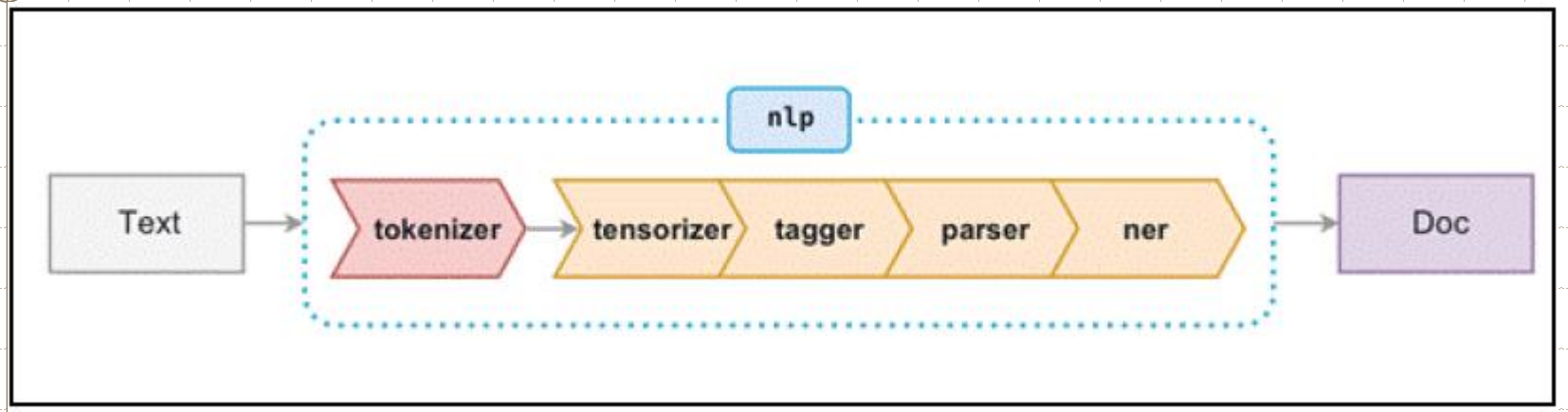
自然言語処理
第11回：
自然言語処理の応用2：
辞書、コーパスベースの技術
テキスト処理

David Ramamonjisoa

目次

- トークン化(tokenization)
- 辞書
- IPALコーパス辞書
- Wordnet, Wordnetのオントロジー
- コーパス
- 京大コーパス
- まとめ
- 課題

テキスト処理の流れ



- トークン化(tokenizer)、単語の品詞付与(tagger)、単語の原形の復元(parser,lemmatizer)、固有名抽出(NER)は一般的なテキスト処理のこと
- **Doc**は応用による様々な形で処理を行う

トークン化/トークナイザ（Tokenizer）

- トークンは一般的に形態素解析の結果の文字列のリスト（日本語）、文から単語と区切りのリスト(英語)
- トークン化

→文字列を分解し、トークンと呼ばれる言語データの一部を構成する識別可能な言語学上の単位に変換する作業である

日本語のトークン化/トークナイザ(Tokenizer)

- JUMAN, JUMAN++ 形態素解析ツール
- ChaSen(奈良先端科学技術大学院で開発された形態素解析ツール)
- TinySegmenter(http://www.chasen.org/~taku/software/TinySegmenter/)
- MeCab:Yet Another Part-of-Speech and Morphological Analyzer
- Janome:辞書内包の形態素解析器
- Nagisa:BLSTMsによる単語分割と品詞タガー
- …

MeCabの形態素解析の例

```
>>> s0
'・車が入ってこられないところには、物資の配給がない。'
>>> print(s)
・      ・      ・      記号-一般
車      クルマ  車      名詞-一般
が      ガ      が      助詞-格助詞-一般
入っ    ハイッ  入る    動詞-自立          五段・ラ行          連用タ接続
て      テ      て      助詞-接続助詞
こ      コ      くる    動詞-非自立        カ変・クル          未然形
られ    ラレ    られる  動詞-接尾          一段        未然形
ない    ナイ    ない    助動詞    特殊・ナイ          基本形
ところ  トコロ  ところ  名詞-非自立-副詞可能
に      ニ      に      助詞-格助詞-一般
は      ハ      は      助詞-係助詞
、      、      、      記号-読点
物資    ブッシ  物資    名詞-一般
の      ノ      の      助詞-連体化
配給    ハイキュウ      配給      名詞-サ変接続
が      ガ      が      助詞-格助詞-一般
ない    ナイ    ない    形容詞-自立        形容詞・アウオ段  基本形
。      。      。      記号-句点
EOS
```

トークンの数：18

英語などのトークン化/トークナイザ（Tokenizer）

- Nltk(python)
- Stanford parser(web application)
- Spacy(python)
- Google nlp api
- Yahoo nlp api
- Syntaxnet(C++)
- Corenlp(java)

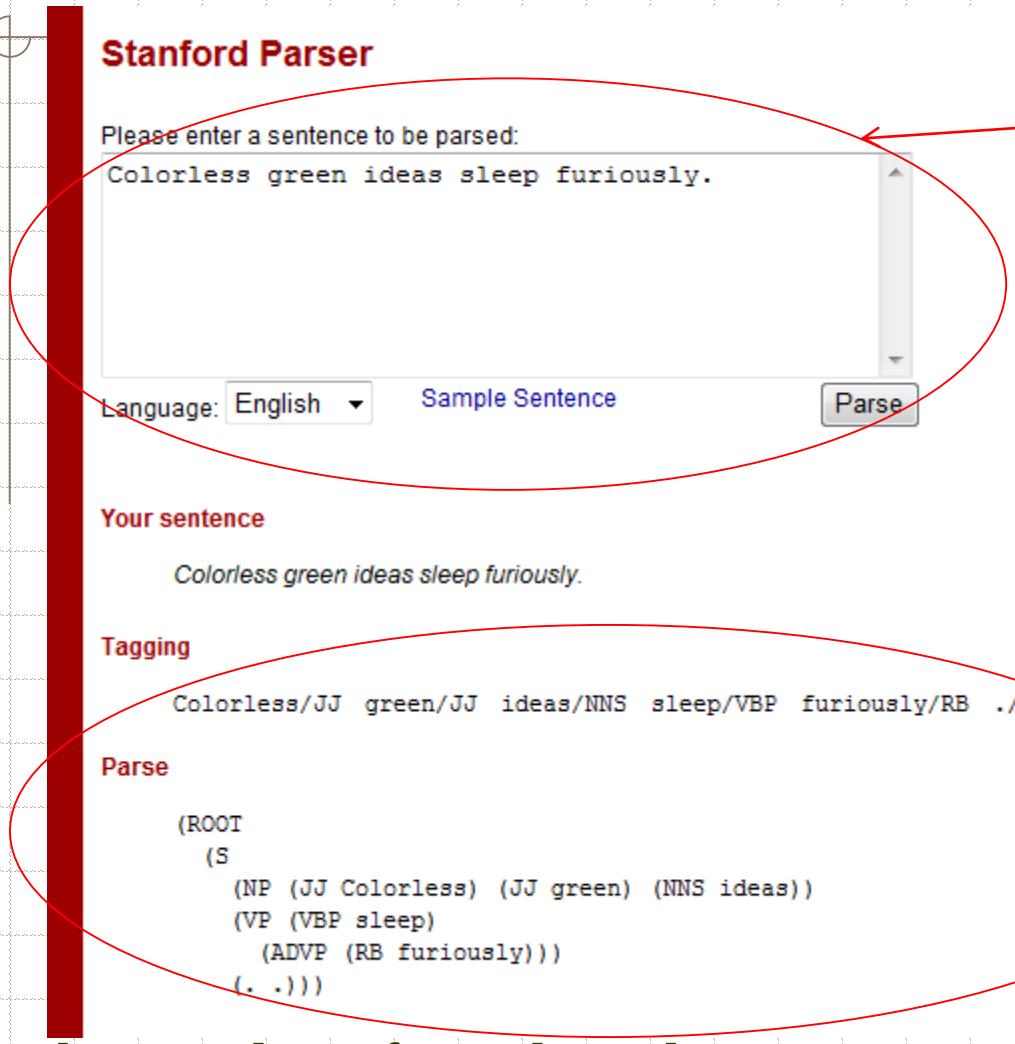
英語などのトークン化/トークナイザ（Tokenizer）

Feature comparison

Here's a quick comparison of the functionalities offered by spaCy, SyntaxNet, NLTK and CoreNLP.

	SPACY	SYNTAXNET	NLTK	CORENLP
Programming language	Python	C++	Python	Java
Neural network models	●	●	●	●
Integrated word vectors	●	●	●	●
Multi-language support	●	●	●	●
Tokenization	●	●	●	●
Part-of-speech tagging	●	●	●	●
Sentence segmentation	●	●	●	●
Dependency parsing	●	●	●	●
Entity recognition	●	●	●	●
Coreference resolution	●	●	●	●

Stanford Parser



入力エリア

トークンの数：6

結果エリア

Tokens: a list of words and punctuation

テキストのトークン化のための正規表現

- 基本的な作業であるが、多くのコーパスが既にトークン化されており、NLTKがいくつかのトークナイザを提供してくれている
  - TreebankWordTokenizer
  - PunctWordTokenizer
  - RegexpTokenizer
    - WordPunctTokenizer

NLTKのトークン化

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

シンプルなアプローチによるトークン化

- re.split(r’ ’, raw)
- re.split(r’[¥t¥n]+’, raw)

→re.split(r’¥s+’, raw)

- re.split(r’¥W+”, raw)

(not などが出てしまう

- re.findall(r’¥w+|¥S¥w*’, raw)

- re.findall(r”¥w+(?:[-’]¥w+) * | ` | [~, (]+ | ¥S¥w*”, raw)

正規表現･正規表現の省略形

- ¥b 単語の境界(幅はゼロ)
- ¥d すべての数字([0-9]と等価)
- ¥D すべての非数字([^0-9]と等価)
- ¥s すべての空白文字([¥t¥n¥r¥f¥v]と等価)
- ¥S すべての非空白文字([^ ¥t¥n¥r¥f¥v]と等価)
- ¥w すべての英文字、数字およびアンダースコア([a-zA-Z0-9]と等価)
- ¥W すべての英文字、数字およびアンダースコア以外の 文字([^ a-zA-Z0-9]と等価)
- ¥t タブ文字
- ¥n 改行文字

NLTKの正規表現トークナイザ

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+         # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*         # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?   # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.             # ellipsis
...   | [][.,;"'?():-_`]   # these are separate tokens
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

u.s.a.の小文字,日付,ドル以外の通貨は抽出されない

トークン化のそのほかの問題

- 何に応用するかによって、何をトークンとしてみなすか異なってきてしまう
- 生のテキストを手作業でトークン化した高品質なトークンを**ゴールドスタンダード**と呼ぶ
- トークン化で「didn't」のような短縮形を
  文の意味を解析しているならば
  「did」と「n't」か「not」と正規化するべきである
  ルックアップテーブルの助けを借りることにより
  可能になるだろう

辞書・事典とは何か―その体系性・網羅性・信頼性を求めて―

- 辞書・事典は人間のもつ概念の体系的表現である。辞書は我々が使う言葉の体系であり、事典はそれ以外の各分野の対象についての説明である。
- 辞書や事典をどう構成するかは辞書学（lexicography）であるが、その構成要素である語の内容は語彙論（lexicology）による。後者の中心的課題は意味をどう表現するかにある。
- 直接的には内包的定義と外延的定義があり、語は常にあいまいさを伴うため、この双方で記述することが必要で ある。

2019年度版 形態素解析器比較表

ツール名	初期リリース	最新リリース	解析手法	分割基準	解析速度	対応言語	pip	Star	特徴
JUMAN	1992	2014	ラティス/人手によるコスト設定	JUMAN基準	○	perl, Python	△	-	豊富な意味表現の付与が可能
ChaSen	1996	2011	ラティス/HMM	ipadic, UniDic, NAIST-dic	△	コマンドラインのみ	×	-	統計処理により形態素解析を可能とした
MeCab	2006	2013	ラティス/CRF	ipadic, UniDic, JUMAN基準, NEologd	◎	C++, Python, その他ラッパー多数	○	469	日本語NLPのデファクトスタンダード
KyTea	2011	2014	点推定/SVM	UniDic	○	C++, Python	△	156	読み推定が可能
Rakuten MA	2014	2015	系列ラベリング/SCW	UniDic	○	JavaScript	×	403	100% JavaScript による実装
JUMAN++	2016	2019	ラティス/ニューラルLM	JUMAN基準	○	Python	△	173	ニューラルLMを用いることで高精度
Sudachi	2017	2019	ラティス/CRF	長・中・短の3種類	○	Java, Python	○	368	分割単位のコントロールが可能
nagisa	2018	2019	系列ラベリング/BLSTMs	UniDic	△	Python	○	135	Pythonで簡単に使える
Stanford NLP	2019	2019	系列ラベリング/BLSTMs	Universal Depndency	×	Python	○	2377	多言語に対応
GiNZA	2019	2019	ラティス/CRF	Universal Depndency	×	Python	○	233	係り受け解析も可能
SentencePiece	2017	2019	UnigramLM	Subword	◎	C++, Python	○	2977	ニューラル言語処理向けのトークナイザ

まとめ

本日の内容は以下の通りであった:

- 単語、単語のクラス、単語のタグ
- 問題:一つ単語が複数タグを持っている:曖昧性
- 機械用辞書
- IPALコーパス辞書:日本語辞書(名詞、動詞、形容詞)
- Wordnet, Wordnetのオントロジー
- 言語コーパス:生コーパス、タグ付きコーパス
- 機械学習を用いて単語分割、形態素解析