GEI1002

Computers and the humanities

Week 5
Working with text, Part I

Natural Language Processing (NLP)

NLP techniques are useful in many areas:

- Search engines
- Social media recommender systems

Often for practical goals:

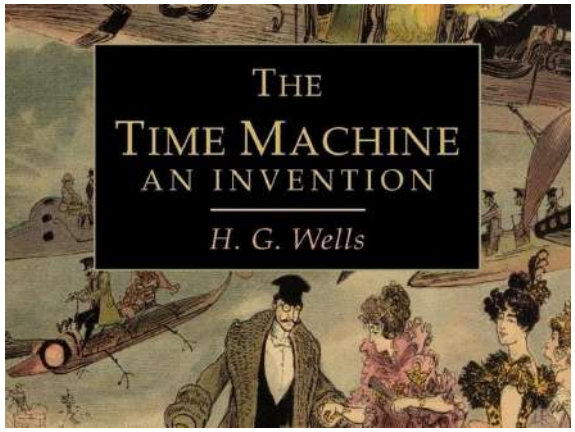
- Flagging spam
- Identify posts that might interest a user

In the Humanities and Social Sciences:

- Description, rather than prediction
- How novels have changed over time
- How a particular newsworthy event is covered differently across national contexts

Simple examples

The Time Machine: An Invention, a novel by H.G. Wells (1895)

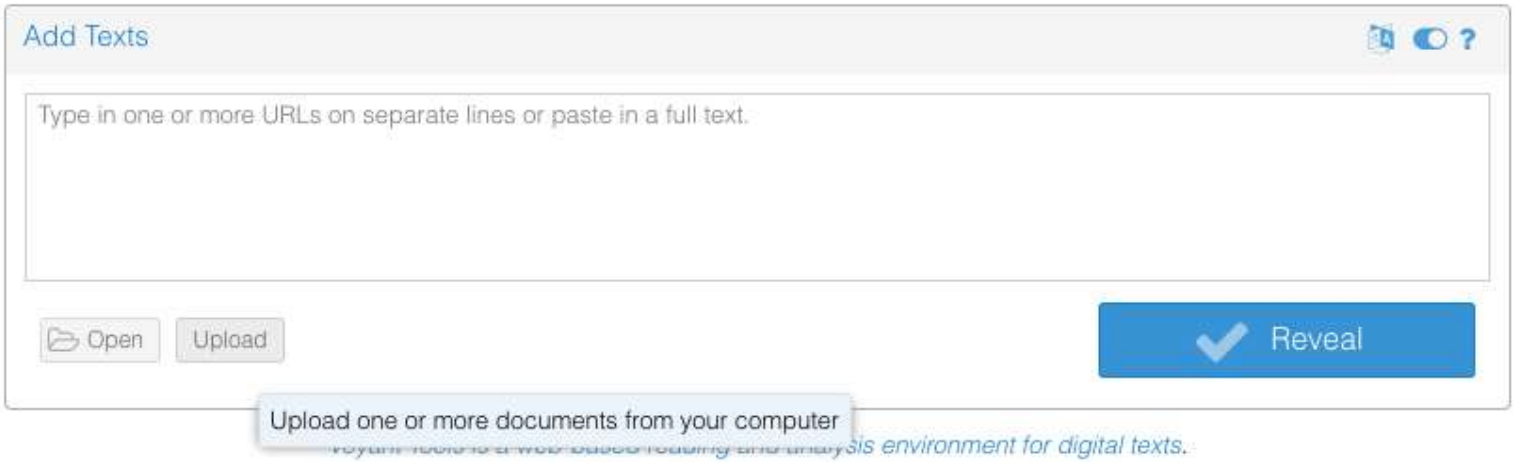


word cloud main use is for brief overview of the words - but not good for precise visualisations

hard to differentiate by sizes, and orientation affects also

Voyant Tools (https://voyant-tools.org)

Voyant Tools



The full text of “The Time Machine” is in the course materials. Or you can simply click here to follow along the examples https://voyant-tools.org/?corpus=931fb5c450f8d231c109b52c41add888

Voyant Tools

Key concepts and tools:

- Word Frequencies
- Concordances
- Collocates
- Vocabulary density

KWIC - keywords in context

Two toy sentences

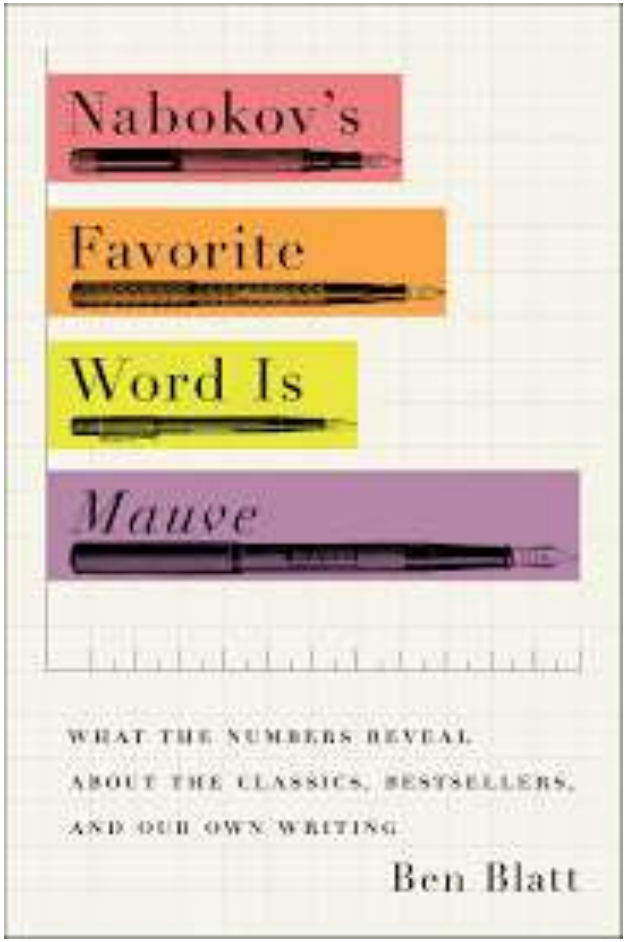
Sentence A: “I am a cat”

Sentence B: “Cat cat cat cat”

density is 1 (4/4) for sentence A
density is 1/4 for sentence B

higher density means higher vocab use

Data is everywhere



Chapter 1: Use sparingly

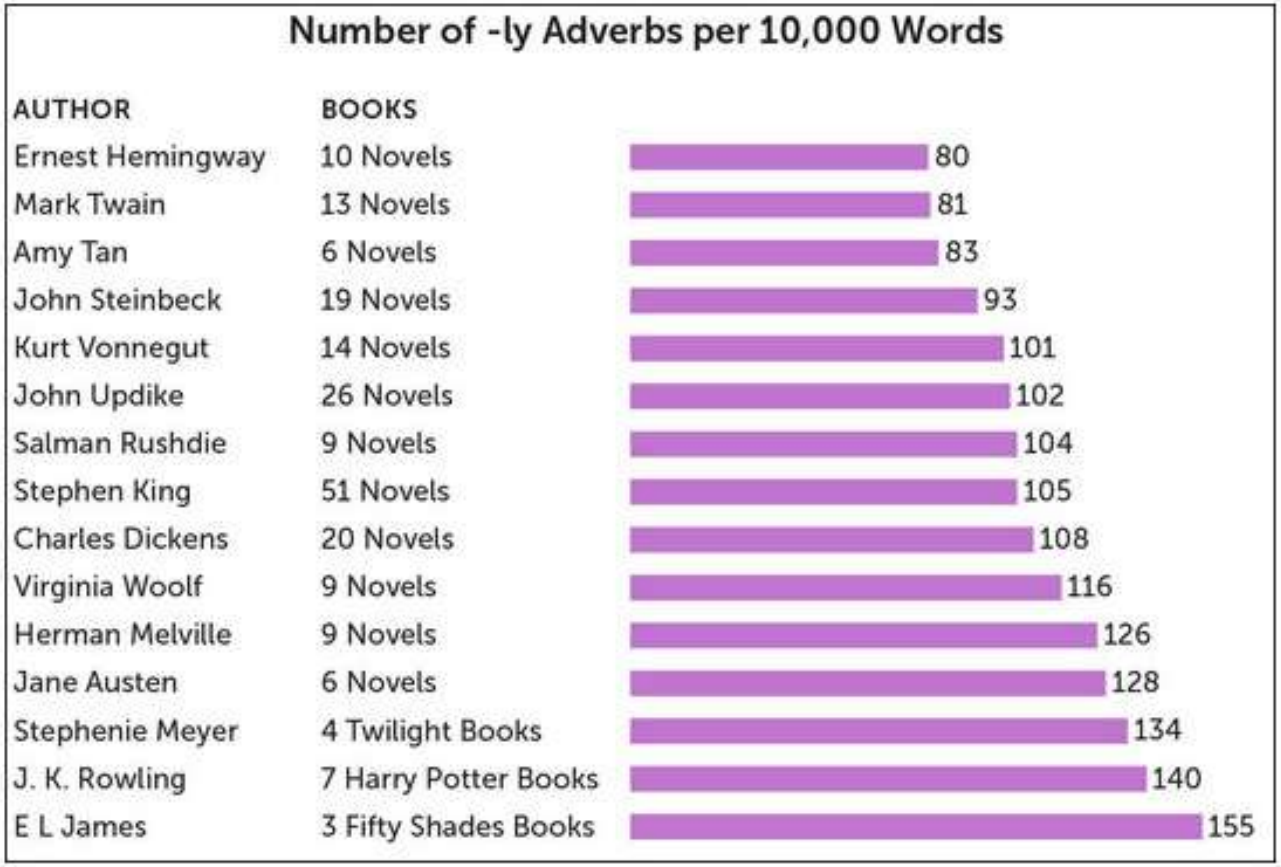
"Don't use too many adverbs!"

Data is everywhere

Sentence A. My cat is extremely quiet.

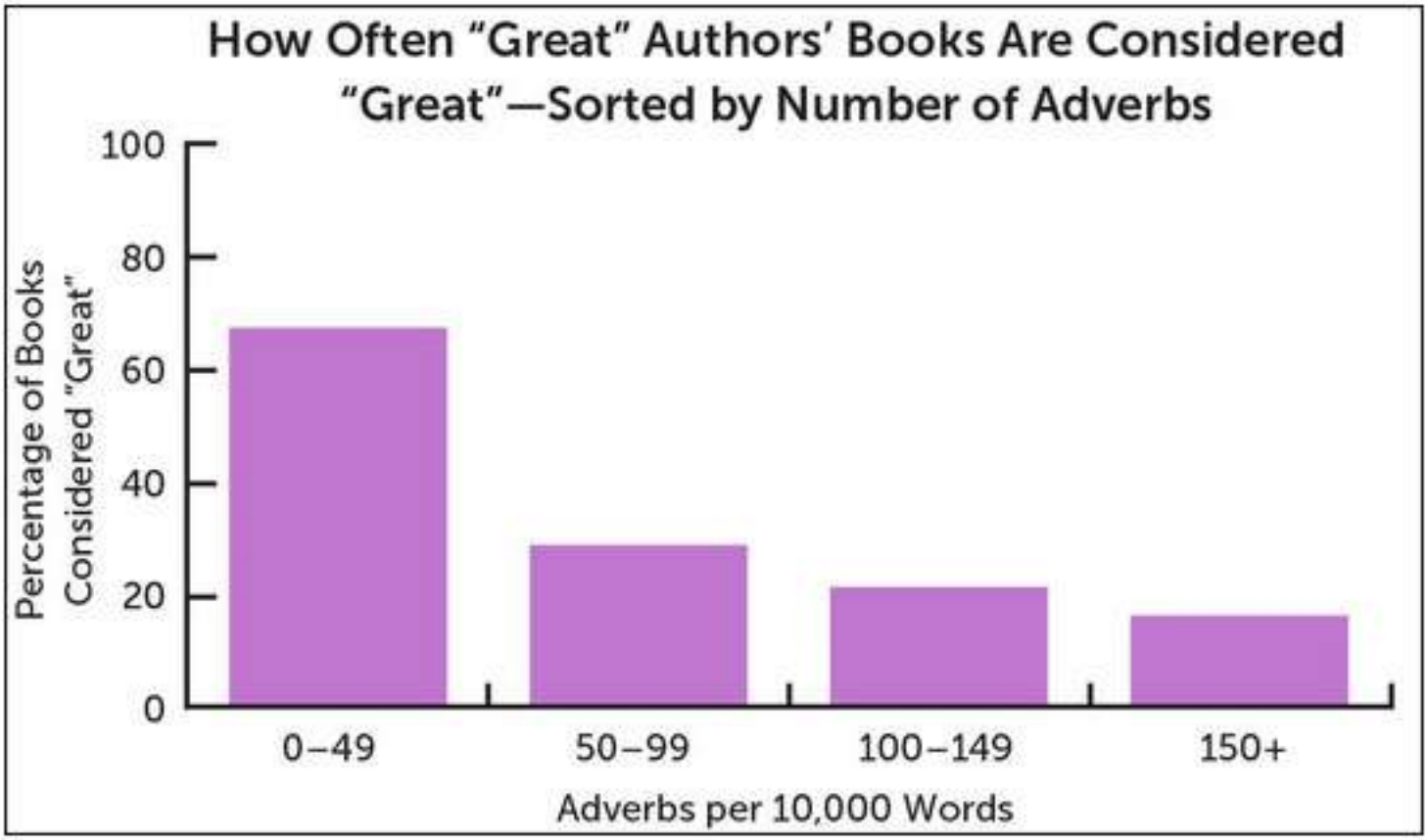
Sentence B. My cat is quiet, and I often mistake it for a pillow.

Data is everywhere

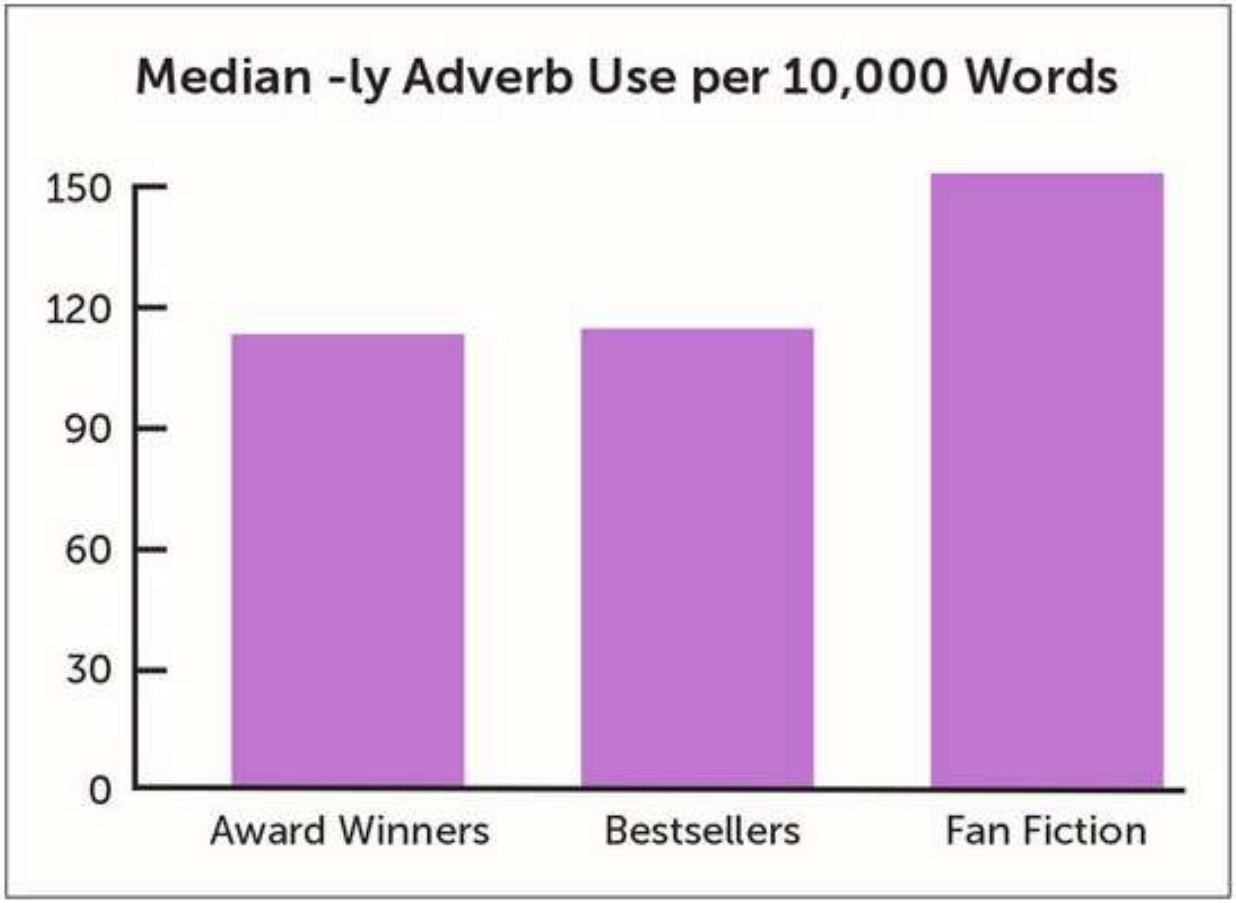


Blatt (2017)

Data and Culture



Culture


Median -ly Adverb Use per 10,000 Words
150
120
90
60
30
0
Award Winners
Bestsellers
Fan Fiction

Using Python packages

Ben Blat used NLTK

We are going to use spaCY

spaCy

Networks of cultural history


```
import spacy
import pandas as pd
```
✓ 7.4s

```
nlp  = spacy.load("en_core_web_lg")
```
✓ 3.2s

```
sentence1 = "My cat is extremely quiet."
sentence2 = "My cat is quiet, and I often mistake it for a pillow."
```
✓ 0.0s

```
for token in nlp(sentence1):
    print(token.text, token.pos_)
```
✓ 0.1s

```
My PRON
cat NOUN
is AUX
extremely ADV
quiet ADJ
. PUNCT
```

This code generates POS tags for the sentences mentioned earlier.

Part of Speech Tags

- ADJ: adjective
- ADP: adposition
- ADV: adverb
- AUX: auxiliary
- CCONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection
- NOUN: noun
- NUM: numeral
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol
- VERB: verb
- X: other

spaCY's core English module uses the Universal Dependencies Framework (https://universaldependencies.org/)

POS tags

```
for token in nlp(sentence2):
    print(token.text, token.pos_)
```

✓ 0.0s

```
My PRON
cat NOUN
is AUX
quiet ADJ
, PUNCT
and CCONJ
I PRON
often ADV
mistake VERB
it PRON
for ADP
a DET
pillow NOUN
. PUNCT
```

This code generates POS tags for the second of the sentences mentioned earlier.

Ocurences per POS tag

As explained in the Spacy Tutorial, we can create matching patterns that only match 'light' when it is a noun.

```
matcher.add("LIGHT_NOUN", [[{"LOWER": "light", "POS":"NOUN"}]])
matches = matcher(doc)
print("Light as a noun:", len(matches))
```

✓ 0.0s Python

```
Light as a noun: 40
```

We need to remove the previous pattern before adding a new one. Then we can match 'light' as and adjective.

```
matcher.remove("LIGHT_NOUN")
matcher.add("LIGHT_ADJECTIVE", [[{"LOWER": "light", "POS":"ADJ"}]])
matches = matcher(doc)
print("Light as an adjective:",len(matches))
```

✓ 0.0s Python

```
Light as an adjective: 1
```

And, finally, as a verb.

```
#light as a verb
matcher.remove("LIGHT_ADJECTIVE")
matcher.add("LIGHT_VERB", [[{"LOWER": "light", "POS":"VERB"}]])
matches = matcher(doc)
print("Light as a verb:",len(matches))
```

✓ 0.0s Python

```
Light as a verb: 2
```

See the Jupyter Notebook **5.2.ipynb**

Ocurences per POS tag


```
for span in matches:
    print(span)
```

✓ 0.0s

```
said
said
said
said
said
say
said
said
said
said
said
said
say
say
```

See the Jupyter Notebook **5.3.ipynb**

Finding lemmas

Sometimes verbs such as “say” would present in different forms in a text, such as “saying” or “said”.

To count all instances of the verb, regardless of the tense, we can access what is called the **LEMMA** or the root form of a word.

Forms of the verb “to say”


```
matcher = Matcher(nlp.vocab)
matcher.add("LIGHT_NOUN", [[{"LEMMA": "say", "POS":"VERB"}]])
matches = matcher(doc,as_spans=True)
print("Forms of 'say':", len(matches))
```

✓ 0.1s

```
Forms of 'say': 189
```

```
for span in matches:
    print(span)
```

✓ 0.0s

```
said
say
said
said
said
said
said
said
said
say
said
say
saying
said
```

See the Jupyter Notebook **5.3.ipynb**

More information on spaCy

You can find an easy-to-follow full tutorial on spaCy here:
https://course.spacy.io/en/