ClauseSplitter 1.0: Multilingual Tool for Partial Syntactic Analysis

01/09/2017

1 Introduction

ClauseSplitter is an open-source program for partial syntactic analysis which performs the following main tasks:

- identification and tagging of verb forms, including analytical forms;
- identification and annotation of clause boundaries within each sentence.

ClauseSplitter is largely language independent. However, it requires the use of language-specific resources (list of patterns for verb form recognition; list of conjunctions; etc.) for each language. So far, it has been applied on Bulgarian and English.

2 Requirements

ClauseSplitter is written in Java and is platform independent. It is distributed as open source and is released in the following forms:

1. as a runnable JAR file (ClauseSplitter.jar to run on Java 1.8) and accompanied by scripts to run it under Windows and Linux;
2. as a runnable JAR file (ClauseSplitter16.jar to run on older versions of Java 1.6, 1.7) with adjustments of the scripts to run it under Windows and Linux;
3. as source code.

Detailed instructions, description of required parameters (below) and examples makes the system easy to use on any OS.

3 Dependencies

The program makes use of the following Java libraries (distributed separately):

- Stanford CoreNLP[1] used for preprocessing of plain English texts – sentence splitting, tokenisation, POS tagging, etc.;

[1] https://stanfordnlp.github.io/CoreNLP/

- `OpenNLP` [2] (alternative to `Stanford CoreNLP`) used for preprocessing plain texts in English and some other languages depending on available trained models – sentence splitting, tokenisation, POS tagging, etc.;
- `HTTPClient` [3] used for handling the preprocessing of plain Bulgarian texts using the web services of the `Bulgarian Language Processing Chain`[4] – sentence splitting, tokenisation, POS tagging, etc.;
- Other supporting libraries for the above – e.g., `Commons Lang`[5], `XOM`[6], etc.

4 How to run `ClauseSplitter`

The system is distributed as a runnable `JAR` (either for `Java` 1.8 or older versions) and is accompanied by scripts for easy use on either `Windows` or `Linux`.

Alternatively, you can build from source code or use within your own language processing programs.

4.1 Required command line arguments

`ClauseSplitter` requires the following arguments:

- Language – two-letter codes for languages are used, and the argument is required in order to select the relevant folder with resources and to apply pre-defined language-specific processing (available for Bulgarian and English);
- Folder containing the text files for processing;
- Output file (an `html` file containing the result of the clause splitting with colouring and annotation);
- Resources folder containing subfolders with resources for each language; each subfolder contains the following files (please keep the same names for corresponding files):
  - `verb-form-models.txt` – the file lists all verb form patterns for the language, in particular analytical verb forms (see section 5.1);
  - `conj.txt` – a list of conjunctions; data is in 5 columns: conjunction, type (coordinate or subordinate), true/false (whether it always introduces a new clause), true/false (whether it requires a comma);
  - `specadv.txt` – a list of special adverbs which attach as modifiers to conjunctions (EN: **only** to, BG: **точно** преди да);
  - `special-verbs.txt` – a list of verbs which do not comprise an independent predicate but form compound predicates with other verbs, e.g. modal and phase verbs, as well as auxiliaries;
  - `strong-punct-exceptions.txt` – a list of punctuation that always enforces a clause boundary, e.g. ';', punctuation introducing direct speech, e.g. '—', etc.

[2] `https://opennlp.apache.org/`
[3] `http://hc.apache.org/httpclient-3.x/tutorial.html`
[4] `http://dcl.bas.bg/dclservices/index.php`
[5] `https://commons.apache.org/proper/commons-lang/`
[6] `http://www.xom.nu/`

- File extension for corpus files: in order to select only files in the relevant language from a parallel corpus where files have a different, language-specific ending/extension attached;
- Type of input:
  - `xml` – processed and annotated text (sentence splitting, tokenisation, lemmatisation and POS tagging) represented in `xml` format (this is the format of the Bulgarian-English Sentence and Clause-Aligned Parallel Corpus[7]),
  - `plain` – accepting as input plain text which will require additional processing and annotation.

4.2 Running under `Linux`

There is a `bash` script file `clausesplitter.sh` which can be used to run the program on `Linux`. The script file needs to be given permission for executing (e.g. `chmod 755 clausesplitter.sh`).

4.3 Running under `Windows`

There is a script file `clausesplitter.cmd` which can be used to run the program on `Windows`. The script file needs to be given permission for executing.

4.4 Command line examples

For Bulgarian plain text with sample user-constructed examples:

```
java -jar ClauseSplitter.jar bg CORPUS/sample/ output/clauses-2017-09-01.html
resources plain bg.txt
```

For Bulgarian to run on the BulEnAC corpus in xml format:

```
java -jar ClauseSplitter.jar bg CORPUS/BulEnAC-full/ output/clauses-2017-09-01.html
resources xml txt
```

For English plain text with sample user-constructed examples:

```
java -jar ClauseSplitter.jar en CORPUS/sample/ output/clauses-2017-09-01.html
resources plain en.txt
```

For English to run on the BulEnAC corpus in xml format:

```
java -jar ClauseSplitter.jar en CORPUS/BulEnAC-full/ output/clauses-2017-09-01.html
resources xml al
```

4.5 Using API in your `Java` programs

The following example demonstrates the use of `ClauseSplitter` for processing a single sentence:

```
// construct the verb form recogniser
VerbFormFormalism vff = new VerbFormFormalism();
vff.readModelsFromFile(modelsFile);
```

[7] `http://metashare.ibl.bas.bg/repository/browse/the-bulgarian-english-sentence-and-clause-aligned-corpus/`

```
// construct the clause splitter
RuleBasedClauseSplitter rbcs = new  RuleBasedClauseSplitter(vff, conjFile,
verbsFile, punctExcFile);

// sentence is an array of type Word
// Word is a type containing: wordform, lemma, pos, etc.
Sentence sent;
// ... get sentence from text ...
// ... get sentence annotation ...

// the result is a sentence annotated with clause boundaries
Sentence resultsent = rbcs.processSentence(sent);
```

For processing a whole corpus, you can apply the clause splitter to an entire directory:

```
// run the clause splitter on texts in xml format
rbcs.processCorpusXML(corpusDirBulEnAC, outfileBulEnAC, fileext);

// run the clause splitter on plain texts
rbcs.processCorpusWithTagging(lang, corpusDirBulEnAC, outfileBulEnAC, fileext);
```

5 Method

5.1 Verb form recognition

Verb forms are recognised via pattern matching. Each verb form pattern is defined as a sequence of:

- lexical components: auxiliary verbs with or without additional morphosyntactic restrictions (*be*, *съм:Vr* - only present tense), particles (*to*, *да*), etc.
- components defined by POS with/without additional morphosyntactic restrictions: verbs (*V*), participles (*Vx* – past participle, *Vq* – passive participle),
- external elements that can break the analytical verb form defined either as 'certain lexical component', 'certain POS', or 'except certain POS', or as 'any word' (*(5,ли,се,си,P,да)* - up to 5 elements of the listed categories; *(5,-C,-V,-U)* - up to 5 elements, anything BUT Conj, Verb, Punct).

Examples:

```
Aorist+Neg+Pass не съм:Vd (5,{ли,се,си,P,да}) Aod не беше ли си му даден

Future ще (5,{ли,се,си,P,да}) Vr ще ли му го дам

Future_Perfect+Neg няма:Vr3s (n,{-V,-U,-C}) да съм:Vr (5,{ли,се,си,P,да}) Vxo
няма ли да съм му го дал
```

5.2 Conjunctions

The list of conjunctions contains the following information:

- lemma,

- main type (coordinate or subordinate),
- true/false denoting whether the conjunction always introduces a clause opening,
- true/false denoting whether the conjunction requires a comma,
- (optional) semantic type of the conjunction (e.g., conditional, temporal, relative, etc.)
- (optional) translation into English.

Example:

```
че Subord TRUE TRUE that

дали Subord TRUE FALSE conditional if; whether

дали да Subord TRUE TRUE if
```

5.3 Clause splitting

The clause splitting is performed by the consecutive application of a set of rules for identification of clause boundaries. The order of the rules can be altered which will result in different outcome. It might be language dependent. Further comparative analyses might be helpful to establish a better order of the rules.

```
# applied only at the beginning
rules.add(new BeginSentenceClauseSplittingRule());

# punctuation that always forces a clause boundary
rules.add(new StrongPunctuationClauseSplittingRule(punctExcFile));

# conjunctions that always open a new clause
rules.add(new ConjunctionClauseSplittingRule(
conjunctions, specadverbs));

# comma closing subordinate clause (Bulgarian)
rules.add(new ClosingCommaClauseSplittingRule(true));

# conjunctions that can open a new clause
rules.add(new OptionalConjunctionClauseSplittingRule(conjunctions));

# punctuation as a clause boundary
# (if no boundary is identified with any of the above)
rules.add(new PunctuationClauseSplittingRule(false));

# arbitrary boundary
# (if no boundary is identified with any of the above)
rules.add(new ArbitraryClauseSplittingRule());

# applied only at the end
rules.add(new EndSentenceClauseSplittingRule());
```

6 Bulgarian-English Sentence- and Clause-Aligned Parallel Corpus

Together with `ClauseSplitter`, we also distribute the BulEnAC corpus (Bulgarian Sentence- and Clause-Aligned Corpus) which is a very useful language resource for the analysis and training of applications for syntactic analysis, machine translation, etc.

BulEnAC consists of 176,397 tokens for Bulgarian and 190,468 for English (366,865 tokens altogether). Sentences are 30,385 (14,667 Bulgarian sentences (12.02 words per sentence on average) and 15,718 English sentences (12.11 words per sentence). The average number of clauses in a sentence in the Bulgarian part is 1.67 compared to 1.85 clauses per sentence for the English part.

BulEnAC has been processed at several levels: tokenisation, sentence splitting, lemmatisation. The processing has been performed using the Bulgarian language processing chain for the Bulgarian part and Apache OpenNLP and Stanford CoreNLP for the English part.

BulEnAC includes five broad categories, called 'styles': administrative, fiction, science, journalism, and subtitles. The corpus is represented in XML format and is supplied with various linguistic annotation – monolingual for both Bulgarian and English (sentence splitting, tokenisation, lemmatisation, POS and grammatical tagging), and parallel (sentence and clause alignment).

7 Examples and results

7.1 Some interesting examples and results

You can either run the `ClauseSplitter` on the BulEnAC corpus (xml format) or construct your own examples in order to test specific cases of clause splitting. At present, the program is distributed with resources for processing Bulgarian and English. Some simple examples have been provided in the files `CORPUS/sample/s01.txt` and `CORPUS/sample/s01-en.txt` for Bulgarian and English, respectively.

The output is displayed in `html` format, where separate clauses within a sentence are coloured with different colours, verb forms are bold and conjunctions are underlined. Each word in the sentence is annotated with a POS tag and grammatical features.

Below the example, the clause boundaries are listed – CO (clause opening), COC (coordinate CO), COS (subordinate CO), CC (clause closing), with their corresponding position in the sentence.

Tuition/NCs fees/NCp in/R England/NHs **are/Vr** under/R intense/A scrutiny/NCs -/U with/R a/DT huge/A amount/NCs of/R confusion/NCs about/R what/PQ/null **is/Vr3s** **happenning/Vy** next/A ./U

COC,0 COS,15 CC,20 CC,20

It/PF **is/Vr3s** a/DT case/NCs of/R the/DT politics/NCp of/R a/DT minority/NCs government/NCs and/C/null power/NCs bases/NCp **are/Vr** **becoming/Vy** more/D important/A than/R policy/NCs ./U

COC,0 CC,10 COC,11 CC,20

No/DT one/NCs **wants/Vr3s** to/TO/null **be/V** **left/Vx** holding/Vy an/DT unpopular/A policy/NCs when/D/null the/DT music/NCs **stops/Vr3s** ./U

COC,0 COS,3 COS,10 CC,14 CC,14 CC,14

7.2 Performance data for the BulEnAC corpus

If you run the `ClauseSplitter` on the BulEnAC corpus, since it is already manually annotated with clause boundaries, it will compare the output of the automatic annotation against the manual annotation and will display the precision and recall of the method in terms of: number of sentences, number of clauses, number of clause boundaries (opening and closing boundaries).

The program can be run on subsets of the corpus and compare results based on style, domain, etc.

Although the statistics are displayed for plain texts as well, in this case they are not valid since the plain texts have not been previously annotated with clause boundaries, so these results should be ignored.

8 Future work

Our future work involves development of additional functionalities for partial syntactic analysis, such as noun phrase recognition, anaphora resolution, etc.