Classical Approaches to Tagging

ESSLLI 2013: Computational Morphology

Jirka Hana & Anna Feldman

Classical tagging techniques

Overview:

- Intro
- Non-statistical approaches to tagging
- Statistical approaches to tagging:
    - Supervised (HMMs in particular)
    - Unsupervised (only the definition)
- TnT (Brants 2000)
- Evaluation

What is morphological tagging?

- Part-of-speech (POS) tagging is the task of labeling each word in a sentence with its appropriate POS information.
- Morphological tagging is a process of labeling words in a text with their appropriate (in context) detailed morphological information.

Ambiguous word types in the Brown corpus

- Most English words are unambiguous, but many of the most common words are ambiguous
- Ambiguity in the Brown corpus
  - 40% of word tokens are ambiguous
  - 12% of word types are ambiguous
  - Breakdown of ambiguous word types:

Unambiguous (1 tag)	35,340
Ambiguous (2–7 tags)	4,100
2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 (“still”)

How bad is the ambiguity problem?

- One tag is usually much more likely than the others,
    - in the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time.
- A tagger **for English** that simply chooses the most likely tag for each word can achieve good performance.
- Any new approach should be compared against the unigram baseline (assigning each token to its most likely tag)

Ambiguity (cont.)

- Problem 1:
  - Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG.
  - All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN.
  - Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 2500/CD.
- Problem 2:
  - cotton/NN sweater/NN;
  - income-tax/JJ return/NN;
  - the/DT Gramm-Rudman/NP Act/NP.
- Problem 3:
  - They were **married/VBN** by the Justice of the Peace yesterday at 5:00.
  - At the time, she was already **married/JJ**.

Two approaches to POS tagging

1. Rule-based tagging
   - Assign each word in the input a list of potential POS tags, then winnow down this list to a single tag using hand-written disambiguation rules
2. Statistical tagging (can be supervised/unsupervised)
   - Probabilistic: Find the most likely sequence of tags T for words W:

     $$\arg\max_T \; P(T|W)$$

   - Transformation-based (Brill) tagging: Get a training corpus of tagged text, and give it to a machine learning algorithm so it will learn its own tagging rules (as in 1).

Supervised vs. Unsupervised tagging

- *Supervised* taggers
  - rely on pretagged corpora
- *Unsupervised* models
  - do not require a pretagged corpus,
  - cluster words by word properties (their shape and context)
  - completely unsupervised models induce their own 'tagset'; but often a seed of examples for each tag is used

Rule-based POS tagging

English Constraint Grammar approach (e.g., Karlsson et al. 1995) and EngCG tagger (Voutilainen, 1995,1999).

- Thousands of rules are applied in steps
- Each rule either *adds*, *removes*, *selects* or *replaces* a tag or a set of grammatical tags in a given sentence context.
- Context conditions are included, both local (defined distances) or global (undefined distances)
- Context conditions in the same rule may be linked, i.e. conditioned upon each other, negated or blocked by interfering words or tags.

An Example

Pavlov had shown that salivation...

- Stage 1:
  - Pavlov **PAVLOV N NOM SG PROPER**
  - had **HAVE V PAST VFIN SVO** / HAVE PCP2 SVO
  - shown **SHOW PCP2 SVOO SVO SV**
  - that ADV / PRON DEM SG/ DET CENTRAL DEM SG / **CS**
  - salivation **N NOM SG**
- Stage 2: Apply constraints (3,744) (used in a negative way to eliminate tags that inconsistent with the context):

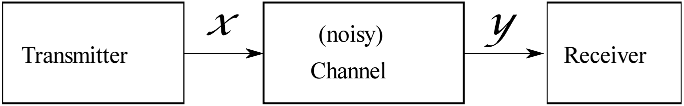
  ADVERBIAL-THAT RULE
  Given input: "that"
  if
  (+1 A/ADV/QANT); if next word is adj, adverb, or quantifier
  (+2 SENT-LIM); and following which is a sentence boundary
  (NOT -1 SVOC/A); and the previous word is not a verb like "consider" which allows adjectives as object complements
  then eliminate non-ADV tags
  else eliminate ADV-tags

Q: How should "that" be analyzed in *I consider that odd.* based on the algorithm?

Noisy Channel

- Tags and words transferred over the noisy channel get corrupted into words
- We want to reconstruct the original message

http://upload.wikimedia.org/wikipedia/commons/4/48/Comm_Channel.svg



Tagging

$W = w_1 \dots w_n$ - words in the corpus (observed)

$T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging

$W = w_1 \ldots w_n$ - words in the corpus (observed)

$T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Bayes rule: $P(T|W) = \frac{P(W|T)*P(T)}{P(W)}$

Tagging

$W = w_1 \dots w_n$ - words in the corpus (observed)

$T = t_1 \dots t_n$ - the corresponding tags (unknown)

Bayes rule: $P(T|W) = \frac{P(W|T)*P(T)}{P(W)}$

tagging = find

$$\text{argmax}_T P(T|W) \tag{1}$$

Tagging

$W = w_1 \dots w_n$ - words in the corpus (observed)

$T = t_1 \dots t_n$ - the corresponding tags (unknown)

Bayes rule: $P(T|W) = \frac{P(W|T)*P(T)}{P(W)}$

tagging = find

$$
\begin{aligned}
& \operatorname{argmax}_T P(T|W) && (1) \\
= \; & \operatorname{argmax}_T \frac{P(W|T) \cdot P(T)}{P(W)} && (2)
\end{aligned}
$$

Tagging

$W = w_1 \dots w_n$ - words in the corpus (observed)

$T = t_1 \dots t_n$ - the corresponding tags (unknown)

Bayes rule: $P(T|W) = \frac{P(W|T) * P(T)}{P(W)}$

tagging = find

$$
\begin{aligned}
& \mathrm{argmax}_T P(T|W) && (1)\\
= \; & \mathrm{argmax}_T \frac{P(W|T) \cdot P(T)}{P(W)} && (2)\\
= \; & \mathrm{argmax}_T P(W|T) \cdot P(T) && (3)
\end{aligned}
$$

Tagging

$W = w_1 \ldots w_n$ - words in the corpus (observed)

$T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Bayes rule: $P(T|W) = \frac{P(W|T)*P(T)}{P(W)}$

tagging = find

$$
\begin{aligned}
& \operatorname{argmax}_T P(T|W) && (1)\\
= \; & \operatorname{argmax}_T \frac{P(W|T) \cdot P(T)}{P(W)} && (2)\\
= \; & \operatorname{argmax}_T P(W|T) \cdot P(T) && (3)\\
= \; & \operatorname{argmax}_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) \cdot P(t_i|t_1 \ldots t_{i-1}) && (4)
\end{aligned}
$$

A simple bigram tagger

Relies on Markov assumption (clearly a simplification)

A simple bigram tagger

Relies on Markov assumption (clearly a simplification)

$$\begin{aligned} & \operatorname{argmax}_T P(T|W) && (5) \\ & \vdots && (6) \\ = \quad & \operatorname{argmax}_T \prod_i P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i) \cdot P(t_i|t_1 \dots t_{i-1}) && (7) \end{aligned}$$

A simple bigram tagger

Relies on Markov assumption (clearly a simplification)

$$
\begin{aligned}
& \text{argmax}_T P(T|W) && (5)\\
& \vdots && (6)\\
= \quad & \text{argmax}_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) \cdot P(t_i|t_1 \ldots t_{i-1}) && (7)\\
\approx \quad & \text{argmax}_T \prod_i P(w_i|t_i) \cdot P(t_i|t_{i-1}) && (8)
\end{aligned}
$$

n-grams are sequences of probabilities based on a limited number of previous categories.

- The bigram model uses $P(t_i|t_{i-1})$ ("first order model")
- The trigram model uses $P(t_i|t_{i-1}, t_{i-2})$ ("second order model")

n-grams

Example text: *a screaming comes across the sky* ($N = 6$)

Unigrams	Bigrams	Trigrams
a		
screaming	a screaming	
comes	screaming comes	a screaming comes
across	comes across	screaming comes across
the	across the	comes across the
sky	the sky	across the sky

Transitions and emissions

- There are two sets of probabilities involved.
  - *Transition probabilities* control the movement from state to state (e.g., $P(t_i|t_{i-1})$)
  - *Emission probabilities* control the emission of output symbols (=words) from the hidden states, e.g., $P(w_i|t_i)$

Sparsity problem

- Standard *n*-gram models must be trained from some corpus
- Any training corpus is finite
- Some perfectly acceptable *n*-grams are bound to be missing from it
- Thus we have a very large number of cases of putative zero-probability *n*-grams that should really have some non-zero

Sparsity problem

- Standard *n*-gram models must be trained from some corpus
- Any training corpus is finite
- Some perfectly acceptable *n*-grams are bound to be missing from it
- Thus we have a very large number of cases of putative zero-probability *n*-grams that should really have some non-zero
- Solution: Smoothing (e.g., Goodman 1996): Assign a non-zero (small) probability to unseen possibilities

TnT tagger (Brants 2000)

- Trigrams'n'Tags (TnT) is a statistical Markov model tagging approach, developed by (Brants 2000).
- Performs very well
- States are tags; outputs are words; transition probabilities depend on the pairs of tags.
- Transitions and output probabilities are estimated from a tagged corpus, using maximum likelihood probabilities, derived from the relative frequencies.

TnT (cont.)

- Special features:
  - *Suffix analysis* for handling unknown words: Tag probabilities are set according to the word's ending because suffixes are word predictors for word classes (e.g., 98% of the words in the Penn Treebank corpus ending in *-able* are adjectives and the rest are nouns).
  - *Capitalization*: probability distributions of tags around capitalized words are different from those not capitalized
  - *Reducing the processing time*
    The processing time of the Viterbi algorithm is reduced by introducing a beam search. While the Viterbi algorithm is guaranteed to find the sequence of states with the highest probability, this is no longer true when beam search is added.

Evaluating POS taggers

- Taggers are evaluated by comparing them with a 'gold standard' (human-labeled) test set, based on percent correct: the percentage of all tags in the test set where the tagger and the gold standard agree
- Most current taggers get about 96% correct (for English)
- Note, however, that human experts don't always agree on the correct tag, which means the 'gold standard' is likely to have errors and 100% accuracy is impossible

Measures of success

The following measures are typically used for evaluating the performance of a tagger:

- Precision $= \frac{\text{Correctly-Tagged-Tokens}}{\text{Tags-generated}}$
  - Precision measures the percentage of predicted tags that were correct.
- Recall $= \frac{\text{Correctly-Tagged-Tokens}}{\text{Tokens-in-data}}$
  - Recall measures the percentage of tags actually present in the input that were correctly identified by the system.
- F-measure $= 2 * \frac{\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$
  - The F-measure provides a way to combine these two measures into a single metric.