Speech and Language Processing

by Jurafsky, Martin

1 Introduction

2 Regular Expressions, Text Normalisation, Edit Distance

Regular expressions are the most important tool for describing text patterns.
Text normalisation is the process of converting text to a more standard form: for example, tokenisation (splitting up text into words), lemmatisation (identifying common roots) or stemming (removing suffixes), and sentence segmentation (splitting up text into sentences).
Edit distance is a metric that measures how similar two strings are.

2.1 Regular Expressions

Formally, a regular expression is an algebraic notation for characterising a set of strings. They are useful for searching in texts.

2.1.1 Basic Regular Expression Patterns

The simplest kind of regular expression is just a sequence of characters, like `/woodchuck/` [slashes are not part of expressions]. Regexes are case sensitive.

We can use braces, which indicate disjunctions, to solve this issue. To match both *woodchucks* and *Woodchucks*, we can use `/[wW]oodchucks/`.
Ranges can be specified using the hyphen; e.g. `/[A-Z]/` matches any uppercase letter.

A caret in the *beginning* of the square braces expression can be used to specify a "negative" pattern: what the character shouldn't be. Thus `/[^0-9]/` matches any character that is not a number (the caret negates the entire expression inside the brackets).

A question marks indicates "the previous character or nothing"; for instance `/woodchucks?/` matches *woodchuck* and *woodchucks*.

The Kleene star or asterisk allows us to match *zero or more* occurrences of a character. Therefore `/a*/` matches any string of 0 or more a's. Similarly, `/aa*/` matches one or more a's; `/[ab]*/` matches all strings consisting only of a's and b's; and so on.
One useful example is `/[0-9][0-9]*/`, which matches any nonnegative integer.
For convenience, the Kleene plus can be used to indicate one or more occurences; thus `/[0-9]+/`.

The period is a wildcard expression that matches any single character *except* a carriage return. Therefore `/beg.n/` matches *begin*, *begun*, *began*, etc.
`/.*/` is used often and means "any string of characters".

Anchors are special characters that indicate where in a string we wish the regex to match. The caret indicates the beginning and the dollar indicates the end. Thus `/^The/` matches *The* if it's the first word, and `/ $/` matches a space at the end of a line.
Other anchors are `\b` for a word boundary, and `\B` for a non-boundary. A word is defined as any sequence of digits, letters or underscores.

2.1.2 Disjunction, Grouping and Precedence

The disjunction operators for strings is the pipe; thus `/cat|dog/` matches either *cat* or *dog*.

Precedence is overridden using ordinary parentheses, as in `/gupp(y|ies)/`, which matches *guppy* and *guppies*.
Note that the Kleene star matches only the preceding character, and therefore needs parens to cover an entire regex. Therefore `/(Column [0-9]+ *)*/` matches any repetition of the word *Column*, followed by a space and an integer, followed by any number of spaces.

The precedence order for operators is `() > *,+,?,{} > ^,seqs > |`.

To avoid ambiguity, regexes are defined to match greedily: they match the largest possible string. However, the `?` qualifier can be used to enforce non-greedy matching.

2.1.3 A Simple Example

To match all occurrences of *the*, we could say `/the/`; but it won't match *The*.
`/[tT]he/` would, but it would match part of *theology* also.
`/\b[tT]he\b/` wouldn't, but words aren't defined the same.
`/[^a-zA-Z][tT]he[^a-zA-Z]/` would work, but it would only match if there was some character preceding *the*, and some character following it.
`/(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/` would work.

The above procedure illustrates two efforts in developing speech and language processing systems:

- increasing precision (reducing false +ves), and
- increasing recall (reducing false –ves).

2.1.4 A More Complex Example