Regular Expressions and Python

Methods in Computational Linguistics I
October 22, 2010
Last Time

- Objects in Python
Today

• Regular Expressions
Matching Strings

- `string.startswith("He")`
- `string.endswith(".")`
- `string.find("the")`
- `string.replace("the", "a")`
- `string.replace("the", "a", 1)`
Regular Expressions

- Regular expressions are a more flexible technique for string matching and replacement.

- For example:
  - Match one or more instances of the same character
  - Matching more than one group of characters
  - Context based replacement.
Regular Expressions

- Match any character.
  - ‘a.b’ matches ‘axb’ or ‘a b’ or ‘a qb’
- Match one or more of a character
  - ‘a+b’ matches ‘ab’ or ‘aaaaab’
- Match zero or more characters
  - ‘a*b’ matches ‘b’ ‘ab’ or ‘aaaaaaaaab’
Regular Expressions

- Matching groups
  - ‘(abc|def)’ matches abc or def
    - this can be extended to ‘(abc|def|ghi)’
- Match one of a set of characters
  - ‘a[bcd]e’ matches ‘abe’ ‘ace’ and ‘ade’
- Match anything but a set of characters
  - ‘a[^bcd]e’ matches ‘ape’ but not ‘ace’
Regular expressions

- Match exactly n of something
  - ‘ab{2}c’ matches ‘abbc’
  - ‘ab{2,4}c’ matches ‘abbc’, ‘abbbbc’ and ‘abbbbbc’
Positional Matching

- In general, regular expressions will match any part of a string.
- e.g. ‘a[bcd]e’ will match ‘gatorade’ and ‘abe lincoln’
- Positional matching allows you to match at the start or end of strings
  - ‘^abc’ matches ‘abc’ but not ‘123abc’
  - ‘abc$’ matches ‘abc’ but not ‘abc123’
Special Characters

- ‘\s’ matches all white space
- ‘\S’ matches no white space
- ‘\w’ matches any “word” character alphanumeric plus underscore ‘_’
- ‘\W’ matches no word characters
- ‘\d’ matches any digit
- ‘\D’ matches any non-digit
Escaping characters

- We have used some characters in regular expressions. Specifically
  - . ? [ ] { } ( ) ^ $ | * \\
- To match these characters prefix them with a backslash
  - ‘Hello\.’ matches ‘Hello.’ but not ‘Hello!’
  - ‘Hello.’ matches ‘Hello.’ and ‘Hellox’
Using regular expressions in Python

- Demoing regular expressions
Topic Modeling and Segmentation

• Identifying words that identify a topic.
• Take some known examples of a topic
  • News documents
  • Emails
  • Blog posts, etc.
• Identify words that are indicative of a topic
Topic modeling

• How do we know which words are specific to a topic?

• What about similar words?
  • e.g. running and marathon.
Topic Segmentation

• Identify where one topic starts and another ends.

• The goal is to make semantically homogenous subdocuments.

• Useful for Information Retrieval.

• Do specific words, syntactic constructions or other linguistic phenomena indicate topic shift?
Topic Modeling for Topic Segmentation

• Evaluate the topic models on consecutive spans of text drawn from the document.
• Identify the point at which a new topic model “fits” the text better.
• This is the point of segmentation.
Speech Information for Topic Segmentation

- Speech offers some additional information for segmentation.
- Speaking rate and duration.
- Pitch changes
- Intensity changes.
Next Time

- Regular Expressions