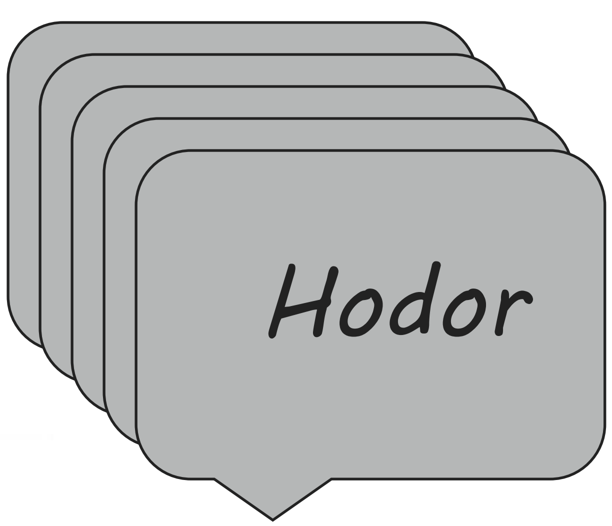
```
rules = {
    RegularExpression["(\\w+)(ss)(es)"] :> "$1$2",
    RegularExpression["(\\w+)(sh)(es)"] :> "$1$2",
    RegularExpression["(\\w+)(ies)"] :> "$1" ~~ "y",
    RegularExpression["(\\w+)(ss)"] :> "$1$2",
    RegularExpression["(\\w+)(us)"] :> "$1$2",
    RegularExpression["(\\w+)(s)"] :> "$1"
  };
```

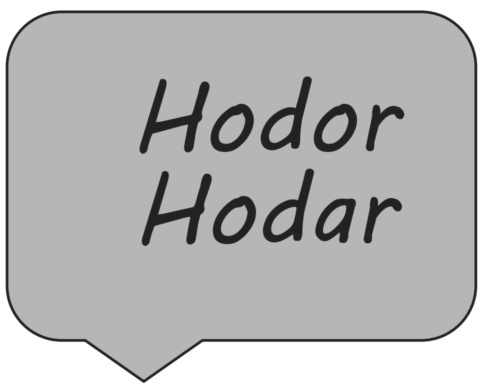
COMP90042 LECTURE 1B

# PREPROCESSING

# DEFINITIONS

- Corpus: a collection of documents
- Document: a collection of tokens
- Token: a particular instance of a type
- Type: a distinct word form
- Lexicon: a grouping of types
- Vocabulary: a list of all word types in the corpus




$V = \{\text{Hodor}\}$

# TEXT NORMALISATION

- Remove unwanted formatting (e.g. HTML)
- Segment structure (e.g. sentences)
- Tokenise words
- Normalise words
- Remove unwanted words

“<p> Hi there. I’m Julian. </p>”

“Hi there. I’m Julian.”

[“Hi there.”, “I’m Julian.”]

[[“Hi”, “there”, “.”], [“I”, “’m”, “Julian”, “.”]]

[[“hi”, “there”, “.”], [“i”, “be”, “julian”, “.”]]

[[],[“julian”]]

# SENTENCE SEGMENTATION

- Naïve approach: break on sentence punctuation ([.?!])
  - But periods are used for abbreviations! (U.S. dollar)
- Second try: use regex to require capital ([.?!] [A-Z])
  - But abbreviations often followed by names (Mr. Brown)
- Better yet: have lexicons
  - But difficult to enumerate all names and abbreviations
- State-of-the-art uses machine learning, not rules

# TOKENISATION: ENGLISH

- Naïve approach: separate out alphabetic strings (\w+)
- Abbreviations (*U.S.A.*)
- Hyphens (*merry-go-round vs. well-respected* vs. *yes-but*)
- Numbers (*1,000,00.01*)
- Dates (3/1/2016)
- Clitics (*n't* in *can't*)
- Internet language (http://www.google.com, #RefugeesWelcome, :-))
- Multiword units (New Zealand)

# TOKENISATION: CHINESE

- Some Asian languages are written without spaces between words
- In Chinese, words often correspond to more than one character

墨大的学生与众不同

| | | | |
|---|---|---|---|
| 墨大 | 的 | 学生 | 与众不同 |
| Unimelb | ’s | student(s) (are) | special |

# TOKENISATION: CHINESE

- Standard approach assumes an existing vocabulary
- MaxMatch algorithm
  - Greedily match longest word in the vocabulary

V = {墨,大,的,学,生,与,众,不,同, 墨大,学生,不同,与众不同}

墨大的学生与众不同

| | | | |
|---|---|---|---|
| match 墨大, | match 的, | match 学生, | match与众不同, |
| move to 的 | move to 学 | move to 与 | done |

# TOKENISATION: CHINESE

- But how do we know what the vocabulary is
- And doesn't always work

去买新西兰花

| | | | |
|---|---|---|---|
| 去 | 买 | 新西兰 | 花 |
| go | buy | New Zealand | flowers |

| | | | |
|---|---|---|---|
| 去 | 买 | 新 | 西兰花 |
| go | buy | new | broccoli |

# WORD NORMALISATION

- Lower casing (Australia -> australia)
- Removing morphology
- Correcting spelling
- Expanding abbreviations

# INFLECTIONAL MORPHOLOGY

- Inflectional morphology creates grammatical variants
- English inflects nouns, verbs, and adjectives
  - Nouns: *number* of the noun (*-s*)
  - Verbs: *number* of the subject (*-s*), the *aspect* (*-ing*) of the action and the *tense* (*-ed*) of the action
  - Adjectives: *comparatives* (*-er*) and *superlatives* (*-est*)
- Many languages have much richer inflectional morphology than English
  - E.g. French inflects nouns for gender (*un chat*, *une chatte*)

# LEMMATISATION

- Lemmatisation means removing any inflection to reach the uninflected form, the *lemma*
  - *speaking* → *speak*
- In English, there are irregularities that prevent a trivial solution:
  - *poked* → *poke* (not pok)
  - *stopping* → *stop* (not *stopp*)
  - *watches* → *watch* (not *watche*)
  - was → be (not *wa*)
- A lexicon of lemmas needed for accurate lemmatisation

# DERIVATIONAL MORPHOLOGY

- Derivational morphology creates distinct words
- English derivational *suffixes* often change the lexical category, e.g.
  - *-ly* (*personal* → *personally*)
  - *-ise* (*final* → *finalise*)
  - *-er* (*write* → *writer*)
- English derivational *prefixes* often change the meaning without changing the lexical category
  - *write* → *rewrite*
  - *healthy* → *unhealthy*

# STEMMING

- Stemming strips off all suffixes, leaving a *stem*
  - *E.g. automate, automatic, automation* $\rightarrow$ *automat*
  - Often not an actual lexical item
- Even less lexical sparsity than lemmatisation
- Popular in information retrieval

# THE PORTER STEMMER

- Most popular stemmer for English
- Applies rewrite rules in stages
  - First strip inflectional suffixes,
    - E.g. *-ies* $\rightarrow$ *-i*
  - Then derivational suffixes, from right to left
    - E.g *-isation* $\rightarrow$ *-ise; -ise* $\rightarrow$

# FIXING SPELLING ERRORS

- Why fix them?
  - Spelling errors create new, rare types
  - Disrupt various kinds of linguistic analysis
  - Very common in internet corpora
  - In web search, particularly important in queries
- How?
  - String distance (Levenshtein, etc.)
  - Modelling of error types (phonetic, typing etc.)
  - Use an $n$-gram language model

# OTHER WORD NORMALISATION

- Normalising spelling variations
  - Normalize → Normalise (or vice versa)
  - U r so coool! → *you are so cool*
- Expanding abbreviations
  - US, U.S. → United States
  - imho → in my humble opinion

# STOP WORDS

- Definition: a list of words to be removed from the document
  - Typical in bag-of-word (BOW) representations
  - Not appropriate when sequence is important
- How to choose them?
  - All *closed-class* or *function* words
    - E.g. *the, a, of, for, he, …*
  - Any high frequency words

# A FINAL WORD

- Preprocessing unavoidable in text analysis
- Can have a major effect on downstream applications
- Exact steps may vary depending on corpus, task
- Simple rule-based systems work well, but rarely perfectly

# FURTHER READING

- J&M3 Ch 2. on Normalisation (includes a review of regex and Levenshtien distance)
- (Optional) details on the Porter Stemmer algorithm (http://snowball.tartarus.org/algorithms/porter/stemmer.html)