Cultural Analytics

ENGL 64.05

Fall 2019

Prof. James E. Dobson

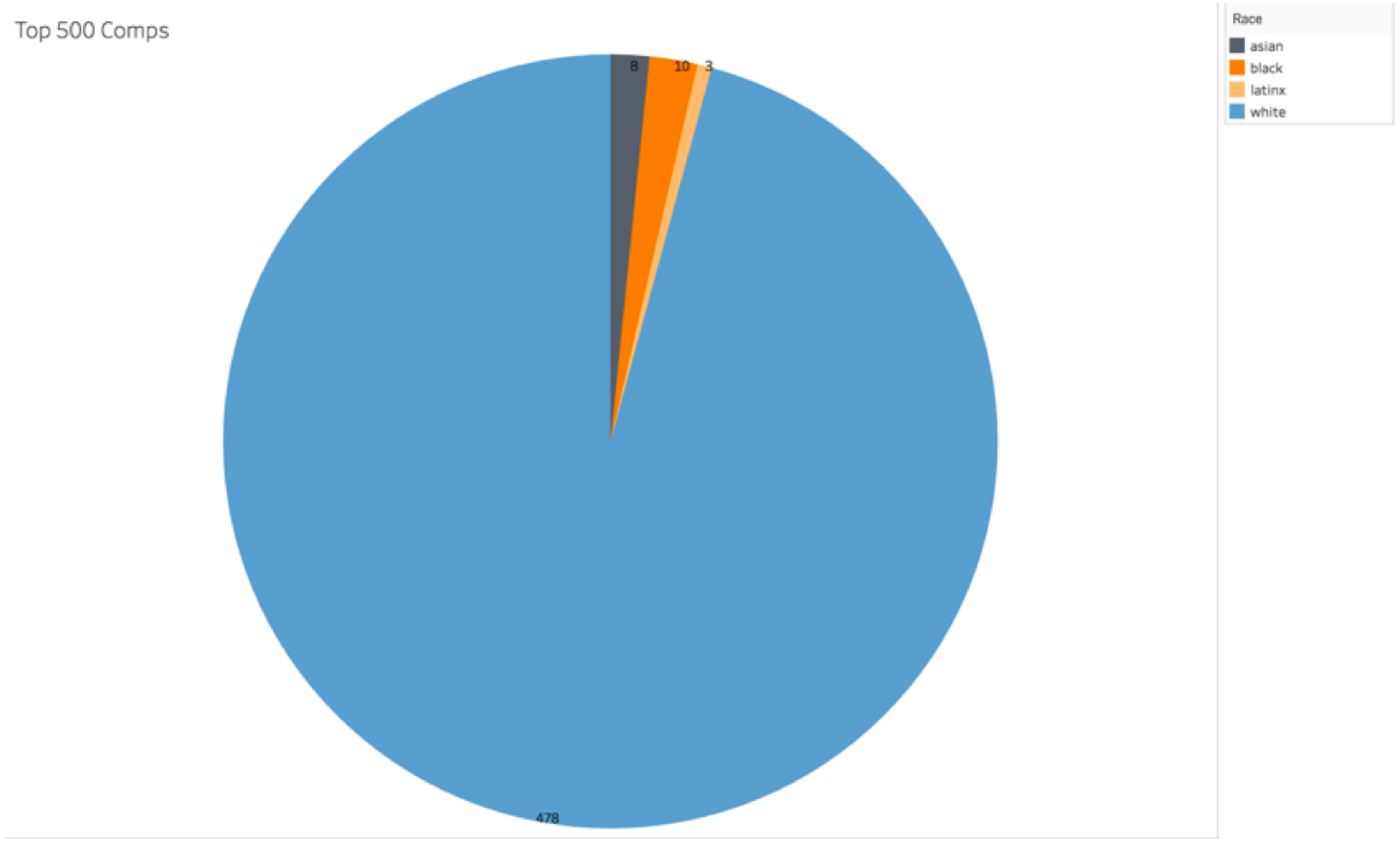
October 7, 2019

Laura B. McGrath
"Comping White"

- What is her argument?
- What is her archive?
- What is her method?

"From 2013 to 2019, publishers identified 31,876 comps — about three comp titles for every new title published. I wanted to know which comps get cited most frequently (and, by extension, communicate high value), so I winnowed the list of 30,000-plus comp titles down to the top 50 most frequently used comps. Because many books have been cited as comps the same number of times, this list is actually comprised of 225 titles. I then worked with my undergraduate research assistant, Jonathan Morales, to research the race of each of the authors whose books are listed here."

Cultural Analytics

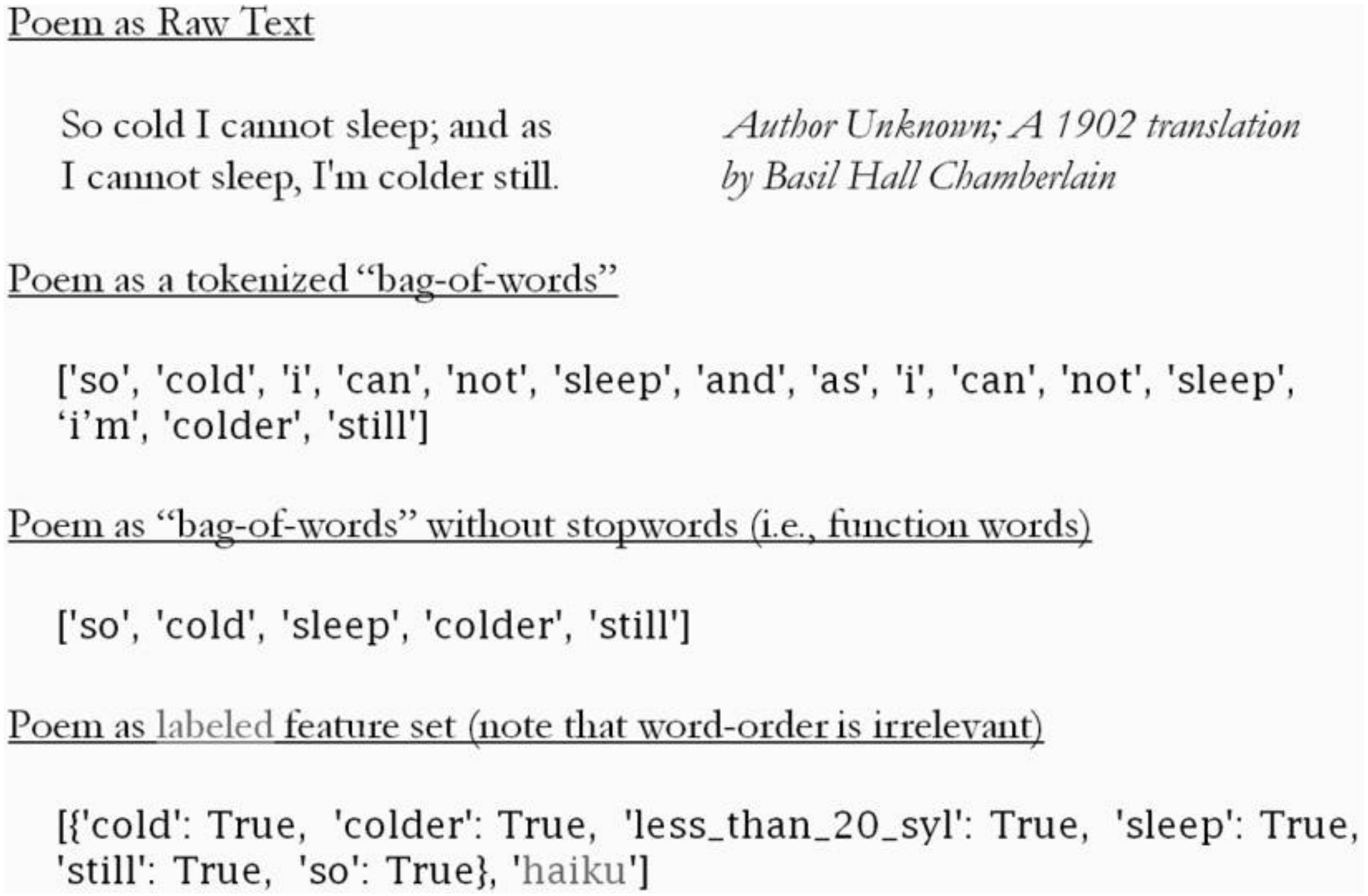


Hoyt Long and Richard Jean So "Literary Pattern Recognition"

- What is their argument?
- What is their archive?
- What is their method?

"The Tasks"

- Categorization: haiku/nonhaiku
- Representation: labeled (binary) feature set from lemmatized nouns
- Learning: Naïve Bayes training
- Classification: Naïve Bayes testing



Poem as Raw Text

So cold I cannot sleep; and as
I cannot sleep, I'm colder still.

Author Unknown; A 1902 translation by Basil Hall Chamberlain

Poem as a tokenized "bag-of-words"

```
['so', 'cold', 'i', 'can', 'not', 'sleep', 'and', 'as', 'i', 'can', 'not', 'sleep',
'i'm', 'colder', 'still']
```

Poem as "bag-of-words" without stopwords (i.e., function words)

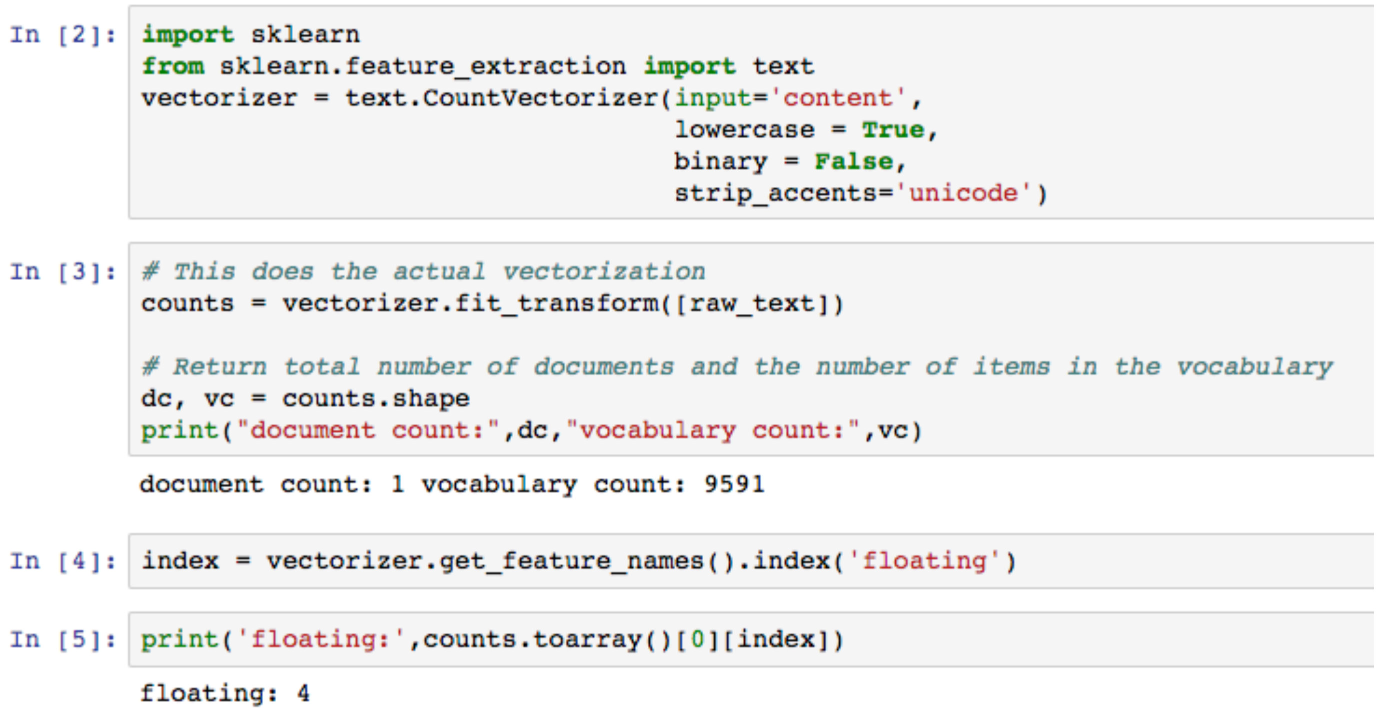
```
['so', 'cold', 'sleep', 'colder', 'still']
```

Poem as labeled feature set (note that word-order is irrelevant)

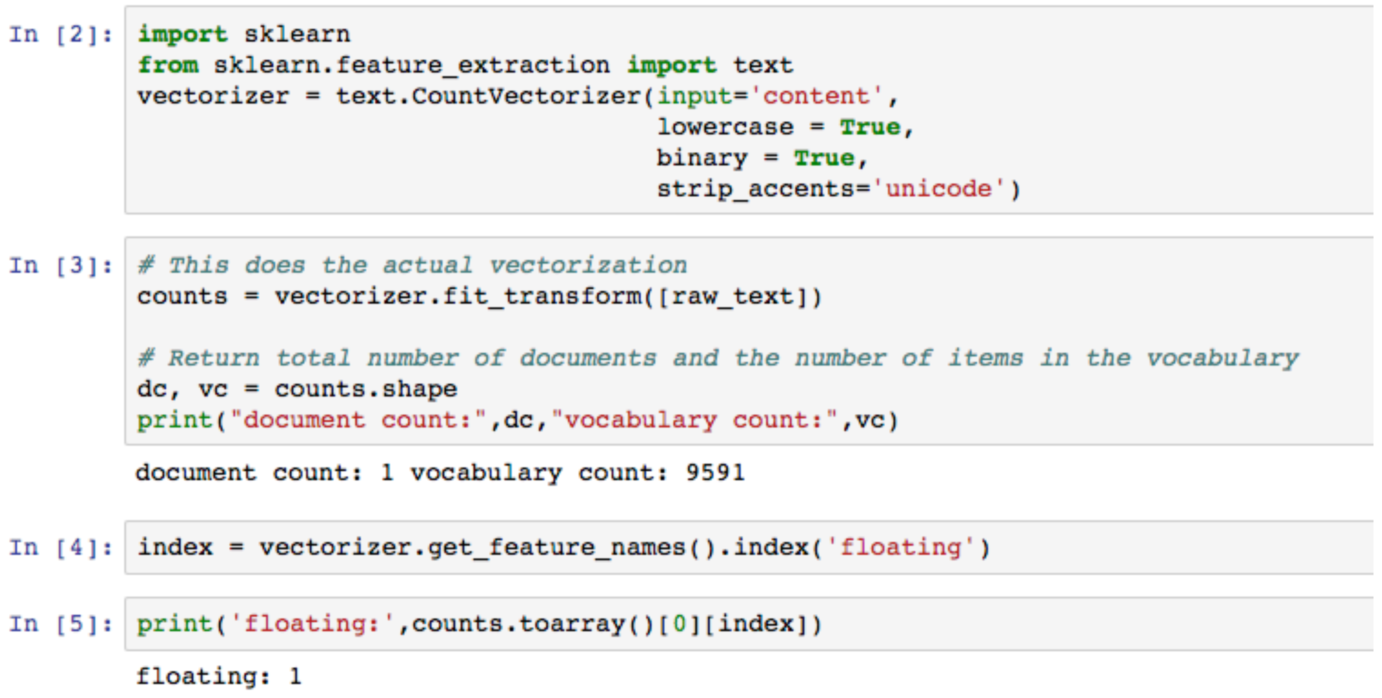
```
[{'cold': True,  'colder': True,  'less_than_20_syl': True,  'sleep': True,
'still': True,  'so': True}, 'haiku']
```

Long and So, "Literary Pattern Recognition," figure 4, pg. 255

Binary Vectors



```
In [2]: import sklearn
        from sklearn.feature_extraction import text
        vectorizer = text.CountVectorizer(input='content',
                                          lowercase = True,
                                          binary = False,
                                          strip_accents='unicode')

In [3]: # This does the actual vectorization
        counts = vectorizer.fit_transform([raw_text])

        # Return total number of documents and the number of items in the vocabulary
        dc, vc = counts.shape
        print("document count:",dc,"vocabulary count:",vc)

        document count: 1 vocabulary count: 9591

In [4]: index = vectorizer.get_feature_names().index('floating')

In [5]: print('floating:',counts.toarray()[0][index])

        floating: 4
```



```
In [2]: import sklearn
        from sklearn.feature_extraction import text
        vectorizer = text.CountVectorizer(input='content',
                                          lowercase = True,
                                          binary = True,
                                          strip_accents='unicode')

In [3]: # This does the actual vectorization
        counts = vectorizer.fit_transform([raw_text])

        # Return total number of documents and the number of items in the vocabulary
        dc, vc = counts.shape
        print("document count:",dc,"vocabulary count:",vc)

        document count: 1 vocabulary count: 9591

In [4]: index = vectorizer.get_feature_names().index('floating')

In [5]: print('floating:',counts.toarray()[0][index])

        floating: 1
```

Akkana Peck's countsyl.py

```python
def count_syllables(word):
    vowels = ['a', 'e', 'i', 'o', 'u']
    on_vowel = False
    in_diphthong = False
    minsyl = 0
    maxsyl = 0
    lastchar = None

    word = word.lower()
    for c in word:
        is_vowel = c in vowels
        if on_vowel == None:
            on_vowel = is_vowel
        # y is a special case
        if c == 'y':
            is_vowel = not on_vowel
        if is_vowel:
            if not on_vowel:
                # We weren't on a vowel before.
                # Seeing a new vowel bumps the syllable count.
                minsyl += 1
                maxsyl += 1
            elif on_vowel and not in_diphthong and c != lastchar:
                # We were already in a vowel.
                # Don't increment anything except the max count,
                # and only do that once per diphthong.
                in_diphthong = True
                maxsyl += 1
        on_vowel = is_vowel
        lastchar = c
    # Some special cases:
    if word[-1] == 'e':
        minsyl -= 1
    # if it ended with a consonant followed by y, count that as a syllable.
    if word[-1] == 'y' and not on_vowel:
        maxsyl += 1
    return minsyl
```

Billy Collins, Five Haiku

Slicing strawberries
this morning, I'm suddenly
slicing strawberries!

A twig in its beak,
a bird disappears into
the town's noon siren.

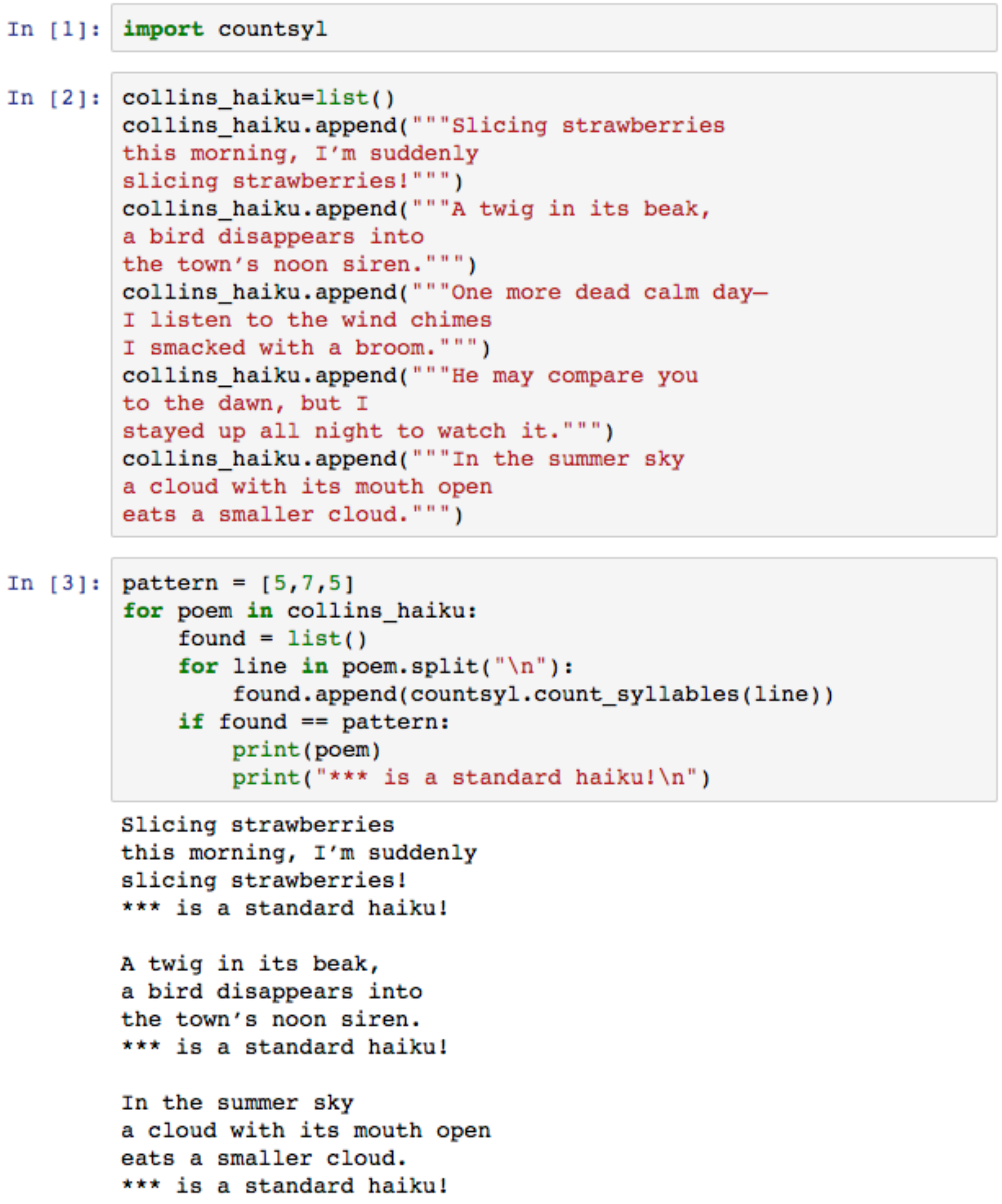
One more dead calm day—
I listen to the wind chimes
I smacked with a broom.

He may compare you
to the dawn, but I
stayed up all night to watch it.

In the summer sky
a cloud with its mouth open
eats a smaller cloud.

Billy Collins writes: "I follow the seventeen syllable limit because it provides me with a pleasurable feeling of push-back, a resistance to whatever literary whims I may have at the time. If you want to create a little flash of illumination the haiku tells us, start by counting on your fingers. A three-line poem with a frog is not necessarily a haiku."

Cultural Analytics



```
In [1]: import countsyl
```

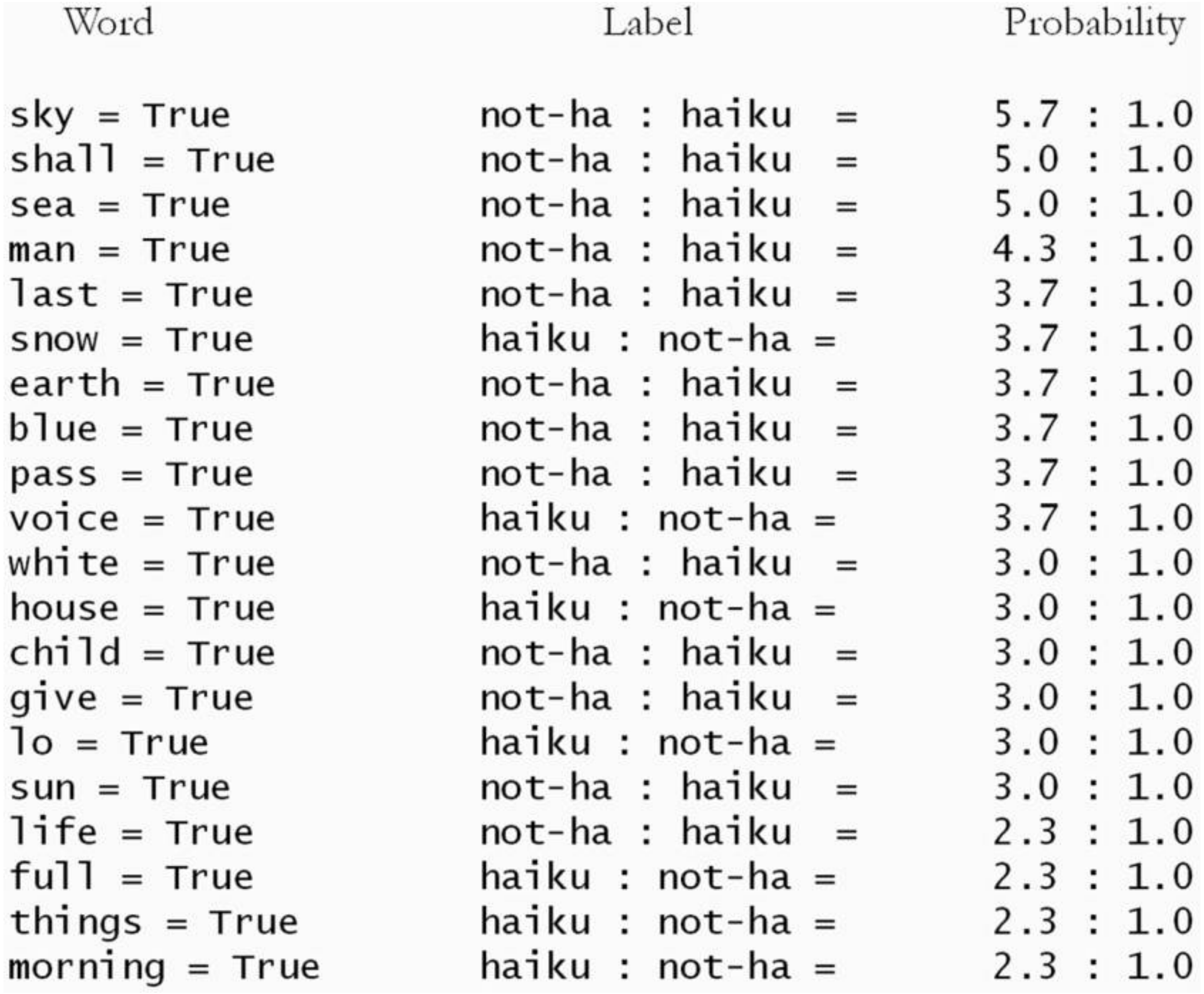
```
In [2]: collins_haiku=list()
        collins_haiku.append("""Slicing strawberries
        this morning, I'm suddenly
        slicing strawberries!""")
        collins_haiku.append("""A twig in its beak,
        a bird disappears into
        the town's noon siren.""")
        collins_haiku.append("""One more dead calm day—
        I listen to the wind chimes
        I smacked with a broom.""")
        collins_haiku.append("""He may compare you
        to the dawn, but I
        stayed up all night to watch it.""")
        collins_haiku.append("""In the summer sky
        a cloud with its mouth open
        eats a smaller cloud.""")
```

```
In [3]: pattern = [5,7,5]
        for poem in collins_haiku:
            found = list()
            for line in poem.split("\n"):
                found.append(countsyl.count_syllables(line))
            if found == pattern:
                print(poem)
                print("*** is a standard haiku!\n")
```

```
Slicing strawberries
this morning, I'm suddenly
slicing strawberries!
*** is a standard haiku!

A twig in its beak,
a bird disappears into
the town's noon siren.
*** is a standard haiku!

In the summer sky
a cloud with its mouth open
eats a smaller cloud.
*** is a standard haiku!
```

Strongest Features?

“Thus we repeated our tests without using syllable count as a feature, and this time included all words except function words (fig. 7). What we found is that accuracy scores dropped considerably, averaging 73 percent for the translations and 65 percent for the adaptations… It turns out that for many of the journals, scores decreased in large part because the classifier was misclassifying many more short poems as haiku. By expanding the set of features with which Naïve Bayes identified textual pattern, we got more misclassifications and thus more evidence of where our haiku and nonhaiku corpora overlap” (260-261).



```
   Word                       Label                     Probability

sky = True               not-ha : haiku  =          5.7 : 1.0
shall = True             not-ha : haiku  =          5.0 : 1.0
sea = True               not-ha : haiku  =          5.0 : 1.0
man = True               not-ha : haiku  =          4.3 : 1.0
last = True              not-ha : haiku  =          3.7 : 1.0
snow = True              haiku : not-ha =           3.7 : 1.0
earth = True             not-ha : haiku  =          3.7 : 1.0
blue = True              not-ha : haiku  =          3.7 : 1.0
pass = True              not-ha : haiku  =          3.7 : 1.0
voice = True             haiku : not-ha =           3.7 : 1.0
white = True             not-ha : haiku  =          3.0 : 1.0
house = True             haiku : not-ha =           3.0 : 1.0
child = True             not-ha : haiku  =          3.0 : 1.0
give = True              not-ha : haiku  =          3.0 : 1.0
lo = True                haiku : not-ha =           3.0 : 1.0
sun = True               not-ha : haiku  =          3.0 : 1.0
life = True              not-ha : haiku  =          2.3 : 1.0
full = True              haiku : not-ha =           2.3 : 1.0
things = True            haiku : not-ha =           2.3 : 1.0
morning = True           haiku : not-ha =           2.3 : 1.0
```