Språk och Datorer (729G49)

Språkteknologi 1: Textextraktion och -segmentering

2/4/2025

Innehåll

- Textextraktion
- Textsegmentering
- Enkla statistiska analyser

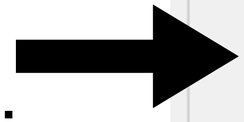
Att bygga en mini-corpus

- För veckans labb vill vi bygga en egen liten korpus till en statistisk analys!
- I denna föreläsning ska vi därför titta på hur man kan:
    - Ladda ner webbsidor.
    - Tokenisera dem med reguljära uttryck.
- Den sista delen handlar om statistiska analyser och lag som ni ska testa i labben.

Extrahera text från webbsidor

“Spindla” webben 

- Exempel: Vi vill ha alla Wikipedia-artiklar i en viss kategori.
- https://sv.wikipedia.org/wiki/Kategori:Marsvinsartade_gnagare
- Vi kan skriva en funktion som hittar alla länkar inom kategorin.
- Sedan kan vi ladda ner dem.



```python
import requests
from bs4 import BeautifulSoup

category_url = "https://sv.wikipedia.org/wiki/Kategori:Marsvinsar

# Get all article links from a category page
def fetch_category_links(url):
    article_links = []
    while url:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        for link in soup.find_all('a', href=True):
            href = link['href']
            if href.startswith('/wiki/') and not href.startswith(
                article_links.append('https://sv.wikipedia.org' +

        next_page = soup.find('a', {'title': 'Nästa sida'})
        url = 'https://sv.wikipedia.org' + next_page['href'] if n

    return article_links

links = fetch_category_links(category_url)
```

Ladda ner en Wikipedia-artikel

- Okej, vi ska göra det lite lättare och bara ladda ner en artikel:
    - https://sv.wikipedia.org/wiki/Kapybara

- Enkelt!

```python
import requests

url = "https://sv.wikipedia.org/wiki/Kapybara"

response = requests.get(url)

with open("Kapybara_article.html", 'w', encoding='utf-8') as file:
    file.write(response.text)
```

Avformatering

- Hur ser den nedladdade artikeln ut?
    - HTML-uppmärkningar.
- Vi vill bara ha råtext i korpusen.
    - Extrahera sidans egentliga innehåll med hjälp av en HTML-parser.


```
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page
<head>
<meta charset="UTF-8">
<title>Kapybara – Wikipedia</title>
<script>(function(){var className="client-js vector-feature-language-in-header-enabled vector-feature-
"wgDefaultDateFormat":"dmy","wgMonthNames":["","januari","februari","mars","april","maj","juni","juli",
"Wikipediaartiklar med identifierare från Fossilworks","Wikipediaartiklar med identifierare från MSW",
"wgULSisCompactLinksEnabled":false,"wgVector2022LanguageInHeader":true,"wgULSisLanguageSelectorEmpty":
"skins.vector.search.codex.styles":"ready","skins.vector.styles":"ready","skins.vector.icons":"ready",
"ext.popups","ext.visualEditor.desktopArticleTarget.init","ext.visualEditor.targetLoader","ext.echo.cen
<script>(RLQ=window.RLQ||[]).push(function(){mw.loader.impl(function(){return["user.options@12s5i",func
}];});});</script>
<link rel="stylesheet" href="/w/load.php?lang=sv&amp;modules=ext.cite.styles%7Cext.uls.interlanguage%7C
<script async="" src="/w/load.php?lang=sv&amp;modules=startup&amp;only=scripts&amp;raw=1&amp;skin=vecto
<meta name="ResourceLoaderDynamicStyles" content="">
<link rel="stylesheet" href="/w/load.php?lang=sv&amp;modules=ext.gadget.babel%2Cerror%2Cgeo%2Cprintonly
<link rel="stylesheet" href="/w/load.php?lang=sv&amp;modules=site.styles&amp;only=styles&amp;skin=vecto
<meta name="generator" content="MediaWiki 1.44.0-wmf.3">
<meta name="referrer" content="origin">
<meta name="referrer" content="origin-when-cross-origin">
<meta name="robots" content="max-image-preview:standard">
<meta name="format-detection" content="telephone=no">
<meta property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/thumb/6/6e/Kapybara_K
<meta property="og:image:width" content="1200">
<meta property="og:image:height" content="817">
<meta property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/thumb/6/6e/Kapybara_K
<meta property="og:image:width" content="800">
<meta property="og:image:height" content="545">
<meta property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/thumb/6/6e/Kapybara_K
<meta property="og:image:width" content="640">
<meta property="og:image:height" content="436">
<meta name="viewport" content="width=1120">
<meta property="og:title" content="Kapybara – Wikipedia">
<meta property="og:type" content="website">
<link rel="preconnect" href="//upload.wikimedia.org">
<link rel="alternate" media="only screen and (max-width: 640px)" href="//sv.m.wikipedia.org/wiki/Kapyba
<link rel="alternate" type="application/x-wiki" title="Redigera" href="/w/index.php?title=Kapybara&amp;
<link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png">
<link rel="icon" href="/static/favicon/wikipedia.ico">
<link rel="search" type="application/opensearchdescription+xml" href="/w/rest.php/v1/search" title="Wik
<link rel="EditURI" type="application/rsd+xml" href="//sv.wikipedia.org/w/api.php?action=rsd">
<link rel="canonical" href="https://sv.wikipedia.org/wiki/Kapybara">
<link rel="license" href="https://creativecommons.org/licenses/by-sa/4.0/deed.sv">
```

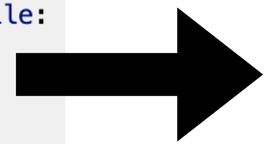
Extrahera alla p-element (paragrafer)

```python
from bs4 import BeautifulSoup

with open("Kapybara_article.html", 'r', encoding='utf-8') as file:
    soup = BeautifulSoup(file, 'html.parser')

paragraphs = soup.find_all('p')
cleaned_text = '\n'.join([para.get_text() for para in paragraphs])

with open("Kapybara_article.txt", 'w', encoding='utf-8') as file:
    file.write(cleaned_text.strip())
```



Kapybara eller vattensvin (Hydrochoerus hydrochaeris) är världens största gnagare. Den påminner i sitt levnadssätt om flodhästar men är närmare släkt med marsvin. Fram till 1990-talet kategoriserades kapybaran som ensam art i släktet Hydrochoerus, som i sin tur placerades som ensamt släkte i familjen Hydrochaeridae. I nyare taxonomiska avhandlingar kategoriseras den norra populationen som den egna arten Hydrochoerus isthmius och de placeras istället i familjen marsvin (Caviidae).[1][2]

Kapybaran når en kroppslängd upp till 130 centimeter och en vikt upp till 61 kilogram. Arten förekommer i stora delar av Sydamerikas slättland samt i angränsande regioner av Centralamerika. Kapybaran jagas för köttets och hudens skull men räknas inte som en hotad art. Dess närmaste släkting, utöver Hydrochoerus isthmius, är klippmarsvinet och listas därför tillsammans som en underfamilj inom familjen marsvin (Caviidae).

Djuret har en stor och robust kropp med korta extremiteter. Kapybaran blir upp till 130 cm lång och 60 cm hög (vid skuldran)[3]. Hanar och honor är lika i utseendet, nosen är dock längre på hanar. Hanen har en oval, naken doftkörtel på nosen, vilken avsöndrar ett vitt, klibbigt sekret. Båda könen har även analkörtlar vars doft gör att de känner igen varandra. På de främre fötterna finns fyra tår och på de bakre fötterna tre. Tårna liknar hovar och mellan dem finns lite simhud. Svansen är bara rudimentär. Honor är lite större än hanar och de förstnämnda har en genomsnittlig vikt på 61 kilogram medan hanarnas genomsnittliga vikt ligger vid 50 kilogram. Allmänt kan vikten ligga på mellan 27 och 80 kilogram.

Pälsen är lång och grov men vid vissa ställen så gles att hude[illegible]r synlig. Färgen varierar mellan rödbrun och grå på ovansidan, buken är gulaktig[illegible]brun. En del individer har svarta fläckar i ansiktet, vid extremiteternas ut[illegible]ler på bålen. Hårens längd ligger mellan 30 och 120 millimeter.

Kapybaran har ett påfallande brett huvud. Jämförd m[illegible]ktingar är nosen förstorad och mera avrundad. Näsborrarna är s[illegible]randra. Öronen är små och avrundade. Även ögonen är små och [illegible]. Som hos flera andra vattendjur ligger öronen, ögone[illegible] ansiktet[3]. På så sätt behöver de bara lyfta en lit[illegible] vattenytan för att iaktta omgivningen.

Tandformeln är 1-0-1-3, kapybaran har alltså per käkha[illegible] och tre molarer, tillsammans 20 tänder. Som hos alla g[illegible]ot och växer hela livet. Efter framtänderna finns en stör[illegible] så kalla[illegible] diastema.

Segmentering och Tokenisering

Textsegmentering

- **Textsegmentering** är uppgiften att dela upp en text i lingvistiskt meningsfulla enheter, t.ex.:
    - Ord,
    - Morfemer,
    - Meningar,
    - Stycken.

Tokenisering (1)

- När enheterna som segmenteras är ord eller ordliknande enheter så kallas det för **tokenisering.**
- Enklaste fall: Tokenisera på vittecken.

```
>>> text = 'Tokenisera mig!'
>>> text.split()
['Tokenisera', 'mig!']
```

  - Men kanske vill vi även separera skiljetecken från ord: [‘Tokenisera’, ’mig’, ‘!’]?
  - …och vill samtidigt behålla ‘Karl-Ove” som én token?

Tokenisering (2)

- Och sen finns det språk som inte använder sig av vittecken…
- …och där tokeniseringen inte alltid är trivial.

A potential overlapping ambiguity in Chinese.

a. * 布什 在 谈话 中指 出
Bush at talk middle-finger out

b. 布什 在 谈话 中 指出
Bush at talk middle point-out
'Bush pointed out in his talk'

Glass et al. 2024:
Language and Computers

Möjliga fel vid tokenisering

- **Undersegmentering**
    - Den automatiska tokeniseringen missar att segmentera en teckensekvens som enligt guldstandard ska segmenteras.
    - [‘Tokenisera’, ’mig!’] — [‘Tokenisera’, ’mig’, ‘!’]
- **Översegmentering**
    - Den automatiska tokeniseringen delar på en teckensekvens som enligt guldstandarden inte ska segmenteras.
    - [‘S’, ‘:’,’t’,‘Lars’,‘Kyrka’] — [‘S:t’,‘Lars’,‘Kyrka’]

Utvärdering av tokeniseringen (1)

- **Guldstandard**: En korpus av högt kvalité, gärna en mänsklig annotation.
    - **Träningsmängden** används för att träna upp tokeniseraren, eller, i vårt fall, utveckla ett reguljärt uttryck.
    - **Testmängden** används för att utvärdera det färdiga systemets förmåga att generalisera utanför träningsmängden.
    - Testmängden får absolut inte vara en del av träningsmängden!

Utvärdering av tokeniseringen (2)

	Hittade vi	Hittade vi inte
Finns i guldstandarden	**Sanna Positiva**	**Falska Negativa**
Finns inte i guldstandarden	**Falska Positiva**	**Sanna Negativa**

Utvärdering av tokeniseringen (3)

- **Precision**: Hur många procent av alla tokens som vi hittade är i guldstandarden?
  - $$\frac{\text{sanna positiva}}{\text{sanna positiva + falska positiva}}$$

- Vår tokeniserare: [‘S’, ‘:’,’t’,’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping.’]
- Guldstandard: [’S:t’, ’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping’, ‘.’]

Utvärdering av tokeniseringen (3)

- **Precision**: Hur många procent av alla tokens som vi hittade är i guldstandarden?
  - $$\frac{\text{sanna positiva}}{\text{sanna positiva + falska positiva}}$$

- Vår tokeniserare: [‘S’, ‘:’,’t’,’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping.’]
- Guldstandard: [’S:t’, ’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping’, ‘.’]

$$\frac{4}{4+4} = 0.5$$

Utvärdering av tokeniseringen (4)

- **Täckning**: Hur många procent av alla tokens i guldstandardet hittade vi?
  - $$\frac{\text{sanna positiva}}{\text{sanna positiva + falska negativa}}$$

- Vår tokeniserare: [‘S’, ‘:’,’t’,’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping.’]
- Guldstandard: [’S:t’, ’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping’, ‘.’]

Utvärdering av tokeniseringen (4)

- **Täckning**: Hur många procent av alla tokens i guldstandardet hittade vi?
  - $$\frac{\text{sanna positiva}}{\text{sanna positiva + falska negativa}}$$

- Vår tokeniserare: [‘S’, ‘:’,’t’,’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping.’]
- Guldstandard: [’S:t’, ’Lars’,’Kyrkan’, ‘är’, ‘i’, ‘Linköping’, ‘.’]

$$\frac{4}{4+3} \approx 0.57$$

Meningssegementering

- Ibland vill vi inte bara identifiera ord utan även större enheter så som meningar eller stycken.
    - Meningssegementering behövs t.ex. for syntaktisk analys
- Meningssegmentering är uppgiften att dela upp en text i meningar.
- Meningssegmentering är svårare än att dela upp en text efter en punkt eller något annat skiljetecken.
    - *I visited the U.S. last year. After that, I went to Canada.*

Regex (reguljära uttryck)

Regex | Reguljära Uttryck | RE

- Ett verktyg för att matcha och manipulera text baserat på specifika mönster.
- Används ofta för att söka, extrahera, eller ersätta text.

```python
import re

text = "Min e-post är test@example.com och mitt postnummer är 12345."

regex_mail = r'[\w_-]+@[\w_-]+\.[A-Za-z]{2,}'
emails = re.findall(regex_mail, text)
print("E-postadresser:", emails)
```

✓ 0.0s

```
E-postadresser: ['test@example.com']
```

Varning

- Det är ovanligt att någon förstå hur man använder regex från en förklaring.
  - Ni måste testa det!
  - Det absolut bästa är att gå genom uppgifterna på https://regexone.com/ och testa vad händer med olika uttryck på https://regex101.com/.
- Här ska jag främst samla den viktigaste informationen som behövs för labben.

Kom ihåg: Tokenisering

- När enheterna som segmenteras är ord eller ordliknande enheter så kallas det för **tokenisering.**
- Enklaste fall: Tokenisera på vittecken.

```
>>> text = 'Tokenisera mig!'
>>> text.split()
['Tokenisera', 'mig!']
```

  - Men kanske vill vi även separera skiljetecken från ord: [‘Tokenisera’, ’mig’, ‘!’]?
  - …och vill samtidigt behålla ”Karl-Ove” som én token?

Tokenisering med reguljära uttryck

- När man måste tokenisera en text på ett visst sätt kan man använda **reguljära uttryck**.
- Mönster som man matchar mot en text och som kan anpassar till sina behov.

```
>>> pattern = r'\S+'
>>> re.findall(pattern, text)
['Tokenisera', 'mig!']
```

```
>>> pattern = r'\w+'
>>> re.findall(pattern, text)
['Tokenisera', 'mig']
```

```
>>> pattern = r'\w+|\S'
>>> re.findall(pattern, text)
['Tokenisera', 'mig', '!']
```

Syntax

Hur skapar jag en regex?

Ett tecken matchar sig själv.	a	S:t Lars Kyrka
…samma gäller en sekvens av tecken.	banan	Jag äter en banan för att jag älskar bananer.
Ett lodstreck (|) är ett logiskt ELLER	a|å|ä	Måns äter bananer.
Hakparenteser[ ]: Matchar ett av tecknen innanför hakarna	S[oö]der	Söder Soder Sder Soöder
Vanliga parenteser skapar en grupp.	S(o|oe|ö)der	Söder Soeder Sder Soöder
Cirkumflex (^) tar inversen av en teckenklass	S[^oö]der	Söder Suder Spder

Operatorer

- Följande operatorer skrivs direkt efter det som ska upprepas.
  - ? Matchar 0 eller 1: colou?r —> colour | color
  - * Matchar 0 eller flera: Go*gle —> Ggle | Gogle | Google | Gooogle | …
  - + Matchar minst 1: Go+gle —> Gogle | Google | Gooogle | …
  - {2,4} Matchar 2-4: Go{2,4}gle —> Google | Gooogle | Goooogle

Speciella teckensekvenser

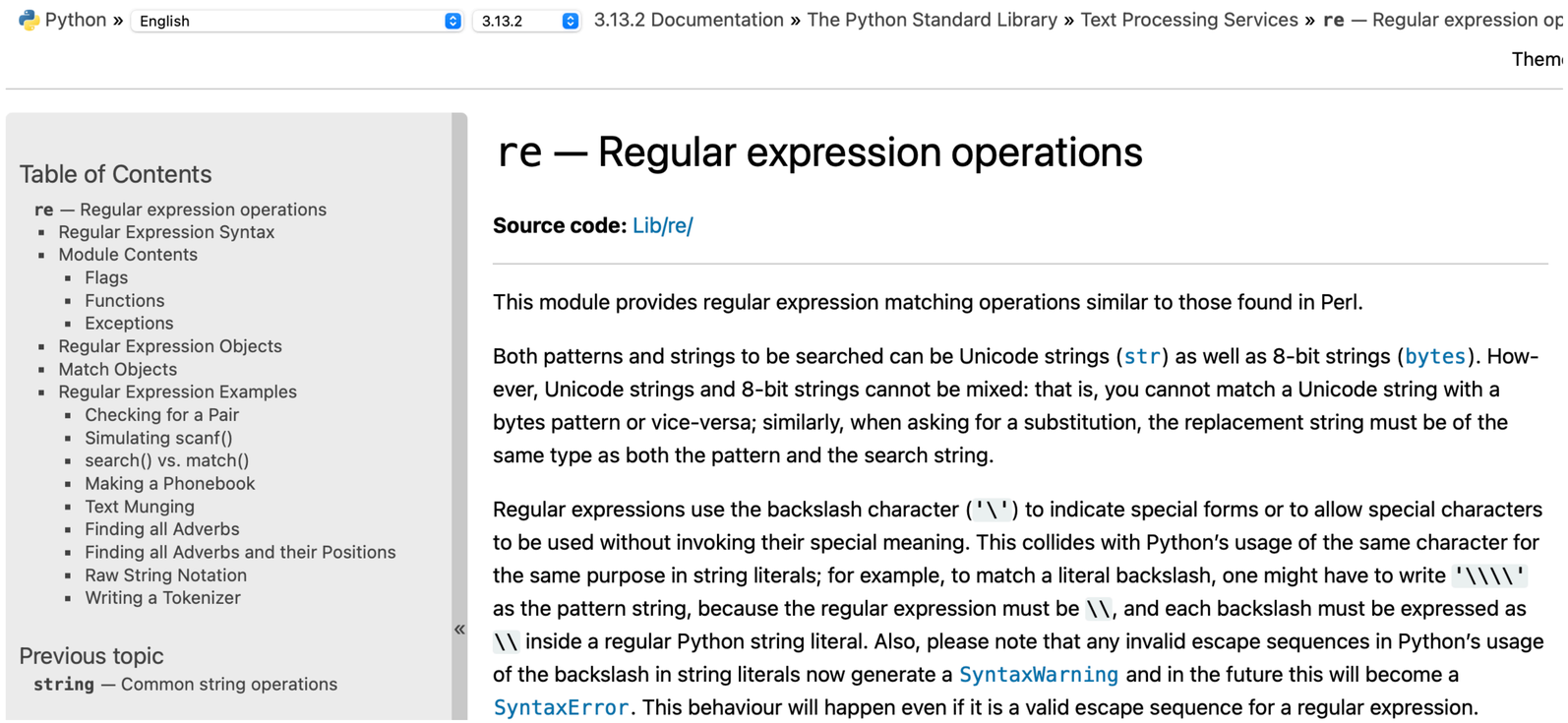
- \w Alla alfanumeriska symboler: [A-Za-z0-9_]
- \W Allt förutom dem alfanumeriska symbolerna:[^A-Za-z0-9_]
- \s Det vita tecknet
- \S Allt förutom det vita tecknet
- \d Alla siffor: [0-9]
- \D Allt förutom siffrorna: [^0-9]

Matcha operatortecken

- För att matcha någon av de speciella tecken ({}[]()^$.|*+?\ ) behöver man en backslash:
    - Punkten **.** skulle matcha alla möjliga tecken. Om man vill bara matcha själva punkten måste man skriva: **\.**

Användning i Python

re



- Modulen re finns i Pythons standardbibliotek
- Dokumentation: https://docs.python.org/3/library/re.html

Funktioner i re

- re.match(pattern, string): Matchar *början* av strängen mot ett mönster. Om matchning hittas returneras ett matchobjekt, annars *None*.
- re.search(pattern, string): Söker efter det första stället där mönstret matchar *någonstans* i strängen. Returnerar ett matchobjekt eller *None*.
- **re.findall(pattern, string): Returnerar en lista med *alla* förekomster av mönstret i strängen.**
- re.sub(pattern, replacement, string): Ersätter alla matchningar av mönstret i strängen med *replacement*.

re.findall()

```
import re

text = "Min e-post är test@example.com och mitt postnummer är 12345. \
    Min första epostaddress var blah-blah@anothertest.se"

regex_mail = r'[\w_-]+@[\w_-]+\.[A-Za-z]{2,}'
emails = re.findall(regex_mail, text)
print("E-postadresser:", emails)
```

✓ 0.0s

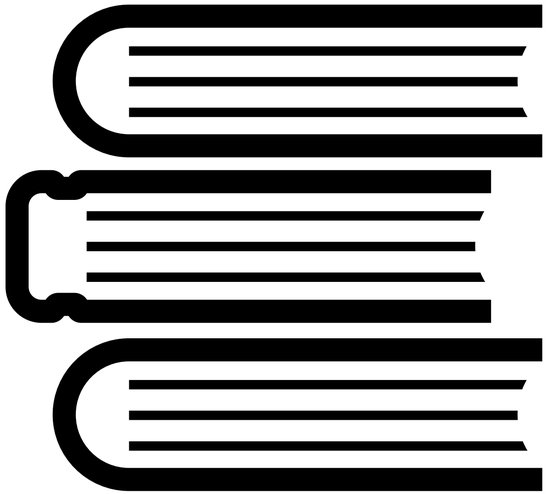
```
E-postadresser: ['test@example.com', 'blah-blah@anothertest.se']
```

Mail-Regex: Saknas det något?

- regex_mail = r'[\w_-]+@[\w_-]+\.[A-Za-z]{2,}'
  - [\w_-]+
  - @
  - [\w_-]+
  - \.
  - [A-Za-z]{2,}

Statistisk korpusanalys

Att jobba med en korpus



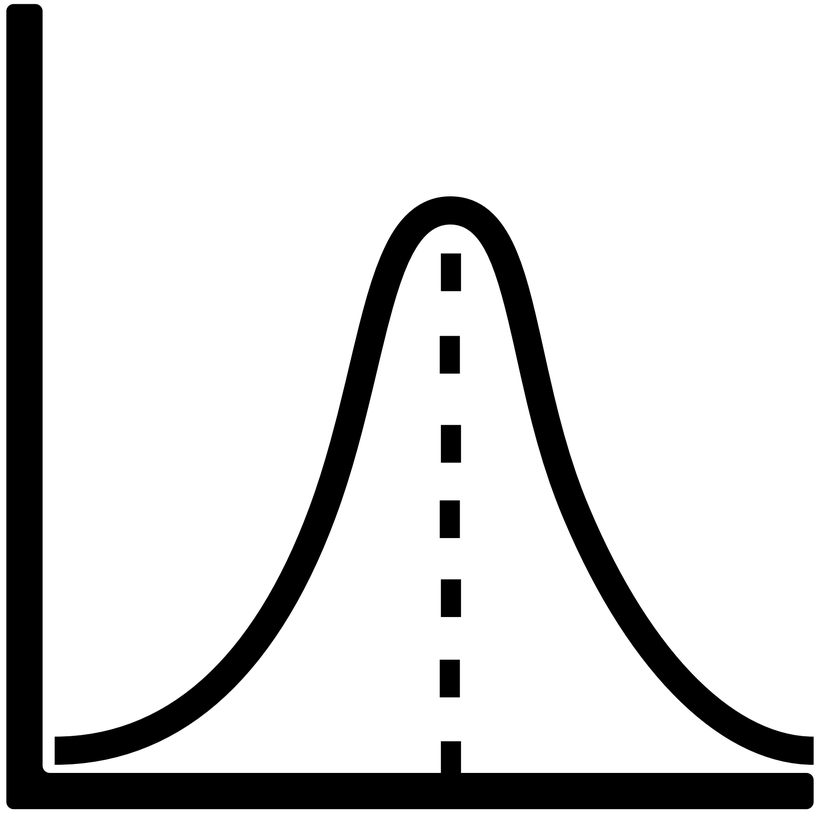
Viktiga frågor att ställa sig (även i labben):

- Hur stor är korpusen?
- …och hur stor måste den vara för uppgiften?
- Hur divers är korpusen?
- Hur representativ är korpusen, eller vad är den representativ för?

Statistiska Evaluationer

- Enklast: Titta på ord i en korpus.
    - Hur många finns det?
    - …och hur många olika finns det?
    - Hur är de fördelade?

- …men vad är det ens, ett ord?



Type Token Ratio (1)

A rose is a rose is a rose

(Gertrude Stein)

- Tre ordformer/types (*a*, *rose*, *is*).
- Åtta ord/tokens (*A*, *rose*, *is*, *a*, *rose*, *is*, *a*, *rose*).
- TTR: $\frac{3}{8} + 0.375$

Type Token Ratio (2)

- Vad betyder en högre TTR?
    - Mer **variation**.
    - Högre **komplexitet**.

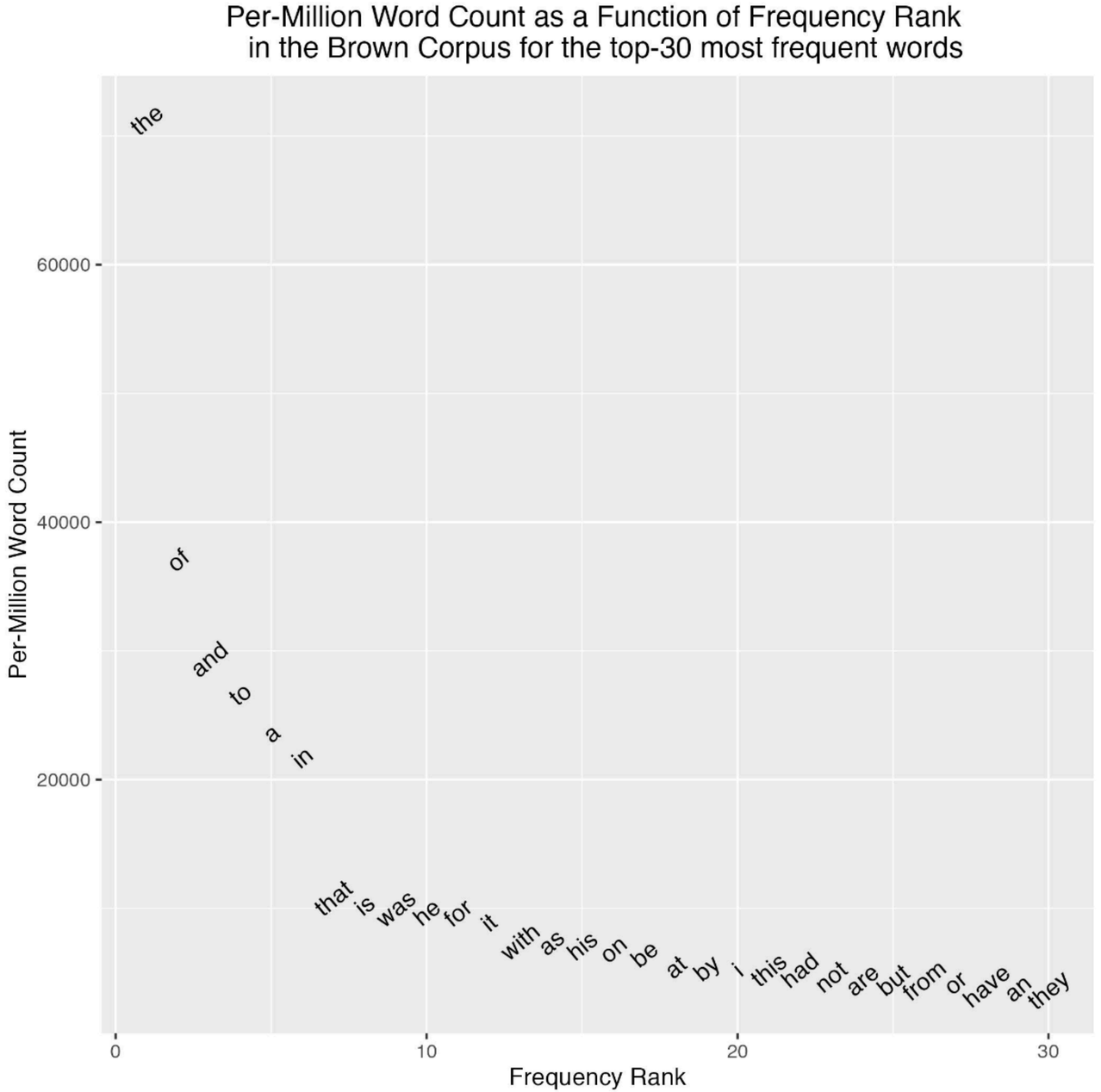
- TTR i konversationer brukar vara lägre än i text.
    - Varför?

Zipf’s Power Law (1)

- Frekvensen av ett visst ord är **omvänt proportionell** mot ordets frekvensranking.
- Det mest frekventa ordet inträffar ungefär dubbelt så ofta som det näst vanligaste ordet, tre gånger så ofta som det tredje vanligaste ordet, etc.
- $$\text{Frekvens} \propto \frac{1}{\text{Frekvensranking}}$$

Zipf’s Power Law (2)

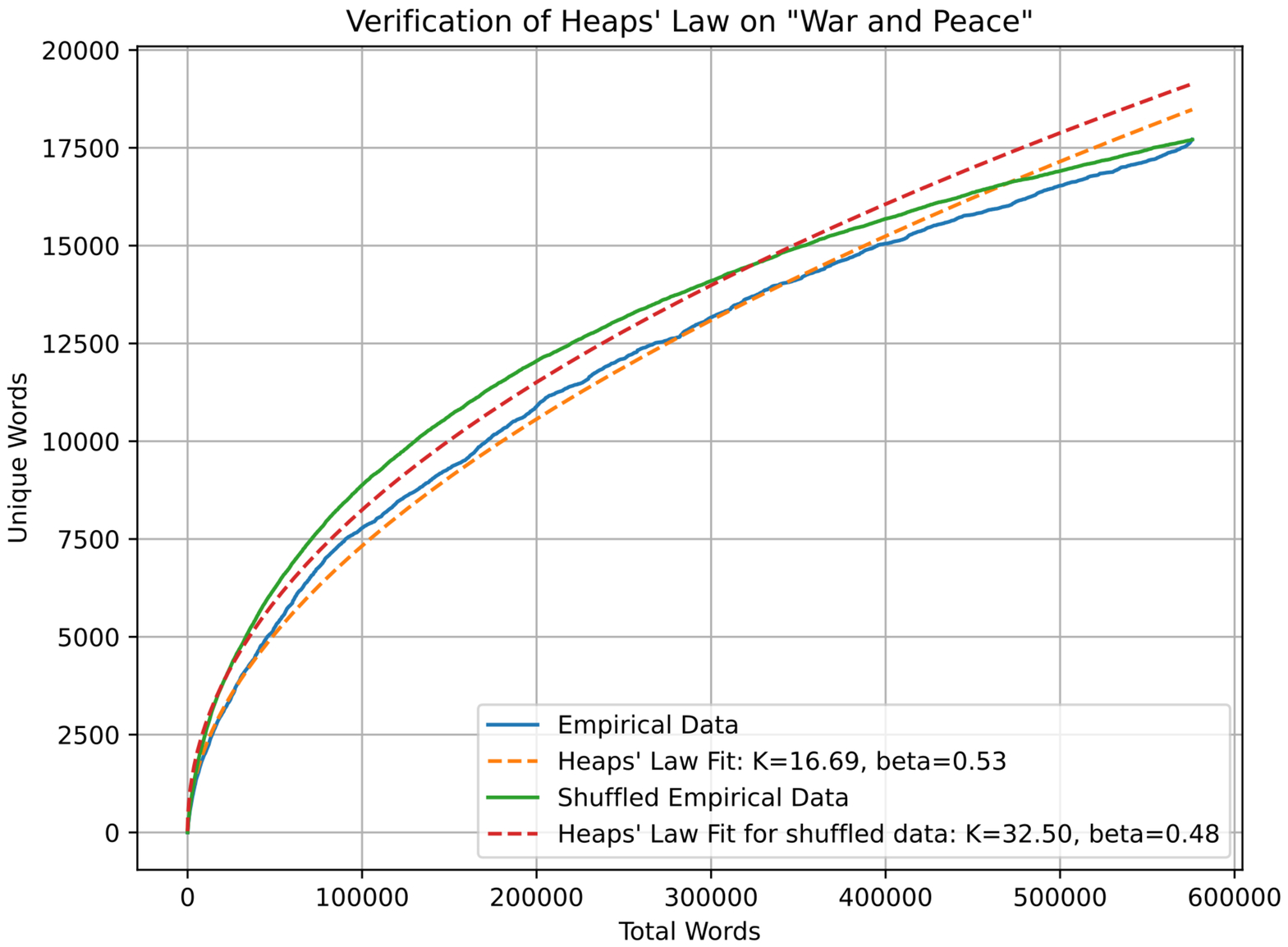
- I Browns korpusen (modern engelska):
  - ”the" är det mest frekventa ordet med en förekomst på 6,9%,
  - ”of” står för drygt 3,6%,
  - "and" står för 2,8%.



Från Glass et al. (2024): Language and Computers

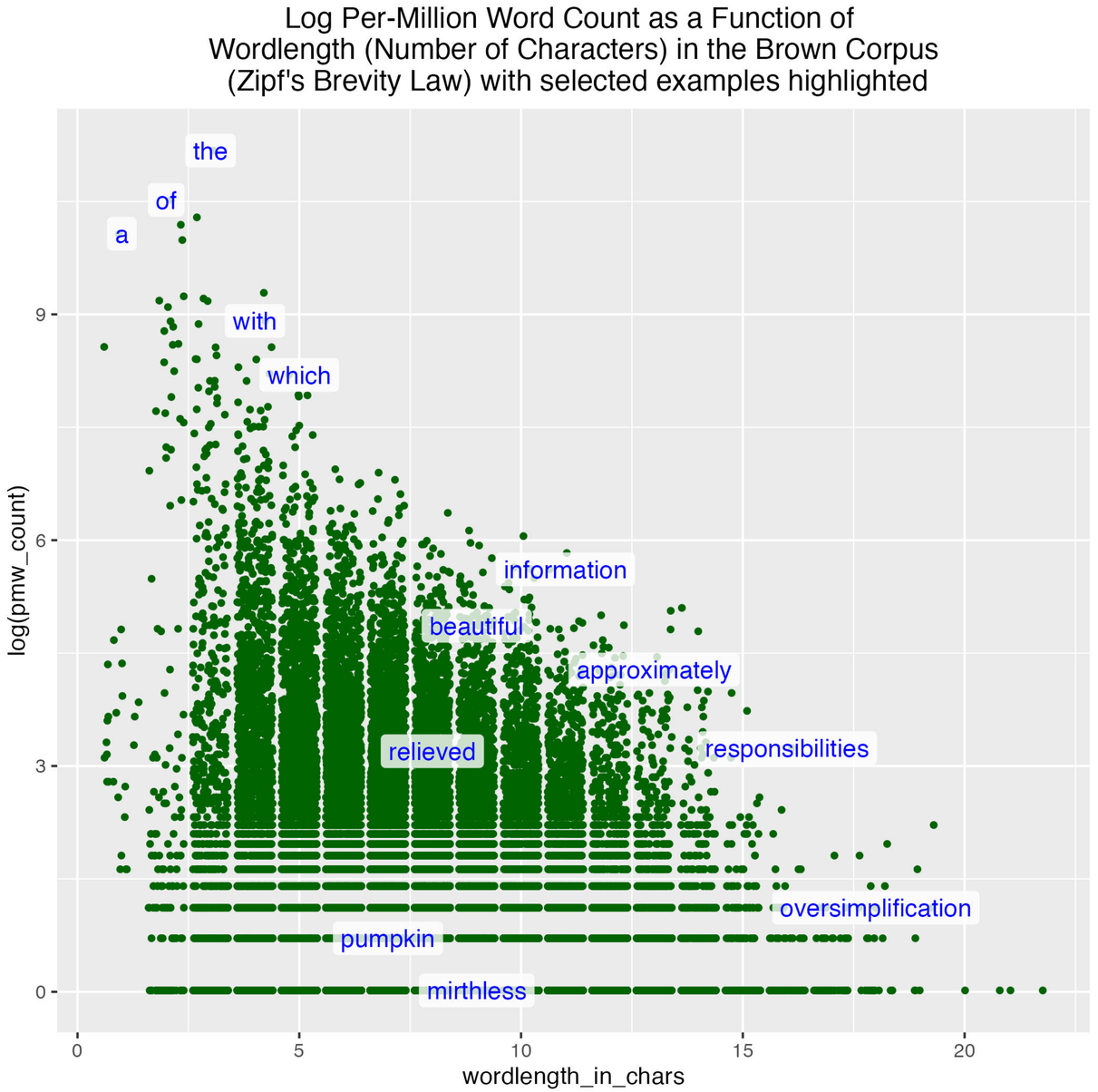
Heap’s Lag

- Antalet ordformer i en korpus är en funktion av korpusens storlek.
- I Brown korpusen förekommer 38% av alla ordformer bara én gång!
- 14% förekommer två gånger.



Zipf’s Brevity Law

- Mer frekventa ord brukar vara kortare.
- Mindre vanliga ord brukar vara längre.
- De vanligaste orden i Brown Corpus: *the*, *be* (i sina olika former), *to*, *of*, *a*.



L1

Tack för idag!

Glöm inte att göra uppgifterna på https://regexone.com/!