EECS 595

Natural Language Processing

Lecture 1: Introduction

Instructor: Joyce Chai

Logistics

- Instructor: Joyce Chai
- In-person lecture: mask mandate
  - Remote session is accommodated
- Office Hours: **Zoom only.** Wednesday 10:45-12:15 or by appointment,
- TA:
  - Peter Yu (kpyu@umich.edu)
  - Shane Storks (sstorks@umich.edu)
- CANVAS
  - Syllabus, lecture notes, lecture videos, assignments
- Piazza for discussions

Structure of the class

- **A graduate-level** introductory course with three goals:
  - Learn the basic principles and theoretical issues underlying natural language processing
  - Learn techniques and tools used to develop practical, robust systems
  - Gain insight into many open research problems in natural language
- A mixture of lectures, reading, hands-on experience
  - Fundamental problems and approaches, and recent research advances
  - 20 lectures + 6 sessions on recent advances

Textbook and Lecture Notes

- *Speech and Language Processing, an introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, third edition (draft)* by Daniel Jurafsky and James Martin, Prentice Hall.

- Optional: Neural Network Methods for Natural Language Processing, Yoav Goldberg, Synthesis Lectures on Human Language Technologies

Programming Requirements

- Proficiency in Python Programming
  - https://pythonprogramming.net/
- NLP toolkit: NLTK
  - https://www.nltk.org/
  - A good tutorial on NLTK: NLTK with Python 3 for Natural Language Processing

Grading

- Four homework assignments: 60%
  - A written portion and a programming portion
  - Written portion: use Latex.
  - Submission through CANVAS

- Final project (40%):
  - 1-2 people
  - You can choose your own topic
  - A list of default topics will be available to you.
  - The scope of the project should be proportional to the effort.

What is NLP?

Dave Bowman: Open the pod bay doors, HAL.

What is NLP?

Dave Bowman: Open the pod bay doors, HAL.

HAL: I’m sorry Dave. I’m afraid I can’t do that.

What is NLP

- The study of human languages and how they can be represented computationally and analyzed and generated algorithmically
  - *The dog likes bacon.* --> like (dog, bacon)
  - like (dog, bacon) --> *The dog likes bacon*
- Studying NLP involves studying natural language, formal representations, and algorithms for their manipulation
- Applications
  - information extraction, question answering, machine translation, conversational systems

Multidisciplinary

- **Linguistics**: how words, phrases, and sentences are formed.
- **Psycholinguistics**: how people understand and communicate using human language
- **Philosophy**: relates to the semantics of language; notation of meaning. NLP requires considerable knowledge about the world

Multidisciplinary

- **Computer Science**: deals with model formation and implementation
- **Mathematics and Statistics**: deals with probabilities, statistical distribution and hypothesis testing of language phenomena
- **Artificial Intelligence**: relates to knowledge representation and reasoning

Language Ambiguities

I made her duck.

- How many different interpretations does the above sentence have?
- How can each ambiguous piece be resolved?

Language Ambiguities

- Lexical ambiguity: when a word has more than one part of speech

  Rice flies like sand.

- Structural ambiguity:

  John saw the boy with a telescope

  John saw the boy with a telescope

Basic levels of language processing

- Phonetics: how words are related to the sounds that realize them.
- Morphology: how words are constructed. *beauty, beautiful*
- Syntax: how words can be put together to form correct sentences, and the role of each plays in the sentence. *John likes Mary*

Basic levels of language processing

- Semantics: the meaning of words and sentences

  bass fishing, bass playing

- Discourse: how the meaning of words and sentences is affected by the surrounding text or utterances

  Mary bought a new computer yesterday. She likes it very much. (pronoun resolution)

- Pragmatics: how sentences are used in different situations (contexts)

  Mary grabbed her umbrella

  A) It is a cloudy day

  B) She was afraid of dogs

Goal: Deep Understanding

- Requires context, linguistic structure, meanings…

Reality: Shallow Matching

- Requires robustness and scale
- Amazing successes, but fundamental limitations

(slide from Dan Klein, Taylor Berg-Kirkpatrick)

Exciting Time for NLP!

- Large data sets and computational resources have become available to build more powerful models.
- Many tools have become available to make real-world applications possible.
  - Play an important role in curbing information explosion on the internet
  - Used for building natural interfaces to databases, machine translations, chatbots
- NLP still remains a challenging problem despite of recent excitement

New AI Model Exceeds Human Performance at Question Answering

Dave Costenaro Follow

Nov 21, 2018 · 5 min read

(BecomingHuman.ai)

AI models from Microsoft and Google already surpass human performance on the SuperGLUE language benchmark

Kyle Wiggers @Kyle_L_Wiggers January 6, 2021 11:04 AM

(The Machine)

AI, ML & DATA ENGINEERING

AI Models from Google and Microsoft Exceed Human Performance on Language Understanding Benchmark

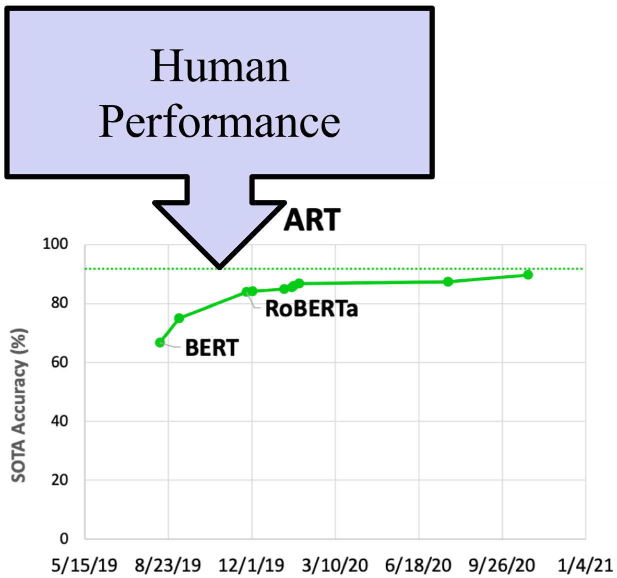
LIKE DISCUSS

JAN 12, 2021 • 3 MIN READ

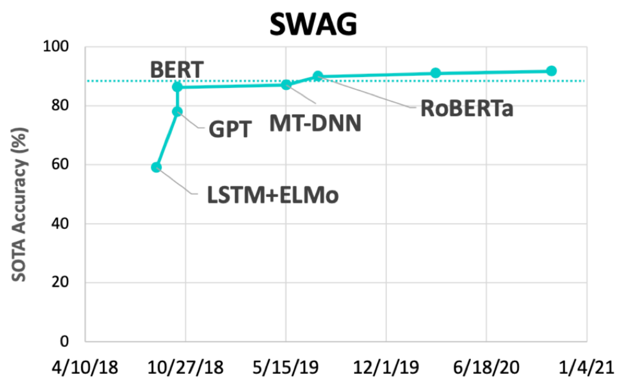
Research teams from Google and Microsoft have recently developed natural language

(InfoQ)

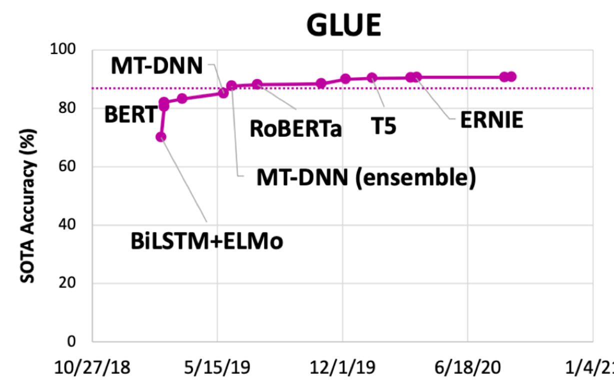
Leaderboard Ranking



https://leaderboard.allenai.org/anli/submissions/public



https://leaderboard.allenai.org/swag/submissions/public

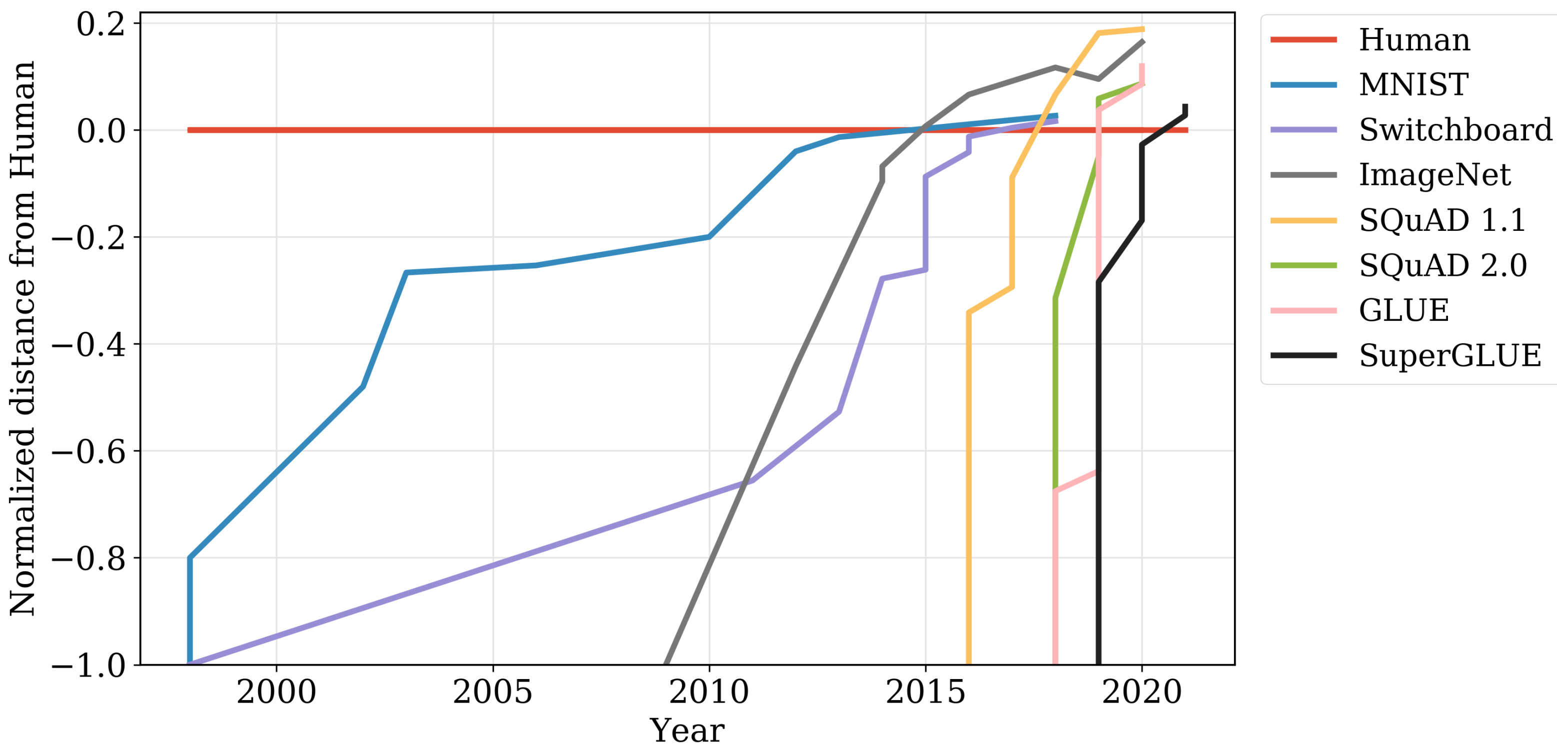


https://gluebenchmark.com/leaderboard

S. Storks, Q. Gao, and J.Y. Chai. Recent advances in natural language inference: a survey of benchmarks, resources, and approaches, arXiv preprint arXiv:1904.01172, 2019.

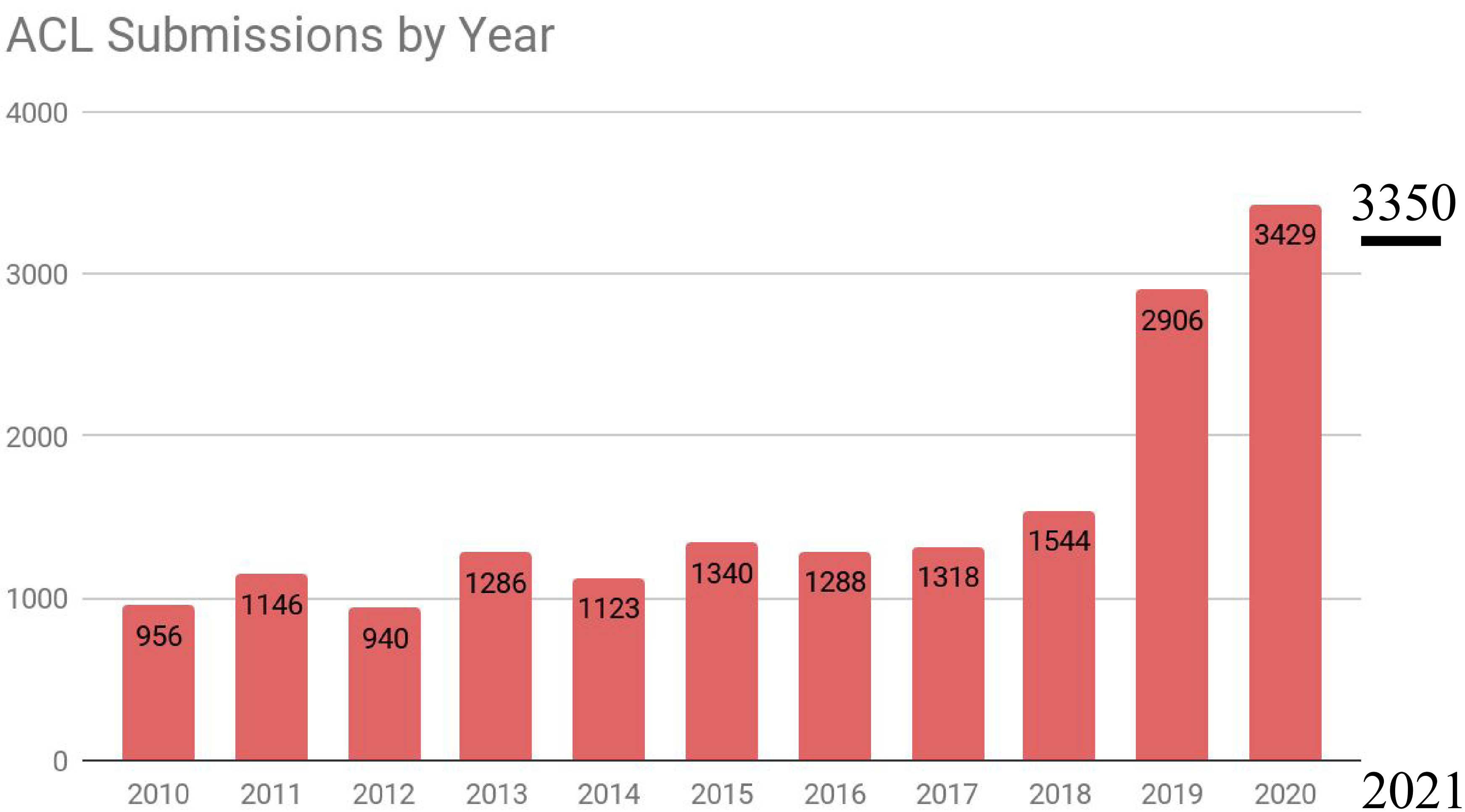
(From Christopher Potts' ACL2021 Keynote speech)

Benchmarks saturate faster than ever



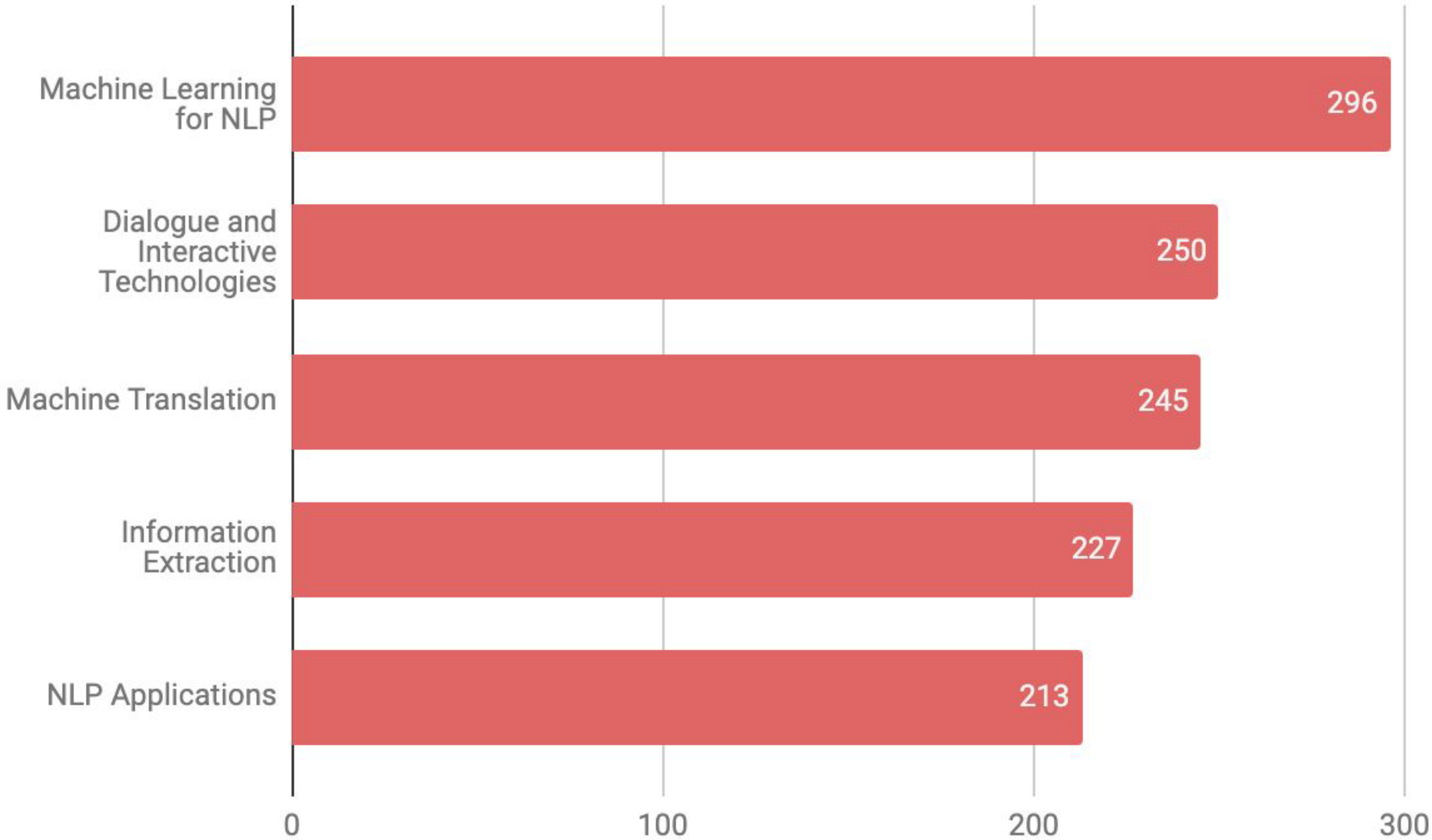
Kiela et al. 2021

Association for Computational Linguistics



(ACL2020)

5 tracks with over 200 submissions! (ACL 2020)



Topics covered in this class

- Three major parts:
  - Linguistic, mathematical, and computational background
  - Levels of linguistic processing: morphology, syntax, semantics, and discourse
  - Applications: sentiment analysis, information extraction, question answering, machine translation, dialogue systems

Today

- Review some of the simple representations and ask ourselves how we might use them to do interesting and useful things
  - Regular Expressions
  - Minimum editing distance

Regular expressions

- A formula in a special language that is used for specifying simple classes of strings
  - A string is a sequence of symbols
  - For text-based search, a string is a sequence of alphanumeric characters (letters, numbers, spaces, tabs, and punctuation)
- Can be used to specify search strings and define a language in a formal way.

Basic Regular Expression Patterns

All modern language have similar library packages for regular expressions

- Case sensitive
- Disjunctions **`[abc]`**
- Ranges **`[A-Z]`**
- Negations **`[^Ss]`**
- Optional characters **`?, +,`** and **`*`**
- Wild cards **`.`**
- Anchors **`^`** and **`$`**, also **\b** and **\B**
- Disjunction, grouping, and precedence |

Regular expressions

- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

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`

Regular Expressions: Negation in Disjunction

- Negations `[^Ss]`
  - Carat 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 ^	`ears`
`a^b`	The pattern a carat b	`Look up a^b now`

Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]ood chuck	

Regular Expressions: ? * + .

Kleene *, Kleene +

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

Regular Expressions: Anchors ^ $

- `^` beginning of the string
- `$` end of the string

Pattern	Matches
`^[A-Z]`	Palo Alto
`^[^A-Za-z]`	1 “Hello”
`\.$`	The end.
`.$`	The end? The end!

RE	Expansion	Match	First Matches
`\d`	`[0-9]`	any digit	Party␣of␣5
`\D`	`[^0-9]`	any non-digit	Blue␣moon
`\w`	`[a-zA-Z0-9_]`	any alphanumeric/underscore	Daiyu
`\W`	`[^\w]`	a non-alphanumeric	!!!!
`\s`	`[␣\r\t\n\f]`	whitespace (space, tab)	
`\S`	`[^\s]`	Non-whitespace	in␣Concord

Figure 2.7 Aliases for common sets of characters.

RE	Match
`*`	zero or more occurrences of the previous char or expression
`+`	one or more occurrences of the previous char or expression
`?`	exactly zero or one occurrence of the previous char or expression
`{n}`	*n* occurrences of the previous char or expression
`{n,m}`	from *n* to *m* occurrences of the previous char or expression
`{n,}`	at least *n* occurrences of the previous char or expression
`{,m}`	up to *m* occurrences of the previous char or expression

Figure 2.8 Regular expression operators for counting.

Python reg exp. package

- >>> import re
- Basic functions:
  - **re.search:** search looks for a pattern anywhere in a string
  - **re.match:** match looks for a match staring at the beginning
  - **re.split**: split a string into several parts based on a pattern
  - **re.sub**: substitutes one string for a pattern
  - **re.findall**: find all from a string that matches a pattern
  - And others you may find useful.

Example

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

  `/the/` Misses capitalized examples

  `/[tT]he/` Incorrectly returns `other` or `theology`

`/[^a-zA-Z][tT]he[^a-zA-Z]`

/\b[tT]he\b/

Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

A more complex example

- Exercise: Write a regular expression that will match "any PC with more than 500MHz and 32 Gb of disk space for less than $1000":

```
/$[0-9]+/
/$[0-9]+\.[0-9][0-9]/
/\b$[0-9]+(\.[0-9][0-9])?\b/
/\b[0-9]+ *(MHz|[Mm]egahertz|Ghz|
    [Gg]igahertz)\b/
/\b[0-9]+ *(Mb|[Mm]egabytes?)\b/
/\b[0-9](\.[0-9]+) *(Gb|[Gg]igabytes?)\b/
```

Another example used in earlier chat bots

Eliza [Weizenbaum, 1966]

Eliza: http://www.manifestation.com/neurotoys/eliza.php3

User: Men are all alike
ELIZA: IN WHAT WAY
User: They're always bugging us about something or other
ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE?
User: Well, my boyfriend made me come here
ELIZA: YOUR BOYFRIEND MADE YOU COME HERE
User: He says I'm depressed much of the time
ELIZA: I AM SORRY TO HEAR THAT YOU ARE DEPRESSED

Substitutions and memory

- Substitutions (Transduction)

  `s/colour/color/`

- Memory (`\1`, `\2`, etc. refer back to matches)

35 boxes => <35> boxes: `s/([0-9]+)/<\1>/`

s/regexp1/pattern/
s/I am feeling (.+)/You are feeling \1?/
s/I gave (.+) to (.+)/Why would you give \2 \1?/

Eliza-style regular expressions

Step 1: replace first person references with second person references
Step 2: use additional regular expressions to generate replies
Step 3: rank possible transformations

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO
  HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU
  THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC
  EXAMPLE/
```

Uses of Regular Expressions in NLP

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations

Minimum Edit Distance

- Much of NLP concern with how similar two strings are
  - Spell checking and correction
  - Word Error Rate for speech recognition
  - machine translation, etc.
- MED s the minimum number of editing operations needed to transform one into the other
  - Insertion
  - Deletion
  - Substitution

Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - Distance between them is 8

Minimum Edit Distance

One possible path

```
i n t e n t i o n
                   ← delete i
n t e n t i o n
                   ← substitute n by e
e t e n t i o n
                   ← substitute t by x
e x e n t i o n
                   ← insert u
e x e n u t i o n
                   ← substitute n by c
e x e c u t i o n
```

There can be many different paths
The problem becomes the search problem to find the path with minimum cost

Defining Min Edit Distance

- For two strings S_1 of len n, S_2 of len m
  - distance(i,j) or D(i,j)
    - means the edit distance of $S_1[1..i]$ and $S_2[1..j]$
    - i.e., the minimum number of edit operations need to transform the first i characters of S_1 into the first j characters of S_2
    - The edit distance of S_1, S_2 is D(n,m)
- We compute D(n,m) by computing D(i,j) for all i ($0 \leq i \leq$ n) and j ($0 \leq$ j $\leq$ m)
- Note the index associated with the source/target string: first is source and second is the target

Defining Min Edit Distance

- Base conditions:
  - $D(i,0) = i$ /* *deletion cost*/
  - $D(0,j) = j$ /* *insertion cost*/

  - Recurrence Relation:
  - $$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{/* cost for deletion*/} \\ D(i,j-1) + 1 & \text{/* cost for insertion*/} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

    /* cost for substitution */

Dynamic Programming

- A tabular computation of D(n,m)
- Bottom-up
  - Compute D(i,j) for smaller i,j
  - Increase i, j to computer D(i,j) using previously computed values based on smaller indexes.

The Edit Distance Table

source

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

target

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$