POS tagging (3)

LING 570

Fei Xia

Week 9: 11/23/09

Outline

- POS tagging with a classifier
- Sequence labeling problem
- Beam search

N-gram POS tagger

$$argmax_{t_1^n} P(t_1^n | w_1^n)$$

$$\approx argmax_{t_1^n} \prod_i P(w_i|t_i) P(t_i|t_{i-N+1}^{i-1})$$

Bigram model: $$\prod_i P(w_i|t_i) P(t_i|t_{i-1})$$

Trigram model: $$\prod_i P(w_i|t_i) P(t_i|t_{i-2}, t_{i-1})$$

Cues for unknown words

- Affixes: unforgettable: un-, -able ➔ JJ
- Capitalization: Hyderabad ➔ NNP
- Word shapes: 123,456 ➔ CD
- The previous word: San _ ➔ NNP

How can we take advantage of these cues?

➔ Treat them as features

An example

- I am going to San Diego next week
- San NNP IsCap 1 PrevW=to 1 ContainNum 0
- Diego NNP IsCap 1 PrevW=San 1 ContainNum 0

Feature templates for POS tagging

- Prev word: w_{-1}
- Current word: w_0
- Next word: w_{+1}
- Prev two words: w_{-2} w_{-1}
- Surrounding words: w_{-1} w_{+1}

- Prev tag: t_{-1}
- Prev two tags: t_{-2} t_{-1}

An example

Mary will come tomorrow

	W_{-1}	W_0	$W_{-1}\ W_0$	W_{+1}	t_{-1}	y
x1 (Mary)	<s>	Mary	<s> Mary	will	BOS	PN
x2 (will)	Mary	will	Mary will	come	PN	V
x3 (come)	will	come	will come	tomorrow	V	V

This can be seen as a shorthand of a much bigger table.

	W_{-1}	W_0	$W_{-1}\ W_0$	W_{+1}	t_{-1}	y
x1 (Mary)	<s>	Mary	<s> Mary	will	BOS	PN
x2 (will)	Mary	will	Mary will	come	PN	V
x3 (come)	will	come	will come	tomorrow	V	V

Mary PN prevW=<s> 1 curW=Mary 1 prevW-curW=<s>-Mary 1
nextW=will 1 prevTag=BOS 1

will V prevW=Mary 1 curW=will 1 prevW-curW=Mary-will 1
nextW=come 1 prevTag=PN 1

come V prevW=will 1 curW=come 1 prevW-curW=will-come 1
nextW=tomorrow 1 prevTag=V 1

Feature templates for POS tagging

- Prev word: w_{-1}
- Current word: w_0
- Next word: w_{+1}
- Prev two words: w_{-2} w_{-1}
- Surrounding words: w_{-1} w_{+1}

- Prev tag: t_{-1}
- Prev two tags: t_{-2} t_{-1}

- How many feature templates?
- How many features? $3|V|+2|V|^2+|T|+|T|^2$

What about unknown words?

Condition	Features
w_i is not rare	$w_i = X$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$
	X is suffix of w_i, $\|X\| \leq 4$
	w_i contains number
	w_i contains uppercase character
	w_i contains hyphen
$\forall\ w_i$	$t_{i-1} = X$
	$t_{i-2}t_{i-1} = XY$
	$w_{i-1} = X$
	$w_{i-2} = X$
	$w_{i+1} = X$
	$w_{i+2} = X$

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

well-heeled JJ pref=w 1 pref=we 1 pref=wel 1 pref=well 1
suf=d 1 suf=ed 1 suf=led 1 suf=eled 1
containsNum 0 containsUppercase 0 containshyphen 1
prevTag=IN 1 prev2Tags=NNS-IN 1 prefW=about 1
pref2W=stories 1 nextW=communities 1 next2W=and 1

Rare words: words that occur less than N_r times in the training data

Feature selection: remove features that appear less than N_f times in the training data

Building a tagger

- training data: w1/t1 w2/t2 … wn/tn
- test data: w1/t1 w2/t2   … wn/tn

- Create train.vectors.txt from training data
- Create test.vectors.txt from test data
- Run info2vectors to convert the vectors into binary format
- Train a model using train.vectors:
  - vectors2train –training-file train.vectors –trainer MaxEnt –output-classifier me_model --report train:accuracy train:confusion > me.stdout 2>me.stderr
- Run the model on test.vectors:
  - classify --testing-file test.vectors --classifier me_model --report test:accuracy test:confusion test:raw > me_dec.stdout 2>me_dec.stderr

	W_{-1}	W_0	W_{-1} W_0	W_{+1}	t_{-1}	y
x1 (Mary)	<s>	Mary	<s> Mary	will	BOS	PN
x2 (will)	Mary	will	Mary will	come	PN	V
x3 (come)	will	come	will come	tomorrow	V	V

Mary PN prevW=<s> 1 curW=Mary 1 prevW-curW=<s>-Mary 1
nextW=will 1 prevTag=BOS 1

will V prevW=Mary 1 curW=will 1 prevW-curW=Mary-will 1
nextW=come 1 prevTag=PN 1

come V prevW=will 1 curW=come 1 prevW-curW=will-come 1
nextW=tomorrow 1 prevTag=V 1

Hw9: Build a maxent tagger

- Q1: maxent_tagger.sh train_file test_file rare_thres feat_thres output_dir
- train_file and test_file: w1/t1 … wn/tn

- Main steps:
  - Create feature vectors for train_file and test_file
  - Run info2vectors to convert feature vectors into binary format
  - Run vectors2train to create a MaxEnt model
  - Run classify to tag the test data

Creating feature vectors

- Features: Table 1 in (Ratnaparkhi, 1996)

- Use rare_thres to identify rare words in the training data and in the test data
- Remove low-frequency features from the feature vectors using feat_thres.
- Replace ``,'' with ``comma'' as Mallet treats ``,'' as a delimiter.

Q2: tagging results

Expt id	rare thres	feat thres	training accuracy	test accuracy	# of feats	# of kept feats	running time
1_1	1	1					
1_3	1	3					
2_1	2	1					
2_3	2	3					
3_3	3	3					
3_5	3	5					
5_10	5	10					

Output files

- Store under res_id/ (e.g., res_1_1/)
  - train_voc: “word freq”
  - train.vectors.feat: “featName freq”
  - kept_feats: “featName freq”
  - final_train.vectors.txt and final_test.vectors.txt
  - me_model: MaxEnt model

- Submission:
  - gzip hw9.tar
  - If still too big, let David know the location of the tar file on patas.

Sequence labeling problem

Sequence labeling problem

- Task: to find the most probable labeling of a sequence.

- Examples:
  - POS tagging
  - NP chunking
  - NE tagging
  - Word segmentation
  - Table detection
  - …

Questions

- Training data: $\{(x_i, y_i)\}$
- What is x_i? What is y_i?
- What are the features?
- How to convert x_i to a feature vector for training data? How to do that for test data?

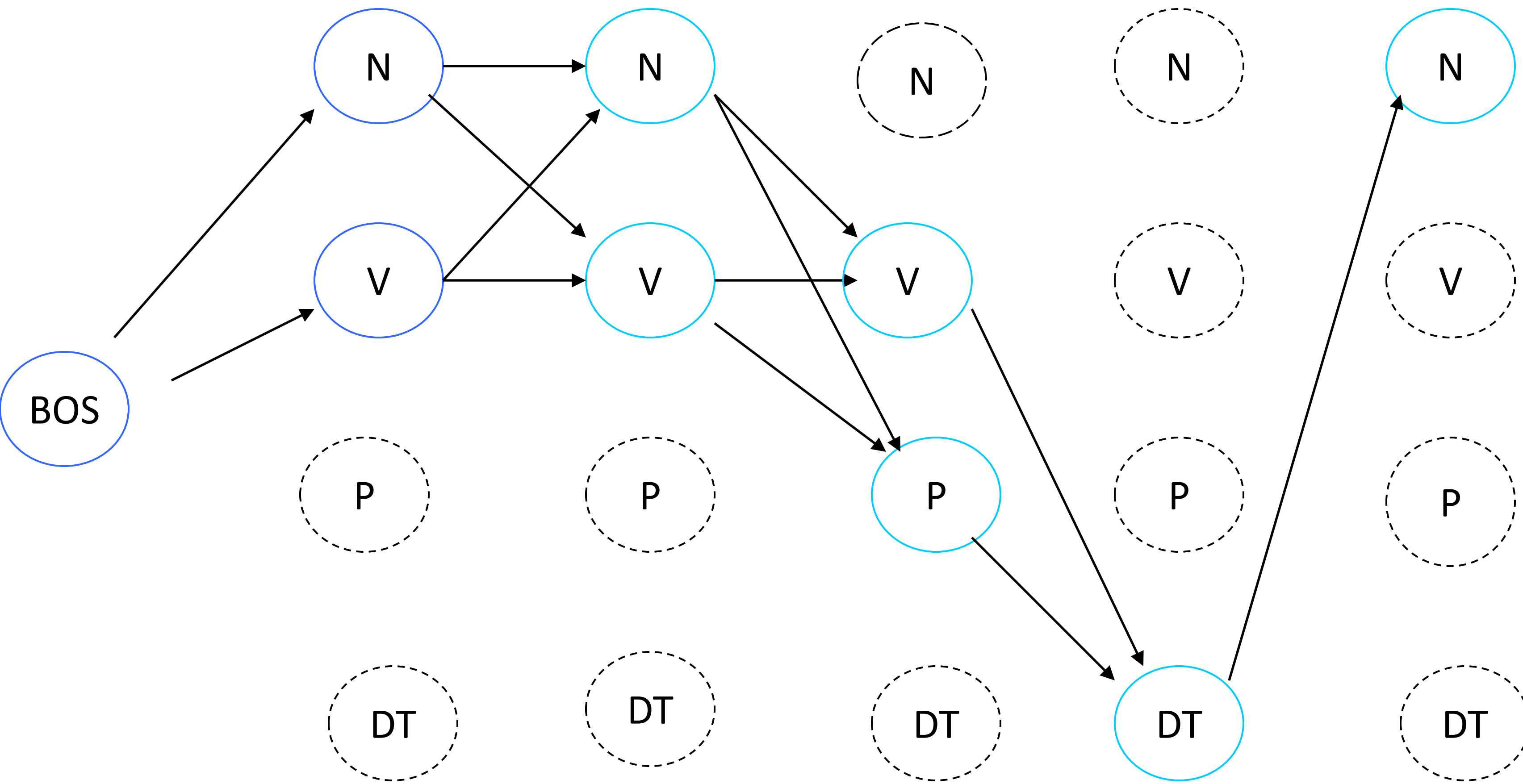
How to solve a sequence labeling problem?

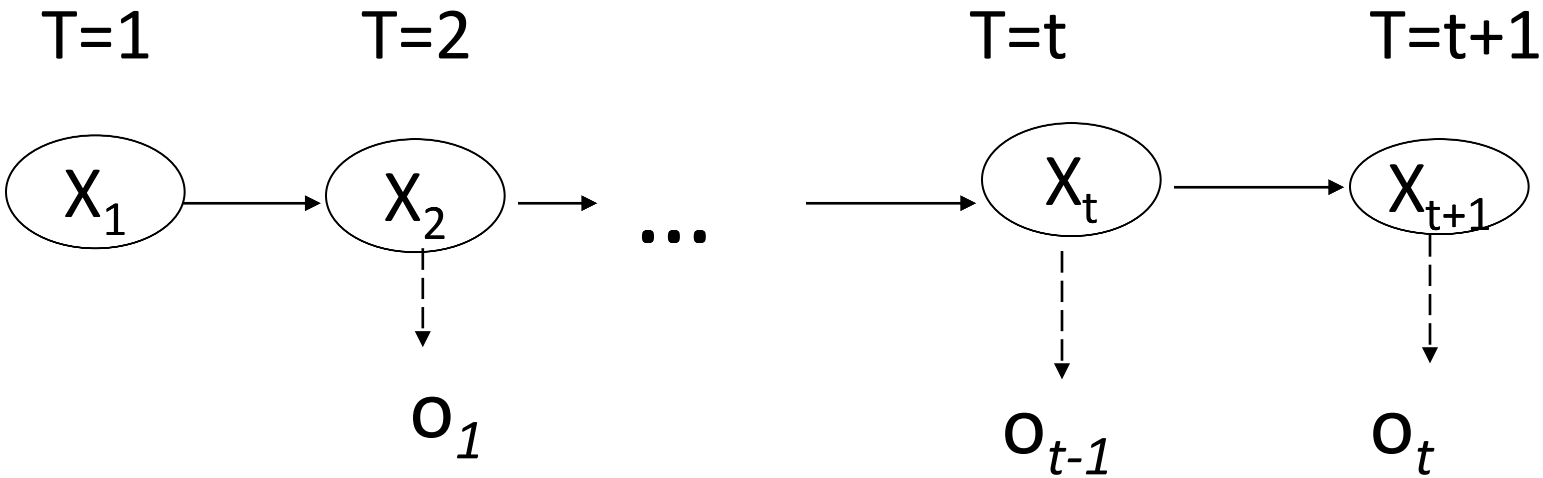
- Using a sequence labeling algorithm: e.g., HMM

- Using a classification algorithm
  - Don't use features that refer to class labels
  - Use those features and get their values by running other processes
  - Use those features and find a good (global) solution.

Viterbi for HMM

time flies like an arrow





$$\delta_j(t) \stackrel{def}{=} \max_{X_{1,t-1}} P(X_{1,t-1}, O_{1,t-1}, X_t = j)$$

$$\delta_j(1) = \pi_j$$

$$\delta_j(t+1) = \max_i \delta_i(t) a_{ij} b_{jo_t}$$

Time complexity: $O(N^2 T)$

Beam search

Why do we need beam search?

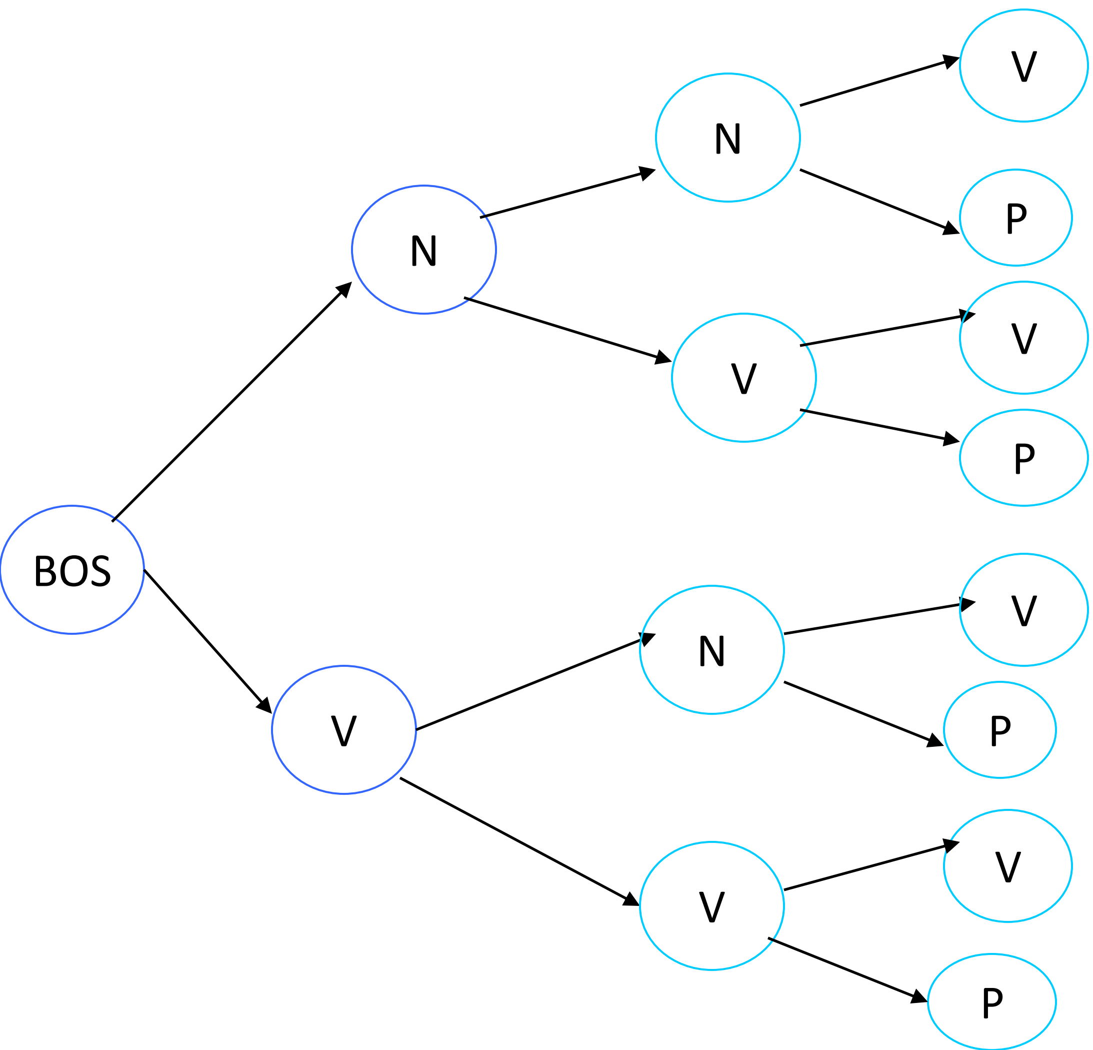
- Features refer to tags of previous words, which are not available for the TEST data.

- Knowing only the best tag of the previous word is not good enough.

- So let's keep multiple tag sequences available during the decoding.

Beam search

time flies like an arrow



Beam search

- Generate m tags for w_1, set s_{1j} accordingly

- For i=2 to n (n is the sentence length)
  - Expanding: For each surviving sequence $s_{(i-1),j}$
    - Generate m tags for w_i, given $s_{(i-1)j}$ as previous tag context
    - Append each tag to $s_{(i-1)j}$ to make a new sequence.
  - Pruning: keep only the top k sequences

- Return highest prob sequence s_{n1}.

Beam search (basic)

- Beam inference:
  - At each position keep the top k complete sequences.
  - Extend each sequence in each local way.
  - The extensions compete for the k slots at the next position.
- Advantages:
  - Fast; and beam sizes of 3–5 are as good or almost as good as exact inference in many cases.
  - Easy to implement (no dynamic programming required).
- Disadvantage:
  - Inexact: the globally best sequence can fall off the beam.

Viterbi search

- Viterbi inference:
  - Dynamic programming or memoization.
  - Requires small window of state influence (e.g., past two states are relevant).
- Advantage:
  - Exact: the global best sequence is returned.
- Disadvantage:
  - Harder to implement long-distance state-state interactions (but beam inference tends not to allow long-distance resurrection of sequences anyway).

Viterbi vs. Beam search

- DP vs. heuristic search
- Global optimal vs. inexact
- Small window vs. big window for features

Summary

- POS tagging with a classifier: use a classifier to determine the class of the word
- Sequence labeling problem: the feature of the current word depends on the tags of previous words
- Beam search: brute-force search with pruning