A Japanese Natural Language Toolset Implementation for ConceptNet

Tyson Roberts, Rafal Rzepka, Kenji Araki

Language Media Laboratory, Graduate School of Information Science and Technology, Hokkaido University
Kita-ku Kita 14, Nishi 9, 060-0814 Sapporo, Japan
{nall, kabura, araki}@media.eng.hokudai.ac.jp

Abstract

In recent years, ConceptNet has gained notoriety in the Natural Language Processing (NLP) as a textual commonsense knowledge base (CSKB) for its utilization of k-lines (Liu and Sing, 2004a) which make it suitable for making practical inferences on corpora (Liu and Sing, 2004b).

However, until now, ConceptNet has lacked support for many non-English languages. To alleviate this problem, we have implemented a software toolset for the Japanese Language that allows Japanese to be used with ConceptNet's concept inference system. This paper discusses the implementation of this toolset and a possible path for the development of toolsets in other languages with similar features.

Introduction

ConceptNet is a freely available commonsense knowledge base (CSKB) currently in its 4th major revision (referred to as ConceptNet4). The English version contains over one million sentences mined from the Open Mind Commonsense (OMC) corpus (Liu and Sing, 2004a). From these sentences, over 300,000 elements of common sense (called "concepts") were extracted (Liu and Sing, 2004b).

While English is well supported at this point, support for other languages is incomplete or non-existent. After English, the next language, Traditional Chinese, contains only 64,000 concepts. Japanese is much further down the list, containing only 9,527 concepts.

Natural Language Interface

In order to accommodate other languages, ConceptNet is broken up into several modules, one of which is the Natural Language Interface. This interface abstracts language features away from the other modules in ConceptNet and allows arbitrary languages to be implemented without modifying other modules. To fulfill the language interface, a set of language tools was created.

Language Tools Implementation

The language tools implementation for Japanese uses a custom B-tree to represent utterances. The top-level node is a special utterance object, with chunk objects forming the second layer, and word objects forming all lower branches. Token objects are used as leaves.

The tree is built using data from CaboCha, a Japanese Dependency Structure Analyzer[1], itself built upon MeCab, a Japanese Part-of-Speech / Morphological Analyzer[2].

The tree structure is built in 5 stages:

Stage 1: The input string is sanitized. First, all input is converted to UTF-8. All numeric digits are converted to full-width digits in Unicode range [0xefbc91, 0xefbc9a]. Finally, other punctuation is converted to their corresponding full-width representation.

Stage 2: The input string is analyzed by CaboCha using the "-f1 -n1" option string to retrieve information in lattice format. From this analysis, chunk and token information is extracted and used to build the skeleton tree structure in a bottom-up manner.

Stage 3: Tokens are combined into word structures by applying operators to the tokens in each chunk. At the end of stage 3, chunk objects contain only word-derived objects as direct children.

Operators:
Op 1. Wrap nouns and noun suffixes into single noun objects so that compound words can be recognized.
Op 2. Wrap adjectives in the *nai*/**negative** inflection form into adjective objects to prevent *nai*/**negative** from being confused as an independent auxiliary verb.

[1] http://chasen.org/~taku/software/cabocha/
[2] http://mecab.sourceforge.net/

Op 3. Wrap nouns followed by the na/**conjuctive particle** into adjective objects to prevent them from being treated as simple nouns.
Op 4. Wrap all forms of *dearu*/**to be** into special "to be" objects to normalize them.
Op 5. Wrap adjacent particles into single particle objects: *dewa*, *deno*, *eno*, *niwa*, *nowa node*, *towa*, *tono*, *demo*, *dewa* to allow compound particle recognition.
Op 6. Wrap all forms of the copula, *desu* into special "to be" objects to normalize them.
Op 7. Wrap verbs and inflections into verb objects to unify access between verbs.
Op 8. Wrap adjectives and inflections into adjective objects to unify access between verbs.
Op 9. Wrap all remaining tokens into word objects according to the token type to unify access.

Stage 4: Operators are applied to word objects to further modify them into more complex objects.

Operators:
Op 1. Wrap *koto* and *mono* particles into special objects when used to nominalize verbs to allow them to be identified as stopwords.
Op 2. Wrap verbal nouns (noun + *suru*/**to do**) into special verbalNoun objects to unify access.
Op 3. Wrap adjectival verbs (adjective + *suru*/**to do** and adjective + *naru*/**to become**) to unify access.

Stage 5: Final operators are applied to do cleanup.

Operators:
Op 1. Change the base form of the token in the verb ending *masen*/**negative polite inflection** from *n* to *nai* to normalize it.
Op 2. Change the *da*/**copula** to *de*/**location particle** in certain cases where it is reported abnormally.

Natural Language Interface Implementation

The Natural Language Interface has 7 required software interfaces and 2 optional ones as shown below.

Required:
is_stopword: Returns whether or not a word is a stopword
stem_word: Returns an inflectionless version of a word
word_split: Splits a word into normalized stem/inflections
normalize: Returns the normalized form of a word
lemma_split: Splits lemmas of a sentence into two strings: one with normalized lemmas, and the second containing stopwords and inflections with placeholders for lemmas
tokenize: Returns surface forms separated by spaces
untokenize: Reverses the tokenize operation

Optional:
lemma_combine: Reverses the lemma_split operation
is_blacklisted: Returns if a word is explicitly disallowed

The Japanese interfaces were implemented by the application of 4 lower-level operations, as follows:

Stopword Detection is performed on a per word object basis. Punctuation, sentence-level particles, adverbs, all forms of *dearu*/**to be** including the copula, and the special case of nominalizing *koto* and *mono* are treated as stopwords.

Inflection Normalization is performed only on verbs and adjectives in ConceptNet. Each inflection is normalized to its canonical form and output inflections are prepended with a *nyoro*/**full-width tilde**.

Tokenization is performed by a simple concatenation of the surface forms of each word in an utterance separated by a space character (Unicode 0x0020).

Lemmatization removes all inflections from a word and outputs its canonical form. There are two exceptions: the inflections *~nai*/**negative inflection** and *~tai*/**desiderative mood indicator** are not removed, but instead appended directly to the verb or adjective's canonical form to preserve polarity and mood during ConceptNet concept extraction.

Future Work

With the natural language toolset complete, work on refining the toolset will continue, possibly by splitting the toolset into its own project independent of ConceptNet.

The direction of forward work includes creation of bilingual assertions within the ConceptNet framework with the goal creating a multilingual ontology applicable to Machine Translation and other fields.

To this end, work has begun on resolving the discrepancy of source sentences between the English and Japanese versions of ConceptNet through text mining in Japanese Wikipedia.

Acknowledgements

The authors would like to express their sincere thanks to Rob Speer of the MIT Media Lab, who was instrumental to the undertaking of this project.

References

Liu, H., Singh, P. 2004. ConceptNet - A Practical Commonsense Reasoning Tool-kit. *BT Technology Journal*, Vol. 22 No 4, October 2004a: 211-227

Liu, H., and Singh, P. 2004b. Commonsense Reasoning in and over Natural Language. *Knowledge-Based Intelligent Information and Engineering Systems* Vol. 3215: 293-306