Text analysis

From a string of characters to a list of words.

Text analysis

- ☐ This is a pen.
- ☐ My cat who lives dangerously has nine lives.
- ☐ He stole \$100 from the bank.
- ☐ He stole 1996 cattle on 25 Nov 1996.
- ☐ He stole \$100 million from the bank.
- ☐ It's 13 St. Andrew St. near the bank.
- ☐ Its a PIII 1.2Ghz, 512MB RAM, 14.5Gb SCSI, 56x cdrom and 17" LCD monitor.
- ☐ My home pgae is `http://www.geocities.com/awb/`.

```
from awb@cstr.ed.ac.uk ("Alan W Black") on Thu 23 Nov 15:30:45:
>
>  ...  but, *I* wont make it :-) Can you tell me who's going?
>
  IMHO I think you should go, but I think the followign are going
     George Bush
     Bill Clinton
     and that other guy

Bob

--
                                                              -------
 +----------------------------------------------------+  |\\   //|
 | Bob Beck  E-mail bob@beck.demon.co.uk              |  | \\ // |
 +----------------------------------------------------+  |  > <  |
                                                         | // \\ |
                                          Alba gu brath |//___\\|
```

Text analysis: subtasks

- ☐ Identify tokens in text.
- ☐ chunk the tokens into reasonably sized sections.
- ☐ identify types for tokens.
- ☐ map tokens to words
- ☐ identify types for words.

Identifying tokens

- ☐ Character encodings:
  - – latin-1, iso8859-1, utf8, utf16
- ☐ Whitespce separated (European languages):
  - – space, tab, newline, cr, vertical tab ...
- ☐ What about punctuation?
  - – `The boy---who was usually late---arrived on time.`
  - – `We have orange/apple/banana flavors.`
- ☐ Festival splits into:
  - – tokens, plus ...
  - – stripped pre and post punctuation
  - – Thus list of items (with features: name, punc etc)

Chunking into utterances

- ☐ Can't analyze the whole file as one.
- ☐ someone is waiting for it to speak.
- ☐ Do it bit by bit:
  - – audio spooling (cf. printer spooling)
  - – time to first sound
  - – What size are the bits?

What size are the chunks?

- ☐ Big enough to do proper synthesis:
  – how much future information do you need
- ☐ Prosodic phrases are bounded (probably)
- ☐ How do you detect prosodic phrases in raw text?
- ☐ "Sentences" are a compromise.

Hi Alan,

I went to the conference. They listed you as Mr. Black when we
know you should be Dr. Black days ahead for their research.
Next month I'll be in the U.S.A. I'll try to drop by C.M.U.
if I have time.

bye
Dorothy
Institute of XYZ
University of Foreign Place
email: dot@com.dotcom.com

End of utterance decision tree

Selection based on look ahead of one token.

```
((n.whitespace matches ".*\n.*\n[.\n]*")
  ((1))
  ((punc in ("?" ":" "!"))
   ((1))
   ((punc is ".")
    ((name matches "\\(.*\\..*\\|[A-Z][A-Za-z]?[A-Za-z]?\\|etc\\)")
     ((n.whitespace is " ")
      ((0))
      ((n.name matches "[A-Z].*")
       ((1))
       ((0))))
     ((n.whitespace is " ")
      ((n.name matches "[A-Z].*")
       ((1))
       ((0)))
      ((1))))
    ((0)))))
```

Non-standard words

textual tokens whose pronunciation is not just through a lexicon (or letter to sound rules).

- ☐ Numbers:
  - 123, 12 March 1994.
- ☐ Abbreviations, contractions, acronyms:
  - approx, mph, ctrl-C, US, pp, lb
- ☐ punctuation conventions:
  - 3-4, +/-, and/or
- ☐ dates, times, URLs

Why do we need to worry about them?

- ☐ Synthesis:
  – must be able to read *everything* aloud.
- ☐ Recognition:
  – text conditioning for building language models
  – text type specific language modelling
  (e.g IRC/IM/SMS for LM training)
- ☐ Information extraction/queries:
  – token normalization
- ☐ Named entity recognition:
  – named entities more likely to use NSWs.

How common are they?

varies over text type:
– words not in lexicon, or with non-alphabetic chars

Text type	% NSW
novels	1.5%
press wire	4.9%
e-mail	10.7%
recipes	13.7%
classifieds	27.9%
IRC	20.1%

What is the distribution NSW types?

In NANTC (North American News Text Corpora)
from 121,464 NSWs.

major type	minor type	%
numeric	number	26%
	year	7%
	ordinal	3%
alphabetic	as word	30%
	as letters	12%
	as abbrev	2%

How difficult are they?

- ☐ Identification:
  - – some homographs “Wed”, “PA”
  - – some false positives: OOV
- ☐ Realization:
  - – simple rule: money, “$2.34”
  - – POS tags: “lives”/“lives”
  - – type identification + rules: numbers
  - – text type specific knowledge
- ☐ Ambiguity, (acceptable multiple answers):
  - – “D.C.” as letters or full words
  - – “MB” as “meg” or “megabyte”
  - – “250”

Existing techniques

- ☐ ignored
- ☐ lexical lookup
- ☐ hacky hand-written rules
- ☐ (not so hacky) hand-written rules
- ☐ statistically trained prediction

Regular Expression Matching

A well-defined matching language
(used in Emacs, grep, sed, perl, libg++ etc).

- `.` matches any character (except newline)
- `X*` zero or more `X`, `X+` one or more `X`
- `[A-Z]*` zero or more capital letters
- `\\(fish\\|chip\\)` matches fish or chip
- `-?[0-9]+` matches an integer
- `[A-Z][a-z]+` matches a capitalized alphabetic string

This is exactly the libg++ Regex language
Festival offers an interface through
`(string-matches ATOM REGEX)`

Decision Trees (CART)

A method for choosing a class, or number based on features (of a stream item).
– (QUESTION YESTREE NOTREE)

```
((name is "the")
 ((n.num_syllables > 1)
   ((0))
   ((1)))
 ((name in ("a" "an"))
   ((0))
   ((1))))
```

Operators: is, matches, in, `<`, `>`, `=`
`wagon` can automatically build trees from data

building CART trees

Requires set of samples with features:
P f0 f1 f2 f3 f4 ...
P may be class or float
`wagon` finds best (local) possible question that minimises the distortion in each partition.
CART is ideal for

- ☐ mixed features
- ☐ no clear general principle
- ☐ a reasonable amount of data
- ☐ can (sort of) deal with very large classes

Homographs

Words with same written form but different pronunciation

- ☐ Different part of speech: project
- ☐ Semantic difference: bass, tear
- ☐ Proper names: Nice, Begin, Said
- ☐ Roman Numerals: Chapter II, James II
- ☐ Numbers: years, days, quantifiers, phone numbers
- ☐ Some symbols: 5-3, high/low, usr/local

How common are they?
– Numbers: email 2.57% novels 0.00013%
– POS/hgs: WSJ 7.6%

Homograph disambiguation (Yarowsky)

Same tokens with different pronunciation

- □ Identify particular class of homographs
  – e.g. numbers, roman numerals, “St”.
- □ Find instances in large db with context
- □ Train decision mechanism to find most distinguished feature

Homograph disambiguation: example

Roman numerals: as cardinals, ordinals, letter

Henry V: Part I Act II Scene XI: Mr X is I believe, V I Lenin, and not Charles I.

- □ Extract examples with context features
- □ Label examples with correct class:
  – king, number, letter
- □ Build decision tree (CART) to predict class

Features

```
class: n(umber) l(etter) c(entury) t(imes)
rex rex_names section_name num_digits p.num_digits n.num_digits
pp.cap p.cap n.cap nn.cap
```

```
n II 0 0 0 11 7 2 3 7 0 0 1 1
n III 0 0 0 3 4 3 3 5 0 0 1 1
c VII 1 0 0 4 9 3 3 3 1 1 0 0
n V 0 0 1 3 4 1 1 2 0 1 0 1
n VII 0 0 1 2 4 3 1 2 0 1 0 1
...
```

```
    ((p.lisp_tok_rex_names is 0)
     ((lisp_num_digits is 5)
      ((number))
      ((lisp_num_digits is 4)
       ((number))
       ((nn.lisp_num_digits is 13)
...
                ((nn.lisp_num_digits is 2)
                 ((letter))
                 ((n.cap is 0) ((letter)) ((number))))))))
...
```

Homograph disambiguation: example

Example data features:
– surrounding words, capitalization, “king-like”, “section-like”

class	ord	let	card	times	total	correct	percent
ord	133	0	15	0	148	133/148	89.865
let	3	40	9	0	52	40/52	76.923
card	7	6	533	0	546	533/546	97.619
times	0	2	1	1	4	1/4	25.000

707/750 94.267% correct

Homograph disambiguation

But it still fails on many obscure (?) cases

- □ William B. Gates III.
- □ Meet Joe Black II.
- □ The madness of King George III.
- □ He's a nice chap. I met him last year.

How many homographs are there?

Very few actually, ...

axes bass Begin bathing bathed bow Celtic close cretan Dr executor jan jean lead live lives Nice No Reading row St Said sat sewer sun tear us wed wind windier windiest windy winds winding windily wound Number num/num num-num Roman_numerals

Plus *many* POS homographs

Letter Sequences?

Letter sequences as words, letters, abbreviations (or mix)

- □ IBM, CIA, PCMCIA, PhD
- □ NASA, NATO, RAM
- □ etc, Pitts, SqH, Pitts. Int. Air.
- □ CDROM, DRAM, WinNT, SunOS, awblack

Require lexicon, or “pronouncability oracle”.
Heuristic: captialization and vowels

Alphabetic tag sub-classification

NSW tag **t** for alphabetic observations **o**
NATO: ASWD, **PCMCIA**: LSEQ, **frplc**: EXPN

- ☐
$$p(\mathbf{t}|\mathbf{o}) = \frac{p_t(\mathbf{o}|\mathbf{t})p(\mathbf{t})}{p(\mathbf{o})}$$
where $\mathbf{t} \in [ASWD, LSEQ, EXPN]$.
- ☐ $p_t(\mathbf{o}|\mathbf{t})$ estimated by a letter trigram model
$$p_t(\mathbf{o}|\mathbf{t}) = \prod_{i=1}^{N} p(l_i|l_{i-1}, l_{i-2}),$$
- ☐ $p(\mathbf{t})$ prior from data or uniform
- ☐ normalized by
$$p(\mathbf{o}) = \sum_t p_t(\mathbf{o}|\mathbf{t})p(\mathbf{t})$$

Text to Speech in Festival

- ☐ Chunk into utterances:
  – using eou tree
- ☐ Apply modules to each chunk
  – Defined `hooks` (list of functions)
  – `utt.synth, utt.play`
- ☐ Can modify what tts is for different needs
  – tokenizer only
  – phoneme generator
  – Word analysis

Partial analysis

```
(define (find-pos utt)
  (Token_POS utt)
  (Token utt)
  (POS utt)
)

(define (output-pos utt)
 (mapcar
  (lambda (word)
    (format t "%l/%l\n" (item.name word) (item.feat word "pos")))
  (utt.relation.items utt 'Word)))
  utt))

;;; Redefine what happens to utterances during text to speech
(set! tts_hooks (list find-pos output-pos))

;;; Do the work
(tts_file "-")
```

Exercises, for March 26th

Only 1 and 2 need be done.

1. Copy *text2pos* and modify it to output the number of nouns (of any type) in a given file.
2. Copy *text2pos* and modify it to output the number of vowels (phoneme vowels not letter vowels) in a given file.
3. Using the sentence "Mr. Green moved to Pittsburgh on 25 May 1976." create a Text utterance and hand synthesize it by applying each of the modules. At each stage identify which relations are created and what new features are added to the items.

Hints for Exercises, for March 26th

1.
   ```
   (set!  total_ns (+ 1 total_ns))
   (format t "Total number of nouns %d\n" total_ns)
   ```
2. See $SPPPDIR/src/festival/lib/synthesis.scm for the definition of `Tokens` UttType for list of extra modules to call. You want to look at the Segment relation
   ```
   (if (string-equals (item.feat seg "ph_vc") "+")
   (set!  total_vs (+ 1 total_vs))
   )
   ```
3. See $SPPPDIR/src/festival/lib/synthesis.scm for the definition of `Text` UttType for list of modules to call.
   The following functions are useful:
   ```
   (utt.relationnames utt)
   (utt.relation.items utt 'Segment)
   ```
   The following actually works for trees and lists
   ```
   (utt.relation_tree utt 'Segment)
   (set!  seg4 (nth 4 (utt.relation.items utt 'Segment)))
   (item.features seg4)
   ```