コラム：単語の前処理について

ここまでの実行例では, 空白で区切られたすべての文字列を単語として扱いましたが, 実際にはテキストには数字や特殊文字 (記号や罫線など) が入っており, これらをそれぞれ別の単語として扱うと, 特に数字によって語彙が爆発的に増えてしまいます. また, 英語では `I'm` や `Alice's` はそれぞれ, `I 'm`, `Alice 's` と分割して, `I` や `Alice` を単語とみなすのが望ましいでしょう.

そこで, 一般にたとえば数字を`#`で置き換えたり, 特殊文字を削除したり, `aaaaaaa` のような同じ文字の連続を 3 文字に縮約して `aaa` とするなどの前処理を行います. こうすると, たとえば `No.123` は `no.###`に, `2021.3.28` は `####.#.##` に, `Gooooooooal` は `goooal` になります. また, 大文字と小文字の違いで別の単語と認識されないよう, ここに示したようにすべて小文字に直すこともよく行われる処理の一つです.[*22]

前処理には決まった手続きがあるわけではありませんが, 英語の場合は教科書 FSNLP [29]のサイトにある `strip.sed`[*23] などを使うのが簡単です. 日本語の場合は, 後で紹介する mecab-NEologd のサイト [*24] に正規化処理についてまとめられているほか, この処理を pip でインストールできるパッケージにした `neologdn`[*25] が公開されています. 表 3.2 は, こうした前処理を行った上で単語を数えたものです.

また, Python の spaCy や R の quanteda といったパッケージに前処理を任せることも可能です.

3.3▽ 単語の統計的フレーズ化

∗25　多くの言語で何を大文字にするかには規則性があるため, 文脈を利用して単語の各文字を小文字から適切に大文字にする分類器を教師あり学習することは容易です. よって, 小文字にしても本質的な情報量が落ちることはあまりありませんが, たとえば `US` を `us` として処理すると誤ることもありますので, すべて小文字化することにはリスクもあることに注意してください.

∗25　`https://nlp.stanford.edu/fsnlp/statest/sed.strip`

∗25　`https://github.com/neologd/mecab-ipadic-neologd/wiki/Regexp.ja`

∗4　`https://github.com/ikegami-yukino/neologdn`
`https://yukinoi.hatenablog.com/entry/2015/10/11/205006`