Introduction Natural Language Processing

3 minute read

- Understand key terms in natural language processing

Announcements

- Written course notes uploaded to Canvas
- Streaming homework hard deadline tonight
- Submit last week's participation if you haven't
- Want midterm next week, or after Spring break?

123 GO - What's something you enjoyed this weekend?

Resources

- Julia TextAnalysis package (https://juliatext.github.io/TextAnalysis.jl/latest/)

> This manual is designed to get you started doing text analysis in Julia. It assumes that you already familiar with the basic methods of text analysis.

Our goal in this lecture is to give you the bare minimum background knowledge necessary to read the documentation.

Even though we're using Julia, the terms we'll learn today are general across NLP (Natural Language Processing).

Motivation

Our eventual goal is to statistically analyze the text contained in the IRS 990 forms, for example, clustering for similar statements. To accomplish this, we need to convert the data from text into a numeric matrix.

The numeric matrix should have rows for each observations, and columns representing features that distinguish or describe this observation. If we represent our data in this way, then we have a wealth of statistical methods available to us.

Consider these two phrases:

1. “Bash has been processing big data forever.”
2. “Don’t bash big data. It’s big fun!”

How are we going to represent these as a table / matrix?

```
    Bash    has     been    processing  big     data    forever don't
1   1       1       1       1           1       1       1       0
2   1       0       0       0           2       1       0       1

... more columns for the rest of second.
```

123 GO: Notice anything that might hinder our analysis?

- misspelling
- tense (past present future)
- bash is a homonym
- sentences and punctuation

A **document term matrix** has rows for every observation (document) and columns with counts for every term (word).

Objects

Text is a sequence of characters. It has many other names: aka free text, plain text, raw text, strings.

Encoding is how exactly the computer stores that text in binary format.

- **ASCII** (American Standard Code for Information Interchange) is old, it can represent the English language with standard punctuation.
- **Unicode** is newer, it can represent other languages, mathematical symbols, and emoticons. The most common Unicode encoding is UTF-8.

```
echo "a,b,c
hello,goodbye,there" > abc.csv

file abc.csv
```

123 GO- when should you encode text data in format other than ASCII or UTF-8?

Only if you're forced to.

Other options:

- UTF-16
- UTF-32
- latin
- japanese specific
- many others

A **document** is one unit of text, for example:

- single sentence
- paragraph
- essay
- book

`StringDocument()`

```
using TextAnalysis

s1 = "Bash has been processing big data forever."
s2 = "Don't bash big data. It's big fun!"

d1 = StringDocument(s1)
d2 = StringDocument(s2)
```

Printing shows some metadata.

A **corpus** is a collection of documents. The plural form is corpora.

`Corpus`

```
c = Corpus([d1, d2])
```

Processing

tokens are the semantically meaningful elements of the text, often just single words. **tokenization**, aka **lexing** breaks a document down into tokens.

`tokens`

```
tokens(d1)
```

123 GO: what doesn't belong?

The period.

`prepare!`, `strip_punctuation`, `remove_case!`

```
prepare!(d1, strip_punctuation)

remove_case!(d1)
```

123 GO: are these updates happening in place? Will the documents inside the corpus be changed?

Yes!

Let's look at the corpus.

```
text(c[1])
text(c[2])
```

"The **lexicon** of a corpus consists of all the terms that occur in any document in the corpus."

`update_lexicon!`

```
update_lexicon!(c)
lexicon(c)
```

---

123 go: How many unique words are in the following example?

```
d3 = StringDocument("help helpful helper helping person")
```

Stemming means converting multiple words to one representative word stem.

`stem!`

```
stem!(d3)

text(d3)
```

Let's stem the whole corpus.

```
stem!(c)

text(c[1])
```

Document Term Matrix

Let's bring it all together now.

```
c = Corpus([StringDocument("Bash has been processing big data forever."),
    StringDocument("Don't bash big data. It's big fun!")])

remove_case!(c)
prepare!(c, strip_punctuation)
stem!(c)
update_lexicon!(c)

text(c[1])
text(c[2])
```

`DocumentTermMatrix`, `dtm`, `:dense`

```
d = DocumentTermMatrix(c)

dtm(d)

dtm(d, :dense)
```

Let’s check that the columns correspond to these terms:

`.term`

```

```

What’s a sparse matrix? Why use it here?

Hint: There are more than 100,000 words in the English language.

Updated: March 8, 2021