BKParser 1.0

1. Introduction

BKParser is a tool for part-of-speech tagging and dependency parsing for Vietnamese. It implemented the paper

Kiem-Hieu Nguyen. 2018. "*BKTreebank: Building a Vietnamese Dependency Treebank*" LREC 2018

It uses machine learning approach, with CRFSuite for POS tagging and MaltParser (Nivre algorithm) for dependency parsing.
Training data are from BKTreebank.
BKParser uses UETSegmenter for word segmentation.
BKParser was written in Java.

2. Contents

BKParser-1.0.jar:	Executable file, also used to import as library into your project.
Models:	Containing models for UETSegmenter and BKParser.
Dictionary:	Containing dictionary for UETSegmenter and BKParser.

3. Usage:

For command-line usage, you need JDK 1.8 (or higher) with following arguments:

java -jar BKParser-1.0.jar -r <operator> {additional arguments}

in which:

-r: operator (tag|parse)

There are two ways for additional arguments:

<u>1) Tagging or parsing an input text entered by user</u>

-t <text>

-t: input text.

Example:

```
java -jar BKParser-1.0.jar -r tag -t "Hôm nay, tôi đi học."
java -jar BKParser-1.0.jar -r parse -t "Hôm nay, tôi đi học."
```

<u>2) Tagging or parsing a text from input file and writing to an ouput file</u>

-i <inputPath> -o <outputPath>

-i: path to input file

-o: path to output file

Example:

```
java -jar BKParser-1.0.jar -r tag -i /home/user/input.txt -o /home/user/output.txt
java -jar BKParser-1.0.jar -r parse -i /home/user/input.txt -o /home/user/output.txt
```

4. APIs:

To use APIs, you need to import BKParser-1.0.jar into your project (BKParser already includes UETSegmenter), and copy ‘dictionary’ and ‘models’ directories into your project. They contain necessary dictionaries and models for word segmentation and dependency parsing.

4.1. Contructors:

BKParser could be initialized in two ways:

<u>1) No argument</u>

This way will load UETSegmenter for word segmentation. Therefore, initialization will be slower. This is used when input text has not been segmented yet.

Syntax:

```
BKParser bkParser = new BKParser();
```

2) With loadUETSegmenter

When loadUETSegmenter = true, it is similar to initialization with no argument.
When loadUETSegmenter = false, BKParser will not load UETSegmenter. Initialization speed will be faster. This is only when input texts are already segmented.

Syntax:

```
BKParser bkParser = new BKParser(boolean loadUETSegmenter);
(loadUETSegmenter = true/false)
```

4.2. APIs:

Three APIs are provided:

a) Word segmentation

This method takes as input a text, executes sentence detection and word segmentation, and returns a list of segmented sentences.

Code sample:

```
String text = "Hôm nay là sáng chủ nhật. Mọi người chuẩn bị lên đường nhé.";
List<String> sentences = parser.segment(text);
```

b) Tagging or parsing a text

It takes as input a text, executes sentence detection, word segmentation, then executes tagging and/or parsing.

Code sample:

```
String text = "Hôm nay, tôi đi học. Ngày mai, tôi đi làm ";
List<List<CONLLToken>> tagResult = bkParser.tag(text);
List<List<CONLLToken>> parseResult = bkParser.parse(text);
```

c) Tagging or parsing from file

It takes as input a path, reads text from file indicated by the path, executes sentence detection and word segmentation, and tags or parses the text

Code sample

```
String path = "/home/user/input.txt";
List<List<CONLLToken>> tagResult = bkParser.tagFile(path);
List<List<CONLLToken>> parseResult = bkParser.parseFile(path);
```

d) Tagging or parsing for a list of sentences

There are two arguments: the first is a list of sentences; the second is a boolean variable isSegmented indicating whether the sentences have been segmented or not. If isSegmented = false, executes word segmentation on the sentences. If isSegmented = true, ignores word segmentation. Then tags or parses the sentences. This API is useful for texts segmented by human annotators or a segmenter other than UETSegmenter.

Code sample:

```
BKParser bkParser = new BKParser(false);
String sentence1 = "Hôm_nay , tôi đi học .";
String sentence2 = "Tất_cả mọi người đều rất vui .";
List<String> sentences = new ArrayList<String>();
sentences.add(sentence1);
sentences.add(sentence2);
List<List<CONLLToken>> tagResult = bkParser.tag(sentences, true);
List<List<CONLLToken>> parseResult = bkParser.parse(sentences, true);
```

Output format:

All the outputs are in List<List<CONLLToken>>.
Each CONLLToken is equivalent to a token in CONLL-U format.
(More details at http://universaldependencies.org/format.html)

Each List<CONLLToken> is a sentence.
Each List<List<CONLLToken>> is a list of sentences.

5. Used libraries:

- *UETSegmenter*: Word segmenter for Vietnamese
  Link: https://github.com/phongnt570/UETsegmenter
- *CRFSuite*: CRF tool.
  Link:
  http://www.chokkan.org/software/crfsuite/
  https://github.com/vinhkhuc/jcrfsuite
- *MaltParser*: Tools for methods on transition-based dependency parsing
  Link: http://www.maltparser.org/

Contact: hieunk@soict.hust.edu.vn