BKParser 1.0

1. Giới thiệu

BKParser là một công cụ gán nhãn từ loại (Part of Speech Tagging) và phân tích cú pháp phụ thuộc (Dependency Parsing) cho tiếng Việt.
Công cụ này được cài đặt theo bài báo:

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

Hướng tiếp cận của nó là học máy, sử dụng công cụ CRFSuite để huấn luyện mô hình gán nhãn từ loại, và công cụ MaltParser (thuật toán Nivre) để huấn luyện mô hình phân tích cú pháp phụ thuộc.
Bộ dữ liệu dùng để huấn luyện mô hình là BKTreebank.
BKParser sử dụng công cụ UETSegmenter để thực hiện tách từ.
BKParser được viết trên ngôn ngữ Java.

2. Các thành phần trong thư mục đi kèm:

BKParser-1.0.jar:	File chạy, sử dụng file này để import thư viện cho project.
Models:	Thư mục chứa các model của công cụ UETSegmenter và BKParser.
Dictionary:	Thư mục chứa các từ điển cần thiết cho UETSegmenter và BKParser.

3. Cách dùng:

Để sử dụng BKParser bằng command line, bạn cần JDK 1.8 (hoặc mới hơn) và chạy với các tham số như sau:

java -jar BKParser-1.0.jar -r <thao tác> {các tham số bổ sung}

Trong đó:

-r: thao tác (tag|parse)

Các tham số bổ sung, có hai cách chạy:

1) Thực hiện phân tích trên đoạn văn bản do người dùng gõ vào

-t <text>

-t: đoạn văn bản cần gán nhãn từ loại hoặc phân tích cú pháp phụ thuộc.

Ví dụ:

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."

2) Thực hiện phân tích trên một tệp văn bản và ghi kết quả ra một tệp văn bản

-i <inputPath> -o <outputPath>

-i: đường dẫn đến tệp văn bản cần gán nhãn từ loại hoặc phân tích cú pháp phụ thuộc.

-o: đường dẫn đến tệp văn bản ghi kết quả đã gán nhãn từ loại hoặc phân tích cú pháp phụ thuộc.

Ví dụ:

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. Các API:

Để sử dụng các API, bạn cần import BKParser-1.0.jar trong project của mình, (trong BKParser đã có UETSegmenter), đồng thời copy hai thư mục dictionary và models vào project, chúng chứa các từ điển và mô hình cần thiết để tách từ và phân tích cú pháp phụ thuộc.

4.1. Contructors:

Có hai cách khởi tạo BKParser:

1) Khởi tạo không tham số

Cách khởi tạo này sẽ tải cả mô hình UETSegmenter để tách từ. Do vậy, thời gian khởi tạo sẽ chậm. Cách này dùng trong trường hợp đưa vào một đoạn văn bản hoặc một tệp văn bản chưa được tách từ.

Cú pháp:

BKParser bkParser = new BKParser();

2) Khởi tạo với tham số loadUETSegmenter

Khi loadUETSegmenter = true, khởi tạo như khởi tạo không tham số.

Khi loadUETSegmenter = false, BKParser sẽ không tải UETSegmenter. Do vậy tốc độ khởi tạo rất nhanh, nhưng chỉ dành cho trường hợp câu đã được tách từ.

Cú pháp:

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

4.2. Các API:

Ba loại API được cung cấp để gán nhãn từ loại hoặc phân tích cú pháp phụ thuộc tiếng Việt.

a) Tách từ

Phương thức này nhận tham số là một đoạn văn bản, tiến hành tách câu và tách từ trên đoạn văn bản đó, trả về một danh sách các câu đã tách từ.

Code mẫu:

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

b) Gán nhãn từ loại hoặc phân tích cú pháp phụ thuộc cho đoạn văn bản

Các phương thức này nhận tham số là một đoạn văn bản, thực hiện tách câu và tách từ trên đoạn văn bản này. Sau đó gán nhãn từ loại và phân tích cú pháp phụ thuộc cho nó.

Code mẫu:

```
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) Gán nhãn từ loại cho hoặc phân tích cú pháp phụ thuộc cho tệp văn bản

Các phương thức này nhận tham số là một đường dẫn đến một tệp văn bản, đọc văn bản từ trong tệp ra, thực hiện tách câu và tách từ trên văn bản này. Sau đó gán nhãn từ loại và phân tích cú pháp phụ thuộc cho nó.

Code mẫu:

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

d) Gán nhãn từ loại hoặc phân tích cú pháp phụ thuộc cho một danh sách các câu.

Các phương thức này nhận hai tham số: tham số đầu tiên là danh sách các câu, tham số thứ hai là một biến boolean isSegmented cho biết các câu này đã được tách từ hay chưa. Nếu isSegmented = false, thực hiện tách từ trên các câu trong danh sách. Nếu isSegmented = true, bỏ qua tách từ. Sau đó gán nhãn từ loại và phân tích cú pháp phụ thuộc cho nó. API thứ ba này cung cấp cho những ai có văn bản được tách từ rất chính xác theo ý của họ, hoặc muốn dùng một bộ tách từ khác ngoài UETSegmenter.

Code mẫu:

```
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);

Lưu ý về đầu ra:

Tất cả các API trên đều cung cấp đầu ra dưới dạng List<List<CONLLToken>>.
Các CONLLToken tương ứng với một từ ở định dạng CONLL-U.
(Xem thêm: http://universaldependencies.org/format.html)
Các List<CONLLToken> tương ứng với một câu.
Các List<List<CONLLToken>> tương ứng với một đoạn văn bản.

5. Các thư viện sử dụng:

- *UETSegmenter*: Công cụ tách từ tiếng Việt.
  Link: https://github.com/phongnt570/UETsegmenter
- *CRFSuite*: Công cụ CRF.
  Link:
  http://www.chokkan.org/software/crfsuite/
  https://github.com/vinhkhuc/jcrfsuite
-*MaltParser*: Công cụ cung cấp các phương pháp xây dựng mô hình phân tích cú pháp phụ thuộc.
  Link: http://www.maltparser.org/

Liên hệ: hieunk@soict.hust.edu.vn