Foundations of Natural Language Processing
Lecture 3b
Text Corpora: Some basic principles of experimental design

Alex Lascarides

School of informatics

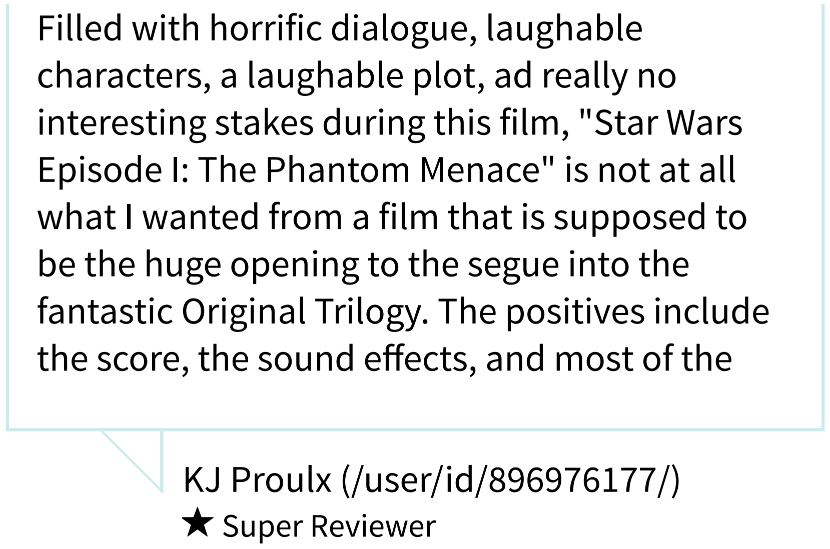
Corpora in NLP

Last time:

- What is a corpus?
- Why do we need corpora to inform NLP tasks?
  - looked at sentiment analysis example

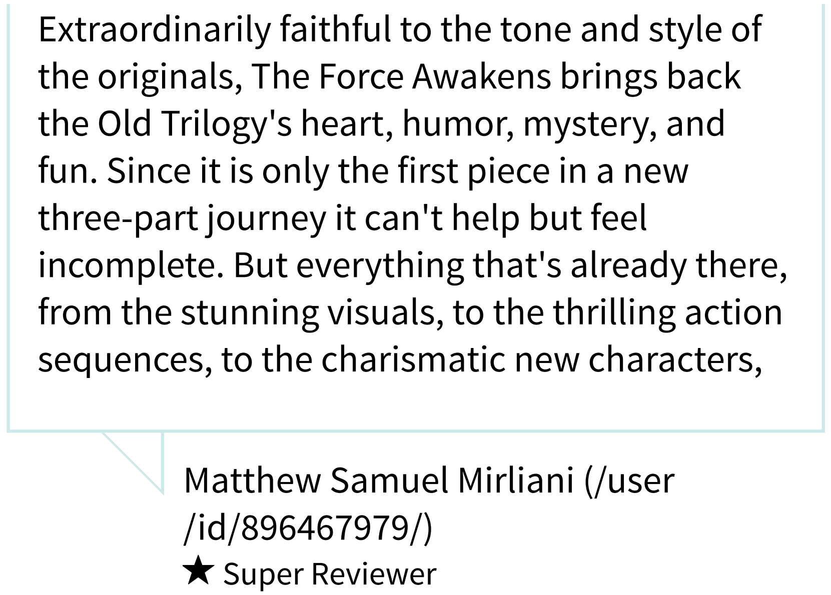
Now: some basic principles for learning to do an NLP task from corpus data
Illustrative application: **sentiment analysis**

Reminder of the example task



Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

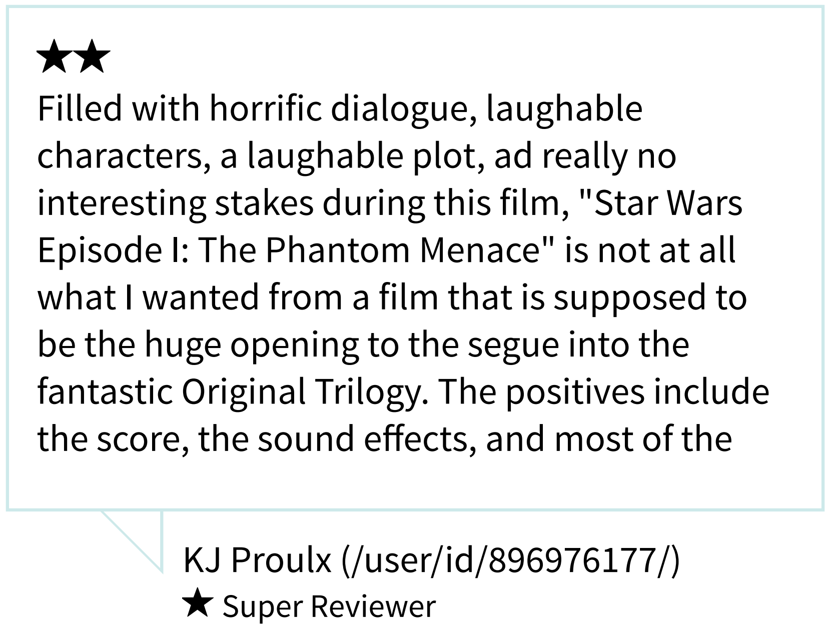


Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user /id/896467979/)
★ Super Reviewer

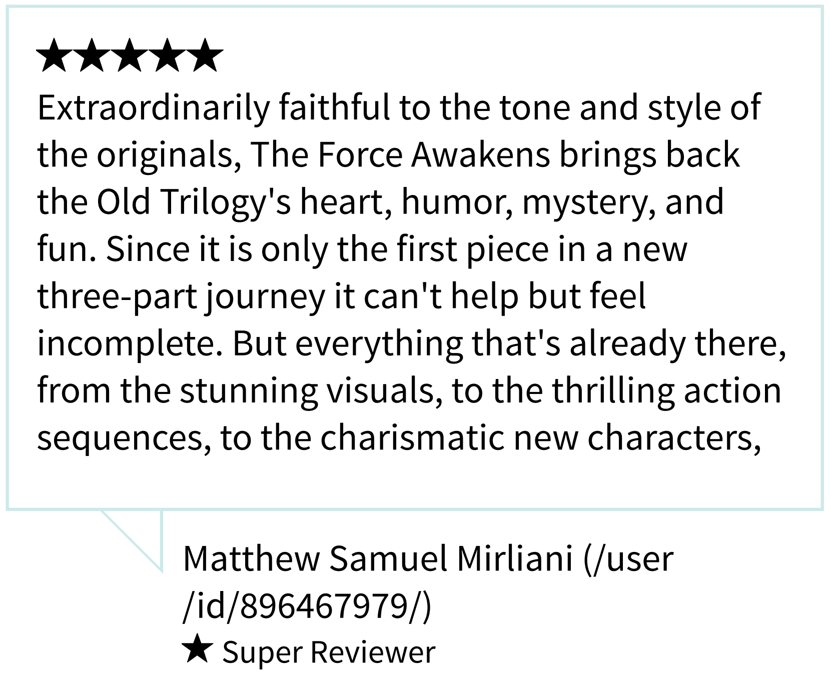
RottenTomatoes.com

Reminder of the example task



★★
Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer



★★★★★
Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user /id/896467979/)
★ Super Reviewer

RottenTomatoes.com

A sentiment analyzer: design decisions!

Questions to ask yourself:

1. What is the input for each prediction? (sentence? full review text? text+metadata?)

2. What are the possible outputs? (+ or − / stars)

3. How will it decide?

4. How will you measure its effectiveness?

BEFORE you build a system, choose a dataset for evaluation!

Why is data-driven evaluation important?

- Good science requires controlled experimentation.
- Good engineering requires benchmarks.
- Your intuitions about typical inputs are probably wrong.

Sometimes you want multiple evaluation datasets: e.g., one for **development** as you hack on your system, and one reserved for final **testing**.

Where can you get a corpus?

- Many corpora are prepared specifically for linguistic/NLP research with text from content providers (e.g., newspapers). In fact, there is an entire subfield devoted to developing new **language resources**.

- You may instead want to collect a new one, e.g., by scraping websites. (There are tools to help you do this.)

Annotations

To evaluate and compare sentiment analyzers, we need reviews with **gold labels** ($+$ or $-$) attached. These can be

- derived automatically from the original data artifact (**metadata** such as star ratings), or
- added by a human annotator who reads the text
  - Issue to consider/measure: How consistent are human annotators? If they often have trouble deciding or agreeing, how can this be addressed?

More on these issues later in the course!

An evaluation measure

Once we have a dataset with gold (correct) labels, we can give the text of each review as input to our system and measure how often its output matches the gold label.

Simplest measure:

$$\mathbf{accuracy} = \frac{\#\ \text{correct}}{\#\ \text{total}}$$

More measures later in the course!

Catching our breath

We now have:

✓ a definition of the sentiment analysis task (inputs and outputs)

✓ a way to measure a sentiment analyzer (accuracy on gold data)

So we need:

- an algorithm for predicting sentiment

A simple sentiment classification algorithm

Use a **sentiment lexicon** to count positive and negative words:

Positive:

absolutely	beaming	calm
adorable	beautiful	celebrated
accepted	believe	certain
acclaimed	beneficial	champ
accomplish	bliss	champion
achieve	bountiful	charming
action	bounty	cheery
active	brave	choice
admire	bravo	classic
adventure	brilliant	classical
affirm	bubbly	clean
...		...

Negative:

abysmal	bad	callous
adverse	banal	can't
alarming	barbed	clumsy
angry	belligerent	coarse
annoy	bemoan	cold
anxious	beneath	collapse
apathy	boring	confused
appalling	broken	contradictory
atrocious		contrary
awful		corrosive
		corrupt
		...

From `http://www.enchantedlearning.com/wordlist/`

Simplest rule: Count positive and negative words in the text. Predict whichever is greater.

Some possible problems with simple counting

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.
   - sense ambiguity
   - sarcasm/irony
   - text could mention expectations or opposing viewpoints, in contrast to author's actual opinon

2. Opinion words may be describing (e.g.) a character's attitude rather than an evaluation of the film.

3. Some words act as semantic modifiers of other opinion-bearing words/phrases, so interpreting the full meaning requires sophistication:

I **can't** stand this movie

vs.

I **can't** believe how great this movie is

What if we have more data?

Perhaps corpora can help address the first objection:

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.

A data-driven method: Use **frequency counts** to ascertain which words tend to be positive or negative.

NLTK

In this course, we will be using Python 2.7 and NLTK, the Natural Language Toolkit (http://nltk.org). NLTK

- is open-source, community-built software
- was designed for teaching NLP: simple access to datasets, reference implementations of important algorithms
- contains wrappers for using (some) state-of-the-art NLP tools in Python

It will help if you familiarise yourself with Python **strings** and methods/libraries for manipulating them. Nathan Schneider has produced a number of useful reference guides for NLP using Python: http://people.cs.georgetown.edu/nschneid/howtos.html

Using an NLTK corpus

```
>>> from nltk.corpus import movie_reviews
>>> movie_reviews.words()
[u'plot', u':', u'two', u'teen', u'couples', u'go', ...]
>>> movie_reviews.sents()
[[u'plot', u':', u'two', u'teen', u'couples', u'go',
  ↪ u'to', u'a', u'church', u'party', u',', u'drink',
  ↪ u'and', u'then', u'drive', u'.'], [u'they',
  ↪ u'get', u'into', u'an', u'accident', u'.'], ...]
>>> print('\n'.join(' '.join(sent) for sent in
  ↪ movie_reviews.sents()[:5]))
plot : two teen couples go to a church party , drink
  ↪ and then drive .
they get into an accident .
one of the guys dies , but his girlfriend continues to
  ↪ see him in her life , and has nightmares .
what ' s the deal ?
watch the movie and " sorta " find out .
```

Using an NLTK corpus: word frequencies

```
>>> from nltk import FreqDist
>>> f = FreqDist(movie_reviews.words())
>>> f.most_common(20)
[(u',', 77717), (u'the', 76529), (u'.', 65876), (u'a',
  ↪ 38106), (u'and', 35576), (u'of', 34123), (u'to',
  ↪ 31937), (u"'", 30585), (u'is', 25195), (u'in',
  ↪ 21822), (u's', 18513), (u'"', 17612), (u'it',
  ↪ 16107), (u'that', 15924), (u'-', 15595), (u')',
  ↪ 11781), (u'(', 11664), (u'as', 11378), (u'with',
  ↪ 10792), (u'for', 9961)]
>>> help(f)
...
```

Using an NLTK corpus: word frequencies

```
>>> f = FreqDist(w for w in movie_reviews.words() if
  ↪ any(c.isalpha() for c in w))
>>> f.most_common(20)
[(u'the', 76529), (u'a', 38106), (u'and', 35576),
  ↪ (u'of', 34123), (u'to', 31937), (u'is', 25195),
  ↪ (u'in', 21822), (u's', 18513), (u'it', 16107),
  ↪ (u'that', 15924), (u'as', 11378), (u'with',
  ↪ 10792), (u'for', 9961), (u'his', 9587), (u'this',
  ↪ 9578), (u'film', 9517), (u'i', 8889), (u'he',
  ↪ 8864), (u'but', 8634), (u'on', 7385)]
```

Using an NLTK corpus: categories

```
>>> movie_reviews.categories()
[u'neg', u'pos']
>>> fpos =
  ↪ FreqDist(movie_reviews.words(categories='pos'))
>>> fneg =
  ↪ FreqDist(movie_reviews.words(categories='neg'))
>>> fMoreNeg = fneg - fpos
>>> help(f.__sub__)
>>> fMoreNeg.most_common(20)
[(u'movie', 721), (u't', 700), (u'i', 685), (u'bad',
  ↪ 673), (u'?', 631), (u'"', 628), (u'have', 421),
  ↪ (u'!', 399), (u'no', 350), (u'plot', 321),
  ↪ (u'there', 318), (u'if', 301), (u'*', 286),
  ↪ (u'this', 282), (u'so', 267), (u'why', 250),
  ↪ (u'just', 221), (u'only', 219), (u'worst', 210),
  ↪ (u'even', 207)]
```

What if we have more data?

Perhaps corpora can help address the first objection:

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.

A data-driven method: Use frequency counts from a **training corpus** to ascertain which words tend to be positive or negative.

- Why separate the training and test data (held-out test set)? Because otherwise, it's just data analysis; no way to estimate how well the system will do on new data in the future.

Summary

- NLP tasks should be empirically grounded

- The system design must have answers to:
  - What's the input? what's the output?
  - What algorithms do I use to map input to output? Are they hand written or learned (or both)?
  - How do I evaluate the system?

- When using corpora to inform the task, expect the unexpected.

- So you may go through several iterations, as you learn to design your system better.

Next time: Preparing a corpus: tokenization