Text Mining “Re-” in Victorian Poetry

Adam Mazel, Digital Publishing Librarian
Scholarly Communication Department, IUB Libraries

2023-11-08

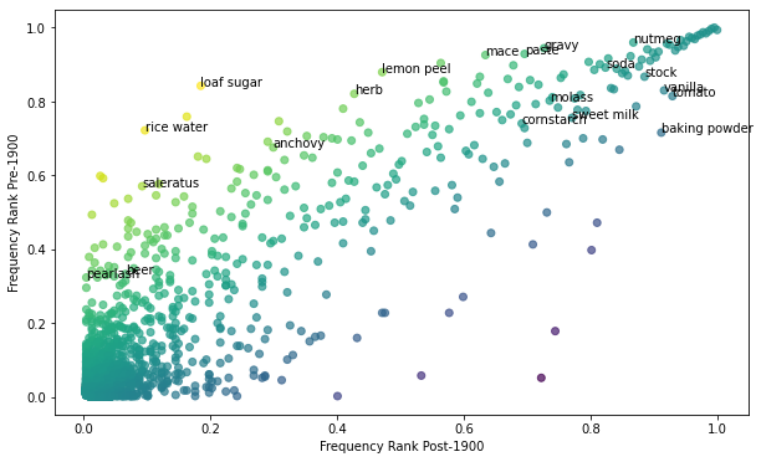
Why “Re-” in VP?

- NAVSA Conference 2023: “**Re**vision, **Re**turn, **Re**form”
  - Panel: “Re: Re: Victorian Poetry”
    - Adela Pinch, Professor of English, University of Michigan
    - Emily Harrington, Assoc. Prof. of English, UC Boulder
    - Naomi Levine, Asst. Prof. of English, Yale University
    - Me!

NAVSA 2023 Conference banner excerpted from conference website

Methodology

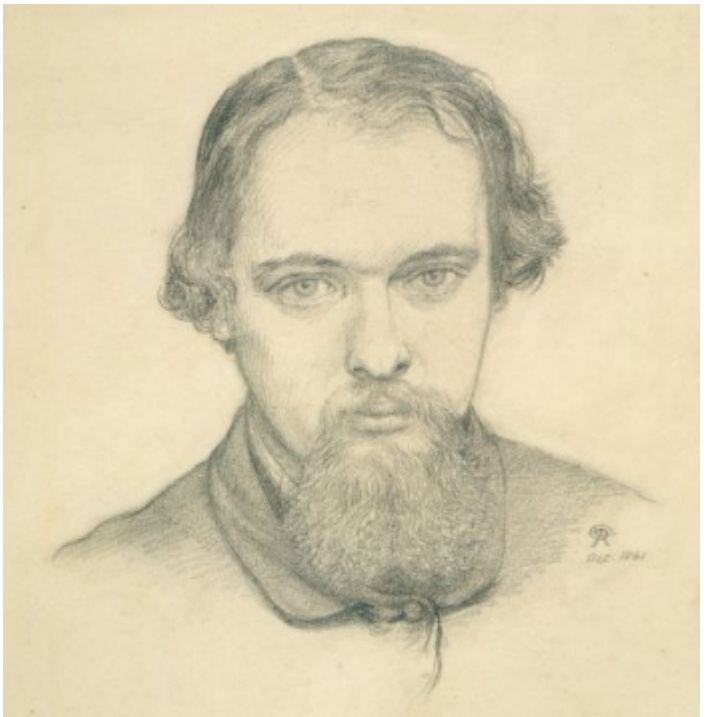
- “Re-” 🤝 Text Mining
- Python
  - Keyword Frequency
  - Key Word in Context (KWIC)
  - Term Frequency over Time
  - Sentiment Analysis
- Exploratory Data Analysis



Python logo used with permission

Data: Dante Gabriel Rossetti

- *Poems* (1870)
- *Poems: A New Edition* (1881)
- *Ballads and Sonnets* (1881)



Self Portrait, 1861

Image from the Rossetti Archive in public domain in US

Data: Algernon Charles Swinburne

- *Atalanta in Calydon* (1865)
- *Poems and Ballads* (1866)
- *Songs Before Sunrise* (1871)
- *Songs of Two Nations* (1875)
- *Erechtheus* (1876)
- *Poems and Ballads, Second Series* (1878)
- *Songs of the Springtides* (1880)
- *Studies in Song* (1880)
- *The Heptalogia, or the Seven against Sense. A Cap with Seven Bells* (1880)
- *Tristram of Lyonesse* (1882)
- *A Century of Roundels* (1883)
- *A Midsummer Holiday and Other Poems* (1884)
- *Poems and Ballads, Third Series* (1889)
- *Astrophel and Other Poems* (1894)
- *The Tale of Balen* (1896)
- *A Channel Passage and Other Poems* (1904)

Data: Michael Field

- *Long Ago* (1889)
- *Sight and Song* (1892)
- *Underneath the Bough* (1893)
- *Wild Honey from Various Thyme* (1908)
- *Poems of Adoration* (1912)
- *Mystic Trees* (1913)
- *Whym Chow: Flame of Love* (1914)

Katherine Bradley & Edith Cooper, aka Michael Field

Image from Wikimedia Commons in public domain in US

Data: Thomas Hardy

- *Wessex Poems and Other Verses* (1898)
- *Poems of the Past and the Present* (1901)
- *Time's Laughingstocks and Other Verses* (1909)
- *Satires of Circumstance* (1914)
- *Moments of Vision* (1917)
- *Late Lyrics and Earlier with Many Other Verses* (1922)
- *Human Shows, Far Phantasies, Songs and Trifles* (1925)

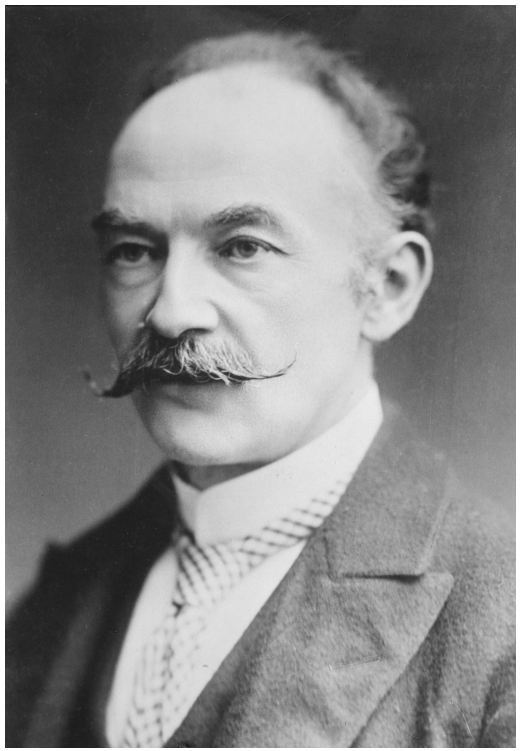


Image from Wikimedia Commons in public domain in US

Who Uses “Re-” Words More / Less?

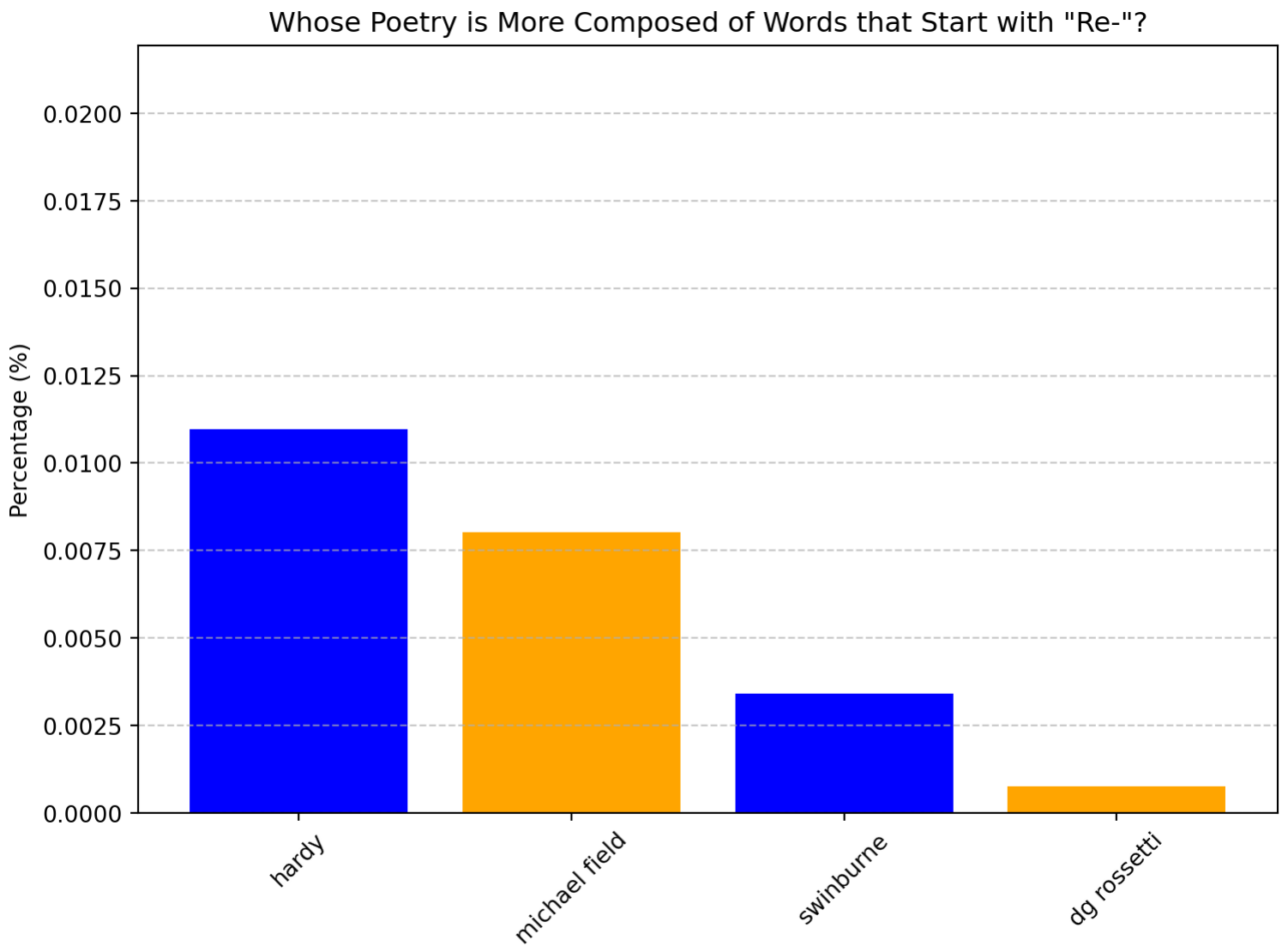
- Keyword Frequency Analysis
  - What percent of each poet’s corpus is composed of words with the prefix “re-”?
  - Compare “Re-” Word Percentages

▼ Code

```
# import software libraries
import nltk
import os
import matplotlib.pyplot as plt

# Download NLTK tokenizer data
nltk.download('punkt')

# Define a function that tokenizes a text and then counts how many of its w
def count_re_words(text):
    words = nltk.word_tokenize(text)
```

```
    return sum(1 for word in words if word.lower().startswith("re-"))

# Specify the directory paths for the two poets' corpora
corpus_directories = {
    'swinburne': '/home/adammazel/Documents/Digital_Scholarship/re-victoria
    'hardy': '/home/adammazel/Documents/Digital_Scholarship/re-victorian-po
    'michael field': '/home/adammazel/Documents/Digital_Scholarship/re-vict
    'dg rossetti': '/home/adammazel/Documents/Digital_Scholarship/re-victor
```

Who Uses “Re-” Words More / Less?



Which “Re-” Words Are Most Frequent?

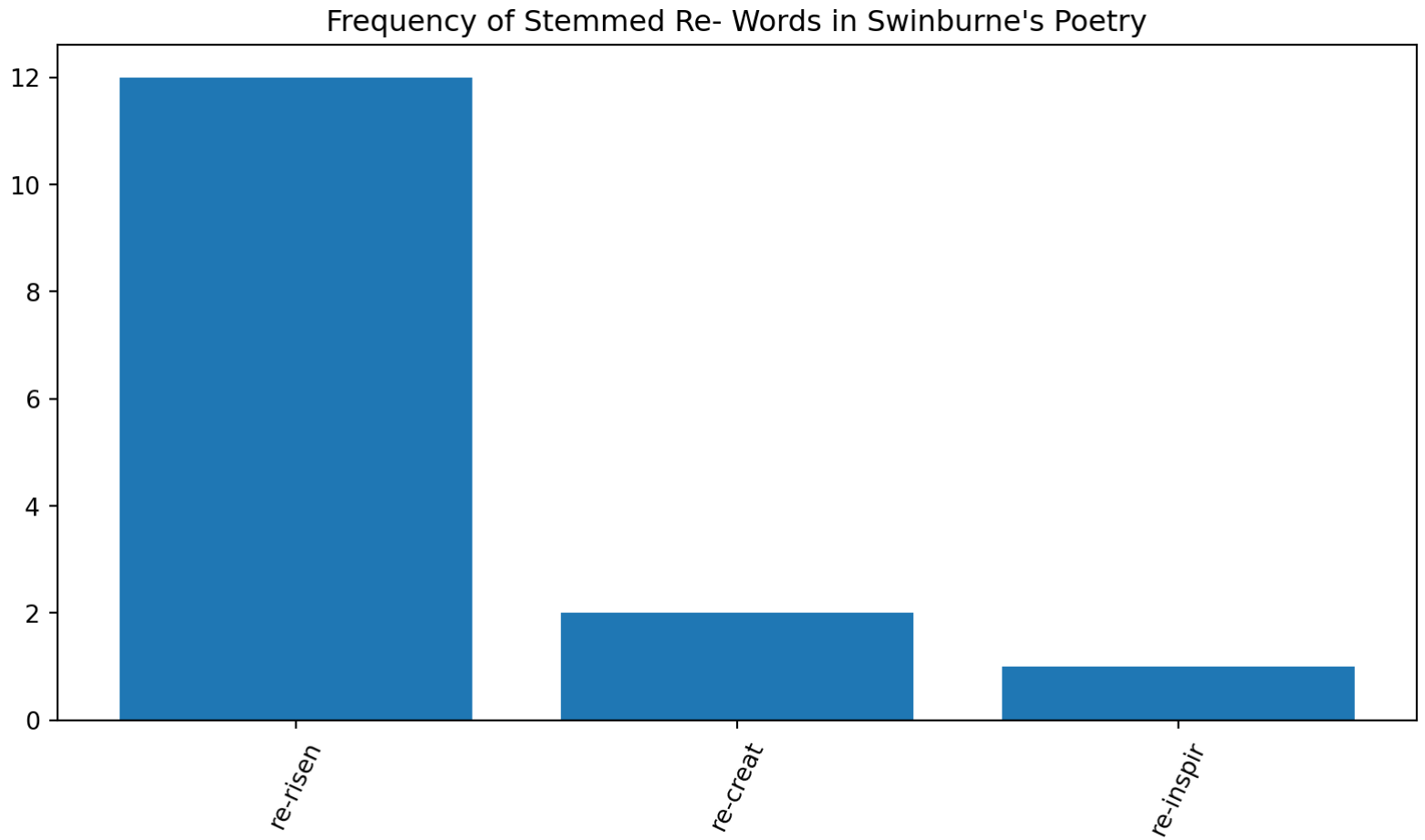
- Keyword Frequency Analysis
  - Which “re-” words are used and how often?

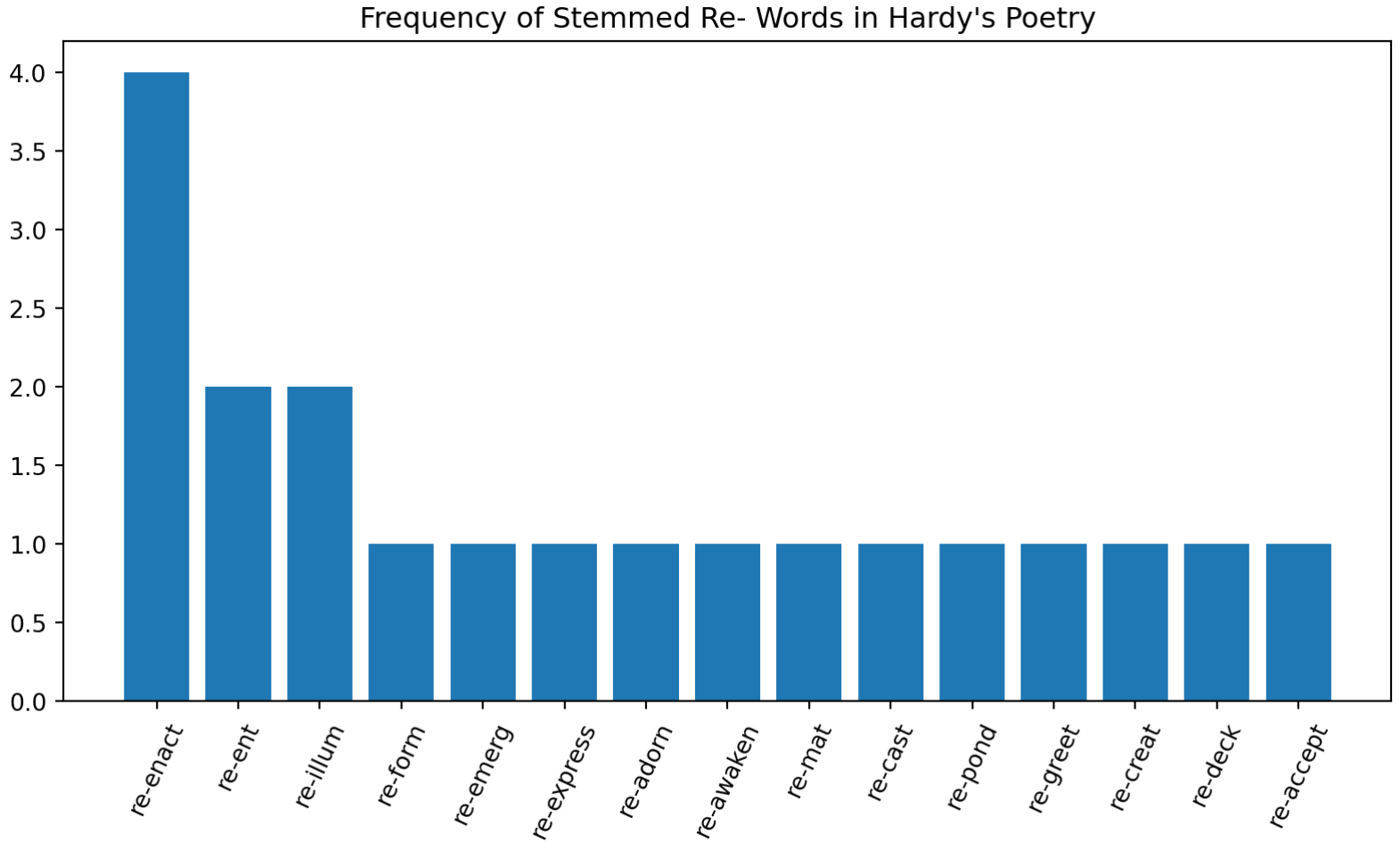
Code

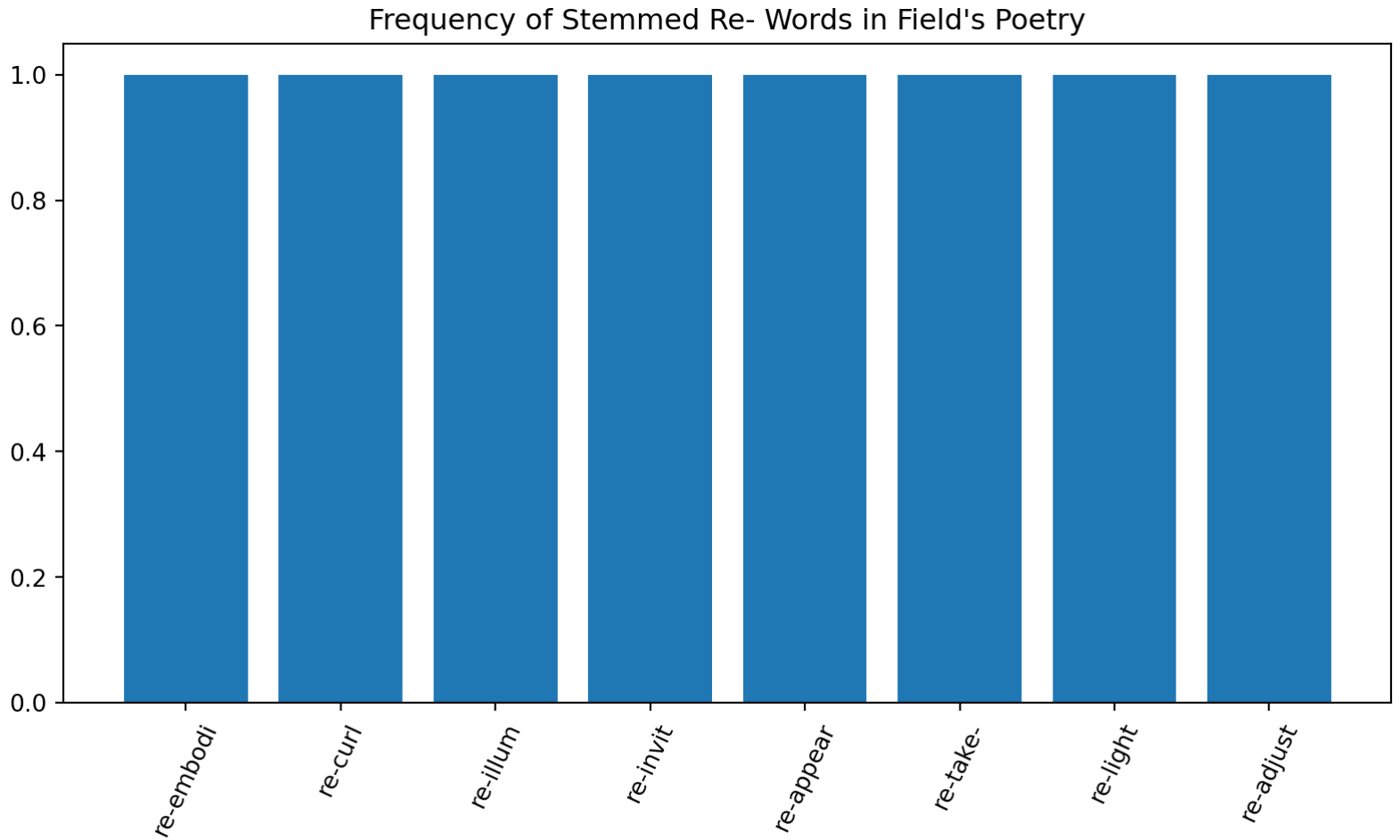
```
# import software libraries / dependencies
import nltk
import os
import re
import matplotlib.pyplot as plt
from collections import Counter
from nltk.stem import SnowballStemmer

# Download NLTK tokenizer data
nltk.download('punkt')

# Initialize Stemmer for English
```

```
stemmer = SnowballStemmer("english")

# create function to preprocess and process files of each directory: create
def process_directory(corpus_directory):
    corpus = []

```

Which “Re-” Words Are Most Frequent?




Frequency of Stemmed Re- Words in Hardy's Poetry
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
re-enact
re-ent
re-illum
re-form
re-emerg
re-express
re-adorn
re-awaken
re-mat
re-cast
re-pond
re-greet
re-creat
re-deck
re-accept


Frequency of Stemmed Re- Words in Field's Poetry
1.0
0.8
0.6
0.4
0.2
0.0
re-embodi
re-curl
re-illum
re-invit
re-appear
re-take-
re-light
re-adjust

Field's and Hardy's "Re-Illume" in Context

- Re-illume
  - Extremely rare
  - Chiefly poetic
  - 1758 – present
- Field: 1x
  - "XXII: Sleeping together: Sleep": *Whym Chow: Flame of Love* (1906, 1914)
- Hardy: 2x
  - "Two Rosalinds": *Time's Laughingstocks and Other Verses* (1909)
  - "For Life I had never cared greatly": *Moments of Vision and Miscellaneous Verses* (1917)

Google Books Ngram Viewer

reillumed+reillume+reilluming+reillumes

1779 - 2019 | British English (2019) | Case-Insensitive | Smoothing

(click on line/label for focus)

Search in Google Books

reillumes	1779 - 1800	1801 - 1839	1840	1841 - 1949	1950 - 2019	English (2019)
reilluming	1779 - 1820	1821 - 1835	1836 - 1839	1840 - 1958	1959 - 2019	English (2019)

Do “Re-” Words and “Re” Words Co-occur in Hardy’s Poetry?

- Key (Re-) Word in Context (KWIC)
- Have Python return Hardy’s sentences that contain word(s) that start with “re-” and word(s) that start with “re”

▼ Code

```
# import libraries
import os # open directories on local machine
import nltk # nlp
import re # reg ex
from nltk.tokenize import sent_tokenize, word_tokenize # break text blob in

# Define regular expressions
re_pattern = r'\b(?:[Rr]e|[Rr]E)\w+\b'  # Matches "re" or "Re" or "rE" or "
re_hyphen_pattern = r'\b(?:[Rr]e-|[Rr]E-)\w+\b'  # Matches "re-" or "Re-" o
# \b matches a word boundary.
```

```
# (?:[Rr]e|[Rr]E) is a non-capturing group that matches either "re" or "Re"
# \w+ matches one or more word characters following "re" or "Re" or "rE" or

def matches_pattern(word, pattern):
    return re.search(pattern, word, re.IGNORECASE) is not None

# Define function to surround matched words with asterisks
def bold_matched_words(match, pattern):
```

Do “Re-” Words and “Re” Words Co-occur in Hardy’s Poetry?

```
Within there
   Too mocking to Love’s **re-expression**
      Was Time’s **repartee**!



IX

      “The words, sir?” cried a creature
   Hovering mid the shine and shade as ’twixt the live world and the
   tomb;
   But the well-known numbers needed not for me a text or teacher
      To **revive** and **re-illume**.
```

When Are “Re-” Words More / Less Frequent?

- Time Series / Term Frequency over Time
  - Y axis: re- word percentage of book
  - X axis: publication year of book

Code

```
# Import software libraries
import nltk # nlp
import os # interact with directories on local machine
import re # reg ex
import matplotlib.pyplot as plt # data visualization
import string # punctuation removal

# Download NLTK tokenizer data
nltk.download('punkt')

```

```
# Define a function that takes each text, tokenizes it into individual word
def count_re_words(text):
    words = nltk.word_tokenize(text)
    return sum(1 for word in words if word.lower().startswith("re-"))

# Define a function to remove all punctuation except hyphens
def remove_punctuation_except_hyphens(text):
    translator = str.maketrans('', '', string.punctuation.replace('-', ''))
```

When Are “Re-” Words More / Less Frequent?

Are “Re-” Words Used Positively or Negatively?

- Sentiment Analysis
  - determines a text’s emotional tone (positive / negative / neutral)

▼ Code

```
# import software libraries
import nltk #nlp
from nltk.sentiment.vader import SentimentIntensityAnalyzer # VADER sentime
import os # enable engagement with directories and files on local machine

# Initialize the VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Function to calculate aggregate sentiment (positive / neutral / negative)
# start counts at zero
```

```
def calculate_aggregate_sentiment(sentences):
    positive_score = 0
    negative_score = 0
    neutral_score = 0
    total_sentences = 0

# for each sentence, calculate sentiment polarity score: if score is positi
    for sentence in sentences:
```

Are “Re-” Words Used Positively or Negatively?

```
AC Swinburne
Total Sentences Analyzed: 15
Average Positive Score: 0.1239333333333335
Average Negative Score: 0.152
Average Neutral Score: 0.7242666666666665
Overall Sentiment: Negative

Thomas Hardy
Total Sentences Analyzed: 21
Average Positive Score: 0.0890476190476190 4
Average Negative Score: 0.1130476190476190 4
Average Neutral Score: 0.797904761904761 9
Overall Sentiment: Negative

Michael Field
Total Sentences Analyzed: 8
```

Conclusion

- Text mining morphemes can:
  - illuminate significant micro-elements of poetic style
  - show how seeminly trivial features contribute to poetic meaning
- Distant Reading 🤝 Close Reading

- **Thank You!**
- Contact Info
  - Adam Mazel
  - Digital Publishing Librarian
  - Indiana University Bloomington
  - amazel@iu.edu