PySBD

Pragmatic Sentence Boundary Disambiguation

2nd Workshop for
Natural Language Processing Open Source Software (NLP-OSS)
@
EMNLP 2020

Authors

Nipun Sadvilkar

Senior Data Scientist @

EPISOURCE

Mark Neumann

Senior Research Engineer @

Allen Institute for AI

SBD : A Solved Problem?

1. SBD is a key underlying task for natural language processing & acts as primary input for downstream tasks such as:
   a. Machine Translation
   b. Named Entity Recognition
   c. Coreference Resolution

2. SBD seems trivial though it complex as per domain e.g medical reports, legal documents, academic literature

3. Treating - “?!:;.” - as end of sentence (EOS) markers won't suffice

4. An ideal SBD system should be able to disambiguate between edge cases and false EOS markers

SBD Approaches

1. **Rules-based**
   Requires hand crafted rules/heuristics
   syntok , Stanford CoreNLP

2. **Supervised Machine Learning**
   Requires annotated datasets
   Palmer and Hearst, 1997
   Evang et al., 2013

3. **Unsupervised Machine Learning**
   Requires distributional statistics derived from raw text corpus
   Kiss and Strunk, 2006

Golden Rules Set over PTB/WSJ corpora

- Benchmarking with respect to *Penn Treebank/Wall Street Journal* corpora
  - Majority sentences end with a regular followed by typical punctuation
    - *PTB* corpus: **~90% same EOS pattern**
    - *WSJ* corpus: **~53% same EOS pattern**
  - Less end of sentence marker variation

Q. What would be a better way?

➔ Introducing **Golden Rules Set (GRS)**

- Hand constructed rules designed to cover sentence boundaries across a variety of domains.
- Keeps track of edge case scenarios

PySBD Python API: Building Blocks

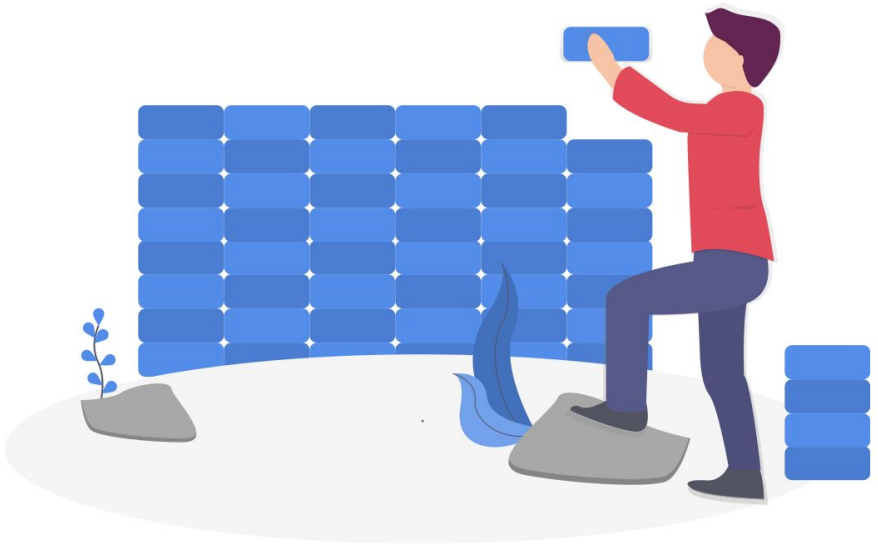
Four key components:

1. **Segmenter**
   Public API to tweak PySBD as per user needs

2. **Processor**
   Core rules engine

3. **Language**
   Makes PySBD multilingual

4. **Cleaner**
   Handles noisy text



Segmenter

Setup segmenter as per user needs:

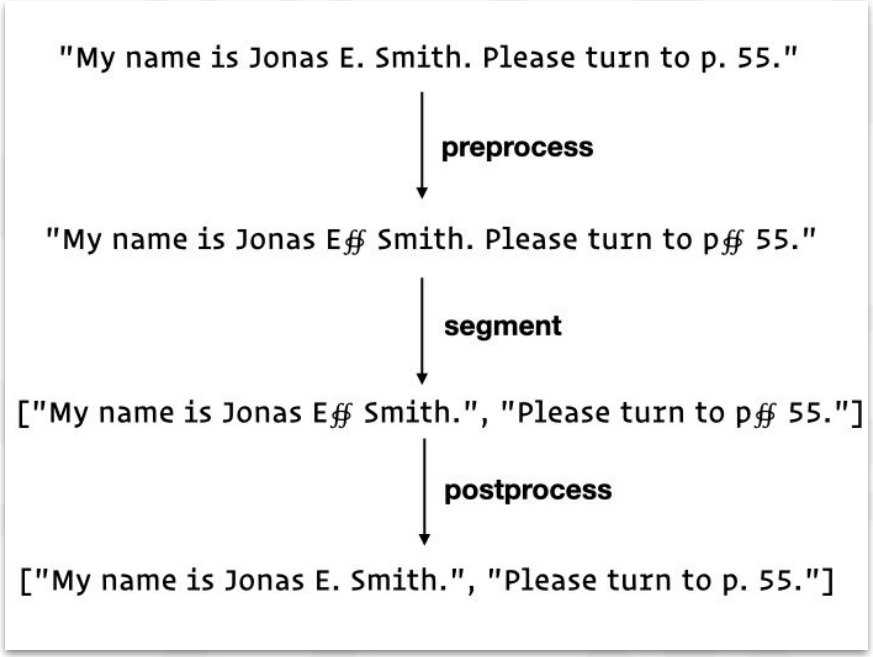
```
import pysbd
text = "My name is Jonas E. Smith. Please turn to p. 55."
seg = pysbd.Segmenter(language="en", clean=False)
print(seg.segment(text))
# ['My name is Jonas E. Smith.', 'Please turn to p. 55.']
```

- **language**
  2 character ISO 639-1 code
- **doc_type**
  Plain text / text obtained from OCRed pdf
- **clean**
  To handle noisy input text
- **char_span**
  Retrieve indices of sentences

Processor

- Core rules engine processes text in 3 stages:
  - **Preprocess**
  - **Segment**
  - **Postprocess**

- Purpose-driven segregated rules set like:
  - **ListItemReplacer**
  - **AbbreviationReplacer**
  - **ExclamationWords**
  - **BetweenPunctuation, etc.**



Language

- Accommodates **22 languages**:
  Amharic, Arabic, Armenian, Bulgarian, Burmese, Chinese, Danish, Deutsch, Dutch, French, Greek, Hindi, Italian, Japanese, Kazakh, Marathi, Persian, Polish, Russian, Spanish, Urdu

- Each language is inherited from two sub-components:
  - **Common**
  - **Standard**

Common Rules Set

AM-PM regex set

Numbers regex set

Generic Sentence Boundary regex

Quotation regex

Parenthesis regex

Standard Rules Set

Generic Punctuations

Default Abbreviations

Geolocation reference regex

Fileformat mentions regex

Ellipsis regex set

Cleaner

Text in the wild can be noisy: extraneous line breaks, unicode characters, uncommon spacing and hangovers from document structure.

- Set of text cleaning rules such as:
  - Irregular newline characters/spacing
  - Table of contents
  - URLs, HTML tags
  - Sentences delimited without any space

- Cleaner is enabled through Segmenter
- Disables **`char_span`** functionality

Benchmarks & Results

Tool	GRS	GENIA
blingfire	75.00	86.95
syntok	68.75	80.90
spaCy	52.08	76.80
spacy dep	54.17	39.20
stanza	72.92	63.40
NLTK	56.25	87.95
PySBD	97.92	97.00

English GRS:
comprising 48 rules

GENIA Corpus:
Linguistically annotated biomedical papers

Tool	Speed(ms)
blingfire	85.24
syntok	1764.11
spaCy	1523.20
spacy dep	26850.69
stanza	48383.46
NLTK	780.49
PySBD	9483.96

Speed benchmark:
on the entire text of
'The adventures of Sherlock Holmes'

Multilingual Support

Accuracy of PySBD's multilingual modules* on the OPUS 100 multilingual corpus test sets, containing 2000 sentences per language.

Language	Accuracy (%)
Amharic	80.95%
Arabic	70.40%
Armenian	63.75%
Bulgarian	93.35%
Burmese	48.05%
Chinese	85.35%
Danish	91.40%
Deutsch	80.95%
Dutch	91.40%
French	91.90%
Greek	91.05%
Hindi	88.50%
Italian	90.55%
Japanese	96.45%
Kazakh	63.20%
Marathi	92.60%
Persian	84.95%
Polish	55.48%
Russian	88.55%
Spanish	92.65%
Urdu	77.55%

*Each language module build with respect to its own GRS.

Contributing Guidelines

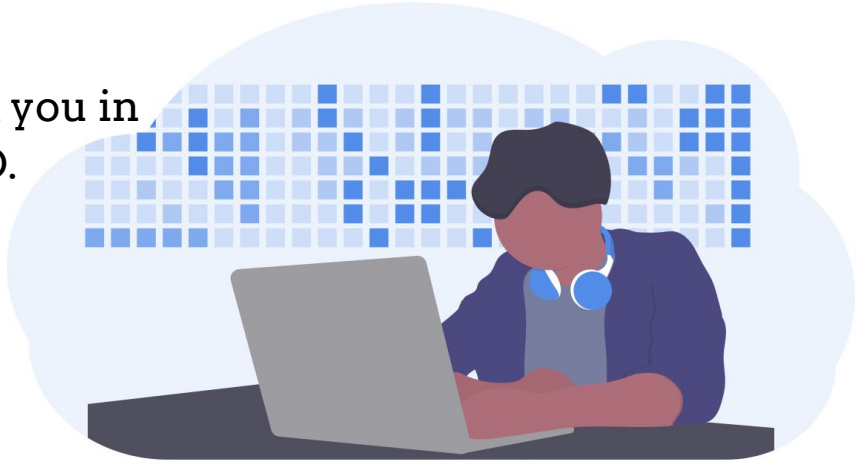
1. **Add a new rule to existing Golden Rules Set (GRS)**

   Existing rules in GRS are by no means exhaustive.
   Contribute by reporting a new rule by an opening an issue on our GitHub repo.

2. **Add support for a new `Language`**

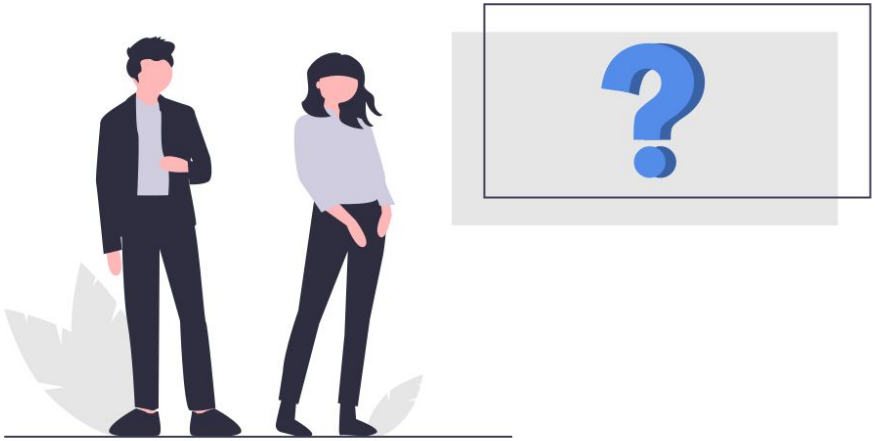
   We would be more than happy to assist you in adding new Language support to PySBD.

   Refer to docs to know more in details.



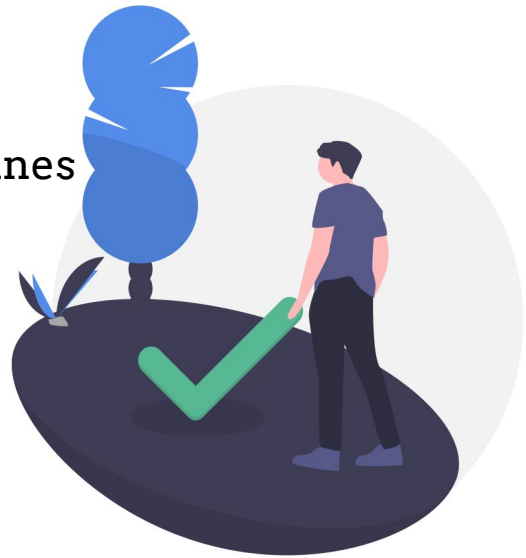
Why PySBD?

- Built by considering various domain edge cases
- Package Development with Test-Driven Development (TDD) ♻️ to ensure robustness 💪🏽
- Non-Destructive Segmentation 👌🏽
- Multilingual Support 🌏



Conclusion

- PySBD has interpretable rules and are easy to modify
- Highly accurate - 97% English GRS - irrespective of domain ✔️
- Robust codebase with 98% test coverage 💚
- Lightweight, easy to integrate with existing NLP pipelines
- Extensible in community driven way
- Already being used by 71 projects*



*as of 22 October 2020