Before we start:

HOMEWORK 1

Goals of Homework 1

- To apply the Information Theory concepts
- To begin to work with text data in python, and apply an open-source NLP package
  - NLTK
- To gain exposure to an openly available NLP research dataset
  - **GLUE Benchmark**

Homework 1 Steps

- Due Weds. September 13 at 11:59pm
- 100 points total
- 7 questions:
  - Download and prepare data
  - Write 4 functions and apply them to data
  - Answer 2 questions on Blackboard

GLUE Benchmark Datasets

- GLUE: <u>**G**</u>eneral <u>**L**</u>anguage <u>**U**</u>nderstanding <u>**E**</u>valuation
- Goal: "favor and encourage models that share general linguistic knowledge across tasks.'
- 9 NLU tasks with a range of sources:
  - Classification: sentiment, linguistic acceptability,
  - Semantic similarity and equivalence
  - Inference: Question answering, entailment, pronoun resolution

https://openreview.net/pdf?id=rJ4km2R5t7

NLTK

- NLTK: Natural Language Toolkit
- Open-source python library for a range of NLP tasks, including Tokenization and Stemming

```
>>> word_tokenize("I wouldn't expect it to split this way!")

['I', 'would', "n't", 'expect', 'it', 'to', 'split', 'this', 'way', '!']
```

Homework 1 Help

- TA Office Hours start Tues Sept 5
  - Virtual and in-person
  - At least one office hour every weekday
  - Schedule posted in Blackboard – see "Content"
- Please bring python and machine setup questions to TA Office Hours
- Blackboard discussion group is available for questions and clarifications about instructions. See "HW 1 Discussion"

Homework 1 Setup

- **<u>Start your Python setup early</u>** if you are new to Python, or not yet set up on machine you will use for this class
- We recommend completing your setup **<u>next week</u>** (week of Sept 5-8). This means you can:

1. Use python interpreter, e.g. `print("Hello World")`
2. Unzip data files
3. Run notebook "HW1_GettingStarted.ipynb"

Homework Submission

- Due Weds. September 13 at 11:59pm on Blackboard
- The homework late policy as posted in course syllabus applies to HW1
- Multiple submissions are allowed; no penalty for resubmission. Only last submission will be graded
- Submit early in case of last-minute technical issues

ILLINOIS INSTITUTE OF TECHNOLOGY

Words and Pattern Matching

CS-585

Natural Language Processing

Sonjia Waxmonsky

Based on slides from Derrick Higgins

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

Regular Expressions

In practice: Applied in Python `re` library

```
[1]: import re

[2]: my_animal = "TYPE_Wood_Chuck"
     re.search("wood.*chuck",my_animal, re.IGNORECASE)

[2]: <re.Match object; span=(5, 15), match='Wood_Chuck'>
```

Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

- Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Negation in Disjunction

- Negations `[^Ss]`
  - Caret means negation only when first in []

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither 'S' nor 's'	`I have no exquisite reason"`
`[^e^]`	Neither e nor ^	`Look here`
`a^b`	The pattern a carat b	`Look up a^b now`

? * + .

Pattern		Matches
colou?r	Optional previous char	color colour
o*h!	0 or more of previous char	h! oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n	Any char	begin begun begun beg3n

Anchors ^ $

Pattern	Matches	
`^[A-Z]`	Palo Alto	Start of string
`^[^A-Za-z]`	1 "Hello"	
`\.$`	The end.	End of string
`.$`	The end? The end!	

Character classes

Pattern	Matches
`\s`	A whitespace character
`\S`	A non-whitespace character
`\d`	A digit (`[0-9]`)
`\D`	A non-digit
`\w`	A “word” character (`[0-9a-zA-Z_]`)
`\W`	A non-word character
`[:upper:]`	An upper-case letter
`[:lower:]`	A lower-case letter

Backreferences (…) …\n

- Sometimes we want to know which part of the text matched a part of a pattern
- We can even use it within the pattern itself, by “capturing” it in parentheses

Pattern	Matches	
`(\d)[a-z]\1`	zsdfg<u>1a1</u>z213	A letter bracketed by the same number on each side
`^(\d)(\d).*\2\1$`	<u>13awdfgasdf31</u>	A line starting with two digits, and ending with those two digits in reverse order

Example

- Find me all instances of the word “the” in a text.

  `the`

  Misses capitalized examples

  `[tT]he`

  Incorrectly returns `other` or `theology`

  `\b[tT]he\b`

Summary

- Regular expressions are surprisingly important
  - Often the first model for any text processing
- For many tasks, we use machine learning
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations

TOKENIZATION

Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

How many words?

- I do uh main- mainly business data processing
  - Fragments, corrections, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many? 15 tokens and 13 types
- Shakespeare - 31k types, 884k tokens

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
- German noun compounds not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

Tokenization: language issues

- Chinese and Japanese -- no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Japanese example: multiple alphabets; dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm ("greedy")

Given a wordlist of Chinese, and a string.

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

Max-match segmentation illustration

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there
  theta bled own there

Doesn't generally work in English!

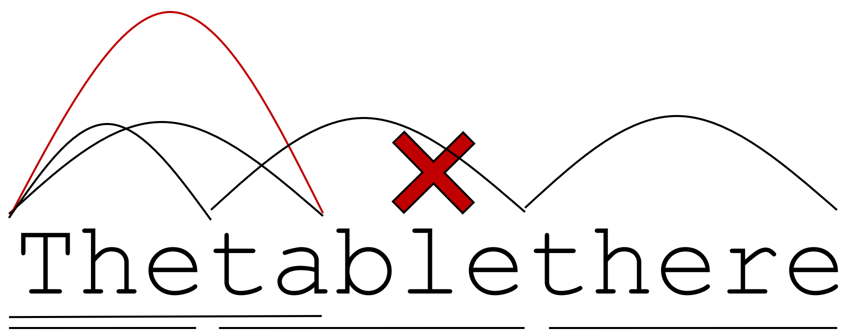
- But works astonishingly well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Modern probabilistic segmentation algorithms even better

Greedy matching

Thetabledownthere

Thetablethere

Backtracking



Dynamic programming

Keep track of ***intermediate results*** (segments of string that can be parsed as a sequence of words)

`Thetablethere`

Location (character indices)	Parse
0-3	the
0-5	theta
0-8	the + table
0-11	the + table + the
0-13	the + table + there

WORD NORMALIZATION & STEMMING

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Case folding

- Applications like Information Retrieval: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAT*** vs. ***sat***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatization

- Reduce inflections or variant forms to ***base*** form
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
- *the boy's cars are different colors → the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

Stemming

- Reduce terms to their stems
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Complex morphology

- Some languages require complex morpheme segmentation
  - Turkish
  - Uygarlaştıramadıklarımızdanmışsınızcasına
  - `(behaving) as if you are among those whom we could not civilize'
  - Uygar `civilized' + laş `become'

    + tır `cause' + ama `not able'

    + dik `past' + lar 'plural'

    + ımız 'p1pl' + dan 'abl'

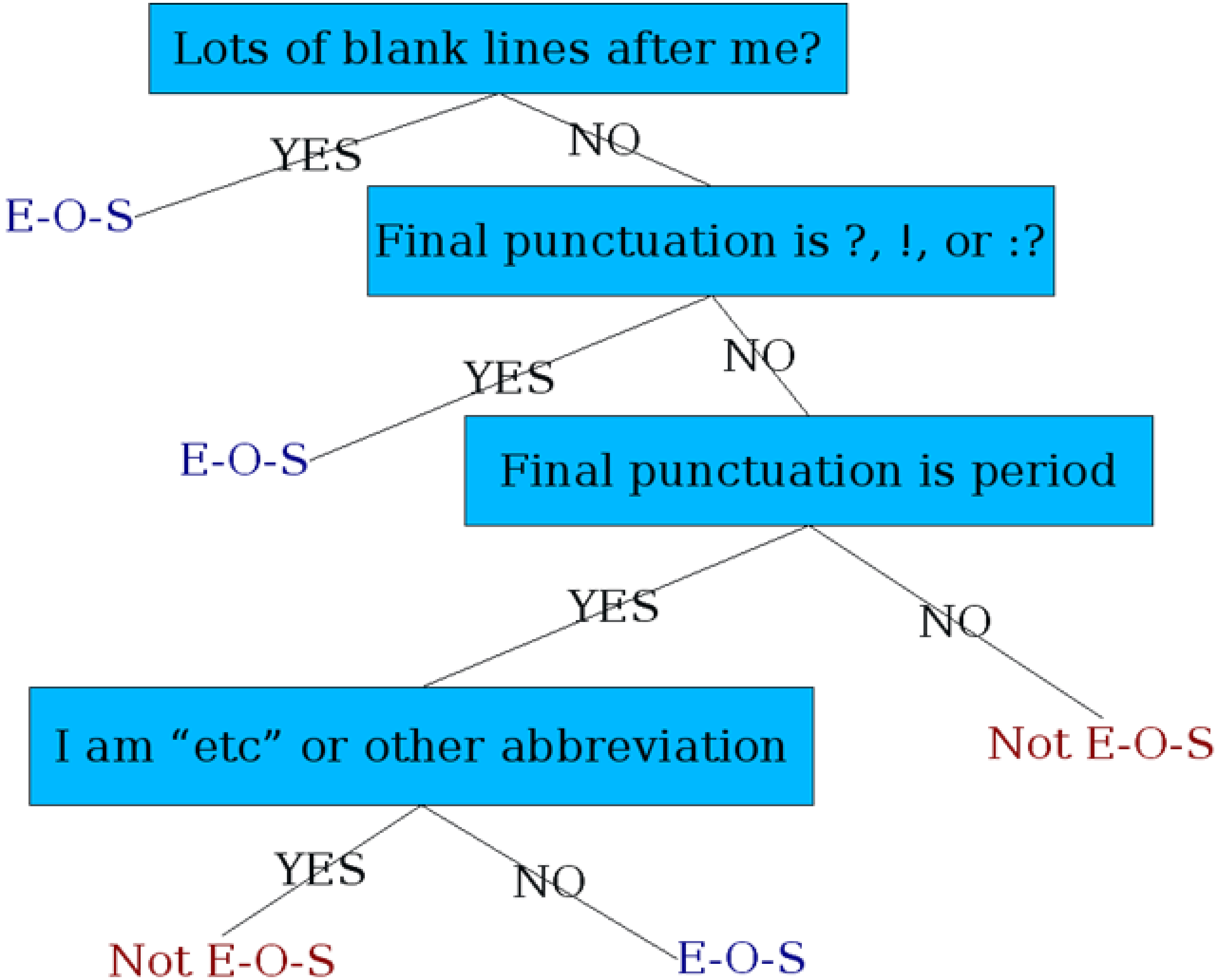
    + mış 'past' + sınız '2pl' + casına 'as if'

SENTENCE SEGMENTATION

Where to break sentences?

- !, ? are relatively unambiguous
- Period "." is <u>very</u> ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a classifier
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

A Decision Tree



More sophisticated features

- Case of word preceding ".":
  Upper, Lower, Cap, Number
- Case of word following ".":
  Upper, Lower, Cap, Number

- Numeric features
  - Length of word preceding "."
  - Probability (word preceding "." occurs at end-of-sent)
  - Probability (word after "." occurs at beginning-of-s)

Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting question is choosing the features

- Setting up the structure is often too hard to do by hand
  - Only possible for very simple features, domains
    - For numeric features, it's too hard to pick each threshold
  - Instead, structure usually learned by machine learning from a training corpus (later in the course)

HOMEWORK 1

Goals of Homework 1

- To apply the Information Theory concepts
- To begin to work with text data in python, and apply an open-source NLP package
  - NLTK
- To gain exposure to an openly available NLP research dataset
  - **GLUE Benchmark**

Homework 1 Steps

- Due Weds. September 13 at 11:59pm
- 100 points total
- 7 questions:
  – Download and prepare data
  – Write 4 functions and apply them to data
  – Answer 2 questions on Blackboard

GLUE Benchmark Datasets

- GLUE: **G**eneral **L**anguage **U**nderstanding **E**valuation
- Goal: "favor and encourage models that share general linguistic knowledge across tasks.'
- 9 NLU tasks with a range of sources:
  - Classification: sentiment, linguistic acceptability,
  - Semantic similarity and equivalence
  - Inference: Question answering, entailment, pronoun resolution

https://openreview.net/pdf?id=rJ4km2R5t7

NLTK

- NLTK: Natural Language Toolkit
- Open-source python library for a range of NLP tasks, including Tokenization and Stemming

```
>>> word_tokenize("I wouldn't expect it to split this
way!")

['I', 'would', "n't", 'expect', 'it', 'to', 'split',
'this', 'way', '!']
```

Homework 1 Help

- TA Office Hours start Tues Sept 5
  - Virtual and in-person
  - At least one office hour every weekday
  - Schedule posted in Blackboard – see "Content"
- Please bring python and machine setup questions to TA Office Hours
- Blackboard discussion group is available for questions and clarifications about instructions. See "HW 1 Discussion"

Homework 1 Setup

- **<u>Start your Python setup early</u>** if you are new to Python, or not yet set up on machine you will use for this class
- We recommend completing your setup **<u>next week</u>** (week of Sept 5-8). This means you can:
  1. Use python interpreter, e.g. `print("Hello World")`
  2. Unzip data files
  3. Run notebook "HW1_GettingStarted.ipynb"

Homework Submission

- Due Weds. September 13 at 11:59pm on Blackboard
- The homework late policy as posted in course syllabus applies to HW1
- Multiple submissions are allowed; no penalty for resubmission. Only last submission will be graded
- Submit early in case of last-minute technical issues