Traitements de Corpus — 4

Background Reading:

Traitements probabilistes et corpus, Traitement automatique des langues Vol. 36, 1995, no. 1–2:

Merialdo (Hidden Markov Model), 7–22; Stein & Schmid (Decision trees), 23–35; Loupy (Transformation rule-based), 37–46

Tokenisation

Sentence boundary detection

Tagging: Hidden Markov Model, transformation rule-based, Constraint Grammar

Chunking: finite-state transducer, memory-based learning, decision tree

Named entity recognition

Tokenisation

A big problem for Chinese etc. (no interword spacing, complex morphology) — not so bad for western European languages

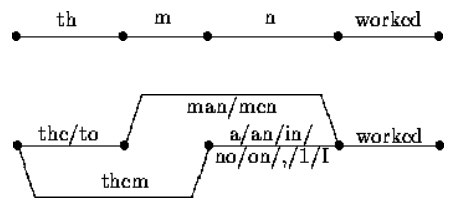
Issues that a tokeniser has to deal with:

- Normalizing quotation marks and identifying their type (i.e. translate ″ into either “ or ”)
- Differentiating ′ as apostrophe from ′ as close single quote
- Differentiating genitive *’s* from *’s* as a permanent portion of a name (e.g. *McDonald’s* the company $\neq$ *McDonald* followed by genitive *’s*, and is quite different from *John’s*)
- Spliting hyphenated words which are not themselves actually ‘words’ (e.g. in *the Apple-IBM joint venture* translate *Apple-IBM* into *Apple - IBM*, but don’t translate *well-known* into *well - known*)
- Handling URLs, e-mail addresses, telephone numbers, SGML, currency amounts, percentages, formulae, numbers sensibly

- Separating contractions into multiple tokens: *you’re* $\rightarrow$ *you ’re*

Occasionally ambiguous: also spelling, typing errors, abbreviatory conventions — ideally return a *lattice* representing all possibilities

th m n worked



Sentence Boundary Detection

Sentence detection: end-of-sentence punctuation marks (., …, ?, !) ambiguous; need context. E.g.

	.	London	?
4 p.m	.	London	?
At 4 p.m	.	London	N
at 4 p.m	.	London	?
at 4 p.m	.	London ,	Y

Some disambiguation techniques:

- regular expression (finite-state) grammars and exception rules
  - language-specific
  - requires only exception lists of abbrevations etc.

- *neural network*, e.g. SATZ system uses part-of-speech probabilities for words in text as input to a neural network trained to label sentence boundaries
  - requires PoS lexicon with frequencies
  - uses most probable parts-of-speech of 3 words before, 3 after
  - easily trainable to new text genres and languages
- *maximum entropy* approach (e.g. Mikheev ACL-98)
  - define a set of features over categories (one or more of abbreviation, proper name, content word, closed-class word) and attributes (capitalised / not) of context words
  - collect and classify training instances
  - compute set of features — or conjunctions of features that best fit training instances

    E.g.

X(+ cap.)	X	p.m	.			N
PREP(+ not cap.)	X	p.m	.	PN	,	Y

Part-of-Speech Tagging

Hard for English: *Buck shot cost double last time prices rose.* — easier in more highly-inflected languages?

Some successful approaches, all 'eliminative' rather than 'constructive':

Probabilistic (HMM) Tagging

Tagsets consist of typically 30–250 extended part-of-speech labels (e.g. verb past tense, plural common noun, comma, ...). Can train either from:

- manually-tagged text, or
- untagged text using a previously existing lexicon

Training maximises sum of probabilities of training sentences →

- lexicon with word-tag frequencies, plus

- n-gram tag transition matrix of probabilities: in practice always bigram ($n = 2$) or trigram ($n = 3$). Zeros replaced by small positive figure so that probability-multiples don't go to zero — sometimes small probabilities are smoothed (e.g. using *Good-Turing* method)

When tagging new text get candidate tags and probabilities for each word, and tag transition scores. Use *Viterbi algorithm* to find best sequence of tags. Gives up to 96–97% correct assignment of tags to words (so a sentence of 25 words would be expected to contain 1 tagging error)

90% accuracy from word freq info, remaining 6–7% from n-grams

N.B. Finite-state automata (FSA) too weak to define natural languages: can't cope with unbounded nested dependencies, e.g. subject of inflected verb may be separated from it by long relative clause.

- But this doesn't imply FSA-type approaches can't do anything useful in NLP. (Defining all-and-only the legal sentences not the ultimate goal)
- Perfect tagging would presumably require full parsing ability; but if

input is 'messy'/semi-ungrammatical, automatic parser may just fall over. An eliminative technique can never fail to yield some analysis, which will may well be correct enough of the time

Robust systems that work on anything may be more use for practical applications than systems that produce perfect results with perfect input and no results with imperfect input

Transformation Rule-based 'Brill' Tagging

Statistical approaches often capture linguistic information in large opaque tables of probabilities: difficult to understand these systems and work out how to improve them. 'Brill' tagging (also parsing etc.) based on transformation rule-based learning:

1. label all words with their most likely tag
2. compare to the 'truth'
3. construct an ordered list of transformations (= rewrite rule + triggering environment) to get nearer the truth

‘change the tag from *MD (modal)* to *NN (noun)* if the preceding tag is *DT (determiner)*’

```
The_DT can_MD rusted_VBD ._.
  → The_DT can_NN rusted_VBD ._.
```

Greedy search: iterate, adding best-scoring transformation to set and updating corpus annotations. Stop when no transformation can improve score

Set of transformations is readable (although interactions between them make them difficult to understand). Demo at <http://bigram.cs.jhu.edu/~demos/tag.html>

Constraint Grammar

Not trained on corpora, but developed manually based almost wholly on corpus evidence

Unordered (non-interacting) collection of rules which eliminate impossible

readings of words in particular contexts, e.g.

‘discard any finite verb reading if the preceding word has a reading with the base form *to*’
(@w =0 VFIN (-1 TO))

‘discard any modal auxiliary reading if the right-hand context contains no infinitive’
(@w =0 AUXMOD (NOT *1 INF))

Can also pick a particular reading as the single correct one in a particular context (much less common). May return multiple labels for words where rules cannot resolve ambiguity

Grammar formalism and impressively accurate wide-coverage system for English (ENGCG) developed at University of Helsinki — demo at <http://www.ling.helsinki.fi/~tapanain/cg/cgdemo.html>

Further refined and used for *shallow parsing* (i.e. assigning subject, main verb, object labels) at Xerox Grenoble, and Conexor Finland <http://www.conexor.fi/analysers.html>

Chunking

Segmenting sentences into syntactically correlated chunks — most commonly focussing on nominal chunks, or *base NPs*. These are just the head noun and preceding material such as determiners and adjectives. E.g.

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only $ 1.8 billion] [PP in] [NP September] .

Both statistical and corpus-based linguistic approaches can work well, with accuracy 90–93% recall and precision for base NPs. See <http://lcg-www.uia.ac.be/conll2000/chunking/>

Finite-state Transducers

NP → D? N* N
VP → V-tns | Aux V-bse

Only the longest match creates an output chunk. If recogniser ‘blocks’

without reaching a final state then starting word is copied to output and recogniser starts again at next word. E.g.

He reckons the current account deficit will narrow to only $ 1.8 billion in September .
N V-tns D N N N Aux V-bse P D N N N P N .
NP –VP ——–NP ———VP P ——–NP P NP .

Memory-based Learning

Classify each word token in training data as the first word of a chunk, the second or subsequent word of a chunk, or not in a chunk. Associate with the classification a feature vector containing e.g. 5 words and PoS tags to the left, the word itself, and 3 words and PoS tags to the right

To run, find nearest learned feature vector to new instance and take its classification. *Distance metric* could simply count number of mismatching feature values, or alternatively take into account differential feature weights

Features are treated as independent. But sometimes combinations of features are good predictors, so could combine features to form complex features. Need search strategy since many possible combinations.

Decision Trees

Create feature vector / classification pairs

V-tns	**D**	N	NP-S (start)
D	**N**	N	NP-I (inside)
.	**N**	V-tns	NP-S
V-tns	**P**	D	NP-O (outside)
.	**P**	N	NP-O

To build the decision tree, select a feature to be the root node. One way to pick root node is to count the 'decisiveness' of each feature. A feature is highly decisive if in every vector with a given value for the feature, classification is the same:

feature	*decisiveness*
previous tag	1/5
current tag	3/5
next tag	2/5

So top node is the current tag feature:

$$\begin{array}{lll} \text{current tag} = & \text{D} \Rightarrow \text{NP-S} & \\ & \text{N} \Rightarrow \text{previous tag} = & \text{D} \Rightarrow \text{NP-I} \\ & & . \Rightarrow \text{NP-S} \\ & \text{P} \Rightarrow \text{NP-O} & \end{array}$$

Preference for picking most decisive attributes first is a type of bias. More general biases in decision tree construction make use of *information theory*.

Named Entity Recognition

Useful as a sub-task of *information extraction*, and also in its own right for document retrieval / routing. Well-established (and detailed) annotation guidelines developed for US MUC (Message Understanding Conference) evaluations

Detect names of people, companies, organisations:

- *Robert Johnson*, *R. Johnson* and *Bob Johnson*
- *Wal-Mart*, *WalMart* and *Wal Mart*
- *Bank of New York*, *BONY* and *BNY* (but not *bony*)

Also places, dates, times of day:

- *Paris / Paris, France / Paris, capital of France*
- *1/1/2000 / Jan. 1st / the first of January, 2000*
- *14:30 / 2.30 p.m. / fourteen-thirty hours GMT*