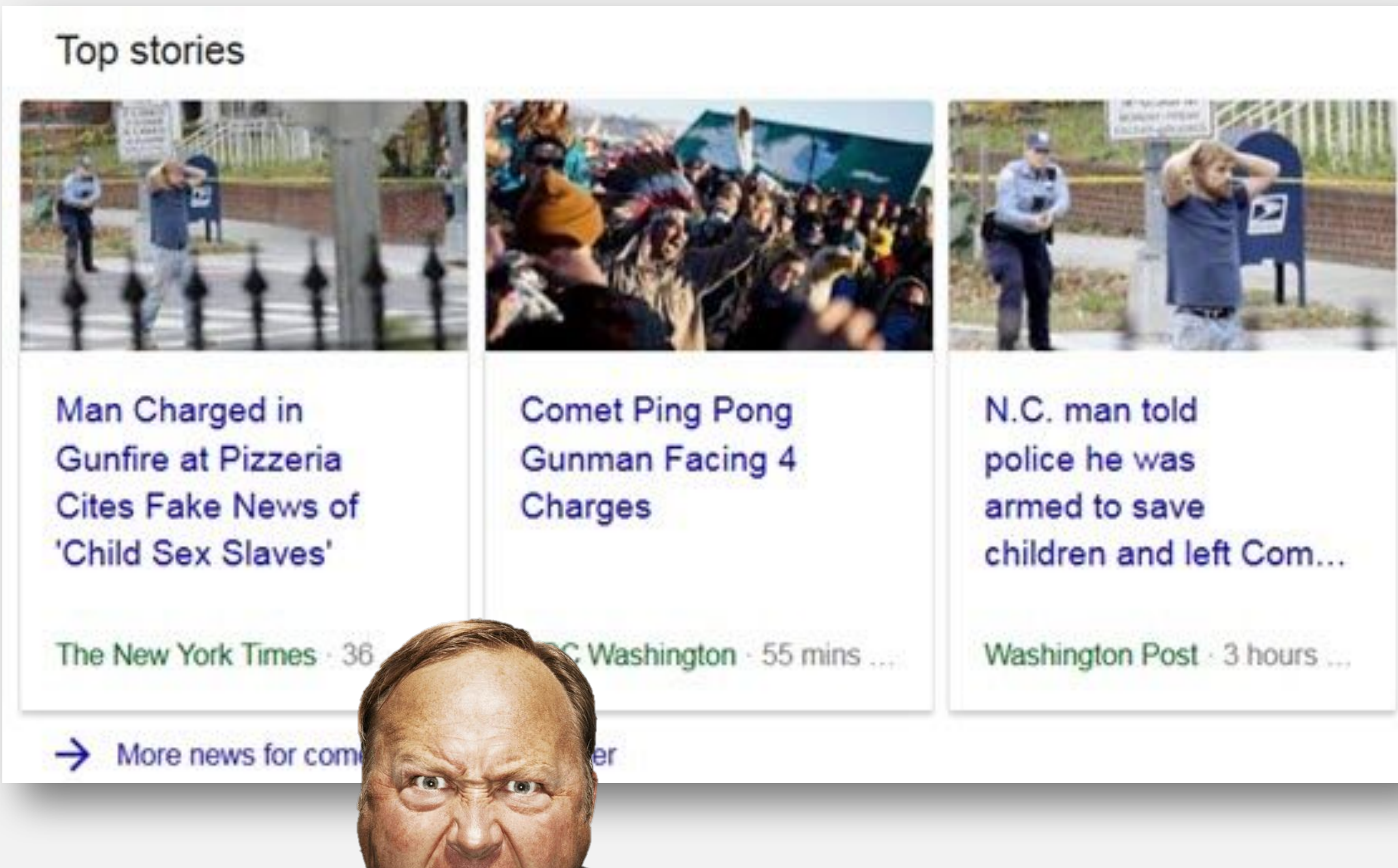
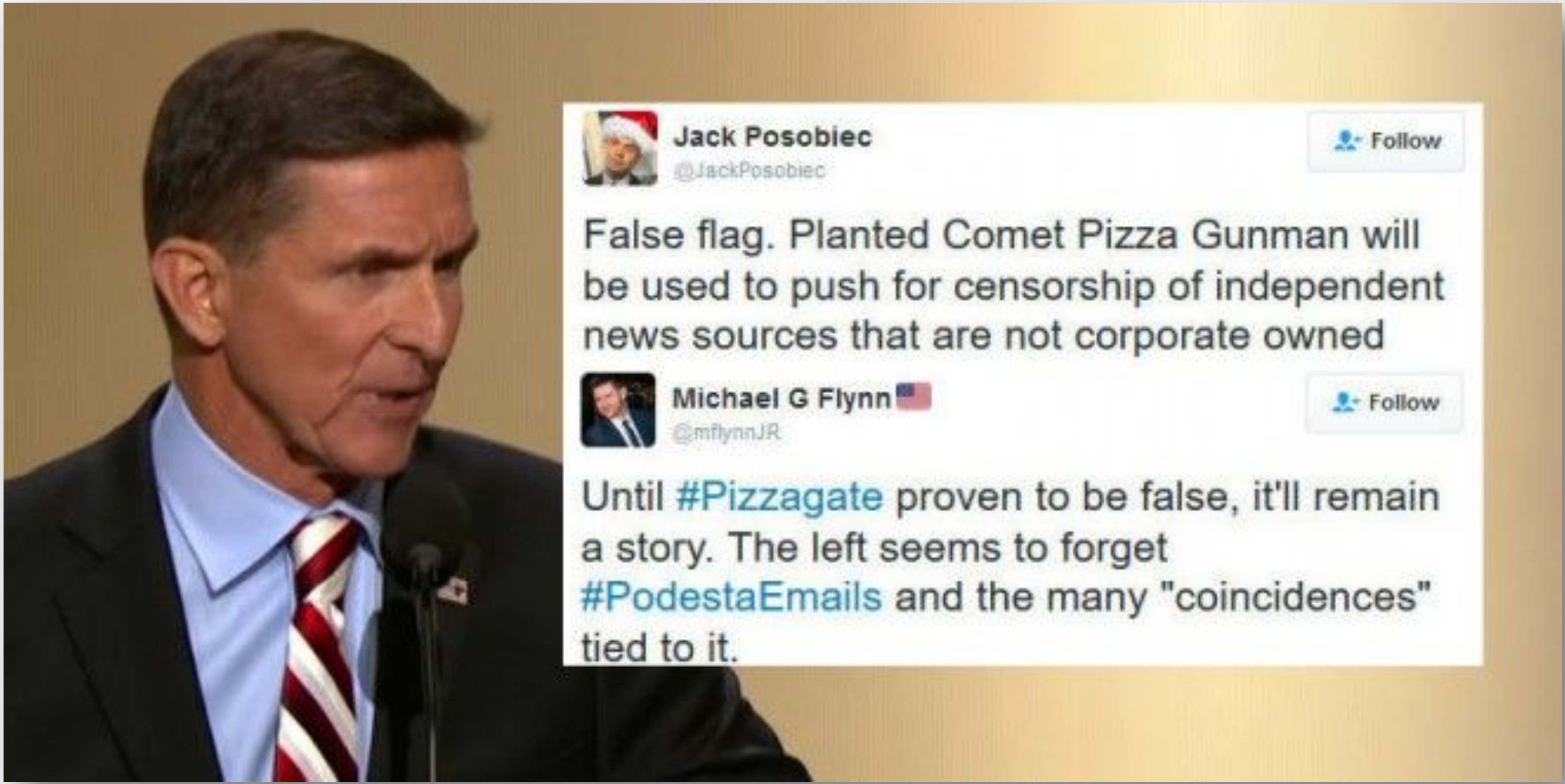
CSC401/2511
ONE OF THE FIRST TUTORIALS EVER, EVERYBODY SAYS SO

- Frank Rudzicz

THE WORLD WE LIVE IN



THE WORLD WE LIVE IN



LOGICAL FALLACIES

strawman
Misrepresenting someone's argument to make it easier to attack.

false cause
Presuming that a real or perceived relationship between things means that one is the cause of the other.

appeal to emotion
Manipulating an emotional response in place of a valid or compelling argument.

the fallacy fallacy
Presuming that because a claim has been poorly argued, or a fallacy has been made, that it is necessarily wrong.

slippery slope
Asserting that if we allow A to happen, then Z will consequently happen too, therefore A should not happen.

ad hominem
Attacking your opponent's character or personal traits instead of engaging with their argument.

tu quoque
Avoiding having to engage with criticism by turning it back on the accuser - answering criticism with criticism.

personal incredulity
Saying that because one finds something difficult to understand that it's therefore not true.

special pleading
Moving the goalposts or making up exceptions when a claim is shown to be false.

loaded question
Asking a question that has an assumption built into it so that it can't be answered without appearing guilty.

burden of proof
Saying that the burden of proof lies not with the person making the claim, but with someone else to disprove.

ambiguity
Using double meanings or ambiguities of language to mislead or misrepresent the truth.

the gambler's fallacy
Believing that 'runs' occur to statistically independent phenomena such as roulette wheel spins.

bandwagon
Appealing to popularity or the fact that many people do something as an attempted form of validation.

no true scotsman
Making what could be called an appeal to purity as a way to dismiss relevant criticisms or flaws of an argument.

genetic
Judging something good or bad on the basis of where it comes from, or from whom it comes.

black-or-white
Where two alternative states are presented as the only possibilities, when in fact more possibilities exist.

begging the question
A circular argument in which the conclusion is included in the premise.

the texas sharpshooter
Cherry-picking data clusters to suit an argument, or finding a pattern to fit a presumption.

middle ground
Saying that a compromise, or middle point, between two extremes is the truth.

appeal to authority
Using the opinion or position of an authority figure, or institution of authority, in place of an actual argument.

composition /division
Assuming that what's true about one part of something has to be applied to all, or other, parts of it.

appeal to nature
Making the argument that because something is 'natural' it is therefore valid, justified, inevitable, or ideal.

anecdotal
Using personal experience or an isolated example instead of a valid argument, especially to dismiss statistics.

WHAT CAN BE DONE?

- There are probably many solutions, including better education and a ramping down of political zealotry from Our Glorious Leaders.

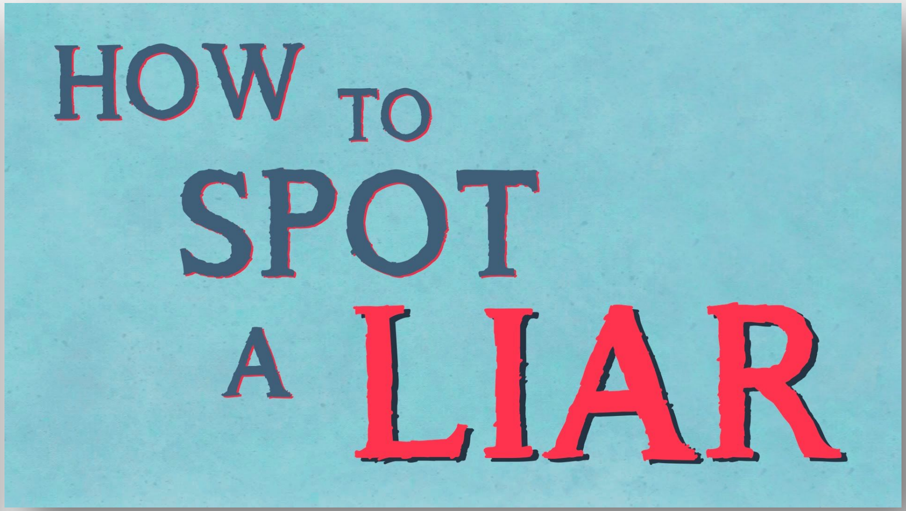
- But this is a class on natural language processing.

- *Can we detect bias automatically from online texts?*

LANGUAGE ANALYSIS AND LYING

"Don't use a big word when a diminutive one would suffice."

https://youtu.be/4ab2ZeZ-krY



https://youtu.be/H0-WkpmTPrM

ASIDE: HOW CAN BIAS DETECTION HELP?

- **Social media platforms:**
  - May want to more closely monitor highly biased groups (e.g. allocate more human annotators to look for ban-able content like inciting violence or doxing).
- **Sociologists and network scientists:**
  - Better understanding of biased online communities can help us address the root causes of bias.
  - How do online communities become biased?
  - Are biased online communities uniformly biased?

REDDIT CORPUS

- We have curated data from Reddit by scraping subreddits, using Pushshift, by perceived political affiliation.

Left (598, 944)	Center (599, 872)	Right (600, 002)	Alt (200, 272)
twoXChromosomes (7, 720, 661)	news (2, 782, 9911)	theNewRight (19, 466)	conspiracy (6, 767, 099)
occupyWallStreet (397, 538)	politics (60, 354, 767)	whiteRights (118, 008)	911truth (79, 868)
lateStageCapitalism (634, 962)	energy (416, 926)	Libertarian (3, 886, 156)	
progressive (246, 435)	canada (7, 225, 005)	AskTrumpSupporters (1, 007, 590)	
socialism (1, 082, 305)	worldnews (38, 851, 904)	The_Donald (21, 792, 999)	
demsocialist (5269)	law (464, 236)	new_right (25, 166)	
Liberal (151, 350)		Conservative (1, 929, 977)	
		tea_party (1976)	

- These data are stored on the teach.cs servers under `/u/cs401/A1/data/`. These files should only be accessed from that directory (and not copied). All data are in the JSON format.

A COMMENT, IN JSON


```
"{"id":"c05os7s", "author":"[deleted]",
"subreddit":"conspiracy", "author_flair_css_class":null,
"ups":-1, "archived":true, "edited":true,
"subreddit_id":"t5_2qh4r", "body":"WAIT! Are you saying
that 9/11 was a *conspiracy*?! Like...an *inside job* or
something?", "score_hidden":false,
"parent_id":"t3_74xuq", "distinguished":null,
"link_id":"t3_74xuq", "author_flair_text":null,
"created_utc":"1223008247",
"retrieved_on":1425887728,"gilded":0,"name":"t1_c05os7s"
,"controversiality":0,"score":-1,"downs":0}",
```


- If you want to experiment a bit, there are some fields of metadata that might be interesting, but the main thing is **`body`**.

THREE STEPS

- In order to **infer** whether the author of a given comment leans a certain way, politically, we use three steps:
  1. **Preprocess** the data, so that we can extract meaningful information, and remove distracting 'noise'.
  2. **Extract** meaningful information.
  3. **Train** classifiers, given labeled data.

****Python 3.9 on CDF****

```
wolf:~$ python --version
Python 2.7.13
wolf:~$ python3 --version
Python 3.10.1
wolf:~$ python3.9 --version
Python 3.9.7
```

SPACY.IO
NLP IN PYTHON


```
import spacy
nlp = spacy.load('en_core_web_sm’)
nlp.add_pipe(“sentencizer")

sentence = “This is a useful library!”
doc = nlp(sentence)

for sent in doc.sents:
     print(sent)
     for token in sent:
          print(token, token.tag_, token.lemma_, token.dep_)
```

PREPROCESSING I

1. Replace all whitespace characters with spaces.
2. Replace HTML character codes (i.e., *&…;*) with their ASCII equivalent.
3. Remove all URLs (i.e., tokens beginning with *http(s)://* or *www.*).
4. Remove duplicate spaces between tokens.
5. Apply the following steps using spaCy:
    1. Tagging with part-of-speech (*dog -> dog/**NN***)
    2. Lemmatization (*word**s**/NNS -> word/NNS*)
    3. Sentence segmentation
        1. ("*I know words. I've got the best words*" **->** "*I know words.***\n***I've got the best words***\n**")

PUTTING IT ALL TOGETHER

I know words. I’ve got the best words.

I/PRP know/VBP word/NNS ./.**\n**I/PRP ‘ve/VBP get/VBN the/DT good/JJS word/NNS ./.**\n**

```
import re, string, html

print(string.whitespace)
print(string.punctuation)

print(re.sub(“spacy”, “spaCy”, “spacy is a python library”))
```

LEMMATIZATION V STEMMING: DAWN OF SPARSENESS

- Both **lemmatization** and **stemming** are often used to transform word tokens to a more base form.
  - This helps to improve sparseness.
  - It also helps in using various resources.
    - (e.g., *funkilicious* might not exist in a norm or embedding, but '*funk*' ought to).

LEMMATIZATION V STEMMING: DAWN OF SPARSENESS

- **lemma**: *n.* an abstract conceptual form of a word that has been mentally selected for utterance in the early stages of speech production.
  - E.g. *lemma best* = *good* (degree)
  - E.g. *lemma*(*houses*) = *house* (number/amount)
  - E.g. *lemma*(*housing*) = *housing*
- **stem**: *n.* usually, a part of a word to which affixes can be attached.
  - E.g. *stem houses* = *stem housing* = *hous*
- We use lemmatization given some of our features, but check out `nltk.stem` in the NLTK package.

PREPROCESSING: YOUR TASK

- Copy the starter template from the drive*. There are two functions you need to modify:
  - In `preproc1`, perform each preprocessing step above.
  - In `main`, replace the lines marked with TODO with the code they describe. Add a new **cat** field with the name of the class
- The program takes three arguments:

1. your student ID (mandatory),
2. the output file (mandatory), and
3. the maximum **number of lines** to sample from each category file (optional; default=10,000).

```
python a1 preproc.py 999123456 -o preproc.json
```

*The CDF folder /u/cs401/A1/code may not be up-to-date with changes made since the release. Check Piazza announcements for details.

PREPROCESSING: SUBSAMPLING

- We provide our student IDs so we each see a different part of the available data.
  - By default, you should only sample 10,000 lines from each of the `Left`, `Centre`, `Right`, and `Alt` files, for a total of 40,000 lines.
  - From each file, start sampling lines at index [ID % len(X)]

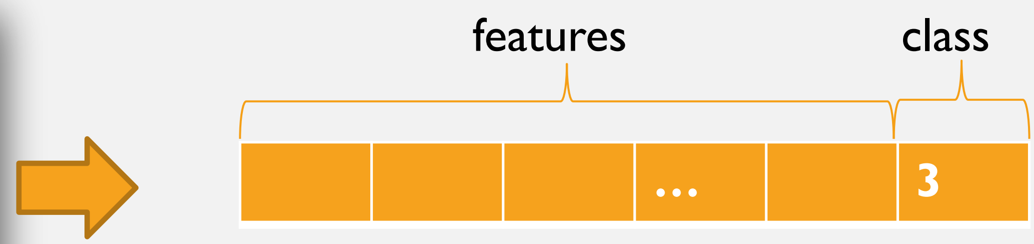
- Feel free to play around with more or less data, *respectful of your peers on the servers*, but this step guarantees it's tractable (and that there's no 'desired' level of accuracy).

FEATURE EXTRACTION

- The `a1_extractFeatures.py` program reads a preprocessed JSON file and extracts features for each comment therein, producing and saving a NumPy array, where the row is the features for the comment, followed by an integer for the class (0: Left, 1: Center, 2: Right, 3: Alt), as per the cat JSON.

```
"{"id":"c05os7s",
"body":"wait ! be you say
that 9 / 11 be a *
conspiracy *?! like ...
an * inside job * or
something ?",cat:"Alt"}",
```

comment in input



row in output

1. Number of words in uppercase (≥ 3 letters long)
2. Number of first-person pronouns
3. Number of second-person pronouns
4. Number of third-person pronouns
5. Number of coordinating conjunctions
6. Number of past-tense verbs
7. Number of future-tense verbs
8. Number of commas
9. Number of multi-character punctuation tokens
10. Number of common nouns
11. Number of proper nouns
12. Number of adverbs
13. Number of *wh-* words
14. Number of slang acronyms
15. Average length of sentences, in tokens
16. Average length of tokens, excluding punctuation-only tokens, in characters
17. Number of sentences.
18. Average of AoA (100-700) from Bristol, Gilhooly, and Logie norms
19. Average of IMG from Bristol, Gilhooly, and Logie norms
20. Average of FAM from Bristol, Gilhooly, and Logie norms
21. Standard deviation of AoA (100-700) from Bristol, Gilhooly, and Logie norms
22. Standard deviation of IMG from Bristol, Gilhooly, and Logie norms
23. Standard deviation of FAM from Bristol, Gilhooly, and Logie norms
24. Average of V.Mean.Sum from Warringer norms
25. Average of A.Mean.Sum from Warringer norms
26. Average of D.Mean.Sum from Warringer norms
27. Standard deviation of V.Mean.Sum from Warringer norms
28. Standard deviation of A.Mean.Sum from Warringer norms
29. Standard deviation of D.Mean.Sum from Warringer norms

30-173. LIWC/Recepti*viti* features

PREPROCESSING:
Searching for Patterns

- Useful tools: regex

```
import re

pattern = re.compile("\d+")
pattern.findall("Highway 401 continues in Quebec as
Autoroute 20")
```

- Useful tools: spaCy documentation
    - https://spacy.io/models/en#en_core_web_sm

```
spacy.explain('VBG')
```

- Useful shortcut:
  https://github.com/explosion/spaCy/blob/master/spacy/glossary.py

- Useful tools: the handout, tables.

1. Number of words in uppercase ($\geq$ 3 letters long)
2. Number of first-person pronouns
3. Number of second-person pronouns
4. Number of third-person pronouns
5. Number of coordinating conjunctions
6. Number of past-tense verbs
7. Number of future-tense verbs
8. Number of commas
9. Number of multi-character punctuation tokens
10. Number of common nouns
11. Number of proper nouns
12. Number of adverbs
13. Number of *wh-* words
14. Number of slang acronyms
15. Average length of sentences, in tokens
16. Average length of tokens, excluding punctuation-only tokens, in characters
17. Number of sentences.
18. Average of AoA (100-700) from Bristol, Gilhooly, and Logie norms
19. Average of IMG from Bristol, Gilhooly, and Logie norms
20. Average of FAM from Bristol, Gilhooly, and Logie norms
21. Standard deviation of AoA (100-700) from Bristol, Gilhooly, and Logie norms
22. Standard deviation of IMG from Bristol, Gilhooly, and Logie norms
23. Standard deviation of FAM from Bristol, Gilhooly, and Logie norms
24. Average of V.Mean.Sum from Warringer norms
25. Average of A.Mean.Sum from Warringer norms
26. Average of D.Mean.Sum from Warringer norms
27. Standard deviation of V.Mean.Sum from Warringer norms
28. Standard deviation of A.Mean.Sum from Warringer norms
29. Standard deviation of D.Mean.Sum from Warringer norms

30-173. LIWC/Receptiv*iti* features

	A	B	C	D	E	F	G	H	I
1		Word	V.Mean.Sum	V.SD.Sum	V.Rat.Sum	A.Mean.Sum	A.SD.Sum	A.Rat.Sum	D.Mean.Sum
2	1	aardvark	6.26	2.21	19	2.41	1.4	22	4.27
3	2	abalone	5.3	1.59	20	2.65	1.9	20	4.95
4	3	abandon	2.84	1.54	19	3.73	2.43	22	3.32
5	4	abandonmer	2.63	1.74	19	4.95	2.64	21	2.64
6	5	abbey	5.85	1.69	20	2.2	1.7	20	5
7	6	abdomen	5.43	1.75	21	3.68	2.23	22	5.15
8	7	abdominal	4.48	1.59	23	3.5	1.82	22	5.32
9	8	abduct	2.42	1.61	19	5.9	2.57	20	2.75

Warringer: These norms measure the valence (V), arousal (A), and dominance (D) of each **lemma**, according to the VAD model of human affect and emotion.

See: Warriner, A.B., Kuperman, V., & Brysbaert, M. (2013). **Norms of valence, arousal, and dominance for 13,915 English lemmas**. *Behavior Research Methods*, **45**:1191-1207.

	A	B	C	D	E	F	G	H	I
1	Source	WORD	AoA (Yrs)	AoA (100-70(	IMG	FAM	Length (Letters)		
2	GL	abandonmer	NA	359	348	359	11		
3	GL	abatement	NA	294	189	294	9		
4	BN	abbey	7.8	480	575	429	5		
5	GL	abdomen	NA	426	548	426	7		

Bristol et al: measure the age-of-acquisition (AoA), imageability (IMG), and familiarity (FAM) of each word, which we can use to measure lexical complexity.

See: Gilhooly, KJ, Logie, RH (1980). Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words *Behavior Research Methods*, **12**(4):394-427.

LIWC/RECEPTIVITI I

- The **Linguistic Inquiry & Word Count** (**LIWC**) tool has been a standard in a variety of NLP research, especially around authorship and sentiment analysis.
  - This tool provides 85 measures mostly related to word choice.
- The company **Receptiv*iti*** provides a superset of these features, which also includes 59 measures of personality derived from text.
- To simplify things, we have already extracted these 144 features for you. Simply copy the pre-computed features from the appropriate uncompressed `npy` files stored `in /u/cs401/A1/feats/`.

LIWC/RECEPTIVITI 2

- Comment IDs are stored in `_IDs.txt` files (e.g., `Alt_IDs.txt`). When processing a comment, find the index (row) of the ID in the appropriate ID text file, for the category, and copy the 144 elements, starting at element , from the associated `feats.dat.npy` file.

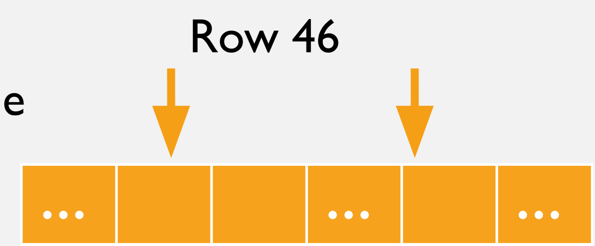
```
"{"id":"c05os7s",
"body":"wait ! be you
say that 9 / 11 be a *
conspiracy *?! like
... an * inside job *
or something ?",
cat:"Alt"}",
```

comment

```
...
c05nn92
c05o811
c05os7s
c05p5vj
c05pbbg
...
```

Alt_IDs.txt

46th line



Alt_feats.dat.npy

```
feats_arr = numpy.load('/u/cs401/A1/feats/Alt_feats.dat.npy')
feats_comment = feats_arr[46]
```

LIWC/RECEPTIVITI 3: FEATURE NAMES

frank

```
liwc_sexual
liwc_shehe
liwc_social
liwc_space
liwc_swear
liwc_tentat
liwc_they
liwc_time
liwc_verb
liwc_we
liwc_work
liwc_you
receptiviti_active
receptiviti_adjustment
receptiviti_adventurous
receptiviti_aggressive
receptiviti_agreeableness
receptiviti_ambitious
receptiviti_anxious
receptiviti_artistic
receptiviti_assertive
receptiviti_body_focus
```

feats.txt

CLASSIFICATION

- Four parts:
  - Compare classifiers
  - Experiment with the amount of training data used
  - Select the best features for classification
  - Do cross-fold validation

CLASSIFICATION 1: COMPARE CLASSIFIERS

- *Randomly* split data into 80% training, 20% testing.

- We have 5 classification methods, which you can consider to be 'black boxes' (input goes in, classes come out).
  1. Support vector machine with linear kernel
  2. Gaussian naïve Bayes classifier.
  3. Random forest classifier
  4. Neural network
  5. Adaboost (with decision tree)

CLASSIFICATION I: COMPARE CLASSIFIERS

- **Accuracy**: the total number of correctly classified instances over all classifications: .
- **Recall**: *for each class k* , the fraction of cases that are truly class k that were classified as class k.
- **Precision**: for each class k, the fraction of cases classified as k that truly are k.

Predicted class

True class

	L	C	R	A
L				
C		8		
R				
A				

number of times class
was classified as class

CLASSIFICATION 2: AMOUNT OF DATA

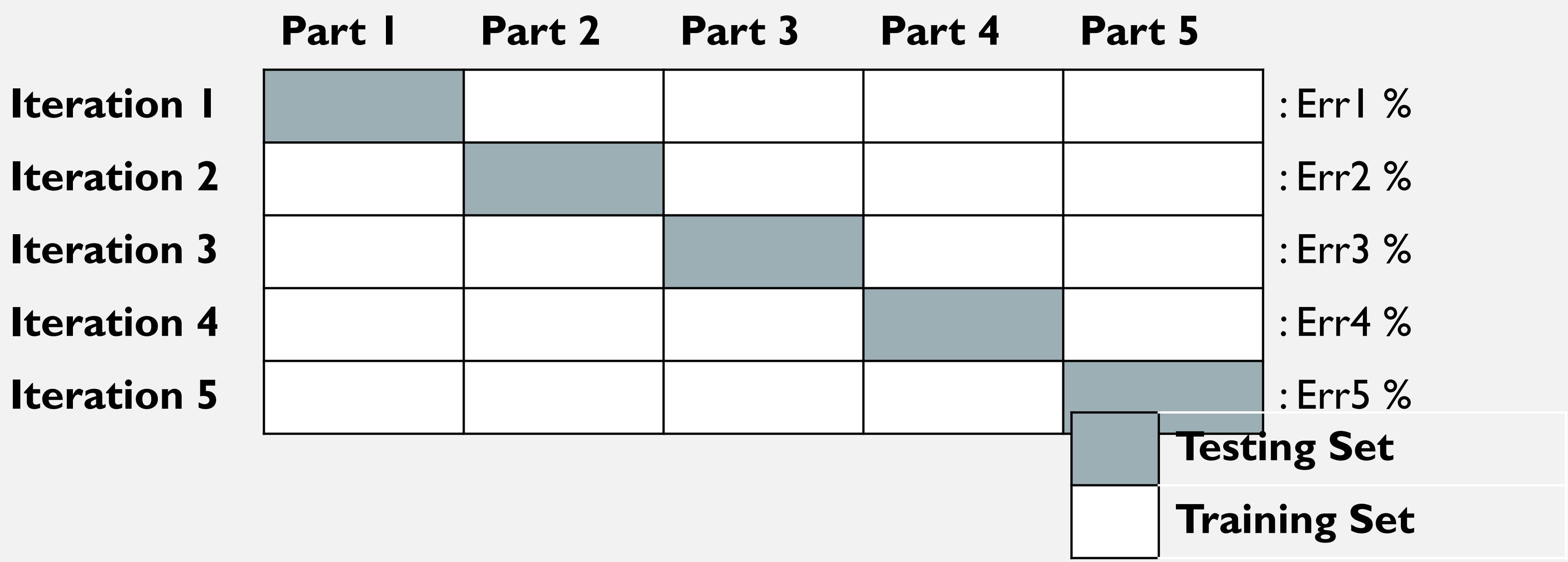
- You previously used a random comments to train.
- Using the classifier with the highest accuracy from Sec3.1, retrain the system using an arbitrary samples from the original train set.

CLASSIFICATION 3: FEATURE ANALYSIS

- Certain features may be more or less useful for classification, and too many can lead to various problems.
- Here, you will select the best features for classification for .
- Train the best classifier from Sec3.1 on just features on both and training samples.
- Are some features always useful? Are they useful to the same degree (*p*-value)? Why are certain features chosen and not others?

CLASSIFICATION 4: CROSS-FOLD VALIDATION

- What if the 'best' classifier from Sec3.1 only appeared to be the best because of a random accident of sampling?
- Test your claims more rigorously.



BONUS

- You have **complete freedom** to expand on this assignment in any way you choose.
- You should have no expectation to the *value* of such an exploration – **check with us** (privately if you want) about the appropriateness of your idea.
- Bonus marks can make up for marks lost in other sections of the assignment, but your overall mark **cannot exceed 100%**.

FESTIVAL DE MIERDA DE TORO

- If things go well, we would love to run a special ‘workshop’ where:
  1. students who did interesting **bonuses** could describe their work
  2. grad students (working around the theme) could present their **projects**
  3. we could hold a **competition** for best systems in A1, A2, A3
- Problem: the instructors and TAs already have a lot on their plates.
- Solution (?): If any of you are interested in spearheading such a get-together at the end of the term (and getting bonus marks), we’d be glad to support.

Remember to…

- Check Piazza regularly for clarifications and announcements.
- Changes to starter code since release:
   1. URL removal (a1_preproc.py, Line 44)
- If you are working in non-CDF environments: use the requirements.txt file to match package versions.
- Sample input and output for Parts 1 and 2 will be out soon (EoD).
- Ask questions: Piazza, tutorials.
- Recommendation: setup up a functioning pipeline, then go back and improve specific sub-modules.