labelling

The corpus search functions (query) can define the patterns of strings, sequences of morphological information, and subtrees of dependencies. The search results can be exported into a Microsoft Excel spreadsheet or CSV file. However, we occasionally need to annotate a label to the searched results.

On the Scripting Panel, we can use Ruby or Python code to execute a labelling action based on the pattern of the query. We can use set of predefined scripts, or write any specific purpose code. The following sample Ruby code assigns the label 'NE' to a region:

Ruby code to assign label 'NE' `CreateSegmentAll.rb`

```
...
records.each do |r|
   svc.Open(corpus, s, nil)
   ...
   c = r.GetCenterCharOffset()
   w = r.GetCenterCharLength()
   svc.SetupProject(0)
   svc.CreateSegment(c, c+w, "NE")
   svc.Commit()
   ...
end
```

The `CreateSegment(startPos, endPos, tagName)` method assigns the label `tagName` to the region between the `startPos` and `endPos-1`. The leftmost offset of the matched pattern can be obtained by the `GetCenterCharOffset` method. The rightmost offset is calculated from the length of the matched pattern given by the `GetCenterCharLength` method.

We perform the following cycle (Figure 8) to assign labels to the corpus. ChaKi.NET enables us to perform this cycle via mouse clicks on the user interface.

[4] `http://taku910.github.io/mecab/`

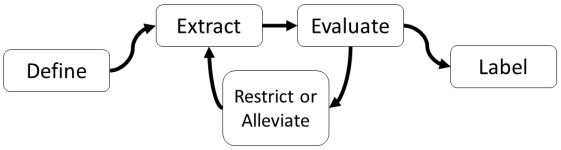


Figure 8: Labeling cycle

1. Define: define the query pattern

2-a. Extract: extract the matched examples

2-b. Evaluate: evaluate the matched examples

2-c. Restrict or Relax: restrict or relax the pattern

3. Label: assign labels to the matched examples

4 Statistical Analysis Aids

The original Collocation functions of ChaKi.NET can extract collocations using various frequencies or statistics, including co-occurrence frequency, MI score, various cooccurrence measures, [5] and N-gram frequency[6].

We can compose a term document matrix without writing a program code by using Word List functions. The demo `https://youtu.be/yWE0z-bd5ME` shows the output of the term document matrix. The original data is from BCCWJ-SUMM, which is a BCCWJ-based summarization corpus containing data from more than one hundred experimental participants.

The matrix can be exported as a Microsoft Excel spreadsheet or R data frame file.

When we define a query of word sequences using a tag search, we can also extract an n-gram/p-mer document matrix (Demo: `https://youtu.be/Ossr5if8cKI`). When we define a subtree query using a dependency search, we can also extract a dependency subtree document matrix (Demo: `https://youtu.be/XwJNEBEzcBw`).

5 Summary and Future Directions

We presented newly installed ChaKi.NET functions. The software is free for any purpose, including commercial use. We hold tutorials of the system periodically in Japan. The copyright-free data for ChaKi.NET can be downloaded from `http://chaki-data.ninjal.ac.jp/`. The BCCWJ-related data can be downloaded from `https://bccwj-data.ninjal.ac.jp/mdl/`. In our future work, we plan to develop new corpus query functions for any annotation, including SEGMENT, LINK, and GROUP.

Acknowledgments

The work reported in this article was supported by the NINJAL research project of the Center for Corpus Development.

References

Masayuki Asahara, Hajime Ono, and Edson T. Miyamoto. 2016. Reading-time annotations for the "Balanced Corpus of Contemporary Written Japanese". In *Proc. of COLING-2016*.

Kikuo Maekawa, Hanae Koiso, Sadaoki Furui, and Hitoshi Isahara. 2000. Spontaneous speech corpus of Japanese. *Proc. LREC2000 (Second International Conference on Language Resources and Evaluation)*, 2:947–952.

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. 2014. Balanced Corpus of Contemporary Written Japanese. *Language Resources and Evaluation*, 48:345–371.

Yuji Matsumoto, Masayuki Asahara, Kou Kawabe, Yurika Takahashi, Yukio Tono, Akira Otani, and Toshio Morita. 2005. ChaKi: An Annotated Corpora Management and Search System. In *Proceedings from the Corpus Linguistics Conference Series*.

[5] MI score, MI3 score, Dice score, log-log score, and Z score.

[6] Also requires a sequence pattern mining algorithm.